



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.

Copyright © 2005-2006 America Online, Inc.; American Express Travel Related Services; Ericsson; France Télécom; Nippon Telegraph and Telephone Corporation; Nokia Corporation; Sun Microsystems, Inc.; Symlabs, Inc.; and Vodafone Group Plc. All rights reserved.

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

Contents

26		
27	1. Introduction	5
28	1.1. Notational Conventions	5
29	1.2. Derivation of ID-SIS-CB from DST and WSF	5
30	1.3. Relation of ID-SIS-CB to vCard Specifications	7
31	1.4. Compliance	8
32	1.5. Namespaces and Preamble	9
33	2. Generic vCard	11
34	2.1. Representing "Self" Card	12
35	2.2. Representing Favorites List	12
36	2.3. Unique Card Identification	12
37	2.4. Representing Distribution Lists	12
38	2.5. List Membership	12
39	2.6. Data Model Extension	12
40	3. Protocol Operations	14
41	3.1. Discovery Option Keywords	14
42	3.1.1. Data Availability Discovery Option Keywords	14
43	3.1.2. Data Update Discovery Option Keywords	14
44	3.1.3. Feature Discovery Option Keywords	14
45	3.2. Query Expressions	15
46	3.2.1. Additional and Value-Added Criteria	15
47	3.2.2. Restrictions on XPath Expressions	17
48	3.3. Query Processing	19
49	3.4. Query Examples	20
50	3.4.1. Picking TEL and EMAIL from Joe Smith's Contact Card	20
51	3.4.2. Returning Entire Cards	20
52	3.4.3. Returning All Cards in the Contact Book	20
53	3.4.4. Returning Some Attributes from All Cards	21
54	3.4.5. Latin 1 Case-Insensitive Matching	21
55	3.4.6. Latin 1 Accent- and Case-Insensitive Matching	21
56	3.4.7. Free Text Search	21
57	3.4.8. Picking by Type	22
58	3.4.9. Picking by Group	22
59	3.4.10. Picking Types in Same Group	22
60	3.4.11. Picking Types of a Given Language	22
61	3.4.12. Querying CATEGORIES	22
62	3.4.13. Finding Distribution Lists	22
63	3.4.14. Finding Members of a Distribution List	23
64	3.5. Sorting Contact Cards	23
65	3.6. Modifying Contact Cards and Attributes	24
66	3.6.1. Add New Cards	25
67	3.6.2. Replace Cards	25
68	3.6.3. Delete Cards	26
69	3.6.4. Add Attributes to a Card	26
70	3.6.5. Replace Attributes in a Card	27
71	3.6.6. Delete Attributes from a Card	27
72	3.6.7. Delete Specific Attributes from a Card	28
73	3.7. Reporting Usage of a Contact Card	28
74	3.8. Manipulating Distribution lists	29
75	4. Processing Rules and Other Considerations	30
76	4.1. Query is not Required to Report Same Data to Repeated Queries	30
77	4.2. Support of Multiple Modification Not Required	30
78	4.3. Simulation (dry-run) Is Required	30

79	4.4. There Can Only Be One SELF Card	30
80	4.5. Multiple Instances of Same Person	30
81	4.6. Error Messages in Case an Attribute Does Not Exist	30
82	4.7. Multiple Modification Is Restricted to One Card	30
83	4.8. Simplifications to Distribution Lists	31
84	4.9. Use of Usage Directives	31
85	5. Mapping MIME-DIR-Based vCard Formats	32
86	5.1. From Generic vCard to vCard 2.1	32
87	5.2. From vCard 2.1 to Generic vCard	32
88	5.3. Mapping vCard 3.0	33
89	6. Mapping vCard Jabber	34
90	6.1. Pure Jabber	34
91	6.2. Generic vCard	34
92	7. Mapping vCard RDF	35
93	8. XML Schemata	36
94	8.1. XML Schema for ID-SIS-CB	36
95	8.2. XML Schema for the Conceptual Data Model	37
96	9. Abstract WSDL for ID-SIS-CB	46
97	10. Mapping Contact Book to Personal Profile	47
98	10.1. Mapping a vCard to Personal Profile	47
99	10.2. Mapping a Personal Profile to a vCard	49
100	10.3. <code>MsgType</code> , <code>MsgMethod</code> , and <code>MsgTechnology</code> Mappings	49
101	10.4. Mapping <code>LegalIdentity</code>	50
102	10.5. Mapping vCards Representing Organizations	50
103	10.6. Mapping <code>N</code> Type to <code>AnalyzedName</code> container	51
104	10.7. Mapping <code>FN</code>	51
105	10.8. Mapping <code>ADR</code> Type to <code>Address</code> Container	51
106	References	54

