Store Kit Framework Reference

Networking & Internet

2009-05-01

Contents

Introduction

Framework	/System/Library/Frameworks/StoreKit.framework
Header file directories	/System/Library/Frameworks/StoreKit.framework/Headers
Companion guide	In App Purchase Programming Guide
Declared in	SKError.h SKPayment.h SKPaymentQueue.h SKPaymentTransaction.h SKProduct.h SKProductsRequest.h SKRequest.h

The Store Kit framework provides classes that allow an application to request payment from a user for additional functionality or content that your application delivers.

Classes

SKMutablePayment Class Reference

Inherits from	SKPayment : NSObject
Conforms to	NSCopying (SKPayment) NSMutableCopying (SKPayment) NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKPayment.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKMutablePayment` class defines a request to the Apple App Store to process payment for additional functionality offered by your application. A payment encapsulates a string that identifies a particular product and the quantity of that item the user would like to purchase.

When a mutable payment is added to the payment queue, the payment queue copies the contents into an immutable request before queueing the request. Your application can safely change the contents of the mutable payment object.

Tasks

Getting and Setting Attributes

`productIdentifier` (page 10) *property*
  A string that identifies a product that can be purchased from within your application.

`quantity` (page 10) *property*
  The number of items the user wants to purchase.

`requestData` (page 10) *property*
  Reserved for future use. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

productIdentifier

A string that identifies a product that can be purchased from within your application.

```
@property(nonatomic, copy, readwrite) NSString *productIdentifier
```

Discussion
The product identifier is a string previously agreed on between your application and the Apple App Store.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

quantity

The number of items the user wants to purchase.

```
@property(nonatomic, readwrite) NSInteger quantity
```

Discussion
The quantity property must be greater than `0`.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

requestData

Reserved for future use. (read-only)

```
@property(nonatomic, copy, readwrite) NSData *requestData
```

Discussion
The default value is `nil`. If `requestData` is not `nil`, your payment will be rejected by the Apple App Store.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

SKPayment Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSMutableCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKPayment.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKPayment` class defines a request to the Apple App Store to process payment for additional functionality offered by your application. A payment encapsulates a string that identifies a particular product and the quantity of those items the user would like to purchase.

Tasks

Creating Instances

`+ paymentWithProduct:` (page 13)
Returns a new payment for the specified product.

`+ paymentWithProductIdentifier:` (page 13)
Returns a new payment with the specified product identifier.

Getting Attributes

`productIdentifier` (page 12) *property*
A string used to identify a product that can be purchased from within your application. (read-only)

`quantity` (page 12) *property*
The number of items the user wants to purchase. (read-only)

`requestData` (page 12) *property*
Reserved for future use. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

productIdentifier

A string used to identify a product that can be purchased from within your application. (read-only)

```
@property(nonatomic, copy, readonly) NSString *productIdentifier
```

Discussion
The product identifier is a string previously agreed on between your application and the Apple App Store.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

quantity

The number of items the user wants to purchase. (read-only)

```
@property(nonatomic, readonly) NSInteger quantity
```

Discussion
Default value is `1`.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

requestData

Reserved for future use. (read-only)

```
@property(nonatomic, copy, readonly) NSData *requestData
```

Discussion
The default value is `nil`. If `requestData` is not `nil`, your payment will be rejected by the Apple App Store.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

Class Methods

paymentWithProduct:

Returns a new payment for the specified product.

```
+ (id)paymentWithProduct:(SKProduct *)product
```

Parameters

`product`
The product the user wishes to purchase.

Return Value
A new payment object.

Discussion
This factory Method in *Cocoa Core Competencies* uses the `productIdentifier` property obtained from the *`product`* parameter to create and return a new payment with that identifier. The quantity property defaults to `1`.

To create a `SKPayment` object with a quantity greater than `1`, create a `SKMutablePayment` object, adjust its `quantity` property and then add it to the payment queue.

```
SKMutablePayment *myPayment = [SKMutablePayment paymentWithProduct: myProduct];
myPayment.quantity = 2;
[[SKPaymentQueue defaultQueue] addPayment:myPayment];
```

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

paymentWithProductIdentifier:

Returns a new payment with the specified product identifier.

```
+ (id)paymentWithProductIdentifier:(NSString *)identifier
```

Parameters

`identifier`
A string that identifies the item to be purchased.

