```
import Combine
import CoreData.CloudKit
import CoreData.CoreDataDefines
import CoreData.CoreDataErrors
import CoreData.NSAtomicStore
import CoreData.NSAtomicStoreCacheNode
import CoreData.NSAttributeDescription
import CoreData.NSBatchDeleteRequest
import CoreData.NSBatchInsertRequest
import CoreData.NSBatchUpdateRequest
import
CoreData.NSCompositeAttributeDescription
import
CoreData.NSCoreDataCoreSpotlightDelegate
import CoreData.NSCustomMigrationStage
import
CoreData.NSDerivedAttributeDescription
import CoreData.NSEntityDescription
import CoreData.NSEntityMapping
import CoreData.NSEntityMigrationPolicy
import CoreData.NSExpressionDescription
import CoreData.NSFetchIndexDescription
import
CoreData.NSFetchIndexElementDescription
import CoreData.NSFetchRequest
import CoreData.NSFetchRequestExpression
import
CoreData.NSFetchedPropertyDescription
import
CoreData.NSFetchedResultsController
import CoreData.NSIncrementalStore
import CoreData.NSIncrementalStoreNode
import
```

```
CoreData.NSLightweightMigrationStage
import CoreData.NSManagedObject
import CoreData.NSManagedObjectContext
import CoreData.NSManagedObjectID
import CoreData.NSManagedObjectModel
import
CoreData.NSManagedObjectModelReference
import CoreData.NSMappingModel
import CoreData.NSMergePolicy
import CoreData.NSMigrationManager
import CoreData.NSMigrationStage
import
CoreData.NSPersistentCloudKitContainer
import
CoreData.NSPersistentCloudKitContainerEve
nt
import
CoreData.NSPersistentCloudKitContainerEve
ntRequest
import
CoreData.NSPersistentCloudKitContainerOpt
ions
import CoreData.NSPersistentContainer
import CoreData.NSPersistentHistoryChange
import
CoreData.NSPersistentHistoryChangeRequest
import CoreData.NSPersistentHistoryToken
import
CoreData.NSPersistentHistoryTransaction
import CoreData.NSPersistentStore
import
CoreData.NSPersistentStoreCoordinator
import
```

```swift
CoreData.NSPersistentStoreDescription
import CoreData.NSPersistentStoreRequest
import CoreData.NSPersistentStoreResult
import CoreData.NSPropertyDescription
import CoreData.NSPropertyMapping
import CoreData.NSQueryGenerationToken
import CoreData.NSRelationshipDescription
import CoreData.NSSaveChangesRequest
import CoreData.NSStagedMigrationManager
import Foundation
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available(swift 5.8)
@available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
extension NSStagedMigrationManager {

    @available(swift 5.8)
    @available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
    public convenience init(_ stages:
[NSMigrationStage])
}

@available(macOS 10.12, iOS 10.0, tvOS
10.0, watchOS 3.0, *)
extension NSManagedObject {

    @objc dynamic public class func
fetchRequest() -> NSFetchRequest<any
NSFetchRequestResult>
```

```
}

@available(macOS 10.15, iOS 13.0, tvOS
13.0, watchOS 6.0, *)
extension NSManagedObject :
ObservableObject {

    /// A publisher that emits before the
object has changed.
    public var objectWillChange:
ObservableObjectPublisher { get }

    /// The type of publisher that emits
before the object has changed.
    @available(iOS 13.0, tvOS 13.0,
watchOS 6.0, macOS 10.15, *)
    public typealias
ObjectWillChangePublisher =
ObservableObjectPublisher
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, watchOS 11.0, *)
extension NSManagedObject {

    public subscript(key: String) -> Any?
}

@available(swift 5.8)
@available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
extension NSLightweightMigrationStage {
    @available(swift 5.8)
    @available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
    public convenience init(_ checksums:
[String])
}

@available(iOS 15.0, macOS 12.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension NSCoreDataCoreSpotlightDelegate
{

    public static let
indexDidUpdateNotification:
Notification.Name
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSMigrationManager {

    public func migrateStore(from
sourceURL: URL, type sourceType:
NSPersistentStore.StoreType, options
sourceOptions: [AnyHashable : Any]? =
nil, mapping: NSMappingModel, to
destinationURL: URL, type
destinationType:
NSPersistentStore.StoreType, options
destinationOptions: [AnyHashable : Any]?