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

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

155 This specification is divided in two levels:

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

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

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

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

```
168  
169 <xs:element name="Card" type="cb:cardType"/>  
170 <xs:complexType name="cardType">  
171   <xs:choice>  
172     <xs:element ref="cb:charData"/>  
173     <xs:any namespace="##other" processContents="lax"/>  
174   </xs:choice>  
175   <xs:attribute ref="cb:format" use="required"/>  
176 </xs:complexType>  
177 <xs:attribute name="format" type="xs:anyURI"/>  
178 <xs:element name="charData" type="cb:charDataType"/>  
179 <xs:simpleType name="charDataType">  
180   <xs:restriction base="xs:string"/> <!-- vCard data in specified format -->  
181 </xs:simpleType>
```

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

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

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

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

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

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

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

197 1.4. Compliance

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

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

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

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

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

216 1.5. Namespaces and Preamble

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

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

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

222 **Table 8. Referenced XML Namespaces**

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

223 The following preamble introduces the namespaces and [LibertyDST20].

```
224 <?xml version="1.0" encoding="UTF-8"?>
225 <xs:schema
226     xmlns="urn:liberty:id-sis-cb:2005-05"
227     xmlns:cb="urn:liberty:id-sis-cb:2005-05"
228     xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"
229     xmlns:xs="http://www.w3.org/2001/XMLSchema"
230     targetNamespace="urn:liberty:id-sis-cb:2005-05"
231     elementFormDefault="qualified">
232   <xs:import
233     namespace="urn:liberty:cb:conceptual-data-model:2005-05"
234     schemaLocation="liberty-id-sis-cb-cdm-v1.0.xsd"/>
235   <xs:include schemaLocation="liberty-idwsf-dst-v2.0.xsd"/>
236   <xs:include schemaLocation="liberty-idwsf-dst-dt-v2.0.xsd"/>
```

2. Generic vCard

237

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

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

249

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

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

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

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

255

```

256 <xs:attributeGroup name="typeAttributes">
257   <xs:attributeGroup ref="localizedLeafAttributes"/>
258   <xs:attribute ref="group"/>
259 </xs:attributeGroup>
260 <xs:attribute name="group">
261   <xs:simpleType>
262     <xs:restriction base="xs:string"/>
263   </xs:simpleType>
264 </xs:attribute>
    
```

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

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

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

273 **2.1. Representing "Self" Card**

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

275 **2.2. Representing Favorites List**

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

277 **2.3. Unique Card Identification**

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

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

283 **2.4. Representing Distribution Lists**

284 A vCard may represent a distribution list. This is indicated by including in `cdm:vCard` a child element:
285 `cdm:DISTRIBUTIONLIST`. Such card may have any attributes that an ordinary card has, but is not a contact by
286 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
287 query and the attributes common to all members of the list to be represented. The distribution list has a `cdm:CARDID`
288 like any other contact card.

289 **2.5. List Membership**

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

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

296 **2.6. Data Model Extension**

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

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

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

310 **Example**

```
311 <cdm:vCard>  
312   <cdm:FN>Zita Lopes</cdm:FN>  
313   <cdm:N>  
314     <cdm:FAMILY>Lopes</cdm:FAMILY>  
315     <cdm:GIVEN>Zita</cdm:GIVEN>  
316   </cdm:N>  
317   <cdm:Extension>  
318     <X-FAVCOLOR>red</X-FAVCOLOR>  
319   </cdm:Extension>  
320 </cdm:vCard>
```

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

```
323 BEGIN:vCard  
324 FN:Zita Lopes  
325 N:Lopes;Zita  
326 X-FAVCOLOR:red  
327 END:vCard
```

328 Note that the extension attribute has a preceding "X-" to indicate that it is a vCard extension, as specified in Section
329 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

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

364 **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

365 3.2. Query Expressions

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

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

```
370
371 <xs:complexType name="SelectType" <!-- connects to DST framework -->
372   <xs:simpleContent>
373     <xs:extension base="xs:string" <!-- XPath expression -->
374       <xs:attribute ref="cb:format"/>
375     </xs:extension>
376   </xs:simpleContent>
377 </xs:complexType>
```

378 A query has the following major components.

- 379 1. Which contact cards and distribution lists to return?
- 380 2. How to sort them? (See Section 3.5.)
- 381 3. Which attributes to return from them?
- 382 4. How many results are desired and what is the starting position in the data set? For specifying these, ID-SIS-CB
 383 uses the pagination mechanism described in Section 4.1.1 of [LibertyDST20].
- 384 5. In which format should the cards be returned? This is specified using cb:format XML attribute. It MUST be
 385 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)

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

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

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

429 3.2.2. Restrictions on XPath Expressions

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

432 1. Query expressions are composed of

433 a. one location path expression and

434 b. zero or one predicates.