Return Value
A new payment object.

Discussion
The product identifier is a string previously agreed on between your application and the Apple App Store. The quantity property defaults to `1`.

To create a `SKPayment` object with a quantity greater than `1`, create a `SKMutablePayment` object, adjust its `quantity` property and then add it to the payment queue:

```
SKMutablePayment *myPayment = [SKMutablePayment paymentWithProductIdentifier:
myIdentifier];
myPayment.quantity = 2;
[[SKPaymentQueue defaultQueue] addPayment:myPayment];
```

Availability
Available in iOS 3.0 and later.

Declared In
`SKPayment.h`

SKPaymentQueue Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKPaymentQueue.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKPaymentQueue` class defines a queue of payment transactions to send to the Apple App Store. To use the payment queue, the application attaches an object that implements the `SKPaymentTransactionObserver` protocol to the payment queue, and then adds one or more payments. When payments are added to the queue, Store Kit connects to the Apple App Store and presents a user interface so that the user can authorize payment. As payments are fulfilled, the payment queue updates transactions and delivers them to its observers.

Tasks

Determining Whether the User Can Make Payments

`+ canMakePayments` (page 17)
Returns whether the user is allowed to make payments.

Getting the Queue

`+ defaultQueue` (page 17)
Returns the singleton payment queue instance.

Adding and Removing the Observer

`– addTransactionObserver:` (page 18)
: Adds an observer to the payment queue.

`– removeTransactionObserver:` (page 19)
: Removes an observer from the payment queue.

Managing Transactions

`transactions` (page 16) *property*
: Returns an array of pending transactions. (read-only)

`– addPayment:` (page 18)
: Adds a payment request to the queue.

`– finishTransaction:` (page 19)
: Completes a pending transaction.

Restoring Purchases

`– restoreCompletedTransactions` (page 20)
: Asks the payment queue to restore previously completed purchases.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

transactions

Returns an array of pending transactions. (read-only)

```
@property(nonatomic, readonly) NSArray *transactions
```

Discussion
The value of this property is undefined when there are no observers attached to the payment queue.

Availability
Available in iOS 3.0 and later.

See Also
`– addTransactionObserver:` (page 18)

Declared In
`SKPaymentQueue.h`

Class Methods

canMakePayments

Returns whether the user is allowed to make payments.

```
+ (BOOL)canMakePayments
```

Return Value
`YES` if the user is allowed to authorize payment. `NO` if they do not have permission.

Discussion
An iPhone can be restricted from accessing the Apple App Store. For example, parents can restrict their children's ability to purchase additional content. Your application should confirm that the user is allowed to authorize payments before adding a payment to the queue. Your application may also want to alter its behavior or appearance when the user is not allowed to authorize payments.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentQueue.h`

defaultQueue

Returns the singleton payment queue instance.

```
+ (SKPaymentQueue *)defaultQueue
```

Return Value
The shared payment queue.

Discussion
Applications do not create a payment queue. Instead, they retrieve the singleton queue by calling this class method.

Special Considerations

The payment queue is not available in Simulator. Attempting to retrieve the payment queue logs a warning.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentQueue.h`

Instance Methods

addPayment:

Adds a payment request to the queue.

```
- (void)addPayment:(SKPayment *)payment
```

Parameters

payment

A payment request.

Discussion

An application should always have at least one observer of the payment queue before adding payment requests.

The payment request must have a product identifier registered with the Apple App Store and a quantity greater than `0`. If either property is invalid, `addPayment:` throws an exception.

When a payment request is added to the queue, the payment queue processes that request with the Apple App Store and arranges for payment from the user. When that transaction is complete or if a failure occurs, the payment queue sends the `SKPaymentTransaction` object that encapsulates the request to all transaction observers.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

addTransactionObserver:

Adds an observer to the payment queue.

```
- (void)addTransactionObserver:(id < SKPaymentTransactionObserver >)observer
```

Parameters

observer

The observer to add to the queue.