= nil) throws
}
```

```
@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSPersistentStore {

    public struct StoreType : Hashable,
Equatable, RawRepresentable {

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
        ///
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue: String
        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
value to use for the new instance.
        public init(rawValue: String)

        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
```

```
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
        public typealias RawValue =
String
    }
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
extension NSPersistentStoreCoordinator {

    public func managedObjectID(for
string: String) -> NSManagedObjectID?
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSPersistentStoreCoordinator {

    public func addPersistentStore(type:
NSPersistentStore.StoreType,
configuration: String? = nil, at
storeURL: URL, options: [AnyHashable :
Any]? = nil) throws -> NSPersistentStore

    public class func
registerStoreClass(_ storeClass:
```

```
AnyClass?, type:
NSPersistentStore.StoreType)

    public class func
metadataForPersistentStore(type
storeType: NSPersistentStore.StoreType,
at storeURL: URL, options: [AnyHashable :
Any]? = nil) throws -> [String : Any]

    public class func setMetadata(_
metadata: [String : Any]?, type
storeType: NSPersistentStore.StoreType,
at storeURL: URL, options: [AnyHashable :
Any]? = nil) throws

    public func migratePersistentStore(_
store: NSPersistentStore, to storeURL:
URL, options: [AnyHashable : Any]? = nil,
type storeType:
NSPersistentStore.StoreType) throws ->
NSPersistentStore

    public func destroyPersistentStore(at
url: URL, type storeType:
NSPersistentStore.StoreType, options:
[AnyHashable : Any]? = nil) throws

    public func replacePersistentStore(at
destinationURL: URL, destinationOptions:
[AnyHashable : Any]? = nil,
withPersistentStoreFrom sourceURL: URL,
sourceOptions: [AnyHashable : Any]? =
nil, type sourceType:
```

```
NSPersistentStore.StoreType) throws
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSPersistentStoreCoordinator {

    @available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
    public func performAndWait<T>(_
block: () throws -> T) rethrows -> T

    @available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
    public func perform<T>(_ block:
@escaping () throws -> T) async rethrows
-> T
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension CocoaError.Code {

    public static var
managedObjectValidation: CocoaError.Code
{ get }

    public static var
validationMultipleErrors: CocoaError.Code
{ get }

    public static var
validationMissingMandatoryProperty:
```

```
CocoaError.Code { get }

    public static var
validationRelationshipLacksMinimumCount:
CocoaError.Code { get }

    public static var
validationRelationshipExceedsMaximumCount
: CocoaError.Code { get }

    public static var
validationRelationshipDeniedDelete:
CocoaError.Code { get }

    public static var
validationNumberTooLarge: CocoaError.Code
{ get }

    public static var
validationNumberTooSmall: CocoaError.Code
{ get }

    public static var
validationDateTooLate: CocoaError.Code {
get }

    public static var
validationDateTooSoon: CocoaError.Code {
get }

    public static var
validationInvalidDate: CocoaError.Code {
get }
```

```
    public static var
validationStringTooLong: CocoaError.Code
{ get }

    public static var
validationStringTooShort: CocoaError.Code
{ get }

    public static var
validationStringPatternMatching:
CocoaError.Code { get }

    public static var
managedObjectContextLocking:
CocoaError.Code { get }

    public static var
persistentStoreCoordinatorLocking:
CocoaError.Code { get }

    public static var
managedObjectReferentialIntegrity:
CocoaError.Code { get }

    public static var
managedObjectExternalRelationship:
CocoaError.Code { get }

    public static var managedObjectMerge:
CocoaError.Code { get }

    public static var
```

```
managedObjectConstraintMerge:
CocoaError.Code { get }

    public static var
persistentStoreInvalidType:
CocoaError.Code { get }

    public static var
persistentStoreTypeMismatch:
CocoaError.Code { get }

    public static var
persistentStoreIncompatibleSchema:
CocoaError.Code { get }

    public static var
persistentStoreSave: CocoaError.Code {
get }

    public static var
persistentStoreIncompleteSave:
CocoaError.Code { get }

    public static var
persistentStoreSaveConflicts:
CocoaError.Code { get }

    public static var coreData:
CocoaError.Code { get }

    public static var
persistentStoreOperation: CocoaError.Code
{ get }
```

```
    public static var
persistentStoreOpen: CocoaError.Code {
get }

    public static var
persistentStoreTimeout: CocoaError.Code {
get }

    public static var
persistentStoreUnsupportedRequestType:
CocoaError.Code { get }

    public static var
persistentStoreIncompatibleVersionHash:
CocoaError.Code { get }

    public static var migration:
CocoaError.Code { get }

    public static var migrationCancelled:
CocoaError.Code { get }

    public static var
migrationMissingSourceModel:
CocoaError.Code { get }

    public static var
migrationMissingMappingModel:
CocoaError.Code { get }

    public static var
migrationManagerSourceStore:
```

```
CocoaError.Code { get }

    public static var
migrationManagerDestinationStore:
CocoaError.Code { get }

    public static var
entityMigrationPolicy: CocoaError.Code {
get }

