Windows Bridge for iOS

[ADDRESSBOOK FRAMEWORK – [FOUNDATION] IMPLEMENT ADDRESSBOOK FRAMEWORK]

DEV DESIGN specification

[**Feature Summary (<50 words)**](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Title%20Page.aspx)

This spec covers the design for implementing the AddressBook framework, addressing gaps and differences.

# [Overview](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Overview.aspx)

## [Summary](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Overview.aspx)

The iOS AddressBook Framework allows developers to interact with the user’s contacts. Specifically, if given permission by the user, they have access to view all of the user’s contacts, modify and delete them, as well as create and add new ones.

These interactions are done with 3 major types – **ABAddressBookRef**, **ABRecordRef**, and **ABMultiValueRef** (and **ABMutableMultiValueRef**, a variant of **ABMultiValueRef**).

**ABAddressBookRef** represents the user’s address book, and has functionality such as asking for a list of the user’s contacts, adding new contacts, and removing existing contacts.

**ABRecordRef** represents a single entry in the user’s address book. It has a specific record type depending on what it represents: **kABPersonType**, **kABGroupType**, and **kABSourceType**. The most common is **kABPersonType**, which represents a single contact/person. It can be queried for various properties (such as first name/last name, emails, birthday, etc). An **ABRecordRef** can have its various properties modified to update information about a specific contact. An **ABRecordRef** can also represent a ‘Group’ or ‘Source’ (**kABGroupType** and **kABSourceType**) – a Group simply represents some grouping of contacts, while a Source represents where the contact originated from (locally to device, social networking site, CardDAV server, etc).

An **ABRecordRef** of type **kABPersonType** will have the columns specified in the table under Gaps/Differences later in the document. One of **kABGroupType** has the ability to copy an array of all persons in said group (an array of **ABRecordRef** of type **kABPersonType**) and one of **kABSourceType** has a column for the source type.

**ABMultiValueRef** represents a column of an **ABRecordRef** that can have multiple values – for example, a phone number or email. When asking for a contact’s phone number, the user is given back an **ABMultiValueRef**, which is essentially a list of phone numbers and some associated label (work, home, etc). An **ABMutableMultiValueRef** is an **ABMultiValueRef** that can be modified – they are usually used when creating a new contact, or adding another field to an existing one (such as adding another phone number or email).

Here are prioritizations of what functionality to implement first:

|  |  |
| --- | --- |
| Priority | Functionality to Implement |
| P0 | Requesting Permission & Getting a List of Contacts |
| P0 | Copying an Individual Contact & Reading Basic Fields (Name, Phone, etc) |
| P1 | Reading More Fields (Jobs, Birthdays, etc) |
| P1 | Creating a New Contact |
| P2 | Modify/Delete Existing Contact (Needs Creation Working Due to Permissions) |
| P2 | **ABGroup**/**ABSource** Functionality |

See the table at the end of the document for prioritization of individual functions.

## [Document Terms](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Overview.aspx" \o "Terms used in this document and not already defined in the functional or other spec.)

|  |  |
| --- | --- |
| Term | Definition |
| Windows Contacts | Windows.ApplicationModel.Contacts namespace |

## [Language Decisions](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Overview.aspx" \o "Describe the patterns, techniques and methodologies used and why they were chosen.)

Objective-C projection will be used to access the Windows Contacts classes. Although the AddressBook Framework types (**ABAddressBookRef**, **ABRecordRef**, etc) are pointers to C structs on the iOS side, they will be implemented as Objective-C objects.

# [Detailed Design](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Detailed%20Design.aspx" \o "A narrative that describes the components of your design at a level of detail that would allow another developer on your team to implement it)

The aforementioned **ABAddressBookRef**, **ABRecordRef**, and **ABMultiValueRef** will be implemented using Objective-C projections of classes from the Windows.ApplicationModel.Contacts namespace. An **ABAddressBookRef** will be implemented using the Windows Contacts classes **ContactManager** and **ContactStore**. A Contact Manager allows a user to ask for a **ContactStore**, which represents some group of contacts on the device (it can represent all of them, provided the necessary permissions are declared in the manifest). A **ContactStore** can give the user a list of all contacts in the **ContactStore**. This mimics much of the functionality in an **ABAddressBookRef**.

**ABMultiValueRef** will be implemented as an **NSArray** of label/value pairs.

An **ABRecordRef** will be 1 of three things: an **ABPerson**, an **ABGroup**, or an **ABSource**, depending on its type (**kABPersonType**, **kABGroupType**, or **kABSourceType**, respectively).