Discussion

Your application should add an observer to the payment queue during application initialization. If there are no observers attached to the queue, the payment queue does not synchronize its list of pending transactions with the Apple App Store, because there is no observer to respond to updated transactions.

If an application quits when transactions are still being processed, those transactions are not lost. The next time the application launches, the payment queue will resume processing the transactions. Your application should always expect to be notified of completed transactions.

If more than one transaction observer is attached to the payment queue, no guarantees are made as to the order they will be called in. It is safe for multiple observers to call `finishTransaction:` (page 19), but not recommended. It is recommended that you use a single observer to process and finish the transaction.

Availability
Available in iOS 3.0 and later.

See Also
`– removeTransactionObserver:` (page 19)
`@property transactions` (page 16)

Declared In
`SKPaymentQueue.h`

finishTransaction:

Completes a pending transaction.

```
- (void)finishTransaction:(SKPaymentTransaction *)transaction
```

Parameters

transaction
The transaction to finish.

Discussion
Your application should call this method from a transaction observer that received a notification from the payment queue. Calling `finishTransaction:` on a transaction removes it from the queue. Your application should call `finishTransaction:` only after it has successfully processed the transaction and unlocked the functionality purchased by the user.

Calling `finishTransaction:` on a transaction that is in the `SKPaymentTransactionStatePurchasing` (page 24) state throws an exception.

Availability
Available in iOS 3.0 and later.

See Also
`– paymentQueue:updatedTransactions:` (page 42)

Declared In
`SKPaymentQueue.h`

removeTransactionObserver:

Removes an observer from the payment queue.

```
- (void)removeTransactionObserver:(id < SKPaymentTransactionObserver >)observer
```

Parameters

observer
The observer to remove.

Discussion
If there are no observers attached to the queue, the payment queue does not synchronize its list of pending transactions with the Apple App Store, because there is no observer to respond to updated transactions.

Availability
Available in iOS 3.0 and later.

See Also

`- addTransactionObserver:` (page 18)

`@property transactions` (page 16)

Declared In

`SKPaymentQueue.h`

restoreCompletedTransactions

Asks the payment queue to restore previously completed purchases.

```
- (void)restoreCompletedTransactions
```

Discussion

Your application calls this method to restore transactions that were previously finished so that you can process them again. For example, your application would use this to allow a user to unlock previously purchased content onto a new device.

When you create a new product to be sold in your store, you choose whether that product can be restored or not. See the *In App Purchase Programming Guide* for more information.

The payment queue will deliver a new transaction for each previously completed transaction that can be restored. Each transaction includes a copy of the original transaction.

After the transactions are delivered, the payment queue calls the observer's `paymentQueueRestoreCompletedTransactionsFinished:` (page 43) method. If an error occurred while restoring transactions, the observer will be notified through its `paymentQueue:restoreCompletedTransactionsFailedWithError:` (page 42) method.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

SKPaymentTransaction Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKPaymentTransaction.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKPaymentTransaction` class defines objects residing in the payment queue. A payment transaction is created whenever a payment is added to the payment queue. Transactions are delivered to your application when the App Store has finished processing the payment. Completed transactions provide a receipt and transaction identifier that your application can use to save a permanent record of the processed payment.

Tasks

Getting Information About the Transaction

`error` (page 22) *property*
An object describing the error that occurred while processing the transaction. (read-only)

`payment` (page 22) *property*
The payment for the transaction. (read-only)

`transactionState` (page 24) *property*
The current state of the transaction. (read-only)

`transactionIdentifier` (page 23) *property*
A string that uniquely identifies a successful payment transaction. (read-only)

`transactionReceipt` (page 23) *property*
A signed receipt that records all information about a successful payment transaction. (read-only)

`transactionDate` (page 23) *property*
The date the transaction was added to the App Store's payment queue. (read-only)

Restored Transactions

`originalTransaction` (page 22) *property*
The transaction that was restored by the App Store. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

error

An object describing the error that occurred while processing the transaction. (read-only)

```
@property(nonatomic, readonly) NSError *error
```

