Introducing the Contacts Framework
For OS X, iOS, and watchOS
Session 223

Bruce Stadnyk iOS Contacts Engineer
Dave Dribin OS X Contacts Engineer
Julien Robert iOS Contacts Engineer
What is Contacts Framework?
What is Contacts Framework?

Objective-C and Swift API
What is Contacts Framework?

Objective-C and Swift API

Optimized for thread-safe, read only usage
What is Contacts Framework?

Objective-C and Swift API
Optimized for thread-safe, read only usage
One API, multiple platforms
What is Contacts Framework?

- Objective-C and Swift API
- Optimized for thread-safe, read only usage
- One API, multiple platforms
- AddressBook API being deprecated
What are Contacts?
What are Contacts?

Everyone has Contacts
What are Contacts?

Everyone has Contacts
Phone, Mail, Messages, …
What are Contacts?

Everyone has Contacts
Phone, Mail, Messages, …
What are Contacts?

Everyone has Contacts
Phone, Mail, Messages, …
Central to the user experience
Contact Properties
Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)
Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)
Contact Properties

John Appleseed
john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)

contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
Contact Properties

John Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)

contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
Contact Properties

John Appleseed
john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)

contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
Contact Properties

John Appleseed

contact.imageData

contact.givenName

contact.familyName

contact.emailAddresses

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

contact.phoneNumbers
Contact Properties

John Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)

contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
Contact Objects
Contact Objects

CNContact
Contact Objects

CNContact

CNMutableContact
Contact Objects

- CNContact
- CNMutableContact
- CNLabeledValue
Creating a New Contact

```python
import Contacts
```
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData

contact.givenName = "John"
contact.familyName = "Appleseed"
Creating a New Contact

Labeled Values

```swift
let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork,
    value: "j.appleseed@icloud.com")
```
Creating a New Contact

Labeled Values

let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork, value: "j.appleseed@icloud.com")

contact.emailAddresses = [homeEmail, workEmail]
Creating a New Contact

Labeled Values

```swift
let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork,
    value: "j.appleseed@icloud.com")

contact.emailAddresses = [homeEmail, workEmail]

contact.phoneNumbers = [CNLabeledValue(
    label: CNLabelPhoneNumberiPhone,
    value: CNPhoneNumber(stringValue: "(408) 555-0126"))]
```
Creating a New Contact

Labeled Values

```swift
let address = CNMutablePostalAddress()
address.street = "774 Loma Vista Ave"
address.city = "Los Gatos"
address.state = "CA"
address.postalCode = "95032"
```
Creating a New Contact

Labeled Values

```swift
let address = CNMutablePostalAddress()
address.street = "774 Loma Vista Ave"
address.city = "Los Gatos"
address.state = "CA"
address.postalCode = "95032"

contact.postalAddresses = [CNLabeledValue(label: CNLabelHome, value: address)]
```
Creating a New Contact

Dates

```swift
let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988  // can omit for a year-less birthday
```
Creating a New Contact

Dates

let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988  // can omit for a year-less birthday

contact.birthday = birthday
Formatting Contact Data
let fullName = CNContactFormatter.stringFromContact(contact, style: .FullName)
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)

print(fullName)
// John Appleseed
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
print(fullName)
// John Appleseed

let postalString = CNPostalAddressFormatter.stringFromPostalAddress(address)
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
print(fullName)
// John Appleseed

let postalString = CNPostalAddressFormatter.stringFromPostalAddress(address)
print(postalString)
// 774 Loma Vista Ave
// Los Gatos, CA 95032
Using Contacts in Your App

Dave Dribin OS X Contacts Engineer
Using Contacts in Your App

CNContactStore
class CNContactStore : NSObject {

    func unifiedContactsMatchingPredicate(
        predicate: NSPredicate,
        keysToFetch: [CNKeyDescriptor]) -> [CNContact] throws

    }

Fetching User's Contacts
Predicates

let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")

