CardDAV

Cyrus Daboo, Apple Inc.

Information

- vCard extensions to WebDAV.
- Modeled after CalDAV (RFC4791).
- Currently draft -04.
- Mailing list: vcarddav@ietf.org

Data Model (1)

- vCard is used for storing address/contact information.
- However, servers MAY support other formats that have the same semantics (i.e. one-to-one XML mapping).
- “Address Object Resource” is stored in an “Address Book Collection”
- MUST contain only one vCard component.

Data Model (2)

- An “Address Book Collection” is a WebDAV collection with a special resource type.
- Address object resources (vCards) are stored inthe address book collection.
- All vCards within a collection MUST have a UID and UIDs MUST all be unique in that collection.
- Address books cannot be contained within another address book.

Protocol Elements (1)

- Extended MKCOL is used to create address book collections.
- draft-daboo-webday-mkcol-00.txt
- That is an MCOL request with an XML body that allows properties to be set at the time of creation.

MKCOL Example

```
MKCOL /home/lisa/addressbook/ HTTP/1.1
Host: addressbook.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol xmlns:D="DAV:"
         xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:set>
    <D:prop>
      <D:resourcetype>
        <D:collection/>
        <C:addressbook/>
      </D:resourcetype>
      <D:displayname>Lisa's Contacts</D:displayname>
      <C:addressbook-description xml:lang="en"
>My address book.</C:addressbook-description>
    </D:prop>
  </D:set>
</D:mkcol>
```

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol-response xmlns:D="DAV:"

xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:propstat>
    <D:prop>
      <D:resourcetype/>
      <D:displayname/>
      <C:addressbook-description/>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:mkcol-response>
```

MKCOL Example

```
MKCOL /home/lisa/addressbook/ HTTP/1.1
Host: addressbook.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol xmlns:D="DAV:"
         xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:set>
    <D:prop>
      <D:resourcetype>
        <D:collection/>
        <C:addressbook/>
      </D:resourcetype>
      <D:displayname>Lisa's Contacts</D:displayname>
      <C:addressbook-description xml:lang="en"
>My address book.</C:addressbook-description>
    </D:prop>
  </D:set>
</D:mkcol>
```

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol-response xmlns:D="DAV:"

xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:propstat>
    <D:prop>
      <D:resourcetype/>
      <D:displayname/>
      <C:addressbook-description/>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:mkcol-response>
```

MKCOL Example

```
MKCOL /home/lisa/addressbook/ HTTP/1.1
Host: addressbook.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol xmlns:D="DAV:"
         xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:set>
    <D:prop>
      <D:resourcetype>
        <D:collection/>
        <C:addressbook/>
      </D:resourcetype>
      <D:displayname>Lisa's Contacts</D:displayname>
      <C:addressbook-description xml:lang="en"
>My address book.</C:addressbook-description>
    </D:prop>
  </D:set>
</D:mkcol>
```

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol-response xmlns:D="DAV:"

xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:propstat>
    <D:prop>
      <D:resourcetype/>
      <D:displayname/>
      <C:addressbook-description/>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:mkcol-response>
```

MKCOL Example

```
MKCOL /home/lisa/addressbook/ HTTP/1.1
Host: addressbook.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol xmlns:D="DAV:"
         xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:set>
    <D:prop>
      <D:resourcetype>
        <D:collection/>
        <C:addressbook/>
      </D:resourcetype>
      <D:displayname>Lisa's Contacts</D:displayname>
      <C:addressbook-description xml:lang="en"
>My address book.</C:addressbook-description>
    </D:prop>
  </D:set>
</D:mkcol>
```

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:mkcol-response xmlns:D="DAV:"

xmlns:C="urn:ietf:params:xml:ns:carddav">
  <D:propstat>
    <D:prop>
      <D:resourcetype/>
      <D:displayname/>
      <C:addressbook-description/>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:mkcol-response>
```

Protocol Elements (2)

- Create an address object resource using PUT.
- Retrieve an address object resource using GET.
- Make use of HTTP ETags to only GET content that has changed and to deal with “lost update” issues on PUT.
- WebDAV “pre-conditions” and corresponding DAV:error codes used to indicate invalid content on PUT (e.g. duplicate UIDs).