Discussion
The `error` property is undefined except when `transactionState` (page 24) is set to `SKPaymentTransactionStateFailed` (page 25). Your application can read the `error` property to determine why the transaction failed.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

originalTransaction

The transaction that was restored by the App Store. (read-only)

```
@property(nonatomic, readonly) SKPaymentTransaction *originalTransaction
```

Discussion
The contents of this property are undefined except when `transactionState` (page 24) is set to `SKPaymentTransactionStateRestored` (page 25). When a transaction is restored, the current transaction holds a new transaction identifier, receipt, and so on. Your application will read this property to retrieve the restored transaction.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

payment

The payment for the transaction. (read-only)

```
@property(nonatomic, readonly) SKPayment *payment
```

Discussion
Each payment transaction is created in response to a payment that your application added to the payment queue.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

transactionDate

The date the transaction was added to the App Store's payment queue. (read-only)

```
@property(nonatomic, readonly) NSDate *transactionDate
```

Discussion
The contents of this property are undefined except when `transactionState` (page 24) is set to `SKPaymentTransactionStatePurchased` (page 25) or `SKPaymentTransactionStateRestored` (page 25).

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

transactionIdentifier

A string that uniquely identifies a successful payment transaction. (read-only)

```
@property(nonatomic, readonly) NSString *transactionIdentifier
```

Discussion
The contents of this property are undefined except when `transactionState` (page 24) is set to `SKPaymentTransactionStatePurchased` (page 25) or `SKPaymentTransactionStateRestored` (page 25). The `transactionIdentifier` is a string that uniquely identifies the processed payment. Your application may wish to record this string as part of an audit trail for App Store purchases. See *In App Purchase Programming Guide* for more information.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

transactionReceipt

A signed receipt that records all information about a successful payment transaction. (read-only)

```
@property(nonatomic, readonly) NSData *transactionReceipt
```

Discussion
The contents of this property are undefined except when `transactionState` (page 24) is set to `SKPaymentTransactionStatePurchased` (page 25).

The receipt is a signed chunk of data that can be sent to the App Store to verify that the payment was successfully processed. This is most useful when designing a store that server separate from the iPhone to verify that payment was processed. For more information on verifying receipts, see *In App Purchase Programming Guide*.

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

transactionState

The current state of the transaction. (read-only)

```
@property(nonatomic, readonly) SKPaymentTransactionState transactionState
```

Availability
Available in iOS 3.0 and later.

Declared In
`SKPaymentTransaction.h`

Constants

Payment Transaction States

The state of a transaction.

```
enum {
SKPaymentTransactionStatePurchasing,
SKPaymentTransactionStatePurchased,
SKPaymentTransactionStateFailed,
SKPaymentTransactionStateRestored
};
typedef NSInteger SKPaymentTransactionState;
```

Constants
`SKPaymentTransactionStatePurchasing`
The transaction is being processed by the App Store.

Available in iOS 3.0 and later.

Declared in `SKPaymentTransaction.h`.

`SKPaymentTransactionStatePurchased`
The App Store successfully processed payment. Your application should provide the content the user purchased.

Available in iOS 3.0 and later.

Declared in `SKPaymentTransaction.h`.

`SKPaymentTransactionStateFailed`
The transaction failed. Check the `error` (page 22) property to determine what happened.

Available in iOS 3.0 and later.

Declared in `SKPaymentTransaction.h`.

`SKPaymentTransactionStateRestored`
This transaction restores content previously purchased by the user. Read the `originalTransaction` (page 22) property to obtain information about the original purchase.

Available in iOS 3.0 and later.

Declared in `SKPaymentTransaction.h`.

SKProduct Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKProduct.h
Companion guide	In App Purchase Programming Guide

Overview

`SKProduct` objects are returned as part of an `SKProductsResponse` object and are used to provide information about a product previously registered with the Apple App Store.