435 2. Location path expressions MUST be either

436 a. simple location path or

437 b. a set of simple location paths joined with `|` operators in a single parenthesized expression

438 and each simple location path MUST start by `/cdm:vCard`.

439 3. The predicate MAY reference location paths that are not anchored at the document root.

440 4. Only abbreviated [XPath], as described in [XPath] Section 2.5, MUST be used.

441 5. In addition to functions listed in Section 3.2.1, only the following functions MUST be used:

```
442     concat()  
443     starts-with()  
444     contains()  
445     substring-before()  
446     substring-after()  
447     substring()  
448     string-length()  
449     normalize-space()  
450     translate()  
451     not()  
452     lang()
```

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

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

```
456 /cdm:vCard
457 /cdm:vCard/cdm:VERSION
458 /cdm:vCard/cdm:CARDID
459 /cdm:vCard/cdm:DISTRIBUTIONLIST
460 /cdm:vCard/cdm:SELF
461 /cdm:vCard/cdm:FAVORITE
462 /cdm:vCard/cdm:FN
463 /cdm:vCard/cdm:N
464 /cdm:vCard/cdm:N/cdm:FAMILY
465 /cdm:vCard/cdm:N/cdm:GIVEN
466 /cdm:vCard/cdm:N/cdm:MIDDLE
467 /cdm:vCard/cdm:N/cdm:PREFIX
468 /cdm:vCard/cdm:N/cdm:SUFFIX
469 /cdm:vCard/cdm:LISTMEMBER
470 /cdm:vCard/cdm:NICKNAME
471 /cdm:vCard/cdm:PHOTO
472 /cdm:vCard/cdm:PHOTO/cdm:EXTVAL
473 /cdm:vCard/cdm:BDAY
474 /cdm:vCard/cdm:ADR
475 /cdm:vCard/cdm:ADR/cdm:HOME
476 /cdm:vCard/cdm:ADR/cdm:WORK
477 /cdm:vCard/cdm:ADR/cdm:POSTAL
478 /cdm:vCard/cdm:ADR/cdm:PARCEL
479 /cdm:vCard/cdm:ADR/cdm:DOM
480 /cdm:vCard/cdm:ADR/cdm:INTL
481 /cdm:vCard/cdm:ADR/cdm:PREF
482 /cdm:vCard/cdm:ADR/cdm:POBOX
483 /cdm:vCard/cdm:ADR/cdm:EXTADR
484 /cdm:vCard/cdm:ADR/cdm:STREET
485 /cdm:vCard/cdm:ADR/cdm:LOCALITY
486 /cdm:vCard/cdm:ADR/cdm:REGION
487 /cdm:vCard/cdm:ADR/cdm:PCODE
488 /cdm:vCard/cdm:ADR/cdm:CTRY
489 /cdm:vCard/cdm:LABEL
490 /cdm:vCard/cdm:LABEL/cdm:HOME
491 /cdm:vCard/cdm:LABEL/cdm:WORK
492 /cdm:vCard/cdm:LABEL/cdm:POSTAL
493 /cdm:vCard/cdm:LABEL/cdm:PARCEL
494 /cdm:vCard/cdm:LABEL/cdm:DOM
495 /cdm:vCard/cdm:LABEL/cdm:INTL
496 /cdm:vCard/cdm:LABEL/cdm:PREF
497 /cdm:vCard/cdm:LABEL/cdm:LINE
498 /cdm:vCard/cdm:TEL
499 /cdm:vCard/cdm:TEL/cdm:HOME
500 /cdm:vCard/cdm:TEL/cdm:WORK
501 /cdm:vCard/cdm:TEL/cdm:VOICE
502 /cdm:vCard/cdm:TEL/cdm:FAX
503 /cdm:vCard/cdm:TEL/cdm:PAGER
504 /cdm:vCard/cdm:TEL/cdm:MSG
505 /cdm:vCard/cdm:TEL/cdm:CELL
506 /cdm:vCard/cdm:TEL/cdm:VIDEO
507 /cdm:vCard/cdm:TEL/cdm:BBS
508 /cdm:vCard/cdm:TEL/cdm:MODEM
509 /cdm:vCard/cdm:TEL/cdm:ISDN
510 /cdm:vCard/cdm:TEL/cdm:PCS
511 /cdm:vCard/cdm:TEL/cdm:PREF
512 /cdm:vCard/cdm:TEL/cdm:NUMBER
513 /cdm:vCard/cdm:EMAIL
514 /cdm:vCard/cdm:EMAIL/cdm:HOME
515 /cdm:vCard/cdm:EMAIL/cdm:WORK
516 /cdm:vCard/cdm:EMAIL/cdm:INTERNET
517 /cdm:vCard/cdm:EMAIL/cdm:PREF
518 /cdm:vCard/cdm:EMAIL/cdm:X400
519 /cdm:vCard/cdm:EMAIL/cdm:USERID
```