John
Appleseed
john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)
April 1, 1988 (birthday)

Jane
Appleseed
jane@example.com (home)
(505) 555-0155 (home)
(408) 555-0166 (work)

Craig
Bromley
cbromley@icloud.com (work)
(465) 555-0199 (iPhone)
Jun 21 (birthday)
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone Numbers</th>
<th>Email Addresses</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>(408) 555-0126 (iPhone)</td>
<td><a href="mailto:john@example.com">john@example.com</a> (home)</td>
<td>April 1, 1988</td>
</tr>
<tr>
<td>Jane</td>
<td>(505) 555-0155 (home)</td>
<td><a href="mailto:jane@example.com">jane@example.com</a> (home)</td>
<td></td>
</tr>
<tr>
<td>Craig</td>
<td>(465) 555-0199 (iPhone)</td>
<td><a href="mailto:cbromley@icloud.com">cbromley@icloud.com</a> (work)</td>
<td>Jun 21</td>
</tr>
</tbody>
</table>
let keysToFetch = ['givenName', 'familyName']

John
Appleseed
john@example.com (home)
(408) 555-0126 (iPhone)
April 1, 1988 (birthday)

Jane
Appleseed
j.appleseed@icloud.com (work)
(505) 555-0155 (home)
(408) 555-0166 (work)

Jane
Appleseed
jane@example.com (home)
(505) 555-0155 (home)
(408) 555-0166 (work)

April 1, 1988 (birthday)
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

**John Appleseed**
- john@example.com (home)
- j.appleseed@icloud.com (work)
- (408) 555-0126 (iPhone)
- April 1, 1988 (birthday)

**Jane Appleseed**
- jane@example.com (home)
- (505) 555-0155 (home)
- (408) 555-0166 (work)

let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
How to Fetch

```swift
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

let store = CNContactStore()
let contacts = try store.unifiedContactsMatchingPredicate(predicate, keysToFetch: keysToFetch)
How to Fetch

```swift
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

let store = CNContactStore()
let contacts = try store.unifiedContactsMatchingPredicate(predicate, keysToFetch: keysToFetch)

for contact in contacts {
    print("\(contact.givenName) \(contact.familyName)")
}
```
Keeping UI Responsive

Main Queue

Background Queue

Fetch Contacts

CNContact
Keeping UI Responsive

- Main Queue
- Background Queue
- Fetch Contacts
- CNContact
Data Privacy
class CNContactStore : NSObject {
    func requestAccessForEntityType(..., completionHandler:)
}
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

John
Appleseed
john@example.com (home)
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)
April 1, 1988 (birthday)
Partial Contacts

let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
Partial Contacts

if (contact.isKeyAvailable(CNContactPhoneNumberKey)) {
    print("\(contact.phoneNumbers)")
}
if (contact.isKeyAvailable(CNContactPhoneNumbersKey)) {
    print("\(contact.phoneNumbers)")
} else {
    let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey,
                       CNContactPhoneNumbersKey]
    var refetchedContact = try store.unifiedContactWithIdentifier(
        contact.identifier, keysToFetch: keysToFetch)
    print("\(refetchedContact.phoneNumbers)")
}
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey, 
                  CNContactNamePrefixKey, CNContactMiddleNameKey, ...]
let keysToFetch = [CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName)]
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressesKey]
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressesKey]

let contacts = try store.unifiedContactsMatchingPredicate(predicate, keysToFetch: keysToFetch)
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter_descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressKey
]

let contacts = try store.unifiedContactsMatchingPredicate(predicate, keysToFetch: keysToFetch)

for contact in contacts {
    let fullName = CNContactFormatter.stringFromContact(contact, style: .FullName) ?? "No Name"
    print("\(fullName): \(contact.emailAddresses)"")
}
Unified Contacts

iCloud

John Appleseed
j.appleseed@icloud.com (work)
(408) 555-0126 (iPhone)

Facebook

John Appleseed
john@example.com (home)

April 1, 1988 (birthday)
Adding a New Contact