    public static var sqlite:
CocoaError.Code { get }

    public static var
inferredMappingModel: CocoaError.Code {
get }

    public static var
externalRecordImport: CocoaError.Code {
get }
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension CocoaError.Code {

    @available(*, deprecated, renamed:
"managedObjectValidation")
    public static var
managedObjectValidationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
```

```
"validationMultipleErrors")
    public static var
validationMultipleErrorsError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationMissingMandatoryProperty")
    public static var
validationMissingMandatoryPropertyError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipLacksMinimumCount"
)
    public static var
validationRelationshipLacksMinimumCountEr
ror: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipExceedsMaximumCoun
t")
    public static var
validationRelationshipExceedsMaximumCount
Error: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipDeniedDelete")
    public static var
validationRelationshipDeniedDeleteError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationNumberTooLarge")
```

```
    public static var
validationNumberTooLargeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationNumberTooSmall")
    public static var
validationNumberTooSmallError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationDateTooLate")
    public static var
validationDateTooLateError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationDateTooSoon")
    public static var
validationDateTooSoonError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationInvalidDate")
    public static var
validationInvalidDateError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationStringTooLong")
    public static var
validationStringTooLongError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"validationStringTooShort")
    public static var
validationStringTooShortError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationStringPatternMatching")
    public static var
validationStringPatternMatchingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectContextLocking")
    public static var
managedObjectContextLockingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreCoordinatorLocking")
    public static var
persistentStoreCoordinatorLockingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectReferentialIntegrity")
    public static var
managedObjectReferentialIntegrityError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectExternalRelationship")
```

```
    public static var
managedObjectExternalRelationshipError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectMerge")
    public static var
managedObjectMergeError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"managedObjectConstraintMerge")
    public static var
managedObjectConstraintMergeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreInvalidType")
    public static var
persistentStoreInvalidTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreTypeMismatch")
    public static var
persistentStoreTypeMismatchError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreIncompatibleSchema")
    public static var
persistentStoreIncompatibleSchemaError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"persistentStoreSave")
    public static var
persistentStoreSaveError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"persistentStoreIncompleteSave")
    public static var
persistentStoreIncompleteSaveError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreSaveConflicts")
    public static var
persistentStoreSaveConflictsError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"coreData")
    public static var coreDataError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreOperation")
    public static var
persistentStoreOperationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreOpen")
    public static var
```

```
persistentStoreOpenError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"persistentStoreTimeout")
    public static var
persistentStoreTimeoutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreUnsupportedRequestType")
    public static var
persistentStoreUnsupportedRequestTypeErro
r: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreIncompatibleVersionHash")
    public static var
persistentStoreIncompatibleVersionHashErr
or: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migration")
    public static var migrationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationCancelled")
    public static var
migrationCancelledError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
```

```
"migrationMissingSourceModel")
    public static var
migrationMissingSourceModelError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationMissingMappingModel")
    public static var
migrationMissingMappingModelError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationManagerSourceStore")
    public static var
migrationManagerSourceStoreError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationManagerDestinationStore")
    public static var
migrationManagerDestinationStoreError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"entityMigrationPolicy")
    public static var
entityMigrationPolicyError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"sqlite")
    public static var sqliteError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"inferredMappingModel")
    public static var