520 /cdm:vCard/cdm:JABBERID
521 /cdm:vCard/cdm:MAILER
522 /cdm:vCard/cdm:TZ
523 /cdm:vCard/cdm:GEO
524 /cdm:vCard/cdm:GEO/cdm:LAT
525 /cdm:vCard/cdm:GEO/cdm:LON
526 /cdm:vCard/cdm:TITLE
527 /cdm:vCard/cdm:ROLE
528 /cdm:vCard/cdm:LOGO
529 /cdm:vCard/cdm:LOGO/cdm:EXTVAL
530 /cdm:vCard/cdm:AGENT
531 /cdm:vCard/cdm:AGENT/cdm:EXTVAL
532 /cdm:vCard/cdm:ORG
533 /cdm:vCard/cdm:ORG/cdm:ORGNAME
534 /cdm:vCard/cdm:ORG/cdm:ORGUNIT
535 /cdm:vCard/cdm:CATEGORIES
536 /cdm:vCard/cdm:CATEGORIES/cdm:KEYWORD
537 /cdm:vCard/cdm:NOTE
538 /cdm:vCard/cdm:PROPID
539 /cdm:vCard/cdm:REV
540 /cdm:vCard/cdm:SORT-STRING
541 /cdm:vCard/cdm:SOUND
542 /cdm:vCard/cdm:SOUND/cdm:PHONETIC
543 /cdm:vCard/cdm:SOUND/cdm:EXTVAL
544 /cdm:vCard/cdm:UID
545 /cdm:vCard/cdm:URL
546 /cdm:vCard/cdm:CLASS
547 /cdm:vCard/cdm:CLASS/cdm:PUBLIC
548 /cdm:vCard/cdm:CLASS/cdm:PRIVATE
549 /cdm:vCard/cdm:CLASS/cdm:CONFIDENTIAL
550 /cdm:vCard/cdm:KEY
551 /cdm:vCard/cdm:KEY/cdm:TYPE
552 /cdm:vCard/cdm:KEY/cdm:CRED
553 /cdm:vCard/cdm:DESC
554 /cdm:vCard/cdm:PHYSICALACCESS

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

556 @cb:group
557 WORK
558 POSTAL
559 PARCEL
560 DOM
561 INTL
562 PREF
563 VOICE
564 FAX
565 PAGER
566 MSG
567 CELL
568 VIDEO
569 BBS
570 MODEM
571 ISDN
572 PCS
573 INTERNET
574 X400
575 CARDID
576 DISTRIBUTIONLIST
577 SELF
578 FAVORITE
579 LISTMEMBER

580 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.

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

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

613 3.4.2. Returning Entire Cards

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

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

617 3.4.3. Returning All Cards in the Contact Book

```
618 /cdm:vCard
```

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

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

625 3.4.4. Returning Some Attributes from All Cards

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

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

629 3.4.5. Latin 1 Case-Insensitive Matching

```
630 /cdm:vCard[contains(translate(/cdm:vCard/cdm:FN,  
631 "AÁÀÃÄÅÄÄBC...",  
632 "aáâãäåäâbc...")  
633 , "joão")]
```

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

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

638 3.4.6. Latin 1 Accent- and Case-Insensitive Matching

```
639 /cdm:vCard[contains(  
640     lowercase(  
641         stripaccents(/cdm:vCard/cdm:FN)  
642     )  
643     , "joao")  
644 ]
```

