Liberty ID-SIS Contact Book Service Specification

Version: 1.0

Editors:
Sampo Kellomäki, Symlabs, Inc.

Contributors:
Rajeev Angal, Sun
Salima Fazal Karim, France Télécom
Sean Franklin, American Express
Ariel Gordon, France Télécom
Jukka Kainulainen, Nokia
Kurt Kolok, IEEE-ISTO
Guillaume Lambert, France Télécom
Rob Lockhart, IEEE-ISTO

Abstract:

The Liberty ID-SIS Contact Book (ID-SIS-CB) specifies a web service offering contact information. ID-SIS-CB is an instance of data-oriented identity web service. ID-SIS-CB is characterized by ability to query and to update attribute data and incorporates from other specifications mechanisms for access control and conveying data validation information and usage directives. Readers of this document should be familiar with SOAP, SAML, XML and vCard. Readers may also wish to familiarize themselves with the Liberty ID-SIS Personal Profile (ID-SIS-PP).

Filename: liberty-id-sis-cb-v1.0.pdf

Notice

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose.** Implementers of this Specification are advised to review the Liberty Alliance Project's website (http://www.projectliberty.org/) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org

Contents

1. Introduction

The Liberty ID-SIS Contact Book (ID-SIS-CB) is a Liberty identity service that allows a Principal to manage contacts for private and business acquaintances, friends, family members, and even for herself. ID-SIS-CB supports communications applications, acting as Principal's phone or address book, allowing her to quickly and easily to locate the information needed to contact people and businesses. Many other applications may also benefit, e.g., an e-commerce application may enable the Principal to ship to any address in the Principal's Contact Book.

ID-SIS-CB is a collection of *Contact Cards* and *Distribution Lists*. ID-SIS-CB Service offers a [LibertyDST20]-compatible remote call interface for accessing and editing these. An ID-SIS-CB Service may, and often will, also offer a user interface that allows the Principal to manipulate her contact cards and distribution lists as well as view and set permissions. Specification of this user interface is, however, not in the scope of this document. It is presumed to be implementation-dependent.

This document is the Liberty ID Services Interface Specification that normatively specifies the Contact Book Service. For fuller understanding, it is also helpful to consult the implementation guidelines [LibertyCBGuide] and documentation for other related services such as the Personal Profile ([LibertyIDPP] and [LibertyIDPPGuide]) and the Employee Profile ([LibertyIDEP] and [LibertyIDEPGuide]). Since ID-SIS-CB is based on vCard, familiarity with [vCard21], vCard 3.0 [RFC2426], MIME-DIR [RFC2425], and [vCardRDF] is highly recommended.

In case of disagreement between the present document and any guidelines or XML schema descriptions, this document is prescriptive. Any published errata is hereby incorporated to this document by reference and as such is normative.

1.1. Notational Conventions

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in IETF [RFC2119]. These keywords are thus capitalized, when used, to specify, unambiguously, requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

> N.B. Formal vCard, or more properly MIME-DIR [RFC2425], terminology is used wherever possible. To avoid confusion, the reader should note that the vCard term "`type`" refers to what is formally an "`element`" in XML terminology and formal PP specs, but what is commonly understood to be an "`attribute`" or field of a record. Similarly, the vCard term "`parameter`" has a role similar to an XML "`attribute`," which is quite different from the common concept of "`attribute`" in the sense of vCard "`type`."

1.2. Derivation of ID-SIS-CB from DST and WSF

The ID-SIS-CB service is an instance of the Data Services Template [LibertyDST20] and all stipulations of [LibertyDST20] are hereby incorporated unless expressly waived or modified in this document.

A service that consults the ID-SIS-CB service MUST use the Liberty architectural framework (see [LibertyIDWSFGuide10]). The Liberty architectural framework ensures that a service acts on behalf of the Principal or that the Principal has consented to sharing the data. A service that consults an ID-SIS-CB-compliant service MUST adhere to the specifications that comprise the Liberty architectural framework (see [LibertyIDWSFOverview11]). A service MUST be able to demonstrate adherence to the specifications.

145 **Table 1. DST General Service Parameters**

Parameter	Value
`ServiceType`	`urn:liberty:id-sis-cb:2005-05`
Discovery Options	See Section 3.1
Data Schema	See [RFC2426] and Section 2 and Section 8.1
`SelectType Element`	See Section 3.2
Query Language	Restricted XPath 1.0 [XPATH], see Section 3.2.1 and Section 3.2.2
Multiple elem uniqueness	Each contact card is labeled with `CARDID` element.
Data Extension Supported	MAY, using vCard extension mechanisms

146 **Table 2. DST Query Parameters**

Parameter	Value
Support querying	MUST
Multiple `Query`	MAY
Multiple `QueryItem`	MUST
Support sorting	SHOULD, request MUST be understood, but actual sorting can be done on "best effort" basis.
`SortType` definition	See Section 3.5
Support `changedSince`	MAY
`changedSince` formats	All
Support `includeCommonAttributes`	MUST
Support pagination	MUST
Support static sets	MAY
Extension in `Query`	MUST NOT

147 **Table 3. DST Modify Parameters**

Parameter	Value
Support Modification	MAY
Multiple `Modify`	MAY
Multiple Modification	MUST, restrictions on modify target
Support partial success	MUST NOT
Support `notChangedSince`	MAY
Extension in `Modify`	MUST NOT

148 **Table 4. DST Subscribe Parameters**

Parameter	Value
Support subscribing	MUST
Use of `Subscribe` element	Creation, renewal, cancellation, and modification of subscriptions MUST be supported.
Multiple `Subscribe`	MAY
`NotifyEndedTo`	MAY be used and MUST be silently skipped, if not implemented
`TypeType` definition	[LibertyDST20] default, i.e., empty
`TriggerType` definition	[LibertyDST20] default, i.e., empty
Start of subscription	MAY, MUST be silently skipped, if not implemented
Subscription expiration	MAY, MUST be silently skipped, if not implemented
Use of expires and duration attributes	MUST support both if subscription expiration is supported
Support `expires==starts`	MUST
Support zero duration	MUST
Extension in `Subscribe`	MUST NOT

149 **Table 5. DST QuerySubscriptions Parameters**

Parameter	Value
Support querying existing subscriptions	MUST
Multiple `QuerySubscription` element	MAY
Extension in `QuerySubscriptions`	MUST NOT

150 **Table 6. DST Notify Parameters**

Parameter	Value
Support Notifications	MAY
Notification acknowledgements	MUST
Extension in `Notify`	MUST NOT

151 **Table 7. DST EndNotify Parameters**

Parameter	Value
Support end notifications	MAY, server side MUST
End notification acknowledgements	MUST
Extension in `Ended`	MUST NOT

152 1.3. Relation of ID-SIS-CB to vCard Specifications

ID-SIS-CB is based on vCard, as specified in [vCard21] and [RFC2426]. All vCard types and their semantics, but not necessarily format, are hereby incorporated and their descriptions will not be repeated here.

This specification is divided in two levels:

a. an abstract vCard data model upon which queries and modifications are applied. We call this "generic vCard" and use it to specify query language and modification operations.

b. mappings of the abstract data model to specific vCard representations such as [vCard21], [RFC2425], or various XML representations. Some mappings are provided in this document, but more mappings may be defined in future versions of this document or in other documents.

The purpose of the generic vCard description combined with the mappings is to enable support for multiple vCard formats to coexist without endorsing any specific format. It is expected that new future vCard formats may be accommodated in this way.

The `cb:Card` element of ID-SIS-CB MUST have as content a vCard. The format of the vCard MUST be specified by `cb:format` XML attribute whose possible values MUST be specified by each mapping from generic vCard to specific format. `cb:Card` is a Liberty CB protocol feature that should not be confused with a conceptual data model, i.e., Jabber, vcard-temp:vCard. The latter may appear within a `cb:Card` element.

```
<xs:element name="Card" type="cb:cardType"/>
<xs:complexType name="cardType">
   <xs:choice>
      <xs:element ref="cb:charData"/>
      <xs:any namespace="##other" processContents="lax"/>
   </xs:choice>
   <xs:attribute ref="cb:format" use="required"/>
</xs:complexType>
<xs:attribute name="format" type="xs:anyURI"/>
<xs:element name="charData" type="cb:charDataType"/>
<xs:simpleType name="charDataType">
   <xs:restriction base="xs:string"/> <!-- vCard data in specified format -->
</xs:simpleType>
```

The `cb:Card` element can contain either XML-formatted vCard data qualified by appropriate name space (i.e., the `xs:any extension`) or `cb:CharData` element containing vCard data in non-XML format, typically either [vCard21] or [RFC2425].

If multiple variants of vCard are offered by a Contact Book Service at a single end point, the variant of vCard is explicitly requested using `cb:format` XML attribute of `Select` element and indicated in each `cb:Card` using `cb:format` XML attribute.

vCard and general Liberty practices regarding naming conventions and cases sensitivity of "attribute" names may appear to conflict, but in fact this is not the case. The vCard is represented as content of `cb:Card` element, thus

1. the CB protocol messages as a whole MUST follow the XML and Liberty conventions, and

2. the contents of the `cb:Card` element MUST follow the conventions of the advertised vCard format.

However, as a special additional requirement to the rule established in item 2, above, any vCard data that could confuse a compliant XML parser MUST be escaped or encoded using vCard conventions such that no confusion can arise, e.g., vCard data contains something that looks like an XML tag, especially a closing vCard tag.

While vCard specifications give latitude in choosing character sets and encoding for the data, the contents of the `cb:Card` element MUST use UTF-8 encoding per general ID-WSF rules.

1.4. Compliance

This specification defines an interface to which an *implementation* and an *instance* (deployment) of ID-SIS-CB service MUST conform. For an AP to be ID-SIS-CB compliant, it MUST adhere to all aspects of the specification.

A minimally-compliant ID-SIS-CB implementation MAY support less than all of the ID-SIS-CB containers or elements or some features (optional containers, elements, and features). Such an implementation may be labeled as an "ID-SIS-CB implementation" provided that publicly-available documentation about the implementation clearly discloses which optional parts of the schema and which features are not supported. All other features and schema are assumed to be supported. A deployment of an implementation that is not capable of supporting the full schema SHOULD only register the discovery option keywords that accurately reflect its capabilities.

An implementation that supports all of the schema and features specified in this document MAY be labeled as a "full ID-SIS-CB implementation." An implementation that is deficient in any feature or part of the schema MUST NOT be labeled as a "full ID-SIS-CB implementation." A "full ID-SIS-CB implementation" deployment MAY restrict, administratively, the schema and features.

A deployment that supports all of the schema and features specified in this document MAY be labeled as a "full ID-SIS-CB deployment" or a "full ID-SIS-CB service." To meet full ID-SIS-CB deployment status, all of the schema and features MUST be supported for all Principals wishing to use them, barring a policy decision excluding some Principal.

A deployment that only supports some subset of ID-SIS-CB may still be labeled as an "ID-SIS-CB deployment" or an "ID-SIS-CB service" provided that the deployment publicly discloses the subset that it supports.

1.5. Namespaces and Preamble

The namespace for the ID-SIS-CB service is designated by the URI:

```
urn:liberty:id-sis-cb:2005-05
```

The Contact Book namespace is abbreviated as "`cb:`" in this document. If the namespace has been omitted at any place in this document, "`cb:`" should be considered the default namespace. The namespace URI is also used as the `ServiceType` designator.

Table 8. Referenced XML Namespaces

Prefix	URI	Description
`xml:`	`http://www.w3.org/TR/REC-xml`	XML Definition [XML] (for `xml:lang`)
`xs:`	`http://www.w3.org/2002/XMLSchema`	XML Schema Definition [Schema1]
`cdm:`	`urn:liberty:cb:conceptual-data-model:2005-05`	ID-SIS-CB conceptual data model, derived from Jabber vcard-temp DTD [JEP0054] [JReg]

The following preamble introduces the namespaces and [LibertyDST20].