Tasks

Getting Product Attributes

`localizedDescription` (page 28) *property*
A description of the product. (read-only)

`localizedTitle` (page 28) *property*
The name of the product. (read-only)

`price` (page 28) *property*
The cost of the product in the local currency. (read-only)

`priceLocale` (page 29) *property*
The locale used to format the price of the product. (read-only)

`productIdentifier` (page 29) *property*
The string that identifies the product to the Apple App Store. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

localizedDescription

A description of the product. (read-only)

```
@property(nonatomic, readonly) NSString *localizedDescription
```

Discussion
The description is localized based on the `currentLocale` property.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProduct.h`

localizedTitle

The name of the product. (read-only)

```
@property(nonatomic, readonly) NSString *localizedTitle
```

Discussion
The description is localized based on the `currentLocale` property.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProduct.h`

price

The cost of the product in the local currency. (read-only)

```
@property(nonatomic, readonly) NSDecimalNumber *price
```

Discussion
Your application can format the price using a number formatter, as shown in the following sample code:

```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
[numberFormatter setFormatterBehavior:NSNumberFormatterBehavior10_4];
[numberFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];
[numberFormatter setLocale:product.priceLocale];
NSString *formattedString = [numberFormatter stringFromNumber:product.price];
```

Availability
Available in iOS 3.0 and later.

See Also
`@property priceLocale` (page 29)

Declared In
`SKProduct.h`

priceLocale

The locale used to format the price of the product. (read-only)

```
@property(nonatomic, readonly) NSLocale *priceLocale
```

Availability
Available in iOS 3.0 and later.

See Also
  `@property price` (page 28)

Declared In
`SKProduct.h`

productIdentifier

The string that identifies the product to the Apple App Store. (read-only)

```
@property(nonatomic, readonly) NSString *productIdentifier
```

Availability
Available in iOS 3.0 and later.

Declared In
`SKProduct.h`

SKProductsRequest Class Reference

Inherits from	SKRequest : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKProductsRequest.h
Companion guide	In App Purchase Programming Guide

Overview

An `SKProductsRequest` object is used to retrieve localized information about a list of products from the Apple App Store. Your application uses this request to present localized prices and other information to the user without having to maintain that list itself.

To use an `SKProductsRequest` object, you initialize it with a list of product identifier strings, attach a delegate, and then call the request's `start` (page 36) method. When the request completes, your delegate receives an `SKProductsResponse` object.

Tasks

Initializing a Products Request

`– initWithProductIdentifiers:` (page 32)
Initializes the request with the set of product identifiers.

Setting the Delegate

`delegate` (page 32) *property*
The delegate for the request.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

delegate

The delegate for the request.

```
@property(nonatomic, assign) id<SKProductsRequestDelegate> delegate
```

Availability
Available in iOS 3.0 and later.

Declared In
`SKProductsRequest.h`

Instance Methods

initWithProductIdentifiers:

Initializes the request with the set of product identifiers.

```
- (id)initWithProductIdentifiers:(NSSet *)productIdentifiers
```

Parameters

`productIdentifiers`
The list of product identifiers for the products you wish to retrieve descriptions of.

Return Value
The initialized request object.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProductsRequest.h`

SKProductsResponse Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKProductsRequest.h
Companion guide	In App Purchase Programming Guide

Overview

An `SKProductsResponse` object is returned by the Apple App Store in response to a request for information about a list of products.

Tasks

Response Information

`products` (page 34) *property*
  A list of products, one product for each valid product identifier provided in the original request. (read-only)

`invalidProductIdentifiers` (page 33) *property*
  An array of product identifier strings that were not recognized by the Apple App Store. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

invalidProductIdentifiers

An array of product identifier strings that were not recognized by the Apple App Store. (read-only)

```
@property(nonatomic, readonly) NSArray *invalidProductIdentifiers
```

Discussion
This list should typically be empty.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProductsRequest.h`

products

A list of products, one product for each valid product identifier provided in the original request. (read-only)

```
@property(nonatomic, readonly) NSArray *products
```

Discussion
The array consists of a list of `SKProduct` objects.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProductsRequest.h`

SKRequest Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKRequest.h
Companion guide	In App Purchase Programming Guide

Overview

`SKRequest` is an abstract class representing a request to the Apple App Store. Subclasses of `SKRequest` represent different kinds of requests.