```swift
let john = CNMutableContact()
john.givenName = "John"
john.familyName = "Appleseed"
```
Adding a New Contact

let john = CNMutableContact()
john.givenName = "John"
john.familyName = "Appleseed"

let saveRequest = CNSaveRequest()
saveRequest.addContact(john, toContainerWithIdentifier: nil)
try store.executeSaveRequest(saveRequest)
Updating an Existing Contact

```swift
let updatedContact = contact.mutableCopy()
let newEmail = CNLabeledValue(label: CNLabelHome,
                               value: "john@example.com")
updatedContact.emailAddresses.append(newEmail)
```
Updating an Existing Contact

```swift
let updatedContact = contact.mutableCopy()
let newEmail = CNLabeledValue(label: CNLabelHome,
    value: "john@example.com")
updatedContact.emailAddresses.append(newEmail)

let saveRequest = CNSaveRequest()
saveRequest.updateContact(updatedContact)
try store.executeSaveRequest(saveRequest)
```
Contacts in the UI

Julien Robert iOS Contacts Engineer
ContactsUI
New framework
ContactsUI
New framework
## ContactsUI

**New framework**

<table>
<thead>
<tr>
<th>iOS</th>
<th>OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNContactPickerController</td>
<td>CNContactPicker</td>
</tr>
</tbody>
</table>
ContactsUI
New framework

<table>
<thead>
<tr>
<th>iOS</th>
<th>OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNContactPickerViewController</td>
<td>CNContactPicker</td>
</tr>
<tr>
<td>CNContactViewController</td>
<td>CNContactViewController</td>
</tr>
</tbody>
</table>
Picking Contacts
CNContactPickerViewController
Picking Contacts
CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController
Picking Contacts

CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController

Must be presented, not pushed
Picking Contacts
CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController
Must be presented, not pushed
Always out of process, no contacts access dialog
Picking Contacts

CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController

Must be presented, not pushed

Always out of process, no contacts access dialog

May return partial contacts
Picking Contacts
CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController
Must be presented, not pushed
Always out of process, no contacts access dialog
May return partial contacts
Behavior based on delegate methods and predicates
Picking Contacts
CNContactPickerViewController

Modern replacement for ABPeoplePickerNavigationController
Must be presented, not pushed
Always out of process, no contacts access dialog
May return partial contacts
Behavior based on delegate methods and predicates
Supports multi-selection
Picking Contacts
Delegate methods
Picking Contacts
Delegate methods

Single contact
Picking Contacts
Delegate methods

Single contact

contactPicker(picker, didSelectContact contact: CNContact)
Picking Contacts

Delegate methods

Single contact

contactPicker(picker, didSelectContact contact: CNContact)

Single property

contactPicker(picker, didSelectContactProperty property: CNContactProperty)
Picking Contacts
Delegate methods

Single contact

contactPicker(picker, didSelectContact contact: CNContact)

Single property

contactPicker(picker, didSelectContactProperty property: CNContactProperty)

class CNContactProperty {
    var contact: CNContact
    var key: NSString
    var value: AnyObject?
    var identifier: NSString?
}
Picking Contacts
Delegate methods
Picking Contacts

Delegate methods

Multiple contacts
Picking Contacts

Delegate methods

Multiple contacts

contactPicker(picker, didSelectContacts contacts: [CNContact])
Picking Contacts

Delegate methods

Multiple contacts

```
contactPicker(picker, didSelectContacts contacts: [CNContact])
```

Multiple properties