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
      xmlns="urn:liberty:id-sis-cb:2005-05"
      xmlns:cb="urn:liberty:id-sis-cb:2005-05"
      xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:liberty:id-sis-cb:2005-05"
      elementFormDefault="qualified">
   <xs:import
         namespace="urn:liberty:cb:conceptual-data-model:2005-05"
         schemaLocation="liberty-id-sis-cb-cdm-v1.0.xsd"/>
   <xs:include schemaLocation="liberty-idwsf-dst-v2.0.xsd"/>
   <xs:include schemaLocation="liberty-idwsf-dst-dt-v2.0.xsd"/>
```

2. Generic vCard

Operations of ID-SIS-CB operate on a generic vCard data model that is expressed in XML. This is to provide a common basis to which query and modification operations are applied. The vCard data on the wire, however, may be in any format for which there is a mapping. Conceptually, the vCard types are represented by XML elements and containers as specified in [JEP0054] and schema in Section 8.2. These elements appear in `cdm:` namespace. Semantics of the vCard types are normatively specified in [RFC2426] and [vCard21]. The semantics of the calendaring-related attributes, namely `CALURI`, `CAPURI`, `CALADRURI`, and `FBURL`, are normatively specified in [RFC2739].

The table "Mapping vCard to XML" summarizes how the mapping is done. Each vCard type corresponds to an XML element of the same name. Element names are in all capital letters. Some vCard types, such as `N` and `ADR`, have useful internal structure which is made available in an analyzed form in the XML representation. Such a type maps to a container which has an element for each useful subcomponent.

Table 9. Mapping vCard to XML

vCard type	Generic vCard XML representation	Notes
`BEGIN`	n/a	
`END`	n/a	
`PROFILE`	n/a	
`NAME`	n/a	
`SOURCE`	n/a	
`N`	`FAMILY`, `GIVEN`, etc.	[JEP0054] Section 2, [DAWSON]
`ADR`	`STREET`, `LOCALITY`, `CTRY`, etc.	[JEP0054] Section 2, [DAWSON]
`ORG`	`ORGNAME`, `ORGUNIT`	[JEP0054] Section 2, [DAWSON]
`JABBERID`	`JABBERID`	Jabber extension
`DESC`	`DESC`	Jabber extension
others in [RFC2426]	An element by same name as type	As specified in [JEP0054] and [DAWSON]
any other (extended)	An element by same name as type	Mapping may specify other representation

The vCard `TYPE` parameter is mapped to XML elements as described in [JEP0054] and [DAWSON]. Note that the philosophy of [JEP0054] and [DAWSON], representing vCard `TYPE` parameters as XML elements, is followed, despite this being in apparent conflict with the [RFC2426] philosophy of representing them as multivalued `TYPE` parameters.

The vCard group feature is represented by `cb:group` XML attribute.

[LibertyDST20]-derived standard attributes are added to all vCard elements.

```
<xs:attributeGroup name="typeAttributes">
   <xs:attributeGroup ref="localizedLeafAttributes"/>
   <xs:attribute ref="group"/>
</xs:attributeGroup>
<xs:attribute name="group">
   <xs:simpleType>
      <xs:restriction base="xs:string"/>
   </xs:simpleType>
</xs:attribute>
```

ID-SIS-CB imposes a restriction on the vCard AGENT type: it MUST NOT contain another vCard. It MAY contain a reference to another vCard. When importing a vCard AGENT type that contains another vCard, the implementation MAY drop the AGENT type entirely. However, it is RECOMMENDED that such AGENT type is handled by creating a vCard corresponding to the content of the AGENT type in the Contact Book and populating a reference to this vCard in the AGENT element.

Human language of a vCard type may be specified by using xml:lang XML attribute in the corresponding element. If the same element appears in multiple languages, multiple instances of the element are generated.

All vCard data is inside top level element cdm:vCard.

2.1. Representing "Self" Card

Self card is indicated by including in cdm:vCard a child element: cdm:SELF.

2.2. Representing Favorites List

A card's membership on the favorites list is indicated by including in cdm:vCard a child element: cdm:FAVORITE.

2.3. Unique Card Identification

Every contact card in a given contact book MUST have unique identification indicated by cdm:CARDID element. The card identification refers to the card itself and is not meant to be identification of the person or business to whom or to which the card refers.

Card identification MUST be unique within a contact book as seen by a WSC. Card identification MUST be different for different WSCs. Card identification MUST NOT be unique across different contact books.

2.4. Representing Distribution Lists

A vCard may represent a distribution list. This is indicated by including in cdm:vCard a child element: cdm:DISTRIBUTIONLIST. Such card may have any attributes that an ordinary card has, but is not a contact by itself - rather it is a place holder for the list. The place holder card allows a distribution list to be returned as part of a query and the attributes common to all members of the list to be represented. The distribution list has a cdm:CARDID like any other contact card.

2.5. List Membership

Membership of a card on a distribution list is indicated by cdm:LISTMEMBER element whose value is the cdm:CARDID of the distribution list place holder card. An implementation MUST support a card belonging to multiple distribution lists. This is indicated by having as many cdm:LISTMEMBER elements as there are memberships.

An implementation MUST permit a card to be recorded multiple times as a member of a distribution list, but MAY collapse multiple memberships to just one membership. A WSC MUST NOT depend on the ability to differentiate between multiple memberships and single membership on a distribution list.

2.6. Data Model Extension

All containers and elements defined in the ID-SIS-CB schema have an Extension element which permits arbitrary schema extension. An implementation MAY support schema extension, but is not required to do so. If an implementation does support schema extension then it MAY register the urn:liberty:dst:can:extend discovery option keyword.

vCard specifications specify extension mechanisms for vCard. In the conceptual data model, the extension is realized by the `Extension` container, which may contain any well-formed XML data. Every extended `TYPE` of a vCard maps to a correspondingly-named element inside the `Extension` container.

Given that [RFC2426] representations of vCard are not capable of expressing name spaces, the extended data elements inside the `cdm:Extension` container are not qualified into any name space, i.e., they appear in the document's default name space. Since it is difficult to foresee extensions, generally, the schema for the default name space will not contain the extended elements. Thus it is recommended that implementations either use an out-of-band method to obtain correct schema and to set the default name space, or simply not to verify the schema for contents of the `cdm:Extension` element.

Example

```
<cdm:vCard>
  <cdm:FN>Zita Lopes</cdm:FN>
  <cdm:N>
    <cdm:FAMILY>Lopes</cdm:FAMILY>
    <cdm:GIVEN>Zita</cdm:GIVEN>
  </cdm:N>
  <cdm:Extension>
    <X-FAVCOLOR>red</X-FAVCOLOR>
  </cdm:Extension>
</cdm:vCard>
```

This Conceptual Data Model document contains an extended attribute called `X-FAVCOLOR` whose value is `red`. The corresponding vCard could be

```
BEGIN:vCard
FN:Zita Lopes
N:Lopes;Zita
X-FAVCOLOR:red
END:vCard
```

Note that the extension attribute has a preceding "`X-`" to indicate that it is a vCard extension, as specified in Section 5.4 of [RFC2425].

3. Protocol Operations

3.1. Discovery Option Keywords

ID-SIS-CB defines a number of discovery keywords to be included as `Option` elements in discovery registrations and queries. (See [LibertyDisco12].) Some of these express the availability of data and others the ability to update data. An attribute provider MAY advertise an ability to update data even if it currently does not have a given data item populated for the Principal.

3.1.1. Data Availability Discovery Option Keywords

The data availability oriented keywords extract selected components from the profile as if an [XPATH] expression were applied. An implementation is not required to use [XPATH] as long as the results are equivalent. Presence of a keyword implies a high probability that the corresponding data can be obtained, if queried. However, the data may not be available due to permissions or race conditions between data removal and updates to the discovery service.

Table 10. Data Availability Discovery Option Keywords

Keyword	Equivalent [XPATH]s	Meaning
`urn:liberty:id-sis-cb`	`/cdm:card`	Has some ID-SIS-CB data

An attribute provider MUST NOT register a data availability discovery option keyword if it is *probable* that the data will not be available. For example, if an AP does not yet have the data, it MUST NOT register the keyword with an intent of gathering the data by the time it is requested or with the intent of gathering the data when requested via the Interaction Service protocol [LibertyInteract11]. An attribute provider SHOULD NOT register a keyword if the Principal has set such permissions on the data that it may not be released under any plausible circumstances.

3.1.2. Data Update Discovery Option Keywords

The data update discovery option keywords express the willingness and ability of the attribute provider to store some data corresponding to the given [XPATH] expression. These keywords do not imply that the AP currently has any data regarding the containers referenced by the keyword.

Table 11. Data Update Discovery Option Keywords

Keyword	Equivalent XPATHs	Meaning
`urn:liberty:id-sis-cb:can`	`/cdm:card`	Can store some ID-SIS-CB data

An implementation MUST NOT register a data update discovery option keyword unless some `Modify` request regarding the data referenced by the keyword can plausibly succeed. For example, if an AP is read-only, it MUST NOT register any data update discovery option keywords. Similarly, if the underlying database is incapable of storing the data, then the keyword MUST NOT be advertised.

An implementation that registers a data update discovery option keyword SHOULD be capable of accepting any `Modify` request (subject to permissions) regarding that category of data and SHOULD support all elements specified in ID-SIS-CB schema for that category.

An implementation MAY also choose to support a read-only service. A read-only service MUST NOT register any data update discovery option keywords.

3.1.3. Feature Discovery Option Keywords

An implementation may indicate support for optional features of this specification using feature discovery option keywords as described in the following table.

Table 12. Feature Discovery Option Keywords

Keyword	Meaning
`urn:liberty:id-sis-cb:changedSince`	`changedSince` is supported for queries
`urn:liberty:id-sis-cb:notChangedSince`	`notChangedSince` is supported for modifies
`urn:liberty:id-sis-cb:sort:mru`	`mru()` criteria is supported for sort
`urn:liberty:id-sis-cb:sort:mfu`	`mfu()` criteria is supported for sort
`urn:liberty:id-sis-cb:sort:fav`	`fav()` criteria is supported for sort
`urn:liberty:id-sis-cb:sort:any`	sorting by any criteria is supported
`urn:liberty:id-sis-cb:format:RFC2426`	vCard 3.0 format is supported
`urn:liberty:id-sis-cb:format:v2.1`	vCard 2.1 format is supported
`urn:liberty:id-sis-cb:format:vcard-temp`	vCard Jabber format is supported
`urn:liberty:id-sis-cb:format:vcard-rdf`	vCard RDF format is supported
`urn:liberty:dst:can:extend`	Data schema extension is supported

3.2. Query Expressions

The [LibertyDST20] specifies a `Query` element that potentially contains several `QueryItem` elements which in turn contain a `Select` element, but leaves the actual `select` expression undefined. ID-SIS-CB uses [XPATH] as the query language, but with some restrictions and modifications.

The `Select` container MUST be present. The `SelectType` is defined as follows:

```
<xs:complexType name="SelectType">    <!-- connects to DST framework -->
   <xs:simpleContent>
      <xs:extension base="xs:string">     <!-- XPath expression -->
         <xs:attribute ref="cb:format"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

A query has the following major components.

1. Which contact cards and distribution lists to return?
2. How to sort them? (See Section 3.5.)
3. Which attributes to return from them?
4. How many results are desired and what is the starting position in the data set? For specifying these, ID-SIS-CB uses the pagination mechanism described in Section 4.1.1 of [LibertyDST20].
5. In which format should the cards be returned? This is specified using `cb:format` XML attribute. It MUST be supplied in queries.

3.2.1. Additional and Value-Added Criteria

Since XPath [XPATH] fails to address some requirements of the ID-SIS-CB service, the following extensions MUST be implemented.

1. Since relational operators (<, >, <=, >=) of [XPATH] perform numeric comparison rather than string comparison, the following new functions are introduced:

   `cb:le(a,b)::` String argument a is string-wise less than or equal to the string argument b.

   `cb:ge(a,b)::` String argument a is string-wise greater than or equal to the string argument b

2. function `cb:soundslike(a,b)` is introduced for sounds-like comparison. All implementations MUST implement `soundslike()` as a syntactical construct, but a ID-SIS-CB implementation MAY implement `soundslike()` as a simple case- and accent-insensitive equality comparison.

3. function `cb:stripaccents(a)` is introduced to make accent-insensitive matching easier. The accents are stripped in the sense of the ISO Latin1 alphabet.

4. function `cb:lowercase(a)` is introduced to make case-insensitive matching easier. Letters are lower-case, including the accented letters, in the sense of the ISO Latin1 alphabet.

5. function `cb:uppercase(a)` is introduced to make case-insensitive matching easier. Letters are upper-case, including the accented letters, in the sense of the ISO Latin1 alphabet.

6. function `cb:avail(node)` is introduced to test whether a given node (an element) is available for the requester. Availability means that if an attempt is made to return the type in result, it will succeed in the sense of *type eligibility criteria* of Section 3.3.

7. function `cb:fav()` is introduced to express "*Favored*" value-added criteria. The implied argument is the current contact card under consideration. This returns `true` for Contact Cards that are on the principal's favorites list. Typically, this list is manually maintained by the principal, but need not be.

8. function `cb:mru()` is introduced to express "*Most Recently Used*" value-added criteria. The implied argument is the current contact card under consideration. This returns `true` if the card is in an automatically-computed set of Contact Cards that reflect recent usage. The number of the cards maintained on the *mru list* MAY be limited by the Contact Book Provider's policy and is often a parameter that the Principal can configure.