To use a request object, initialize a subclass of `SKRequest` and set the `delegate` (page 36) property, then call the `start` (page 36) method.

Tasks

Controlling the Request

– `start` (page 36)
    Sends the request to the Apple App Store.

– `cancel` (page 36)
    Cancels a previously started request.

Accessing the Delegate

`delegate` (page 36) *property*
    The delegate of the request object.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

delegate

The delegate of the request object.

```
@property(nonatomic, assign) id<SKRequestDelegate> delegate
```

Discussion
The delegate must adopt the `SKRequestDelegate` protocol, although most subclasses of `SKRequest` provide a more specific protocol to implement.

Availability
Available in iOS 3.0 and later.

Declared In
`SKRequest.h`

Instance Methods

cancel

Cancels a previously started request.

```
- (void)cancel
```

Discussion
When you cancel a request, the delegate is not called with an error.

Availability
Available in iOS 3.0 and later.

Declared In
`SKRequest.h`

start

Sends the request to the Apple App Store.

```
- (void)start
```

Discussion
The results for a request are sent to the request’s delegate.

Availability
Available in iOS 3.0 and later.

Declared In

`SKRequest.h`

Protocols

PART II

Protocols

SKPaymentTransactionObserver Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKPaymentQueue.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKPaymentTransactionObserver` protocol declares methods that are implemented by observers of an `SKPaymentQueue` object.

An observer is called when transactions are updated by the queue or removed from the queue. An observer should process all successful transactions, unlock the functionality purchased by the user, and then finish the transaction by calling the payment queue's `finishTransaction:` (page 19) method.

Tasks

Handling Transactions

`– paymentQueue:updatedTransactions:` (page 42) *required method*
Tells an observer that one or more transactions have been updated. (required)

`– paymentQueue:removedTransactions:` (page 42)
Tells an observer that one or more transactions have been removed from the queue.

Handling Restored Transactions

`– paymentQueue:restoreCompletedTransactionsFailedWithError:` (page 42)
Tells the observer that an error occurred while restoring transactions.

`– paymentQueueRestoreCompletedTransactionsFinished:` (page 43)
Tells the observer that the payment queue has finished sending restored transactions.

Instance Methods

paymentQueue:removedTransactions:

Tells an observer that one or more transactions have been removed from the queue.

```
- (void)paymentQueue:(SKPaymentQueue *)queue  removedTransactions:(NSArray
    *)transactions
```

Parameters

queue

The payment queue that updated the transactions.

transactions

An array of the transactions that were removed.

Discussion

Your application does not typically need to implement this method but might implement it to update its own user interface to reflect that a transaction has been completed.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

paymentQueue:restoreCompletedTransactionsFailedWithError:

Tells the observer that an error occurred while restoring transactions.

```
- (void)paymentQueue:(SKPaymentQueue *)queue
    restoreCompletedTransactionsFailedWithError:(NSError *)error
```

Parameters

queue

The payment queue that was restoring transactions.

error

The error that occurred.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

paymentQueue:updatedTransactions:

Tells an observer that one or more transactions have been updated. (required)

```
- (void)paymentQueue:(SKPaymentQueue *)queue  updatedTransactions:(NSArray
    *)transactions
```

Parameters

queue

The payment queue that updated the transactions.

transactions

An array of the transactions that were updated.

Discussion

The application should process each transaction by examining the transaction’s `transactionState` (page 24) property. If `transactionState` (page 24) is `SKPaymentTransactionStatePurchased`, payment was successfully received for the desired functionality. The application should make the functionality available to the user. If `transactionState` (page 24) is `SKPaymentTransactionStateFailed`, the application can read the transaction’s error property to return a meaningful error to the user.

Once a transaction is processed, it should be removed from the payment queue by calling the payment queue’s `finishTransaction:` (page 19) method, passing the transaction as a parameter.

Important: Once the transaction is finished, Store Kit can not tell you that this item is already purchased. It is important that applications process the transaction completely before calling `finishTransaction:`.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

paymentQueueRestoreCompletedTransactionsFinished:

Tells the observer that the payment queue has finished sending restored transactions.

```
- (void)paymentQueueRestoreCompletedTransactionsFinished:(SKPaymentQueue *)queue
```

Parameters

queue

The payment queue that restored the transactions.