inferredMappingModelError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"externalRecordImport")
    public static var
externalRecordImportError:
CocoaError.Code { get }
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension CocoaError {

    public static var
managedObjectValidation: CocoaError.Code
{ get }

    public static var
validationMultipleErrors: CocoaError.Code
{ get }

    public static var
validationMissingMandatoryProperty:
CocoaError.Code { get }

    public static var
validationRelationshipLacksMinimumCount:
CocoaError.Code { get }
```

```
    public static var
validationRelationshipExceedsMaximumCount
: CocoaError.Code { get }

    public static var
validationRelationshipDeniedDelete:
CocoaError.Code { get }

    public static var
validationNumberTooLarge: CocoaError.Code
{ get }

    public static var
validationNumberTooSmall: CocoaError.Code
{ get }

    public static var
validationDateTooLate: CocoaError.Code {
get }

    public static var
validationDateTooSoon: CocoaError.Code {
get }

    public static var
validationInvalidDate: CocoaError.Code {
get }

    public static var
validationStringTooLong: CocoaError.Code
{ get }
```

```
    public static var
validationStringTooShort: CocoaError.Code
{ get }

    public static var
validationStringPatternMatching:
CocoaError.Code { get }

    public static var
managedObjectContextLocking:
CocoaError.Code { get }

    public static var
persistentStoreCoordinatorLocking:
CocoaError.Code { get }

    public static var
managedObjectReferentialIntegrity:
CocoaError.Code { get }

    public static var
managedObjectExternalRelationship:
CocoaError.Code { get }

    public static var managedObjectMerge:
CocoaError.Code { get }

    public static var
managedObjectConstraintMerge:
CocoaError.Code { get }

    public static var
persistentStoreInvalidType:
```

```
CocoaError.Code { get }

    public static var
persistentStoreTypeMismatch:
CocoaError.Code { get }

    public static var
persistentStoreIncompatibleSchema:
CocoaError.Code { get }

    public static var
persistentStoreSave: CocoaError.Code {
get }

    public static var
persistentStoreIncompleteSave:
CocoaError.Code { get }

    public static var
persistentStoreSaveConflicts:
CocoaError.Code { get }

    public static var coreData:
CocoaError.Code { get }

    public static var
persistentStoreOperation: CocoaError.Code
{ get }

    public static var
persistentStoreOpen: CocoaError.Code {
get }
```

```
    public static var
persistentStoreTimeout: CocoaError.Code {
get }

    public static var
persistentStoreUnsupportedRequestType:
CocoaError.Code { get }

    public static var
persistentStoreIncompatibleVersionHash:
CocoaError.Code { get }

    public static var migration:
CocoaError.Code { get }

    public static var migrationCancelled:
CocoaError.Code { get }

    public static var
migrationMissingSourceModel:
CocoaError.Code { get }

    public static var
migrationMissingMappingModel:
CocoaError.Code { get }

    public static var
migrationManagerSourceStore:
CocoaError.Code { get }

    public static var
migrationManagerDestinationStore:
CocoaError.Code { get }
```

```
    public static var
entityMigrationPolicy: CocoaError.Code {
get }

    public static var sqlite:
CocoaError.Code { get }

    public static var
inferredMappingModel: CocoaError.Code {
get }

    public static var
externalRecordImport: CocoaError.Code {
get }
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension CocoaError {

    @available(*, deprecated, renamed:
"managedObjectValidation")
    public static var
managedObjectValidationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationMultipleErrors")
    public static var
validationMultipleErrorsError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"validationMissingMandatoryProperty")
    public static var
validationMissingMandatoryPropertyError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipLacksMinimumCount"
)
    public static var
validationRelationshipLacksMinimumCountEr
ror: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipExceedsMaximumCoun
t")
    public static var
validationRelationshipExceedsMaximumCount
Error: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationRelationshipDeniedDelete")
    public static var
validationRelationshipDeniedDeleteError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationNumberTooLarge")
    public static var
validationNumberTooLargeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
```

```
"validationNumberTooSmall")
    public static var
validationNumberTooSmallError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationDateTooLate")
    public static var
validationDateTooLateError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationDateTooSoon")
    public static var
validationDateTooSoonError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationInvalidDate")
    public static var
validationInvalidDateError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationStringTooLong")
    public static var
validationStringTooLongError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationStringTooShort")
    public static var
validationStringTooShortError:
```

```
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"validationStringPatternMatching")
    public static var
validationStringPatternMatchingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectContextLocking")
    public static var
managedObjectContextLockingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreCoordinatorLocking")
    public static var