9. function `cb:mfu()` is introduced to express "*Most Frequently Used*" value-added criteria. The implied argument is the current contact card under consideration. This returns `true` if the card is in an automatically-computed set of Contact Cards that reflect frequent usage. The number of the cards maintained on the *mfu list* or the time frame used for *mfu* computation MAY be limited by the Contact Book Provider's policy and MAY be a parameter that the Principal can configure.

Value-Added Matching and Sorting criteria are criteria that utilize implementation-specific metadata, algorithms, or other out-of-band means to return a more relevant result set. They include:

1. "*most recently used*" criteria (within a time frame)

2. "*most frequently used*" criteria (within a time frame)

3. "*favorites*" criteria (usually explicitly indicated by the Principal)

An implementation MAY return less than the requested number of cards if a parameter governing value-added criteria (e.g., `mfu` time frame, `mru` list length) makes it difficult to return the requested number of cards. CB MUST NOT return more than the requested number of cards (as specified by `count` XML attribute of the `QueryItem`).

An implementation that does not support the requested value-added criteria MAY ignore the criteria or use another similar criteria in its place (e.g., return `mru` for `mfu` if the latter is not supported). The implementation MUST indicate which criteria it actually used. This allows the WSC to detect if the criteria was honored.

Value-added criteria MUST be freely combinable with other criteria.

3.2.2. Restrictions on XPath Expressions

In an effort to simplify implementations, the following restrictions are placed on [XPATH] expressions that a WSC may send and a WSP must support.

1. `Query` expressions are composed of
    a. one location path expression and
    b. zero or one predicates.
2. `Location` path expressions MUST be either
    a. simple `location` path or
    b. a set of simple `location` paths joined with `|` operators in a single parenthesized expression

   and each simple `location` path MUST start by `/cdm:vCard`.
3. The predicate MAY reference `location` paths that are not anchored at the document root.
4. Only abbreviated [XPATH], as described in [XPATH] Section 2.5, MUST be used.
5. In addition to functions listed in Section 3.2.1, only the following functions MUST be used:

```
concat()
starts-with()
contains()
substring-before()
substring-after()
substring()
string-length()
normalize-space()
translate()
not()
lang()
```

6. The predicate MUST NOT contain more than 5 levels of nested subexpressions.

Only `locations` paths, subject to qualification by or omission of namespace identifiers as permitted by XML specifications, that an implementation MUST support are:

```
/cdm:vCard
/cdm:vCard/cdm:VERSION
/cdm:vCard/cdm:CARDID
/cdm:vCard/cdm:DISTRIBUTIONLIST
/cdm:vCard/cdm:SELF
/cdm:vCard/cdm:FAVORITE
/cdm:vCard/cdm:FN
/cdm:vCard/cdm:N
/cdm:vCard/cdm:N/cdm:FAMILY
/cdm:vCard/cdm:N/cdm:GIVEN
/cdm:vCard/cdm:N/cdm:MIDDLE
/cdm:vCard/cdm:N/cdm:PREFIX
/cdm:vCard/cdm:N/cdm:SUFFIX
/cdm:vCard/cdm:LISTMEMBER
/cdm:vCard/cdm:NICKNAME
/cdm:vCard/cdm:PHOTO
/cdm:vCard/cdm:PHOTO/cdm:EXTVAL
/cdm:vCard/cdm:BDAY
/cdm:vCard/cdm:ADR
/cdm:vCard/cdm:ADR/cdm:HOME
/cdm:vCard/cdm:ADR/cdm:WORK
/cdm:vCard/cdm:ADR/cdm:POSTAL
/cdm:vCard/cdm:ADR/cdm:PARCEL
/cdm:vCard/cdm:ADR/cdm:DOM
/cdm:vCard/cdm:ADR/cdm:INTL
/cdm:vCard/cdm:ADR/cdm:PREF
/cdm:vCard/cdm:ADR/cdm:POBOX
/cdm:vCard/cdm:ADR/cdm:EXTADR
/cdm:vCard/cdm:ADR/cdm:STREET
/cdm:vCard/cdm:ADR/cdm:LOCALITY
/cdm:vCard/cdm:ADR/cdm:REGION
/cdm:vCard/cdm:ADR/cdm:PCODE
/cdm:vCard/cdm:ADR/cdm:CTRY
/cdm:vCard/cdm:LABEL
/cdm:vCard/cdm:LABEL/cdm:HOME
/cdm:vCard/cdm:LABEL/cdm:WORK
/cdm:vCard/cdm:LABEL/cdm:POSTAL
/cdm:vCard/cdm:LABEL/cdm:PARCEL
/cdm:vCard/cdm:LABEL/cdm:DOM
/cdm:vCard/cdm:LABEL/cdm:INTL
/cdm:vCard/cdm:LABEL/cdm:PREF
/cdm:vCard/cdm:LABEL/cdm:LINE
/cdm:vCard/cdm:TEL
/cdm:vCard/cdm:TEL/cdm:HOME
/cdm:vCard/cdm:TEL/cdm:WORK
/cdm:vCard/cdm:TEL/cdm:VOICE
/cdm:vCard/cdm:TEL/cdm:FAX
/cdm:vCard/cdm:TEL/cdm:PAGER
/cdm:vCard/cdm:TEL/cdm:MSG
/cdm:vCard/cdm:TEL/cdm:CELL
/cdm:vCard/cdm:TEL/cdm:VIDEO
/cdm:vCard/cdm:TEL/cdm:BBS
/cdm:vCard/cdm:TEL/cdm:MODEM
/cdm:vCard/cdm:TEL/cdm:ISDN
/cdm:vCard/cdm:TEL/cdm:PCS
/cdm:vCard/cdm:TEL/cdm:PREF
/cdm:vCard/cdm:TEL/cdm:NUMBER
/cdm:vCard/cdm:EMAIL
/cdm:vCard/cdm:EMAIL/cdm:HOME
/cdm:vCard/cdm:EMAIL/cdm:WORK
/cdm:vCard/cdm:EMAIL/cdm:INTERNET
/cdm:vCard/cdm:EMAIL/cdm:PREF
/cdm:vCard/cdm:EMAIL/cdm:X400
/cdm:vCard/cdm:EMAIL/cdm:USERID
```

```
/cdm:vCard/cdm:JABBERID
/cdm:vCard/cdm:MAILER
/cdm:vCard/cdm:TZ
/cdm:vCard/cdm:GEO
/cdm:vCard/cdm:GEO/cdm:LAT
/cdm:vCard/cdm:GEO/cdm:LON
/cdm:vCard/cdm:TITLE
/cdm:vCard/cdm:ROLE
/cdm:vCard/cdm:LOGO
/cdm:vCard/cdm:LOGO/cdm:EXTVAL
/cdm:vCard/cdm:AGENT
/cdm:vCard/cdm:AGENT/cdm:EXTVAL
/cdm:vCard/cdm:ORG
/cdm:vCard/cdm:ORG/cdm:ORGNAME
/cdm:vCard/cdm:ORG/cdm:ORGUNIT
/cdm:vCard/cdm:CATEGORIES
/cdm:vCard/cdm:CATEGORIES/cdm:KEYWORD
/cdm:vCard/cdm:NOTE
/cdm:vCard/cdm:PRODID
/cdm:vCard/cdm:REV
/cdm:vCard/cdm:SORT-STRING
/cdm:vCard/cdm:SOUND
/cdm:vCard/cdm:SOUND/cdm:PHONETIC
/cdm:vCard/cdm:SOUND/cdm:EXTVAL
/cdm:vCard/cdm:UID
/cdm:vCard/cdm:URL
/cdm:vCard/cdm:CLASS
/cdm:vCard/cdm:CLASS/cdm:PUBLIC
/cdm:vCard/cdm:CLASS/cdm:PRIVATE
/cdm:vCard/cdm:CLASS/cdm:CONFIDENTIAL
/cdm:vCard/cdm:KEY
/cdm:vCard/cdm:KEY/cdm:TYPE
/cdm:vCard/cdm:KEY/cdm:CRED
/cdm:vCard/cdm:DESC
/cdm:vCard/cdm:PHYSICALACCESS
```

In predicates following additional (relative) `location` paths MUST be supported

```
@cb:group
WORK
POSTAL
PARCEL
DOM
INTL
PREF
VOICE
FAX
PAGER
MSG
CELL
VIDEO
BBS
MODEM
ISDN
PCS
INTERNET
X400
CARDID
DISTRIBUTIONLIST
SELF
FAVORITE
LISTMEMBER
```

However, an implementation MAY support additional paths in the interest of supporting vCard extensions.

3.3. Query Processing

For each *matching* contact card, a `cdm:vCard` element is constructed and populated with data in the requested format, i.e., the ID-SIS-CB service will perform translation from the internal format (generic XML vCard) to the requested format.

This vCard will only have those types that simultaneously satisfy the following *type eligibility criteria*:

1. requested in the [XPATH] expression supplied in the `QueryItem` element,
2. has a nonempty value,
3. is permissible for WSC-given CB provider's policy, and
4. is permissible for WSC-given preferences set by the Principal.

If a card matches, but none of its types match, an empty place-holder card MUST be returned.

Finally, before returning the contact cards to WSC, the ID-SIS-CB service SHOULD sort them according to criteria requested by the WSC.

3.4. Query Examples

3.4.1. Picking TEL and EMAIL from Joe Smith's Contact Card

```
( /cdm:vCard/cdm:TEL
| /cdm:vCard/cdm:EMAIL )
[  /cdm:vCard/cdm:N/cdm:FAMILY="Smith"
and contains(/cdm:vCard/cdm:FN, "Joe") ]
```

N.B. Since this [XPATH] expression was rather long, it was wrapped on several lines and indented for clarity. In general, [XPATH] expressions are not sensitive to presence or absence of whitespace.

The [XPATH] is formed of two parts:

1. the union of the elements that are picked to be included in the result is expressed using the parenthesized part. Here `TEL` and `EMAIL` attributes are desired.
2. the criteria that select appropriate contact card(s) is expressed using the predicate in the square brackets. The predicate can be arbitrarily complex and can use comparisons as well as functions defined by [XPATH]. For example, this predicate would also match Joel's contact card as long as her last name was Smith because the `contains()` function will accept any substring match. In general, the predicate can match any number of contact cards.

A common reading of this [XPATH] might be "Give me TEL and EMAIL for all cards whose family is 'Smith' and formatted name has 'Joe' somewhere in it."

Note that although only `TEL` and `EMAIL` were requested, they will still be wrapped in a `cdm:vCard` container when they are returned.

3.4.2. Returning Entire Cards

```
/cdm:vCard[/cdm:vCard/cdm:N/cdm:FAMILY="Smith"]
```

This query returns all contact cards whose `FAMILY` is "Smith." The entire content of these cards is returned, except that permissions and policy may filter out some attributes.

3.4.3. Returning All Cards in the Contact Book

```
/cdm:vCard
```

Since no predicate is used to limit which cards to return, all cards are returned. However, permissions or policy may prevent some cards from being returned.

It is RECOMMENDED that the ID-SIS-CB Provider maintain policies that stop wholesale harvesting of Contact Books and limits a WSC's access to a legitimate "*need to know*" basis. It is RECOMMENDED that the Principal controls access to her Contact Book such that only select WSCs that have a legitimate "*need to know*" will be able to get all Contact Cards.

3.4.4. Returning Some Attributes from All Cards

```
( /cdm:vCard/cdm:N/cdm:GIVEN | /cdm:vCard/cdm:TEL )
```

Returns first name and phone number from all cards of the contact book. All attributes extracted from one card are wrapped in the same `cdm:vCard` container in the result, thus the data stays together.

3.4.5. Latin 1 Case-Insensitive Matching

```
/cdm:vCard[contains(translate(/cdm:vCard/cdm:FN,
                      "AÁÀÂÄÃÅBc...",
                      "aáàâäãåbc...")
              , "joão")]
```

As can be seen, the author of a query expression must be aware of how to construct a case-insensitive match for a given language. While this is cumbersome and prone to implementation error, this is the way recommended in [XPATH]'s Section 4.2.

The `translate()` function can also be used to implement accent-insensitive matching.

3.4.6. Latin 1 Accent- and Case-Insensitive Matching

```
/cdm:vCard[contains(
              lowercase(
                     stripaccents(/cdm:vCard/cdm:FN)
                    )
              , "joao")
             ]
```

This query expression is much simpler than the one in the previous example. Thanks to use of built-in functions `lowercase()` and `stripaccents()`, the cumbersome translation table may be omitted. This solution only works for the ISO Latin 1 alphabet.