An **ABPerson** will be implemented using a Contact, which is the way Windows Contacts represent a single contact. Contact supports functionality to query different fields of the contact similar to what an **ABRecordRef** with type **kABPersonType** can do.

An **ABGroup** has no equivalent on the Windows side.

An **ABSource** can store the source name and type (obtained from a **ContactList**).

These three types can either be subclasses of **ABRecordRef**, or they can be separate classes, with the type **ABRecordRef** keeping an internal pointer to the appropriate type depending on it being a **kABPersonType**, **kABGroupType**, or **kABSourceType.**

**Scenarios:**

**Getting a List of All Contacts**

This is done by talking to the **ABAddressBookRef**, which will query its internal **ContactStore** (which was created by its internal **ContactManager**) to get a list of all Contacts (formatted as **ABPerson** **ABRecordRefs**). Note that these contacts will be read-only. An extra API to get read/write contacts will be provided, since Windows doesn’t give full read/write permissions to all contacts (only those created by the app).

**Reading a Contact**

Reading values from a contact in iOS is done through an **ABRecordRef** with type **kABPersonType** -- this will be an **ABPerson** backed by a Windows **Contact**. The various columns of an iOS contact will be mapped to the Windows equivalent (see later table for mappings).

**Adding a Contact**

Adding a contact is done using **ABPersonCreate** (which will create an **ABPerson** **ABRecordRef**), updating the contact as desired, and then using the **ABAddressBookRef** to actually add the contact (which will add the **ABPerson's** internal **Contact** to the **ABAddressBookRef's** internal **ContactStore’s ContactList**).

**Removing a Contact**

Removing a contact is similar to adding a **Contact** in terms of interacting with the **ABAddressBookRef** -- the main difference is that rather than creating a new **ABPerson**, an existing one will be used.

**Updating a Contact**

Updating a contact is similar to adding a new one -- any modifications will be reflected to the internal **Contact** of the **ABPerson** **ABRecordRef**.

**Working with a Source**

The closes thing to a Source is the **ContactList** that a contact came from – this is tricky when the contact is an aggregate, because it will be composed of multiple raw contacts that potentially come from difference sources. Sources are not used in very many apps, especially compared to the other scenarios, so they are very low priority.

**Working with a Group**

Windows contacts don’t have an equivalent for Groups.

**Gaps/Differences:**

There are several gaps/differences between the iOS AddressBookFramework and Windows Contacts:

1. Permissions are handled differently. In iOS, the developer asks the user for address book permission using **ABAddressBookRequestAccessWithCompletion**, and if it is granted, they have full read/write access to the user’s entire address book. In Windows, the developer can only have write access to contacts that their app created – they only get read access to the rest of the user’s contacts. Apps usually only want read-access to contacts, so this shouldn’t be a major issue. (There is a way to get full read/write access, but it requires asking for special permission before being allowed in the Windows Store.) In order for an app to get this read-only access to all contacts, it must update its manifest to declare that it is using contacts. Specifically, it must add this to the app’s package manifest:   
     
   <Capabilities><uap:Capability Name="contacts"/></Capabilities>  
     
   To address this gap, **ABAddressBookRequestAccessWithCompletion**/ **ABAddressBookGetAuthorizationStatus** will check to see if the user has denied privacy permissions manually – they still need to ensure that they update their manifest, but this will allow them to detect if a user explicitly goes and removes permissions. Developers will have read/write access to contacts they create, but read-only access to all other contacts.  
     
   Because of this permissions issue, and due to the way Aggregate and Raw contacts are stored in Windows, the method to get all contacts (**ABAddressBookCopyArrayOfAllPeople**) will return read-only contacts, but will contain all contacts on the device; if the user wishes to modify or remove contacts, they can call the new method **ABAddressBookCopyArrayOfAllUserAppPeople** to get the subset of all contacts that the user’s app has created and, therefore, has read/write permissions on.
2. In iOS, making individual additions/deletions to an **ABAddressBookRef** are reflected when **ABAddressBookSave** is called. In Windows Contacts, each individual addition/deletion results in the address book being immediately updated. The address book’s manager will maintain a list of “pending” changes whenever the **ABAddressBookRef** is updated, and then when **ABAddressBookSave** is called, it will go through and actually apply them.
3. Specific fields/properties of a single contact don’t perfectly line up. The major categories (names, addresses, emails, phones, birthdays, notes, etc) line up the same, but iOS has a few fields that Windows Contacts do not. Examples include contact creation date, alternate birthdays, and address country codes. Another difference is that many Windows Contacts properties have a length limit – for example, names are limited to 64 characters on the Windows side, addresses are limited to 1024 characters, and so on. The length limit will be checked, and if violated, an appropriate error will be returned to the user. See the below table for a mapping of fields, and any limits in length imposed by Windows:

|  |  |  |
| --- | --- | --- |
| iOS Field/Property Name | Windows Field/Property Name | Windows Character Limit |
| kABPersonFirstNameProperty | Contact.FirstName | 64 |
| kABPersonLastNameProperty | Contact.LastName | 64 |
| kABPersonMiddleNameProperty | Contact.MiddleName | 64 |
| kABPersonPrefixProperty | Contact.HonorificNamePrefix | 32 |
| kABPersonSuffixProperty | Contact.HonorificNameSuffix | 32 |
| kABPersonNicknameProperty | Contact.Nickname |  |
| kABPersonFirstNamePhoneticProperty | YOMI |  |
| kABPersonLastNamePhoneticProperty | YOMI |  |
| kABPersonMiddleNamePhoneticProperty | - |  |
| kABPersonOrganizationProperty | Contact.JobInfo.CompanyName | 64 |
| kABPersonJobTitleProperty | Contact.JobInfo.Title | 1024 |
| kABPersonDepartmentProperty | Contact.JobInfo.Department | 100 |
| kABPersonEmailProperty | Contact.Emails.ContactEmail |  |
| kABPersonBirthdayProperty | Contact.ImportantDates.ContactDateKind |  |
| kABPersonNoteProperty | Contact.Notes | 4096 |
| kABPersonCreationDateProperty | - |  |
| kABPersonModificationDateProperty | - |  |
|  |  |  |
| kABPersonAddressProperty | Contact.Addresses |  |
| kABPersonAddressStreetKey | ContactAddress.StreetAddress | 1024 |
| kABPersonAddressCityKey | ContactAddress.(Region or Locality) | 1024 |
| kABPersonAddressStateKey | ContactAddress.(Region or Locality) | 1024 |
| kABPersonAddressZIPKey | ContactAddress.PostalCode | 1024 |
| kABPersonAddressCountryKey | ContactAddress.Country | 1024 |
| kABPersonAddressCountryCodeKey | - |  |
|  |  |  |
| kABPersonDateProperty | Contact.ImportantDates |  |
| kABPersonAnniversaryLabel | ContactDateKind |  |
|  |  |  |
| kABPersonKindProperty | - |  |
|  |  |  |
| kABPersonPhoneProperty | Contact.Phones |  |
| kABPersonPhoneMobileLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhoneIPhoneLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhoneMainLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhoneHomeFAXLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhoneWorkFAXLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhoneOtherFAXLabel | ContactPhone.Number/Kind/Description |  |
| kABPersonPhonePagerLabel | ContactPhone.Number/Kind/Description |  |
|  |  |  |
| kABPersonInstantMessageProperty | Contact.ConnectedServiceAccounts |  |
| kABPersonSocialProfileProperty |  |  |
|  |  |  |
| kABPersonURLProperty | Contact.Websites |  |
|  |  |  |
| kABPersonRelatedNamesProperty | Contact.SignificantOthers |  |
|  |  |  |
| kABPersonAlternateBirthdayProperty | - |  |

# [Functional and Unit Testing](http://portals/wpblue/blue/wpbluewiki/Pages/TestValidation.aspx" \o "Communicates the strategy for creating and maintaining the reliability of the feature)

## Test Approach

Testing will primarily be done with unit tests. The AddressBookSample test app will also be used – using the app is a way of ensuring that the AddressBook Framework works as desired, and the app running on iOS can be a reference of the desired behavior.