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

648 3.4.7. Free Text Search

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

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

653 3.4.8. Picking by Type

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

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

662 3.4.9. Picking by Group

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

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

671 3.4.10. Picking Types in Same Group

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

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

678 3.4.11. Picking Types of a Given Language

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

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

686 3.4.12. Querying CATEGORIES

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

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

689 3.4.13. Finding Distribution Lists

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

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

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

695 3.4.14. Finding Members of a Distribution List

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

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

699 3.5. Sorting Contact Cards

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

```
703  
704 <xs:complexType name="SortType" >  
705   <xs:sequence>  
706     <xs:element ref="By" maxOccurs="unbounded" />  
707   </xs:sequence>  
708   <!-- connects to DST framework -->  
709 </xs:complexType>  
710 <xs:element name="By" type="cb:ByType" />  
711 <xs:complexType name="ByType" mixed="true">  
712   <xs:attribute ref="sortAlg" />  
713   <xs:attribute ref="sortWeight" />  
714 </xs:complexType>  
715 <xs:attribute name="sortAlg">  
716   <xs:simpleType>  
717     <xs:restriction base="xs:string">  
718       <xs:enumeration value="asc" />  
719       <xs:enumeration value="desc" />  
720       <!-- Ascending order, e.g., alphabetic order -->  
721       <!-- Descending order, e.g., inverse alphabet -->  
722     </xs:restriction>  
723   </xs:simpleType>  
724 </xs:attribute>  
725 <xs:attribute name="sortWeight">  
726   <xs:simpleType>  
727     <xs:restriction base="xs:integer" />  
728   </xs:simpleType>  
729 </xs:attribute>
```

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

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

733 An implementation **MUST** support sorting by the following elements.

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

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

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

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

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

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

748 3.6. Modifying Contact Cards and Attributes

749 The modify operation is comprised of

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

753 Different modalities of operation are specified as follows

754 **Table 13. Flexible Modify Operation**

<i>Semantic</i>	<i>Cards</i>	<i>Attributes</i>	<i>vCard data</i>	<i>Notes</i>
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

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

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

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

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

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

773 3.6.1. Add New Cards

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

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

778 Example

```
779 <cb:Modify>
780   <cb:Modification>
781     <cb>Select/>
782     <cb>NewData>
783       <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
784         <cb:CharData>vCard:BEGIN
785         FN:Sampo Kellomaki
786         N:Kellomaki;Sampo
787         vCard:END
788         </cb:CharData>
789       </cb:Card>
790       <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
791         <cb:CharData>vCard:BEGIN
792         FN:Jukka Kainulainen
793         N:Kainulainen;Jukka
794         vCard:END
795         </cb:CharData>
796       </cb:Card>
797     </cb>NewData>
798   </cb:Modification>
799 </cb:Modify>
```

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

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

805 3.6.2. Replace Cards

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

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

814 Example

```

815 <cb:Modify>
816   <cb:Modification cb:overrideAllowed="1">
817     <cb:Select>/cdm:vCard
818       [/cdm:vCard/cdm:N/cdm:GIVEN="Sampo" ]
819   </cb:Select>
820   <cb:NewData>
821     <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
822       <cb:CharData>vCard:BEGIN
823       FN:Tapani Kellomaki
824       N:Kellomaki;Tapani
825       vCard:END
826     </cb:CharData>
827   </cb:Card>
828   <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
829     <cb:CharData>vCard:BEGIN
830     FN:Jukka Kainulainen
831     N:Kainulainen;Jukka
832     vCard:END
833   <cb:CharData>
834   </cb:Card>
835 </cb:NewData>
836 </cb:Modification>
837 </cb:Modify>
  
```

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

840 3.6.3. Delete Cards

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

843 Example

```

844 <cb:Modify>
845   <cb:Modification overrideAllowed="1">
846     <cb:Select>/cdm:vCard
847       [/cdm:vCard/cdm:N/cdm:GIVEN="Jukka" ]
848   </cb:Select>
849 </cb:Modification>
850 </cb:Modify>
  
```

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

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

856 3.6.4. Add Attributes to a Card

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

861 Example

```

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

```

866     </cb:Select>
867     <cb:NewData>
868         <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
869             <cb:CharData>vCard:BEGIN
870 NOTE:One new attribute
871 NOTE:Another new attribute
872 TITLE;language=fi:Jestelmrkki tehti
873 vCard:END
874             </cb:CharData>
875         </cb:Card>
876     </cb:NewData>
877 </cb:Modification>
878 </cb:Modify>

```

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

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

882 3.6.5. Replace Attributes in a Card

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

887 Example