3.4.7. Free Text Search

```
/cdm:vCard[contains(/cdm:vCard, "Smith")]
```

This query returns all Contact Cards that have the string "Smith" anywhere in them (i.e., in any attribute). This takes advantage of the XPath feature described in [XPATH]'s Section 5.2 that causes the *string-value* of a container to be a concatenation of all the contained text nodes to any depth.

3.4.8. Picking by Type

```
( /cdm:vCard/cdm:TEL
| /cdm:vCard/cdm:ADR
| /cdm:vCard/cdm:PHYSICALACCESS )
[  /cdm:vCard/cdm:N/cdm:FAMILY="Smith"
and cdm:PREF ]
```

This query is keyed on the vCard *type* parameter. It first selects all "Smith" family cards and then returns their preferred contact information. Note that the `cdm:PREF` clause applies to the current context node of the [XPATH] expression, i.e., to the nodes selected by the parenthesized expressions: `TEL`, `ADR`, and `PHYSICALACCESS`.

3.4.9. Picking by Group

```
( /cdm:vCard/cdm:TEL
| /cdm:vCard/cdm:ADR
| /cdm:vCard/cdm:PHYSICALACCESS )
[  /cdm:vCard/cdm:N/cdm:FAMILY="Smith"
and @cb:group="home" ]
```

This query is keyed on vCard's *group*. It first selects all "Smith" family cards and then returns their home contact information. Note that the `@cb:group` clause applies to the current context node of the [XPATH] expression, i.e., to the nodes selected by the parenthesized expressions: `TEL`, `ADR`, and `PHYSICALACCESS`.

3.4.10. Picking Types in Same Group

```
( /cdm:vCard/cdm:ADR
| /cdm:vCard/cdm:PHYSICALACCESS )
[  /cdm:vCard/cdm:N/cdm:FAMILY="Smith"
and @cb:group=/cdm:vCard/cdm:PHYSICALACCESS/@cb:group ]
```

The novelty of this query is that the group need not be known ahead of time: we simply want the physical address and the access information that goes with it.

3.4.11. Picking Types of a Given Language

```
( /cdm:vCard/cdm:FN
| /cdm:vCard/cdm:LABEL )
[  /cdm:vCard/cdm:N/cdm:FAMILY="Smith"
and lang("jp") ]
```

Here we use the [XPATH]`lang()` function to determine the context node's (i.e., `FN` or `LABEL`) language. The language must have been specified by the `xml:lang` XML attribute in the element or in any parent element of the element.

3.4.12. Querying `CATEGORIES`

```
/cdm:vCard/cdm:CATEGORIES/cdm:KEYWORD
```

This query returns all categories in use by any contact card in the Principal's contact book.

3.4.13. Finding Distribution Lists

```
/cdm:vCard [cdm:DISTRIBUTIONLIST]
```

Since distribution list place holder cards are marked with the `DISTRIBUTIONLIST` indicator, we formulate a query by requesting all vCards such that they have a `DISTRIBUTIONLIST` child element.

> N.B. There is nothing to stop distribution list place holder cards from being returned in normal queries as well. They will simply appear mixed with other cards.

3.4.14. Finding Members of a Distribution List

```
/cdm:vCard [cdm:LISTMEMBER=cardid]
```

This query requests all vCards that are members of a distribution list represented by the place holder card whose `CARDID` is `cardid`. This query effectively expands a distribution list.

3.5. Sorting Contact Cards

The [LibertyDST20] specifies a `Query` element that potentially contains several `QueryItem` elements which in turn may contain a `Sort` element, but leaves the actual `sort-by` expression undefined. ID-SIS-CB defines the `SortType` as follows:

```
<xs:complexType name="SortType">
   <xs:sequence>
      <xs:element ref="By" maxOccurs="unbounded"/>
   </xs:sequence>
   <!-- connects to DST framework -->
</xs:complexType>
<xs:element name="By" type="cb:ByType"/>
<xs:complexType name="ByType" mixed="true">
   <xs:attribute ref="sortAlg"/>
   <xs:attribute ref="sortWeight"/>
</xs:complexType>
<xs:attribute name="sortAlg">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="asc"/>
         <xs:enumeration value="desc"/>
         <!-- Ascending order, e.g., alphabetic order -->
         <!-- Descending order, e.g., inverse alphabet -->
      </xs:restriction>
   </xs:simpleType>
</xs:attribute>
<xs:attribute name="sortWeight">
   <xs:simpleType>
      <xs:restriction base="xs:integer"/>
   </xs:simpleType>
</xs:attribute>
```

If a `Sort` container is not specified, an implementation MAY return the Contact Cards in any order.

If a `Sort` container is specified, an implementation SHOULD honor it as best as it can, according to the `By` elements that are available. It is RECOMMENDED that at least one `By` element (the one with `sortWeight="1"`) is honored.

An implementation MUST support sorting by the following elements.

```
/cdm:vCard/cdm:N/cdm:FAMILY
/cdm:vCard/cdm:N/cdm:GIVEN
```

An implementation MAY partially ignore or reorder other sort criteria or substitute criteria by other similar sorting criteria.

If `Sort` is specified, then at least one `By` MUST be present. If more than one is present, all of them MUST have the `sortWeight` XML attribute specified and exactly one of them MUST have `sortWeight` set to one (1). Two `By` elements MUST NOT have equal `sortWeight`.

Sorting is performed by processing the `By` elements in ascending order of `sortWeight`. Lowest `sortWeight` specifies most significant criterion for sorting. For each `By`, the sorting is performed in the order specified by the `sortAlg` XML attribute, if present, or in ascending order, otherwise.

When an [XPATH] expression in `By` matches several types in one card (e.g., the type is multivalued), the type with the value that causes the earliest possible placement in the sort SHOULD be used.

If the predicate in `By` invokes `fav()`, `mfu()`, or `mru()` function, then sorting by the specified value-added criteria SHOULD be invoked.

3.6. Modifying Contact Cards and Attributes

The modify operation is comprised of

1. selection of card(s) to modify,
2. optional selection of attribute(s) to modify, and
3. new vCard data, which could be partial.

Different modalities of operation are specified as follows

Table 13. Flexible Modify Operation

Semantic	***Cards***	***Attributes***	***vCard data***	***Notes***
Add new cards	-	-	1-N entire vCards	Replace empty set with new cards
Replace cards	1-N	-	1-N entire vCards	Replace existing cards with new ones
Delete cards	1-N	-	-	Replace existing cards with emptiness
Add attributes	1	0	Partial vCard data	Add attributes from data
Replace attributes	1	1-N	Partial vCard data	Replace attributes with new data
Delete attributes	1	1-N	-	Replace attributes with emptiness

> As can be seen, there are two major modes of operation: card-level operations and attribute-level operations. The two modes are distinguished by the type of the [XPATH] expression. If the location path component of the expression is `/cdm:vCard`, then card-level operation is meant. Longer or more complex location paths mean attribute-level operation.

For card-level operations, the [XPATH] location path component MUST be `/cdm:vCard` and vCard data MUST represent complete vCards.

For attribute-level operations, the [XPATH] predicate MUST match exactly one card or elements of exactly one card and the location path MUST specify a descendant of `/cdm:vCard`, even if it does not pick any actual attributes. The `/cdm:vCard/cdm:ADD` location path MUST be used to request pure "*add attributes*" functionality. In attribute-level operations, the vCard data MAY be partial (and typically is). If the [XPATH] predicate clause picks some specific attributes for replacement or deletion, then only those are replaced or deleted and the selected card is considered to be the card that contains those attributes.

Section 5.1 of [LibertyDST20] specifies that the `Modification` container may contain a `NewData` container whose content can be any XML data. However, ID-SIS-CB further restricts this such that `NewData` MUST either be omitted or contain one or more `cb:Card` element(s), where each element represents either an entire vCard or a partial vCard according to the modality of the modification.

The `cb:format` XML attribute MUST be supplied for each `cb:Card` container. It MUST NOT be supplied in a `Select` element accompanying the `Modification`.

3.6.1. Add New Cards

Add new cards functionality is invoked by specifying an empty `Select` element in the `Modification` and specifying, in `NewData`, `cb:Card` elements representing complete vCards.

If an `overrideAllowed` XML attribute (see [LibertyDST20] Sections 5.1 and 5.3) is specified in the `Modification`, an `add` is performed even if cards identical to the ones being added already exist.

Example

```
<cb:Modify>
  <cb:Modification>
    <cb:Select/>
    <cb:NewData>
      <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
        <cb:CharData>vCard:BEGIN
FN:Sampo Kellomaki
N:Kellomaki;Sampo
vCard:END
        </cb:CharData>
      </cb:Card>
      <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
        <cb:CharData>vCard:BEGIN
FN:Jukka Kainulainen
N:Kainulainen;Jukka
vCard:END
        </cb:CharData>
      </cb:Card>
    </cb:NewData>
  </cb:Modification>
</cb:Modify>
```

Note the empty `Select` element in the `Modification`. This does not select any card, thus all vCard data is meant to add new vCards.

The ID-SIS-CB Provider MAY refuse an `add` operation that would create duplicate cards unless an `overrideAllowed` XML attribute with a `true` value is specified. The ID-SIS-CB Provider MAY accept an `add` operation that would create duplicate cards.

3.6.2. Replace Cards

Replace cards functionality is invoked by specifying `Select` in the `Modification` and specifying in `NewData` `cb:Card` elements representing complete vCards. The operation is processed by first deleting the cards that match the `Select` and then adding the cards specified in the `NewData`. The `Select` MUST contain [XPATH] whose location path component is `/cdm:vCard`.

If an `overrideAllowed` XML attribute with a `true` value is specified in the `Modification`, the new cards are added, even if cards identical to the ones being added already exist, after deleting the cards that match the `Select`. If no `overrideAllowed` is specified or it has a `false` value, the ID-SIS-CB Provider MAY refuse the `modify` or it MAY accept it.

Example

```
<cb:Modify>
  <cb:Modification cb:overrideAllowed="1">
    <cb:Select>/cdm:vCard
      [/cdm:vCard/cdm:N/cdm:GIVEN="Sampo"]
    </cb:Select>
    <cb:NewData>
      <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
        <cb:CharData>vCard:BEGIN
FN:Tapani Kellomaki
N:Kellomaki;Tapani
vCard:END
        </cb:CharData>
      </cb:Card>
      <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
        <cb:CharData>vCard:BEGIN
FN:Jukka Kainulainen
N:Kainulainen;Jukka
vCard:END
        <cb:CharData>
      </cb:Card>
    </cb:NewData>
  </cb:Modification>
</cb:Modify>
```

Assuming that the example of Section 3.6.1 has already happened, this example will first delete Sampo's card and then add two new cards, one of them being a duplicate of Jukka's card.

3.6.3. Delete Cards

Delete cards functionality is invoked by specifying `Select` in the `Modification` and omitting `NewData`. The `Select` MUST contain [XPATH] whose location path component is `/cdm:vCard`.

Example

```
<cb:Modify>
  <cb:Modification overrideAllowed="1">
    <cb:Select>/cdm:vCard
      [/cdm:vCard/cdm:N/cdm:GIVEN="Jukka"]
    </cb:Select>
  </cb:Modification>
</cb:Modify>
```

Assuming that the examples of Section 3.6.1 and Section 3.6.2 have already happened, this example deletes two cards out of the three present. The deleted cards are the ones belonging to Jukka (i.e., both duplicated card match). Thus, the only card left after the operation is "`Tapani`."

Note that [LibertyDST20] Section 5.3 specifies a processing rule that requires XML-attribute `overrideAllowed="1"` to be specified.

3.6.4. Add Attributes to a Card

Add attributes functionality is invoked by specifying `Select` in the `Modification`, specifying a location path that refers to one of the attributes to be added in the `Select`, and providing the new attribute data `cb:Card` in a `NewData` element. In this case, the `cb:Card` MAY (and often will) contain a partial vCard. There MUST be exactly one `cb:Card` element. The [XPATH] predicate clause MUST match exactly one card.

Example

```
<cb:Modify>
  <cb:Modification>
    <cb:Select>/cdm:vCard/cdm:ADD
      [/cdm:vCard/cdm:N/cdm:GIVEN="Tapani"]
```

```
    </cb:Select>
    <cb:NewData>
      <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
        <cb:CharData>vCard:BEGIN
NOTE:One new attribute
NOTE:Another new attribute
TITLE;language=fi:Jestelmrkkitehti
vCard:END
      </cb:CharData>
    </cb:Card>
  </cb:NewData>
 </cb:Modification>
</cb:Modify>
```

This example first selects Tapani's card and then adds three new attributes to it. The new attributes are added even if identical attributes existed previously.

If there had been two cards for Tapani, the operation would have failed.