persistentStoreCoordinatorLockingError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectReferentialIntegrity")
    public static var
managedObjectReferentialIntegrityError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"managedObjectExternalRelationship")
    public static var
managedObjectExternalRelationshipError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
```

```
"managedObjectMerge")
    public static var
managedObjectMergeError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"managedObjectConstraintMerge")
    public static var
managedObjectConstraintMergeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreInvalidType")
    public static var
persistentStoreInvalidTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreTypeMismatch")
    public static var
persistentStoreTypeMismatchError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreIncompatibleSchema")
    public static var
persistentStoreIncompatibleSchemaError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreSave")
    public static var
persistentStoreSaveError: CocoaError.Code
```

```
{ get }

    @available(*, deprecated, renamed:
"persistentStoreIncompleteSave")
    public static var
persistentStoreIncompleteSaveError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreSaveConflicts")
    public static var
persistentStoreSaveConflictsError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"coreData")
    public static var coreDataError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreOperation")
    public static var
persistentStoreOperationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreOpen")
    public static var
persistentStoreOpenError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"persistentStoreTimeout")
```

```
    public static var
persistentStoreTimeoutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreUnsupportedRequestType")
    public static var
persistentStoreUnsupportedRequestTypeErro
r: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"persistentStoreIncompatibleVersionHash")
    public static var
persistentStoreIncompatibleVersionHashErr
or: CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migration")
    public static var migrationError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationCancelled")
    public static var
migrationCancelledError: CocoaError.Code
{ get }

    @available(*, deprecated, renamed:
"migrationMissingSourceModel")
    public static var
migrationMissingSourceModelError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"migrationMissingMappingModel")
    public static var
migrationMissingMappingModelError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationManagerSourceStore")
    public static var
migrationManagerSourceStoreError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"migrationManagerDestinationStore")
    public static var
migrationManagerDestinationStoreError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"entityMigrationPolicy")
    public static var
entityMigrationPolicyError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"sqlite")
    public static var sqliteError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"inferredMappingModel")
    public static var
inferredMappingModelError:
```

```
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"externalRecordImport")
    public static var
externalRecordImportError:
CocoaError.Code { get }
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension CocoaError {

    /// Object that failed to validate
for a validation error.
    public var validationObject: Any? {
get }

    /// Key that failed to validate for a
validation error.
    public var validationKey: String? {
get }

    /// For predicate-based validation,
the predicate for the condition
    /// that failed to validate.
    public var validationPredicate:
NSPredicate? { get }

    /// The value for the key that failed
to validate for a validation
    /// error.
    public var validationValue: Any? {
```

```
get }

    /// Stores prompting an error.
    public var affectedStores:
[AnyObject]? { get }

    /// Objects prompting an error.
    public var affectedObjects:
[AnyObject]? { get }

    /// The conflicts that arise from a
persistent store.
    public var
persistentStoreSaveConflicts:
[NSMergeConflict]? { get }
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSPersistentContainer {

    public class var defaultDirectoryURL:
URL { get }
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSPersistentContainer {

    @available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
    public func
performBackgroundTask<T>(_ block:
```

```
@escaping (NSManagedObjectContext) throws
-> T) async rethrows -> T
}

@available(swift 5.8)
@available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
extension NSCustomMigrationStage {

    @available(swift 5.8)
    @available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
    public convenience init(migratingFrom
currentModel:
NSManagedObjectModelReference, to
nextModel: NSManagedObjectModelReference)

    @available(swift 5.8)
    @available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
    public var willMigrateHandler: ((_
migrationManager:
NSStagedMigrationManager, _
migrationStage: NSCustomMigrationStage)
throws -> Void)?

    @available(swift 5.8)
    @available(macOS 14, iOS 17, tvOS 17,
watchOS 10, *)
    public var didMigrateHandler: ((_
migrationManager:
NSStagedMigrationManager, _
migrationStage: NSCustomMigrationStage)
```

```
throws -> Void)?