Discussion

This method is called after all restorable transactions have been processed by the payment queue. Your application is not required to do anything in this method.

Availability

Available in iOS 3.0 and later.

Declared In

`SKPaymentQueue.h`

SKProductsRequestDelegate Protocol Reference

Conforms to	SKRequestDelegate
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKProductsRequest.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKProductsRequestDelegate` protocol declares methods that are implemented by the delegate of an `SKProductsRequest` object. The delegate receives the product information that the request was interested in.

Tasks

Receiving the Response

`– productsRequest:didReceiveResponse:` (page 45) *required method*
    Called when the Apple App Store responds to the product request. (required)

Instance Methods

productsRequest:didReceiveResponse:

Called when the Apple App Store responds to the product request. (required)

```
- (void)productsRequest:(SKProductsRequest *)request
    didReceiveResponse:(SKProductsResponse *)response
```

Parameters

request
    The product request sent to the Apple App Store.

`response`
Detailed information about the list of products.

Availability
Available in iOS 3.0 and later.

Declared In
`SKProductsRequest.h`

SKRequestDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/StoreKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	SKRequest.h
Companion guide	In App Purchase Programming Guide

Overview

The `SKRequestDelegate` protocol declares common methods that are implemented by delegates for any subclass of the `SKRequest` abstract class.

Tasks

Completing Requests

`– requestDidFinish:` (page 48)
Called when the request has completed.

Handling Errrors

`– request:didFailWithError:` (page 47)
Called if the request failed to execute.

Instance Methods

request:didFailWithError:

Called if the request failed to execute.

```
- (void)request:(SKRequest *)request didFailWithError:(NSError *)error
```

Parameters

`request`
: The request that failed.

`error`
: The error that caused the request to fail.

Discussion

When the request fails, your application should release the request. The `requestDidFinish:` (page 48) method is not called after this method is called.

Availability

Available in iOS 3.0 and later.

Declared In

`SKRequest.h`

requestDidFinish:

Called when the request has completed.

```
- (void)requestDidFinish:(SKRequest *)request
```

Parameters

`request`
: The request that completed.

Discussion

This method is called after all processing of the request has been completed. Typically, subclasses of `SKRequest` require the delegate to implement additional methods to receive the response. When this method is called, your delegate receives no further communication from the request and can release it.

Availability

Available in iOS 3.0 and later.

Declared In

`SKRequest.h`

Constants

Store Kit Constants Reference

Framework:	StoreKit/SKError.h
Declared in	SKError.h
Companion guide	In App Purchase Programming Guide

Overview

This document describes the constants defined in the Store Kit framework and not described in a document for an individual class.

Constants

SKErrorDomain

This constant defines the Store Kit framework error domain.

```
NSString * const SKErrorDomain;
```

Constants

`SKErrorDomain`

Indicates an error occurred in Store Kit.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

Store Kit Errors

Error codes for the Store Kit error domain.

```
enum {
    SKErrorUnknown,
    SKErrorClientInvalid,
    SKErrorPaymentCancelled,
    SKErrorPaymentInvalid,
    SKErrorPaymentNotAllowed
};
```

Constants

`SKErrorUnknown`

Indicates that an unknown or unexpected error occurred.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

`SKErrorClientInvalid`

Indicates that the client is not allowed to perform the attempted action.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

`SKErrorPaymentCancelled`

Indicates that the user cancelled a payment request.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

`SKErrorPaymentInvalid`

Indicates that one of the payment parameters was not recognized by the Apple App Store.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

`SKErrorPaymentNotAllowed`

Indicates that the user is not allowed to authorize payments.

Available in iOS 3.0 and later.

Declared in `SKError.h`.

Document Revision History

This table describes the changes to *Store Kit Framework Reference*.

Date	Notes
2009-05-01	Updated to add new classes and protocols used to request information from the Apple App Store.
2009-03-12	New document that describes the API for implementing built-in store functionality in an iPhone application.