3.6.5. Replace Attributes in a Card

Replace attributes functionality is invoked by specifying `Select` in the `Modification`, specifying a location path that picks the attributes to be replaced, and providing the new attribute data `cb:Card` in a `NewData` element. In this case, the `cb:Card` MAY (and often will) contain a partial vCard. There MUST be exactly one `cb:Card` element. The [XPATH] predicate clause MUST match exactly one card.

Example

```
<cb:Modify>
 <cb:Modification>
  <cb:Select>/cdm:vCard/cdm:NOTE
    [/cdm:vCard/cdm:N/cdm:GIVEN="Tapani"]
  </cb:Select>
  <cb:NewData>
    <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
      <cb:CharData>vCard:BEGIN
NOTE:Replacement note
TITLE;language=en:Systems Architect
vCard:END
      </cb:CharData>
    </cb:Card>
  </cb:NewData>
 </cb:Modification>
</cb:Modify>
```

This example first selects Tapani's card, deletes from it all `NOTE` attributes, and then adds to it a `NOTE` and a second `TITLE` attribute. The new attributes are added even if identical attributes existed previously.

N.B. We do not need to specify the `overrideAllowed` XML-attribute here because all previous `NOTE`s are removed before inserting the new one and because the `TITLE` is permitted to be multivalued and the new value is different.

Specifying `overrideAllowed` is not meaningful here since, if override is intended, the [XPATH] should pick the attributes that are to be overridden.

3.6.6. Delete Attributes from a Card

Delete attributes functionality is invoked by specifying `Select` in the `Modification`, specifying a location path that picks the attributes to be deleted, and omitting the `NewData` element. The [XPATH] predicate clause MUST match exactly one card.

Example

```
<cb:Modify>
  <cb:Modification overrideAllowed="1">
    <cb:Select> ( /cdm:vCard/cdm:TITLE
             | /cdm:vCard/cdm:NOTE )
             [/cdm:vCard/cdm:N/cdm:GIVEN="Tapani"]
    </cb:Select>
  </cb:Modification>
</cb:Modify>
```

This example first selects Tapani's card and then deletes from it all `TITLE` and `NOTE` attributes. Note that [LibertyDST20] Section 5.3 specifies a processing rule that requires XML-attribute `overrideAllowed="1"` to be specified.

3.6.7. Delete Specific Attributes from a Card

Here we illustrate a predicate that picks only some attributes

Example

```
<cb:Modify>
  <cb:Modification overrideAllowed="1">
    <cb:Select> /cdm:vCard/*
             [   /cdm:vCard/cdm:N/cdm:GIVEN="Tapani"
              and lang("fi")
              and @cb:group = /cdm:vCard/cb:TITLE/@cb:group]
    </cb:Select>
  </cb:Modification>
</cb:Modify>
```

This deletes from Tapani's card (assuming there is only one Tapani) all attributes that are in Finnish and that belong to a group that contains a `TITLE`. Note how location path is used to select all attributes, but how the predicate is used to further refine that selection.

3.7. Reporting Usage of a Contact Card

An ID-SIS-CB provider MUST support `ReportUsage` operation for reporting when a contact has been used. The usage is considered for purposes of most frequently used (`MFU`) and most recently used (`MRU`) value-added criteria.

A WSC SHOULD report use of a contact card to provide a service when the usage of the card was triggered by interaction of the Principal. Mere querying of a card (e.g., to display choices to the end user) does not constitute usage. A WSC SHOULD NOT report usage based on noninteractive processing or standing orders. The ID-SIS-CB provider and WSC MAY establish in business agreements what types of usage to report, and the ID-SIS-CB Provider MAY have policies or restrictions that cause it to ignore some usage reports. The exact effect of the usage reports to `MRU` and `MFU` functionality are implementation-dependent.

Usage is reported using `ReportUsage` messages that are reciprocated with `ReportUsageResponses` whose schemata follow:

```

  <xs:element name="ReportUsage" type="cb:ReportUsageType"/>
  <xs:complexType name="ReportUsageType">
     <xs:sequence>
        <xs:group ref="ResourceIDGroup" minOccurs="0"/>
        <xs:element ref="cdm:CARDID"/>
        <xs:element ref="cb:UsageType" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
  <xs:element name="UsageType" type="UsageTypeType"/>
  <xs:complexType name="UsageTypeType">
     <xs:simpleContent>
```

```
         <xs:extension base="cb:DSTURI">
            <xs:attribute name="success" type="xs:boolean" default="1"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:element name="ReportUsageResponse" type="ReportUsageResponseType"/>
   <xs:complexType name="ReportUsageResponseType">
      <xs:sequence>
         <xs:element ref="Status"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>
```

The `CARDID` identifies which card was used. Only one card may be reported per `ReportUsage` message.

The optional multivalued `UsageType` element allows a usage report to specify how the contact card was used. The possible values held in this element are URIs that act as enumerators designating the uses. Specification of these enumerators is outside the scope of this document, but presumably some business agreement between the WSC and the ID-SIS-CB provider will specify them.

The `UsageType` element MAY have a `success` boolean XML attribute specifying whether the attempted usage was successful.

3.8. Manipulating Distribution lists

No special protocol support is needed for distribution lists. The `DISTRIBUTIONLIST` and `LISTMEMBER` elements of the conceptual data model support modification and querying of distribution lists in a generic way using simple `Query` and `Modify` operations.

The query expression matches Contact Cards and Distribution Lists meeting the criteria. When a Contact Card matches the criteria, it is added to the result set. However, if a distribution list should be expanded, the WSC must make a separate query to expand it, see Section 3.4.14.

Creation and deletion of distribution lists are done using the same operations as creation and deletion of regular contact cards.

To change distribution membership, perform a modification to the `LISTMEMBER` elements of the card that is joining or leaving the list.

4. Processing Rules and Other Considerations

4.1. Query is not Required to Report Same Data to Repeated Queries

An ID-SIS-CB instance is NOT REQUIRED to report the same results to two instances of the same query. An ID-SIS-CB instance SHOULD report the same results to the same query made by the same client, unless an update (`Modify` or out-of-band) has occurred in the interim. An ID-SIS-CB instance MAY use the Interaction Service protocol [LibertyInteract11] or out-of-band means to determine the data to return.

Data to be returned in response to a query is determined by the ID-SIS-CB provider, guided by its policies, the permissions the Principal has set, and the interaction with the Principal. Clients should use the data based on the data's semantic meaning as specified here and further qualified by the `acc` (Attribute Collection Context) XML attributes [LibertyDST20] that may be present in the query response. A client SHOULD NOT attempt to use ID-SIS-CB as a transparent data store as there may be multiple updates, permission, and policy reasons that impede the transparency.

4.2. Support of Multiple Modification Not Required

The Modify operation functions, as described in [LibertyDST20] with the additional relaxation of a minimally-compliant ID-SIS-CB implementation, MAY refuse a `Modify` request with multiple `Modification` elements provided all processing rules specified in [LibertyDST20] are followed regarding failure to support multiple `Modification` elements. Thus a minimally-compliant implementation is not required to support multiple `Modification` elements.

Implementations SHOULD support multiple `Modification` elements when feasible. If an implementation supports multiple `Modification` elements, it MAY register the discovery option keyword `urn:liberty:dst:multipleModification`.

A minimally-compliant ID-SIS-CB implementation MUST support multiple `QueryItem` elements as specified in [LibertyDST20].

4.3. Simulation (dry-run) Is Required

The simulation method using a `ProcessingContext` header with value "`urn:liberty:sb:2003-08:ProcessingContext:Simulate`," as specified in [LibertySOAPBinding12], MUST be supported. If the simulated operation succeeds, a similar actual operation SHOULD have a high probability of succeeding within next 30 minutes.

This feature allows a WSC to test whether a modification is plausible prior to invoking the user interface to query data from the Principal, thus avoiding the Principal having to supply nonactionable data unnecessarily.

4.4. There Can Only Be One SELF Card

A contact book can only have one card marked with `cdm:SELF`. The purpose of the self marker is to enable the Principal to identify which card should be synchronized with her Personal Profile service.

4.5. Multiple Instances of Same Person

An ID-SIS-CB MUST allow multiple cards describing the same person to exist. However, the ID-SIS-CB service does not have to be able to understand that the cards indeed represent the same person.

If multiple contact cards have identical content, the ID-SIS-CB service MAY retain only one copy.

4.6. Error Messages in Case an Attribute Does Not Exist

If an attribute is requested and would be available except for permissions, the ID-SIS-CB Provider MUST generate an error response with code `cb:ActionNotAuthorized`.

4.7. Multiple Modification Is Restricted to One Card

An implementation that supports `modify` operations MUST support multiple `Modification` elements and MUST NOT allow partial modifies. However, in interest of simplification, multiple `Modification` elements within the same `Modify` element MUST contain `Select` elements that refer to the same contact card.

When multiple `Modification` elements are present, they MUST be processed in the order that they appear in the XML document. For example, if one `Modification` adds an attribute, but a later `Modification` deletes it, the net effect is as if the attribute had never been added.

4.8. Simplifications to Distribution Lists

Distribution lists MUST NOT be directly or indirectly recursive.

An implementation MUST be able to cope with a distribution list place holder card being a member of another distribution list, but MAY ignore such instances. A WSC SHOULD NOT depend on being able to query or manipulate distribution lists as members of other distribution lists.

4.9. Use of Usage Directives

Typical use of ID-SIS-CB involves WSC querying the Contact Book for a contact card for a particular use. The ID-SIS-CB provider MAY invoke the interaction service [LibertyInteract11] to ask the Principal to choose from many cards. The Principal will need to know the purpose of the request to make an intelligent choice. Knowing the identity of the WSC may not always be sufficient to understand probable use of the data.

When querying, a WSC MUST include usage directives that indicate the reason for the request and the type of use the data will be put. If the ID-SIS-CB provider invokes the service [LibertyInteract11] service, it SHOULD display this information, usually after localization, to the Principal.

The actual format of the usage directives MUST be agreed out-of-band (e.g., in business agreement between WSC and ID-SIS-CB provider).

5. Mapping MIME-DIR-Based vCard Formats

The [vCard21] mapping is indicated by specifying the `cb:format` XML attribute with value "`urn:liberty:cb:format:v2.1`." In this case, the content MUST conform to [vCard21].

The vCard 3.0 ([RFC2426]) mapping is indicated by specifying the `cb:format` XML attribute with value "`urn:liberty:cb:format:RFC2426`." In this case, the content MUST conform to [RFC2426].

5.1. From Generic vCard to vCard 2.1

1. Map `N` by linearizing the values of elements `FAMILY`, `GIVEN`, `OTHER`, `PREFIX`, and `SUFFIX` in order, separating values of different attributes by semicolons. If any of the elements are missing, an empty value shall appear in its place.

2. Map `ADR` by linearizing the values of elements `POBOX`, `EXTADR`, `STREET`, `LOCALITY`, `REGION`, `CTRY`, and `CTRY` in order, separating values of different attributes by semicolons. If any of the elements are missing, an empty value shall appear in its place.

3. Map `ORG` by linearizing the values of elements `ORGNAME` and `ORGUNIT` in order, separating values of different attributes by semicolons. If any of the elements are missing, an empty value shall appear in its place.

4. Map all other attributes by generating type with name and value of the corresponding element.

5. If an element has a `cb:group` XML attribute, the value of this attribute, concatenated with a period ("`.`"), MUST be prefixed to the type name.

6. Map `xml:lang` XML attribute to `language` parameter.

7. Map all elements representing types (e.g., `HOME`, `WORK`) to corresponding [vCard21] parameters.

5.2. From vCard 2.1 to Generic vCard

1. Map `N` vCard type by generating an `N` container with contained elements `FAMILY`, `GIVEN`, `OTHER`, `PREFIX`, and `SUFFIX`, using semicolon-separated subfields of `N` type. An empty subfield need not generate any element.

2. Map `ADR` vCard type by generating an `ADR` container with contained elements `POBOX`, `EXTADR`, `STREET`, `LOCALITY`, `REGION`, `CTRY`, and `CTRY`, using semicolon-separated subfields of `ADR` type. An empty subfield need not generate any element.

3. Map `ORG` vCard type by generating an `ORG` container with contained elements `ORGNAME` and `ORGUNIT`, using semicolon separated subfields of `ORG` type. An empty subfield need not generate any element.

4. Map extended types (i.e., the ones that are not specified in conceptual data model schema) by generating, for each type, an element with name and value of the corresponding type. The name of the element MUST be upper-case regardless of how it appeared as a type name. The elements that correspond to extended types are generated as child elements of the `cdm:Extension` container.

5. Map all other types by generating, for each type, an element with name and value of the corresponding type. The name of the element MUST be upper-case regardless of how it appeared as a type name.

6. If type name has a group prefix, the corresponding element name appears without the group prefix, but has a `cb:group` XML attribute whose value is the group prefix with period ("`.`") removed.