# [Prioritization](http://windowsblue/docs/home/Windows%20Spec%20Wiki/Dev%20Appendices.aspx" \o "Update the table in this section by right-clicking on it and selecting Update Field (the Update Entire Table)) of Functions

|  |  |
| --- | --- |
| Function | Priority |
| ABAddressBook |  |
|  |  |
| ABAddressBookCreate | P0 |
| ABAddressBookCreateWithOptions | P0 |
| ABAddressBookGetAuthorizationStatus | P0 |
| ABAddressBookRequestAccessWithCompletion | P0 |
| ABAddressBookHasUnsavedChanges | P0 |
| ABAddressBookSave | P0 |
| ABAddressBookRevert | P0 |
|  |  |
| ABAddressBookAddRecord | P1 |
| ABAddressBookRemoveRecord | P2 |
|  |  |
| ABAddressBookRegisterExternalChangeCallback | P2 |
| ABAddressBookUnregisterExternalChangeCallback | P2 |
|  |  |
| ABAddressBookCopyLocalizedLabel | P2 |
|  |  |
| ABMultiValue |  |
|  |  |
| ABMultiValueCopyValueAtIndex | P0 |
| ABMultiValueCopyArrayOfAllValues | P0 |
| ABMultiValueGetCount | P0 |
| ABMultiValueGetFirstIndexOfValue | P0 |
| ABMultiValueCopyLabelAtIndex | P0 |
|  |  |
| ABMultiValueGetIdentifierAtIndex | P0 |
| ABMultiValueGetIndexForIdentifier | P0 |
|  |  |
| ABMultiValueGetPropertyType | P0 |
|  |  |
| ABMutableMultiValue |  |
|  |  |
| ABMultiValueCreateMutable | P1 |
|  |  |
| ABMultiValueCreateMutableCopy | P1 |
|  |  |
| ABMultiValueAddValueAndLabel | P1 |
| ABMultiValueReplaceValueAtIndex | P2 |
| ABMultiValueReplaceLabelAtIndex | P2 |
| ABMultiValueInsertValueAndLabelAtIndex | P1 |
| ABMultiValueRemoveValueAndLabelAtIndex | P2 |
|  |  |
| ABRecord |  |
|  |  |
| ABRecordGetRecordID | P0 |
| ABRecordGetRecordType | P0 |
|  |  |
| ABRecordSetValue | P1 |
| ABRecordCopyValue | P0 |
| ABRecordRemoveValue | P2 |
| ABRecordCopyCompositeName | P2 |
|  |  |
| ABPerson |  |
|  |  |
| ABPersonCreate | P1 |
| ABPersonCreateInSource | P2 |
|  |  |
| ABPersonComparePeopleByName | P2 |
|  |  |
| ABPersonGetTypeOfProperty | P2 |
| ABPersonCopyLocalizedPropertyName | P2 |
|  |  |
| ABPersonSetImageData | P2 |
| ABPersonCopyImageData | P2 |
| ABPersonCopyImageDataWithFormat | P2 |
| ABPersonHasImageData | P2 |
| ABPersonRemoveImageData | P2 |
|  |  |
| ABAddressBookGetPersonCount | P1 |
| ABAddressBookGetPersonWithRecordID | P1 |
| ABAddressBookCopyArrayOfAllPeople | P0 |
| ABAddressBookCopyArrayOfAllPeopleInSource | P2 |
| ABAddressBookCopyArrayOfAllPeopleInSourceWithSortOrdering | P2 |
| ABAddressBookCopyPeopleWithName | P2 |
| ABPersonCopyArrayOfAllLinkedPeople | P2 |
| ABPersonCopySource | P2 |
|  |  |
| ABPersonCopyCompositeNameDelimiterForRecord | P2 |
| ABPersonGetSortOrdering | P2 |
| ABPersonGetCompositeNameFormat | P2 |
| ABPersonGetCompositeNameFormatForRecord | P2 |
|  |  |
| ABPersonCreatePeopleInSourceWithVCardRepresentation | P2 |
| ABPersonCreateVCardRepresentationWithPeople | P2 |
|  |  |
| ABSource |  |
|  |  |
| ABAddressBookCopyDefaultSource | P2 |
| ABAddressBookGetSourceWithRecordID | P2 |
| ABAddressBookCopyArrayOfAllSources | P2 |
|  |  |
| ABGroup |  |
|  |  |
| ABGroupCreate | P2 |
| ABGroupCreateInSource | P2 |
|  |  |
| ABGroupCopyArrayOfAllMembers | P2 |
| ABGroupCopyArrayOfAllMembersWithSortOrdering | P2 |
| ABGroupAddMember | P2 |
| ABGroupRemoveMember | P2 |
|  |  |
| ABAddressBookGetGroupWithRecordID | P2 |
| ABAddressBookGetGroupCount | P2 |
| ABAddressBookCopyArrayOfAllGroups | P2 |
| ABAddressBookCopyArrayOfAllGroupsInSource | P2 |
| ABGroupCopySource | P2 |