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSExpressionDescription {

    public var resultType:
NSAttributeDescription.AttributeType
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSAttributeDescription {

    public struct AttributeType :
Hashable, Equatable, RawRepresentable {

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
```

```
        ///
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue:
NSAttributeType

        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
value to use for the new instance.
        public init(rawValue:
NSAttributeType)

        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
        public typealias RawValue =
NSAttributeType
    }

    public var type:
NSAttributeDescription.AttributeType
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension NSManagedObjectContext {

    @available(macOS 10.4, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public func fetch<T>(_ request:
NSFetchRequest<T>) throws -> [T] where
T : NSFetchRequestResult

    @available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
    public func fetch(_ request:
NSFetchRequest<any NSFetchRequestResult>)
throws -> [Any]

    @available(macOS 10.5, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public func count<T>(for request:
NSFetchRequest<T>) throws -> Int where
T : NSFetchRequestResult
}

@available(macOS 10.4, iOS 3.0, tvOS 3.0,
watchOS 1.0, *)
extension NSManagedObjectContext {

    @available(macOS 10.5, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public static let
willSaveObjectsNotification:
Notification.Name

    @available(macOS 10.4, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public static let
didSaveObjectsNotification:
Notification.Name
```

```
    @available(macOS 10.4, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public static let
didChangeObjectsNotification:
Notification.Name

    @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
    public static let
didSaveObjectIDsNotification:
Notification.Name

    @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
    public static let
didMergeChangesObjectIDsNotification:
Notification.Name

    @available(macOS 10.4, iOS 3.0, tvOS
3.0, watchOS 1.0, *)
    public enum NotificationKey : String
{

        @available(macOS 10.12, iOS 10.0,
tvOS 10.0, watchOS 3.0, *)
        case queryGeneration

        @available(macOS 10.5, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case invalidatedAllObjects

        @available(macOS 10.4, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case insertedObjects

        @available(macOS 10.4, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case updatedObjects

        @available(macOS 10.4, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case deletedObjects

        @available(macOS 10.5, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case refreshedObjects

        @available(macOS 10.5, iOS 3.0,
tvOS 3.0, watchOS 1.0, *)
        case invalidatedObjects

        @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
        case insertedObjectIDs

        @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
        case updatedObjectIDs

        @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
        case deletedObjectIDs

        @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
```

```
        case refreshedObjectIDs

        @available(macOS 10.12, iOS 10.3,
tvOS 10.2, watchOS 3.2, *)
        case invalidatedObjectIDs

        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
value to use for the new instance.
        public init?(rawValue: String)
        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 3.0, tvOS 3.0,
watchOS 1.0, macOS 10.4, *)
        public typealias RawValue =
String

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
        ///
```

```
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue: String { get
}
    }
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSManagedObjectContext {

    public struct ConcurrencyType :
Hashable, Equatable, RawRepresentable {

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
        ///
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue:
NSManagedObjectContextConcurrencyType

        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
```

```
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
value to use for the new instance.
        public init(rawValue:
NSManagedObjectContextConcurrencyType)

        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 15.0, tvOS 15.0,
watchOS 8.0, macOS 12.0, *)
        public typealias RawValue =
NSManagedObjectContextConcurrencyType
    }

    public enum ScheduledTaskType {

        case immediate

        case enqueued

        /// Returns a Boolean value
indicating whether two values are equal.
```

```
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
NSManagedObjectContext.ScheduledTaskType,
b:
NSManagedObjectContext.ScheduledTaskType)
-> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }
    public convenience init(_ type:
NSManagedObjectContext.ConcurrencyType)
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, watchOS 8.0, *)
extension NSManagedObjectContext {

    public func performAndWait<T>(_
block: () throws -> T) rethrows -> T

    public func perform<T>(schedule:
NSManagedObjectContext.ScheduledTaskType
= .immediate, _ block: @escaping ()
throws -> T) async rethrows -> T
}


// MARK: - CloudKit Additions

import CloudKit
import CloudKit.CKDatabase
import CoreData.CloudKit
import CoreData.CloudKit

// Available when CloudKit is imported
with CoreData
public typealias __NSConstantString =
__NSConstantString_tag

// Available when CloudKit is imported
with CoreData
```

```
public typealias __builtin_ms_va_list =
UnsafeMutablePointer<CChar>

// Available when CloudKit is imported
with CoreData
public typealias __builtin_va_list =
UnsafeMutablePointer<CChar>

// Available when CloudKit is imported
with CoreData
@available(macOS 11.0, iOS 14.0, tvOS
14.0, watchOS 7.0, *)
extension
NSPersistentCloudKitContainerOptions {

    @available(macOS 11.0, iOS 14.0, tvOS
14.0, watchOS 7.0, *)
    public var databaseScope:
CKDatabase.Scope
}
```