7. If type has a `language` parameter, it is mapped to `xml:lang` XML attribute on the element corresponding to the type.

8. If type has other parameters, they are mapped to corresponding elements (e.g., `HOME`, `WORK`) contained within the main element.

9. All elements, except the extended elements, generated by the mapping MUST be in the `cdm` namespace.

5.3. Mapping vCard 3.0

The same [vCard21] mapping is used except that type parameters are mapped differently. It should be noted that the generic to [vCard21] mapping observes all the restrictions required to generate a valid vCard 3.0 ([RFC2426])

Instead of rule 5.1.7, the elements representing types are mapped to vCard 3.0 ([RFC2426]) type parameter `TYPE` such that the value of the parameter corresponds to the name of the element.

Instead of rule 5.2.8, the vCard 3.0 [RFC2426]) type parameters whose name is `TYPE` are mapped to corresponding XML elements such that the value of the parameter is upper-case and used as the element name.

6. Mapping vCard Jabber

6.1. Pure Jabber

This mapping is indicated by specifying `cb:format` XML attribute with value "`urn:liberty:cb:format:vcard-temp`."

The content shall be formatted as specified in [JEP0054]. Since the contact book conceptual data model is derived from [JEP0054], the mapping is almost one-to-one (i.e., null mapping) with the only exception being that the implementation MAY remove `cb:group` and `xml:lang` XML attributes.

6.2. Generic vCard

This mapping is indicated by specifying `cb:format` XML attribute with value "`urn:liberty:id-sis-cb:2005-05`." This mapping is a null mapping (i.e., the conceptual data model is directly used on the wire as well). This mapping differs from the Jabber mapping in that `cb:group` and `xml:lang` XML attributes MUST NOT be removed.

7. Mapping vCard RDF

This mapping is indicated by specifying `cb:format` XML attribute with value "`http://www.w3.org/2001/vcard-rdf/3.0#`." The content shall be formatted as specified in [vCardRDF].

Each generic element is mapped "as-is," except that the `vCard:group` XML attribute MAY be ignored or MAY be mapped to an RDF bag construct. Subelements of `N`, `ADR`, and `ORG` are converted to the [vCardRDF] naming convention by converting to lower-case all but the first letter.

8. XML Schemata

8.1. XML Schema for ID-SIS-CB

Formal XML schema for the ID-SIS-CB follows.

Note that this specifies the protocol aspect of the ID-SIS-CB service and relies heavily on the [LibertyDST20] schema. This schema does not specify the actual data representation (for that see the various mapping chapters of this specification).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
      xmlns="urn:liberty:id-sis-cb:2005-05"
      xmlns:cb="urn:liberty:id-sis-cb:2005-05"
      xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:liberty:id-sis-cb:2005-05"
      elementFormDefault="qualified">
   <xs:import
         namespace="urn:liberty:cb:conceptual-data-model:2005-05"
         schemaLocation="liberty-id-sis-cb-cdm-v1.0.xsd"/>
   <xs:include schemaLocation="liberty-idwsf-dst-v2.0.xsd"/>
   <xs:include schemaLocation="liberty-idwsf-dst-dt-v2.0.xsd"/>
   <xs:element name="Card" type="cb:cardType"/>
   <xs:complexType name="cardType">
      <xs:choice>
         <xs:element ref="cb:charData"/>
         <xs:any namespace="##other" processContents="lax"/>
      </xs:choice>
      <xs:attribute ref="cb:format" use="required"/>
   </xs:complexType>
   <xs:attribute name="format" type="xs:anyURI"/>
   <xs:element name="charData" type="cb:charDataType"/>
   <xs:simpleType name="charDataType">
      <xs:restriction base="xs:string"/> <!-- vCard data in specified format -->
   </xs:simpleType>
   <xs:complexType name="SelectType">    <!-- connects to DST framework -->
      <xs:simpleContent>
         <xs:extension base="xs:string">     <!-- XPath expression -->
            <xs:attribute ref="cb:format"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="SortType">
      <xs:sequence>
         <xs:element ref="By" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- connects to DST framework -->
   </xs:complexType>
   <xs:element name="By" type="cb:ByType"/>
   <xs:complexType name="ByType" mixed="true">
      <xs:attribute ref="sortAlg"/>
      <xs:attribute ref="sortWeight"/>
   </xs:complexType>
   <xs:attribute name="sortAlg">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="asc"/>
            <xs:enumeration value="desc"/>
            <!-- Ascending order, e.g., alphabetic order -->
            <!-- Descending order, e.g., inverse alphabet -->
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="sortWeight">
```

```
      <xs:simpleType>
         <xs:restriction base="xs:integer"/>
      </xs:simpleType>
   </xs:attribute>
   <xs:attributeGroup name="typeAttributes">
      <xs:attributeGroup ref="localizedLeafAttributes"/>
      <xs:attribute ref="group"/>
   </xs:attributeGroup>
   <xs:attribute name="group">
      <xs:simpleType>
         <xs:restriction base="xs:string"/>
      </xs:simpleType>
   </xs:attribute>
   <xs:element name="ReportUsage" type="cb:ReportUsageType"/>
   <xs:complexType name="ReportUsageType">
      <xs:sequence>
         <xs:group ref="ResourceIDGroup" minOccurs="0"/>
         <xs:element ref="cdm:CARDID"/>
         <xs:element ref="cb:UsageType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>
   <xs:element name="UsageType" type="UsageTypeType"/>
   <xs:complexType name="UsageTypeType">
      <xs:simpleContent>
         <xs:extension base="cb:DSTURI">
            <xs:attribute name="success" type="xs:boolean" default="1"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:element name="ReportUsageResponse" type="ReportUsageResponseType"/>
   <xs:complexType name="ReportUsageResponseType">
      <xs:sequence>
         <xs:element ref="Status"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>
   <xs:complexType name="TypeType">
      <xs:complexContent>
         <xs:restriction base="EmptyType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="TriggerType">
      <xs:complexContent>
         <xs:restriction base="EmptyType"/>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

8.2. XML Schema for the Conceptual Data Model

The purpose of this schema is to act as a conceptual data model against which the [XPATH] expressions can be applied. Actual data on the wire is usually in another format.

This XML schema was derived from the DTD described in [JEP0054] which in turn was derived from [DAWSON]. The Liberty Alliance has altered the Jabber work by adding `cb:group` and `xml:lang` support as well as support for many attributes defined in [LibertyDST20]. To support identification of card objects, `CARDID` was added as were `PHYSICALACCESS`, `SELF`, and `FAVORITE`. To support distribution lists, `DISTRIBUTIONLIST` and `LISTMEMBER` elements were added. Finally, `CALURI`, `CAPURI`, `CALADRURI`, and `FBURL` were added from [RFC2739].

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
      xmlns="urn:liberty:cb:conceptual-data-model:2005-05"
      xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"
```

```
      xmlns:cb="urn:liberty:id-sis-cb:2005-05"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:liberty:cb:conceptual-data-model:2005-05">
   <xs:import
         namespace="urn:liberty:id-sis-cb:2005-05"
         schemaLocation="liberty-id-sis-cb-proto-v1.0.xsd"/>
   <xs:element name="vCard">
      <xs:complexType>
         <xs:sequence>
            <xs:sequence>
               <xs:element ref="cdm:VERSION"/>
               <xs:element ref="cdm:CARDID"/>
               <xs:element ref="cdm:DISTRIBUTIONLIST" minOccurs="0"/>
               <xs:element ref="cdm:SELF" minOccurs="0"/>
               <xs:element ref="cdm:FAVORITE" minOccurs="0"/>
               <xs:element ref="cdm:FN" maxOccurs="unbounded"/>
               <xs:element ref="cdm:N" maxOccurs="unbounded"/>
               <xs:element ref="cdm:LISTMEMBER" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element ref="cdm:NICKNAME" minOccurs="0"/>
               <xs:element ref="cdm:PHOTO" minOccurs="0"/>
               <xs:element ref="cdm:BDAY" minOccurs="0"/>
               <xs:element ref="cdm:ADR" minOccurs="0"/>
               <xs:element ref="cdm:LABEL" minOccurs="0"/>
               <xs:element ref="cdm:TEL" minOccurs="0"/>
               <xs:element ref="cdm:EMAIL" minOccurs="0"/>
               <xs:element ref="cdm:JABBERID" minOccurs="0"/>
               <xs:element ref="cdm:MAILER" minOccurs="0"/>
               <xs:element ref="cdm:TZ" minOccurs="0"/>
               <xs:element ref="cdm:GEO" minOccurs="0"/>
               <xs:element ref="cdm:TITLE" minOccurs="0"/>
               <xs:element ref="cdm:ROLE" minOccurs="0"/>
               <xs:element ref="cdm:LOGO" minOccurs="0"/>
               <xs:element ref="cdm:AGENT" minOccurs="0"/>
               <xs:element ref="cdm:ORG" minOccurs="0"/>
               <xs:element ref="cdm:CATEGORIES" minOccurs="0"/>
               <xs:element ref="cdm:NOTE" minOccurs="0"/>
               <xs:element ref="cdm:PRODID" minOccurs="0"/>
               <xs:element ref="cdm:REV" minOccurs="0"/>
               <xs:element ref="cdm:SORT-STRING" minOccurs="0"/>
               <xs:element ref="cdm:SOUND" minOccurs="0"/>
               <xs:element ref="cdm:UID" minOccurs="0"/>
               <xs:element ref="cdm:URL" minOccurs="0"/>
               <xs:element ref="cdm:CLASS" minOccurs="0"/>
               <xs:element ref="cdm:KEY" minOccurs="0"/>
               <xs:element ref="cdm:DESC" minOccurs="0"/>
               <xs:element ref="cdm:PHYSICALACCESS" minOccurs="0"/>
               <xs:element ref="cdm:CALURI" minOccurs="0"/>
               <xs:element ref="cdm:CAPURI" minOccurs="0"/>
               <xs:element ref="cdm:CALADRURI" minOccurs="0"/>
               <xs:element ref="cdm:FBURL" minOccurs="0"/>
               <xs:element ref="cdm:Extension" minOccurs="0"/>
            </xs:sequence>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="VERSION">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="FN">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
```

```
   </xs:element>
   <xs:element name="N">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:FAMILY" minOccurs="0"/>
            <xs:element ref="cdm:GIVEN" minOccurs="0"/>
            <xs:element ref="cdm:MIDDLE" minOccurs="0"/>
            <xs:element ref="cdm:PREFIX" minOccurs="0"/>
            <xs:element ref="cdm:SUFFIX" minOccurs="0"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="FAMILY">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="GIVEN">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="MIDDLE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="PREFIX">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="SUFFIX">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="NICKNAME">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="PHOTO">
      <xs:complexType>
         <xs:choice>
            <xs:sequence>
               <xs:element ref="cdm:TYPE"/>
               <xs:element ref="cdm:BINVAL"/>
            </xs:sequence>
            <xs:element ref="cdm:EXTVAL"/>
         </xs:choice>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="BDAY">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="ADR">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:HOME" minOccurs="0"/>
            <xs:element ref="cdm:WORK" minOccurs="0"/>
            <xs:element ref="cdm:POSTAL" minOccurs="0"/>
            <xs:element ref="cdm:PARCEL" minOccurs="0"/>
```

```
            <xs:choice minOccurs="0">
               <xs:element ref="cdm:DOM"/>
               <xs:element ref="cdm:INTL"/>
            </xs:choice>
            <xs:element ref="cdm:PREF" minOccurs="0"/>
            <xs:element ref="cdm:POBOX" minOccurs="0"/>
            <xs:element ref="cdm:EXTADR" minOccurs="0"/>
            <xs:element ref="cdm:STREET" minOccurs="0"/>
            <xs:element ref="cdm:LOCALITY" minOccurs="0"/>
            <xs:element ref="cdm:REGION" minOccurs="0"/>
            <xs:element ref="cdm:PCODE" minOccurs="0"/>
            <xs:element ref="cdm:CTRY" minOccurs="0"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="POBOX">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="EXTADR">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="STREET">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="LOCALITY">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="REGION">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="PCODE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="CTRY">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="LABEL">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:HOME" minOccurs="0"/>
            <xs:element ref="cdm:WORK" minOccurs="0"/>
            <xs:element ref="cdm:POSTAL" minOccurs="0"/>
            <xs:element ref="cdm:PARCEL" minOccurs="0"/>
            <xs:choice minOccurs="0">
               <xs:element ref="cdm:DOM"/>
               <xs:element ref="cdm:INTL"/>
            </xs:choice>
            <xs:element ref="cdm:PREF" minOccurs="0"/>
            <xs:element ref="cdm:LINE" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
```