Protocol Elements (3)

- WebDAV ACLs used as the basis for “sharing” address books and content.
- CARDDAV:addressbook-home-set principal property tells a client where a user’s address books are.
- CARDDAV:principal-address principal property points to a vCard for the user represented by the principal (“my card”).

Protocol Elements (4)

- CARDDAV:addressbook-query REPORT allows a client to query for matching vCards.
- Client can request specific WebDAV properties be returned for each match.
- Client can request all data or some parts of the vCard data be returned for each match (e.g. only return email address and full name - I don't care about the photos).

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

RFC4790 Collations

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

Query Example (1)

```
REPORT /home/bernard/addressbook/ HTTP/1.1
Host: addressbook.example.com
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:addressbook-query xmlns:D="DAV:"
                     xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:prop>
  <D:getetag/>
  <C:address-data>
      <C:prop name="VERSION"/>
      <C:prop name="UID"/>
      <C:prop name="NICKNAME"/>
      <C:prop name="EMAIL"/>
      <C:prop name="FN"/>
  </C:address-data>
 </D:prop>
 <C:filter>
  <C:prop-filter name="NICKNAME">
      <C:text-match collation="i;unicode-casemap" match-type="equals"
      >me</C:text-match>
  </C:prop-filter>
 </C:filter>
</C:addressbook-query>
```

Query Example (1)

```
HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
                        xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:response>
   <D:href>/home/bernard/addressbook/v102.vcf</D:href>
   <D:propstat>
        <D:prop>
          <D:getetag>"23ba4d-ff11fb"</D:getetag>
          <C:address-data>BEGIN:VCARD
VERSION:3.0
NICKNAME:me
UID:34222-232@example.com
FN:Cyrus Daboo
EMAIL:daboo@example.com
END:VCARD
</C:address-data>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
 </D:response>
</D:multistatus>
```

Query Example (1)

```
HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
                        xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:response>
   <D:href>/home/bernard/addressbook/v102.vcf</D:href>
   <D:propstat>
         <D:prop>
           <D:getetag>"23ba4d-ff11fb"</D:getetag>
           <C:address-data>BEGIN:VCARD
VERSION:3.0
NICKNAME:me
UID:34222-232@example.com
FN:Cyrus Daboo
EMAIL:daboo@example.com
END:VCARD
</C:address-data>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
 </D:response>
</D:multistatus>
```

Query Example (1)

```
HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
                        xmlns:C="urn:ietf:params:xml:ns:carddav">
 <D:response>
   <D:href>/home/bernard/addressbook/v102.vcf</D:href>
   <D:propstat>
        <D:prop>
          <D:getetag>"23ba4d-ff11fb"</D:getetag>
          <C:address-data>BEGIN:VCARD
VERSION:3.0
NICKNAME:me
UID:34222-232@example.com
FN:Cyrus Daboo
EMAIL:daboo@example.com
END:VCARD
</C:address-data>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
 </D:response>
</D:multistatus>
```

Protocol Elements (5)

- CARDDAV:addressbook-multiget REPORT: retrieval of multiple vCards in a single request.
- Client can request specific WebDAV properties be returned for each match.
- Client can request all data or some parts of the vCard data be returned for each match.
- Optimization (does make a big difference in CalDAV).

Additional Work

- Both CalDAV and CardDAV and indeed WebDAV in general needs better synchronization of collection contents.
  - One approach: draft-daboo-webdav-sync-00.
- Also, better notification of changes: something akin to the goals of IMAP IDLE that help avoid the need to poll.

To Do

- Some extended MKCOL feedback needs to be addressed.
- Ready for last call.
- Work on synchronization.
- Work on notifications (more important to CalDAV).