```
contactPicker(picker, didSelectContactProperties properties: [CNContactProperty])
```
Picking Contacts

Predicates
Picking Contacts

Predicates

predicateForEnablingContact

• Which contacts are available
• Evaluated on CNContact
Picking Contacts

Predicates

predicateForEnablingContact

- Which contacts are available
- Evaluated on CNContact
Picking Contacts
Predicates

predicateForEnablingContact
• Which contacts are available
• Evaluated on CNContact

let predicate = NSPredicate(format: "familyName LIKE[c] 'parker'")
Picking Contacts

Predicates

predicateForEnablingContact

• Which contacts are available
• Evaluated on CNContact

let predicate = NSPredicate(format: "familyName LIKE[cd] 'parker'")

contactPicker.predicateForEnablingContact = predicate
Picking Contacts
Predicates

predicateForSelectionOfContact

• Which contacts are returned when tapped, others push the card
Picking Contacts
Predicates

predicateForSelectionOfContact
• Which contacts are returned when tapped, others push the card

predicateForSelectionOfProperty
• Which properties are returned when tapped, others perform the default action
• Evaluated on CNContactProperty
Picking Contacts

Predicates

\text{predicateForSelectionOfContact}

\begin{itemize}
  \item Which contacts are returned when tapped, others push the card
\end{itemize}

\text{predicateForSelectionOfProperty}

\begin{itemize}
  \item Which properties are returned when tapped, others perform the default action
  \item Evaluated on \text{CNContactProperty}
\end{itemize}

Coherence between predicates and delegate methods
Viewing Contacts
CNContactViewController
Viewing Contacts

CNContactViewController

One class to replace

• ABPersonViewController
• ABNewPersonViewController
• ABUnknownPersonViewController
Viewing Contacts
CNContactViewController
Viewing Contacts

CNContactViewController

Use appropriate creation method
Viewing Contacts

CNContactViewController

Use appropriate creation method

- viewControllerForContact:
Viewing Contacts

CNContactViewController

Use appropriate creation method

- viewControllerForContact:
- viewControllerForNewContact:
Viewing Contacts
CNContactViewController

Use appropriate creation method
• viewControllerForContact:
• viewControllerForNewContact:
• viewControllerForUnknownContact:
Viewing Contacts

CNContactViewController

Use appropriate creation method

- viewControllerForContact:
- viewControllerForNewContact:
- viewControllerForUnknownContact:
Viewing Contacts

CNContactViewController

Use appropriate creation method

• viewControllerForContact:
• viewControllerForNewContact:
• viewControllerForUnknownContact:

Always out of process
Viewing Contacts

CNContactViewController

Use appropriate creation method

- viewControllerForContact:
- viewControllerForNewContact:
- viewControllerForUnknownContact:

Always out of process

Contact must be fetched with descriptorForRequiredKeys
let contact = try contactStore.unifiedContactWithIdentifier(identifier, keysToFetch: [CNContactViewController.descriptorForRequiredKeys])
let contact = try contactStore.unifiedContactWithIdentifier(identifier, keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)
let contact = try contactStore.unifiedContactWithIdentifier(identifier, keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self
let contact = try contactStore.unifiedContactWithIdentifier(identifier, keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self

self.pushViewController(viewController)
Viewing Contacts

Example

```swift
let contact = try contactStore.unifiedContactWithIdentifier(identifier, keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self

self.pushViewController(viewController)

func contactViewController(vc, didFinishWithContact: contact) {
    // do something with the modified contact
}
```
Demo
Picking and Viewing Contacts

Meow
Summary

A new modern Contacts API
Common across all platforms
Adopt now!
More Information

Documentation
Contacts Framework Reference
ContactsUI Framework Reference
http://developer.apple.com/library

Technical Support
Apple Developer Forums
http://developer.apple.com/forums

General Inquiries
Paul Marcos, App Frameworks Evangelist
pmarcos@apple.com
<table>
<thead>
<tr>
<th>Labs</th>
<th>Frameworks Lab A</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contacts, Calendar and</td>
<td></td>
<td>Thursday 4:30PM</td>
</tr>
<tr>
<td>Reminders Lab</td>
<td>Frameworks Lab A</td>
<td>Friday 9:00AM</td>
</tr>
</tbody>
</table>