```
   </xs:element>
   <xs:element name="LINE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="TEL">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:HOME" minOccurs="0"/>
            <xs:element ref="cdm:WORK" minOccurs="0"/>
            <xs:element ref="cdm:VOICE" minOccurs="0"/>
            <xs:element ref="cdm:FAX" minOccurs="0"/>
            <xs:element ref="cdm:PAGER" minOccurs="0"/>
            <xs:element ref="cdm:MSG" minOccurs="0"/>
            <xs:element ref="cdm:CELL" minOccurs="0"/>
            <xs:element ref="cdm:VIDEO" minOccurs="0"/>
            <xs:element ref="cdm:BBS" minOccurs="0"/>
            <xs:element ref="cdm:MODEM" minOccurs="0"/>
            <xs:element ref="cdm:ISDN" minOccurs="0"/>
            <xs:element ref="cdm:PCS" minOccurs="0"/>
            <xs:element ref="cdm:PREF" minOccurs="0"/>
            <xs:element ref="cdm:NUMBER"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="NUMBER">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="EMAIL">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:HOME" minOccurs="0"/>
            <xs:element ref="cdm:WORK" minOccurs="0"/>
            <xs:element ref="cdm:INTERNET" minOccurs="0"/>
            <xs:element ref="cdm:PREF" minOccurs="0"/>
            <xs:element ref="cdm:X400" minOccurs="0"/>
            <xs:element ref="cdm:USERID"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="USERID">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="JABBERID">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="MAILER">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="TZ">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="GEO">
      <xs:complexType>
```

```
         <xs:sequence>
            <xs:element ref="cdm:LAT"/>
            <xs:element ref="cdm:LON"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="LAT">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="LON">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="TITLE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="ROLE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="LOGO">
      <xs:complexType>
         <xs:choice>
            <xs:sequence>
               <xs:element ref="cdm:TYPE"/>
               <xs:element ref="cdm:BINVAL"/>
            </xs:sequence>
            <xs:element ref="cdm:EXTVAL"/>
         </xs:choice>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="AGENT">
      <xs:complexType>
         <xs:choice>
            <xs:element ref="cdm:vCard"/>
            <xs:element ref="cdm:EXTVAL"/>
         </xs:choice>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="ORG">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:ORGNAME"/>
            <xs:element ref="cdm:ORGUNIT" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="cb:commonAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="ORGNAME">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="ORGUNIT">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
```

```
<xs:element name="CATEGORIES">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="cdm:KEYWORD" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attributeGroup ref="cb:commonAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="KEYWORD">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="NOTE">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:typeAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="PRODID">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="REV">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="SORT-STRING">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:typeAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="SOUND">
   <xs:complexType>
      <xs:choice>
         <xs:element ref="cdm:PHONETIC"/>
         <xs:element ref="cdm:BINVAL"/>
         <xs:element ref="cdm:EXTVAL"/>
      </xs:choice>
      <xs:attributeGroup ref="cb:commonAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="PHONETIC">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:typeAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="UID">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="URL">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="DESC">
   <xs:complexType mixed="true">
      <xs:attributeGroup ref="cb:typeAttributes"/>
   </xs:complexType>
</xs:element>
<xs:element name="CLASS">
   <xs:complexType>
      <xs:choice>
         <xs:element ref="cdm:PUBLIC"/>
```

```
            <xs:element ref="cdm:PRIVATE"/>
            <xs:element ref="cdm:CONFIDENTIAL"/>
         </xs:choice>
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="PUBLIC">
      <xs:complexType/>
   </xs:element>
   <xs:element name="PRIVATE">
      <xs:complexType/>
   </xs:element>
   <xs:element name="CONFIDENTIAL">
      <xs:complexType/>
   </xs:element>
   <xs:element name="KEY">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="cdm:TYPE" minOccurs="0"/>
            <xs:element ref="cdm:CRED"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="CRED">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="HOME">
      <xs:complexType/>
   </xs:element>
   <xs:element name="WORK">
      <xs:complexType/>
   </xs:element>
   <xs:element name="POSTAL">
      <xs:complexType/>
   </xs:element>
   <xs:element name="PARCEL">
      <xs:complexType/>
   </xs:element>
   <xs:element name="DOM">
      <xs:complexType/>
   </xs:element>
   <xs:element name="INTL">
      <xs:complexType/>
   </xs:element>
   <xs:element name="PREF">
      <xs:complexType/>
   </xs:element>
   <xs:element name="VOICE">
      <xs:complexType/>
   </xs:element>
   <xs:element name="FAX">
      <xs:complexType/>
   </xs:element>
   <xs:element name="PAGER">
      <xs:complexType/>
   </xs:element>
   <xs:element name="MSG">
      <xs:complexType/>
   </xs:element>
   <xs:element name="CELL">
      <xs:complexType/>
   </xs:element>
   <xs:element name="VIDEO">
      <xs:complexType/>
   </xs:element>
```

```
   <xs:element name="BBS">
      <xs:complexType/>
   </xs:element>
   <xs:element name="MODEM">
      <xs:complexType/>
   </xs:element>
   <xs:element name="ISDN">
      <xs:complexType/>
   </xs:element>
   <xs:element name="PCS">
      <xs:complexType/>
   </xs:element>
   <xs:element name="INTERNET">
      <xs:complexType/>
   </xs:element>
   <xs:element name="X400">
      <xs:complexType/>
   </xs:element>
   <xs:element name="TYPE">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="BINVAL">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="EXTVAL">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:leafAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="PHYSICALACCESS">
      <xs:complexType mixed="true">
         <xs:attributeGroup ref="cb:typeAttributes"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="SELF">
      <xs:complexType/>
   </xs:element>
   <xs:element name="FAVORITE">
      <xs:complexType/>
   </xs:element>
   <xs:element name="DISTRIBUTIONLIST">
      <xs:complexType/>
   </xs:element>
   <xs:element name="CALURI" type="cdm:QualifiedURIType"/>
   <xs:element name="CAPURI" type="cdm:QualifiedURIType"/>
   <xs:element name="CALADRURI" type="cdm:QualifiedURIType"/>
   <xs:element name="URI" type="cb:DSTURI"/>
   <xs:element name="FBURL" type="cdm:QualifiedURIType"/>
   <xs:complexType name="QualifiedURIType" mixed="true">
      <xs:sequence>
         <xs:element ref="cdm:PREF" minOccurs="0"/>
         <xs:element ref="cdm:URI"/>
      </xs:sequence>
      <xs:attributeGroup ref="cb:leafAttributes"/>
   </xs:complexType>
   <xs:element name="CARDID" type="xs:anyURI"/>
   <xs:element name="LISTMEMBER" type="cb:DSTURI"/>
   <xs:element name="Extension" type="cb:extensionType"/>
</xs:schema>
```

9. Abstract WSDL for ID-SIS-CB

The abstract Web Services Description Language (WSDL) declaration for ID-SIS Contact Book follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
      xmlns:typens="urn:liberty:id-sis-cb:2005-05:wsdl:interface"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:cb="urn:liberty:id-sis-cb:2005-05"
      targetNamespace="urn:liberty:id-sis-cb:2005-05:wsdl:interface"
      name="id-sis-cb_2005-05_interface">
   <types>
      <xsd:schema>
         <xsd:import
            namespace="urn:liberty:id-sis-cb:2005-05"
            schemaLocation="liberty-id-sis-cb-proto-v1.0.xsd"/>
      </xsd:schema>
   </types>
   <message name="Query">
      <part name="body" element="cb:Query"/>
   </message>
   <message name="QueryResponse">
      <part name="body" element="cb:QueryResponse"/>
   </message>
   <message name="Modify">
      <part name="body" element="cb:Modify"/>
   </message>
   <message name="ModifyResponse">
      <part name="body" element="cb:ModifyResponse"/>
   </message>
   <message name="Subscribe">
      <part name="body" element="cb:Subscribe"/>
   </message>
   <message name="SubscribeResponse">
      <part name="body" element="cb:SubscribeResponse"/>
   </message>
   <message name="QuerySubscriptions">
      <part name="body" element="cb:QuerySubscriptions"/>
   </message>
   <message name="Subscriptions">
      <part name="body" element="cb:Subscriptions"/>
   </message>
   <portType name="ContactbookPort">
      <operation name="ContactbookQuery">
         <input message="typens:Query"/>
         <output message="typens:QueryResponse"/>
      </operation>
      <operation name="ContactbookModify">
         <input message="typens:Modify"/>
         <output message="typens:ModifyResponse"/>
      </operation>
      <operation name="ContactbookSubscribe">
         <input message="typens:Subscribe"/>
         <output message="typens:SubscribeResponse"/>
      </operation>
      <operation name="ContactbookQuerySubscriptions">
         <input message="typens:QuerySubscriptions"/>
         <output message="typens:Subscriptions"/>
      </operation>
   </portType>
</definitions>
```

10. Mapping Contact Book to Personal Profile

An ID-SIS-CB service MAY be implemented without any relation to [LibertyIDPP] service. If, however, a system or process maps Contact Cards to or from Personal Profile ([LibertyIDPP]), it SHOULD use the mappings defined here.

10.1. Mapping a vCard to Personal Profile

When mapping a vCard (either version) to [LibertyIDPP], the following correspondence between vCard types, possibly qualified with parameters, and [LibertyIDPP] containers and elements SHOULD be used.

Table 14. Mapping vCard to PP

vCard Attribute	*PP attribute*	*Notes*
`NAME`	?	Possibly CN could be used
`PROFILE`	-	Omitted or synthesized as needed.
`SOURCE`	?	
`FN`	`CN`	
`FN;language=XX`	`LCN`	
`N`	`AnalyzedName`	
`N;language=XX`	`AnalyzedName`	"`L`"-prefixed elements are likely to be used.
`NICKNAME`	`InformalName`	Also: `MsgContact/Nick` and `AddressCard/Nick`
`NICKNAME;language=XX`	`LInformalName`	
`PHOTO`	`MugShot`	
`BDAY`	`DOB`	
`ADR`	`Address`	`TYPE` parameter maps to `AddrType` element
`ADR;language=XX`	`Address`	"`L`"-prefixed elements are likely to be used.
`LABEL`	?	
`TEL`	`MsgContact`	`TYPE` parameter maps to `MsgType`, `MsgMethod`, and `MsgTechnology` elements
`EMAIL`	`MsgContact`	`TYPE` parameter maps to `MsgType`, `MsgMethod`, and `MsgTechnology` elements
`MAILER`	?	Mail User Agent software model and version

1839 **Table 15. Mapping vCard to PP (continued)**

vCard Attribute	*PP attribute*	*Notes*
TZ	TimeZone	
GEO	?	out of scope
TITLE	JobTitle or InternalJobTitle	N.B. This is NOT PersonalTitle (which is the honorific prefix)
TITLE;language=XX	LJobTitle or LInternalJobTitle	
ROLE	?	May be InternalJobTitle or JobTitle?
LOGO	MugShot	
ORG	O or AltO + OU	
ORG;language=XX	LO or LAltO	
CATEGORIES	?	
NOTE		
PRODID	?	Software model and version that created vCard
REV	modificationTime	See [LibertyDST20], sec 2.4.1
SORT-STRING	?	
SOUND	NamePronounced	
UID	?	Globally Unique ID of vCard
URL	WebSite	
VERSION	-	Omitted or synthesized as needed.
CLASS	?	[LibertyIDPP] (v1.1). For v1.0, an extension MAY be used or the type MAY be omitted.
KEY (unknown)	SignKey	EncryptKey MAY be used if out-of-band or context indicates this to be the purpose of the key.
KEY;TYPE=x-sign	SignKey	
KEY;TYPE=x-enc	EncryptKey	

1840 Whenever the language vCard parameter is supplied and the mapping specifies either an element without or with "L"
1841 prefix, the "L"-prefixed variant MUST be used and the lang XML attribute set. If, however, the value of the type is
1842 entirely representable using the Latin 1 alphabet, then the element without "L" prefix MAY be used instead of, or in
1843 addition to, the "L" -prefixed variant.

1844 The lang XML attribute MUST be set according to the language vCard parameter.

1845 Any vCard types that use a private extension naming convention (i.e., start with "X-" or "x-") SHOULD be mapped to
1846 Personal Profile ([LibertyIDPP]) extensions using a convenient names space of the mapper's choice. Such extended
1847 types MAY be omitted.

1848 Any other vCard types that do not appear in the "Mapping vCard to PP" or "Mapping PP to vCard Extensions" tables
1849 SHOULD be mapped to Personal Profile ([LibertyIDPP]) extensions using a convenient names space of the mapper's
1850 choice. Such extended types MAY be omitted.

The result of the conversion MUST conform to ID-SIS-PP [LibertyIDPP]. This, among other requirements, means that:

- Any line-folding or escaping permitted or specified by vCard MUST be removed and XML conventions for escaping MUST be applied.
- All values must be represented using the UTF-8 character set

10.2. Mapping a Personal Profile to a vCard

When mapping a PP ([LibertyIDPP]) to a vCard (either version), the correspondence specified in the table "Mapping vCard to PP" should be used. For those containers and elements that do not appear above, the extended vCard types, possibly qualified by parameters, specified in the table "Mapping PP to vCard Extensions," below, SHOULD be used. The mapping MAY also choose to omit the containers and elements that do not have correspondence in the vCard standard.

Table 16. Mapping PP to vCard Extensions

vCard Attribute	*PP attribute*	*Notes*
?	Age	Can be calculated from `BDay` or `DOB`
?	`Agegroup`	Can be calculated from `BDay` or `DOB`
?	`Birthday`	Can be calculated from `BDay` or `DOB`

Any extensions of the Personal Profile ([LibertyIDPP]) SHOULD be mapped to nonstandard vCard types whose names start by "`x-`" as described in [RFC2426], Section 3.8, unless the creator of the extension has registered an appropriate vCard type that has same semantics.

Despite the destination vCard format possibly allowing free choice of character set, the vCard MUST be produced in the UTF-8 character set if it is to appear as a `cb:Card` element returned by an ID-SIS-CB service.

The result of the conversion MUST be a valid vCard according to the version of vCard that is to be produced. Among other requirements, this means

- any values MUST be escaped and line-folded according to the requirements of the vCard version that is to be produced, cf. [RFC2426] Sections 2.5 and 2.6.

10.3. `MsgType`, `MsgMethod`, and `MsgTechnology` Mappings

vCard `EMAIL` and `TEL` types are represented by PP ([LibertyIDPP]) `MsgContact` containers. vCard allows the specification of `TYPE` parameters to further qualify the contact. In the `MsgContact` container, the `MsgType`, `MsgMethod`, and `MsgTechnology` perform similar roles. In order to establish an interoperable basis, the following mappings MUST be used. If a mapping is not specified here, then the implementation MAY

1. coerce to one of the defined mappings,
2. register an extension mapping, or
3. use private mapping, prefixed by "`x-`" where applicable.

1880 It is RECOMMENDED that implementations only use the mappings defined here.

1881 **Table 17. Mapping vCard `EMAIL` and `TEL` Types to `MsgContact` Container**

vCard Type	*`MsgType`*	*`MsgMethod`*	*`MsgTechnology`*	*`Notes`*
`EMAIL`	`personal`	`email`	`email`	
`EMAIL;TYPE=internet`	`personal`	`email`	`email`	
`EMAIL;TYPE=x400`	`personal`	`email`	?	
`EMAIL;TYPE=pref`	*	*		Indicates preference
`TEL`	`personal`	`voice`	`pots`, `voip`	
`TEL;TYPE=voice`	*	`voice`	`pots`, `voip`	
`TEL;TYPE=fax`	*	`fax`	`fax`	
`TEL;TYPE=cell`	`mobile`	`voice`	`pots`	
`TEL;TYPE=video`	*	?	(`mms`?)	
`TEL;TYPE=pager`	*	`pager`	`sms`,`mms`,`pager`	
`TEL;TYPE=bbs`	*	?		
`TEL;TYPE=modem`	*	?		
`TEL;TYPE=car`	(`mobile`)			
`TEL;TYPE=isdn`	*	`voice`	`pots`	
`TEL;TYPE=pcs`	*	?		
`TEL;TYPE=msg`	*	`voice`		
?	*	`im`	aol,icq, ...	
`TEL;TYPE=home`	`personal`			
`TEL;TYPE=work`	`work`			
`TEL;TYPE=pref`	*	*		Indicates preference
?	`vacation`	*		
?	`emergency`	*		

1882 In this table, an asterisk (*) indicates that any value is eligible. A question mark indicates a gap in the mapping.

1883 Multiple `TEL` and `EMAIL` types generate multiple `MsgContact` containers and vice versa.

1884 10.4. Mapping `LegalIdentity`

1885 vCard is not intended to convey rigorously formal legal names. Therefore, when producing vCards from PP
1886 ([LibertyIDPP]), the naming elements from the `CommonName` container SHOULD be used. If an element is missing
1887 from the `CommonName` container, then the corresponding element in the `LegalIdentity` container MAY be used. In
1888 absence of the `CN` element, the `LegalName` element MAY be used.

1889 When producing PP ([LibertyIDPP]) from vCard, the `CommonName` container SHOULD be populated from the `FN` and
1890 `N` types. The `LegalIdentity` container SHOULD NOT be altered or created if it did not previously exist.

10.5. Mapping vCards Representing Organizations

Since Personal Profile ([LibertyIDPP]) represents an individual, vCards that represent organizations or business entities can not be mapped.

10.6. Mapping `N` Type to `AnalyzedName` container

vCard `N` type has an ordered list of naming components as follows.

Table 18. Mapping `N` to `AnalyzedName`

Ordinal	*CB XML*	*PP XML*	*Meaning*
1	`FAMILY`	`SN`, `LSN`	Family name (surname)
2	`GIVEN`	`FN`, `LFN`	First name
3	`MIDDLE`	`MN`, `LMN`	Other names (middle names)
4	`PREFIX`	`PersonalTitles`, `LPersonalTitle`	Honorific Prefixes
5	`SUFFIX`	?	Honorific suffixes

N.B. It is an unfortunate historical fact that the `FN` type in vCard has a conflicting meaning to the `FN` element in Personal Profile ([LibertyIDPP] and, in fact, to many other common specifications).

[RFC2426] requires that the `N` type is always present whereas Personal Profile ([LibertyIDPP]) defines the `AnalyzedName` container as optional. If `AnalyzedName` is not available, then the following special value SHOULD be used

```
N:;;;;
```

as a place holder. If an implementation is able to analyze `CN` or otherwise knows or can guess some fields of `N` type, it MAY do so.

10.7. Mapping FN

If Personal Profile ([LibertyIDPP]) does not have a `CN` element, but has `AnalyzedName`, an implementation MAY compose a `FN` type from data in the `AnalyzedName` container in an implementation-dependent way.

When producing [RFC2426]-compliant output, if the `CN` element is not available and the implementation is not able to synthesize it from the `AnalyzedName` container, it SHOULD provide the `FN` type with the following special value

```
FN:N.N.
```

as a place holder.

10.8. Mapping `ADR` Type to `Address` Container

Mapping the `ADR` type to the `Address` container via the following:

1914 **Table 19. Mapping ADR to Address**

Ordinal	*CB XML*	*PP XML*	*Notes*
1	POBOX	?	Post Office Box
2	EXTADR	?	Extended Address
3	STREET	PostalAddress, LPostalAddress	Street Address
4	LOCALITY	L, LL	Locality
5	REGION	St, LSt	Region (Province or State)
6	PCODE	PostalCode	Zip code
7	CTRY	C	Country

1915 [RFC2426] specifies the seventh field as "country name" where as [LibertyIDPP] specifies that the ISO country code
1916 is used. Mapping from ISO country code to country name is easy (though possibly language-dependent), but the
1917 inverse mapping is impossible to make reliably. It is RECOMMENDED that implementations use heuristics to cover
1918 common cases and when heuristics fail, populate in C the value of the seventh field prefixed by string "??? " (three
1919 question marks and a space).

1920 When mapping from the ADR type to the Address container and fields 1 and 2 are nonempty, they MUST be appended
1921 to the PostalAddress container.

1922 When mapping from the Address container to the ADR type, fields 1 and 2 MUST be left empty.

1923 Multiple ADR types generate multiple AddressCard containers and vice versa.

1924 The TYPE parameter maps to the AddrType element as follows.

1925 **Table 20. Mapping ADR TYPE to AddrType**

TYPE	*AddrType*	*Notes*
(none)	work	None represents the default of TYPE=intl,postal,parcel,work
home	domicile	legal residence
home	home	everyday home
work	work	work address, the office where the person works
?	vacation	holiday address
?	emergency	structured emergency contact
dom	?	
intl	?	
postal	?	
parcel	?	
pref	-	Indicates preference

1926 As can be seen, many TYPE parameters do not have a mapping. In these cases, the mapping is lossy. When generating
1927 the ADR type from the AddressCard container, an implementation MAY provide TYPE parameters such as intl or
1928 postal that it knows match the data.

The group construct defined in [RFC2425], Section 5.8.2–penultimate paragraph, and in [vCard21], Section 2.1.4.2, SHOULD be used to group together types derived from the same `AddressCard` container. The group label MUST be picked such that it is unique across multiple `AddressCard` containers that may be present in one Personal Profile ([LibertyIDPP]). When mapping from vCard to Personal Profile ([LibertyIDPP]), the types with the same group MUST be mapped to elements within the same `AddressCard` container.

References

Normative

[DAWSON] Dawson, F., Hoffman, P., eds. (15 October, 1998). "The vCard v3.0 XML DTD," expired draft-dawson-vcard-xml-dtd-01.txt, Internet Engineering Task Force/Watersprings.org *http://www.watersprings.org/pub/id/draft-dawson-vcard-xml-dtd-01.txt*

[JEP0054] Saint-Andre, Peter, eds. (26 March, 2003). "Jabber Enhancement Proposal 0054: vcard-temp," v1.1, Jabber Software Foundation *http://www.jabber.org/jeps/jep-0054.html*

[LibertyDisco12] Sergent, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty Alliance Project (12 December 2004). *http://www.projectliberty.org/specs/*

[LibertyDST20] Kainulainen, Jukka, Ranganathan, Aravindan, eds. "Liberty ID-WSF Data Services Template Specification," Version 2.0, Liberty Alliance Project (23 March, 2005). *http://www.projectliberty.org/specs*

[LibertyIDEP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Employee Profile Service Specification," Version 1.1, Liberty Alliance Project (29 September, 2005). *http://www.projectliberty.org/specs*

[LibertyIDPP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Specification," Version 1.1, Liberty Alliance Project (29 September, 2005). *http://www.projectliberty.org/specs*

[LibertyInteract11] Aarts, Robert, eds. "Liberty ID-WSF Interaction Service Specification," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyReg] Kemp, John, eds. "Liberty Enumeration Registry Governance," Version 1.1, Liberty Alliance Project (14 December, 2004). *http://www.projectliberty.org/specs*

[LibertySOAPBinding12] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. "Liberty ID-WSF SOAP Binding Specification," Version 1.2, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs/*

[RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt*

[RFC2425] Howes, T., Smith, M., Dawson, F., eds. (September 1998). "A MIME Content-Type for Directory Information," RFC 2425, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2425.txt*

[RFC2426] Dawson, F., Howes, T., eds. (September 1998). "vCard MIME Directory Profile," RFC 2426, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2426.txt*

[RFC2739] Small, T., Hennessy, D., Dawson, F., eds. (January 2000). "Calendar Attributes for vCard and LDAP," RFC 2739, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2739.txt*

[Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May 2001). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/*

[vCard21] Consortium, Versit, eds. (18 September, 1996). "vCard, The Electronic Business Card," Version 2.1, Internet Mail Consortium *http://www.imc.org/pdi/vcard-21.txt*

[vCardRDF] Iannella, Renato, eds. (22 February, 2001). "Representing vCard Objects in RDF/XML," W3C NOTE-vcard-rdf-20010222, World Wide Web Consortium *http://www.w3.org/TR/vcard-rdf*

[XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve, Yergeau, Francois, eds. (04 February 2004). "Extensible Markup Language (XML) 1.0 (Third Edition)," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/2004/REC-xml-20040204*

[XPATH] Clark , J., DeRose , S., eds. (16 November 1999). "XML Path Language (XPath) Version 1.0 ," Recommendation, W3C *http://www.w3.org/TR/xpath [August 2003]*.

Informative

[JReg] Software Foundation, Jabber, eds. (2004). "Jabber :: Registrar," , Jabber Software Foundation *http://www.jabber.org/registrar/*

[LibertyCBGuide] Lambert, Guillaume, eds. "Liberty ID-SIS Contact Book Service Implementation Guidelines," Version 1.0-08, Liberty Alliance Project (14 February, 2006). *http://www.projectliberty.org/specs*

[LibertyIDEPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Employee Profile Service Implementation Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005). *http://www.projectliberty.org/specs*

[LibertyIDPPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Implementation Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005). *http://www.projectliberty.org/specs*

[LibertyIDWSFGuide10] Weitzel, David, eds. (22 May 2005). "Liberty ID-WSF Implementation Guideline," Draft v1.0-12, Liberty Alliance Project *http://www.projectliberty.org/specs/*

[LibertyIDWSFOverview11] Tourzan, Jonathan, Koga, Yuzo, eds. "Liberty ID-WSF Web Services Framework Overview," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*