```

888 <cb:Modify>
889 <cb:Modification>
890 <cb:Select>/cdm:vCard/cdm:NOTE
891 [ /cdm:vCard/cdm:N/cdm:GIVEN="Tapani" ]
892 </cb:Select>
893 <cb:NewData>
894 <cb:Card cb:format="urn:liberty:cb:format:RFC2426">
895 <cb:CharData>vCard:BEGIN
896 NOTE:Replacement note
897 TITLE;language=en:Systems Architect
898 vCard:END
899 </cb:CharData>
900 </cb:Card>
901 </cb:NewData>
902 </cb:Modification>
903 </cb:Modify>

```

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

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

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

910 3.6.6. Delete Attributes from a Card

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

914 Example

```

915 <cb:Modify>
916   <cb:Modification overrideAllowed="1">
917     <cb>Select> ( /cdm:vCard/cdm:TITLE
918               | /cdm:vCard/cdm:NOTE )
919               [ /cdm:vCard/cdm:N/cdm:GIVEN="Tapani" ]
920   </cb>Select>
921 </cb:Modification>
922 </cb:Modify>
  
```

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

926 3.6.7. Delete Specific Attributes from a Card

927 Here we illustrate a predicate that picks only some attributes

928 Example

```

929 <cb:Modify>
930   <cb:Modification overrideAllowed="1">
931     <cb>Select> /cdm:vCard/*
932           [ /cdm:vCard/cdm:N/cdm:GIVEN="Tapani"
933             and lang("fi")
934             and @cb:group = /cdm:vCard/cb:TITLE/@cb:group ]
935   </cb>Select>
936 </cb:Modification>
937 </cb:Modify>
  
```

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

941 3.7. Reporting Usage of a Contact Card

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

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

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

```

952
953 <xs:element name="ReportUsage" type="cb:ReportUsageType"/>
954 <xs:complexType name="ReportUsageType">
955   <xs:sequence>
956     <xs:group ref="ResourceIDGroup" minOccurs="0"/>
957     <xs:element ref="cdm:CARDID"/>
958     <xs:element ref="cb:UsageType" minOccurs="0" maxOccurs="unbounded"/>
959   </xs:sequence>
960   <xs:attribute name="id" type="xs:ID"/>
961 </xs:complexType>
962 <xs:element name="UsageType" type="UsageTypeType"/>
963 <xs:complexType name="UsageTypeType">
964   <xs:simpleContent>
  
```

```
965         <xs:extension base="cb:DSTURI">
966             <xs:attribute name="success" type="xs:boolean" default="1"/>
967         </xs:extension>
968     </xs:simpleContent>
969 </xs:complexType>
970 <xs:element name="ReportUsageResponse" type="ReportUsageResponseType"/>
971 <xs:complexType name="ReportUsageResponseType">
972     <xs:sequence>
973         <xs:element ref="Status"/>
974     </xs:sequence>
975     <xs:attribute name="id" type="xs:ID"/>
976 </xs:complexType>
```

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

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

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

984 3.8. Manipulating Distribution lists

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

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

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

993 To change distribution membership, perform a modification to the LISTMEMBER elements of the card that is joining or
994 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

1032 If an attribute is requested and would be available except for permissions, the ID-SIS-CB Provider MUST generate an
1033 error response with code `cb:ActionNotAuthorized`.

1034 **4.7. Multiple Modification Is Restricted to One Card**

1035 An implementation that supports `modify` operations MUST support multiple `Modification` elements and MUST
1036 NOT allow partial modifies. However, in interest of simplification, multiple `Modification` elements within the
1037 same `Modify` element MUST contain `Select` elements that refer to the same contact card.

1038 When multiple `Modification` elements are present, they MUST be processed in the order that they appear in the
1039 XML document. For example, if one `Modification` adds an attribute, but a later `Modification` deletes it, the net
1040 effect is as if the attribute had never been added.

1041 **4.8. Simplifications to Distribution Lists**

1042 Distribution lists MUST NOT be directly or indirectly recursive.

1043 An implementation MUST be able to cope with a distribution list place holder card being a member of another
1044 distribution list, but MAY ignore such instances. A WSC SHOULD NOT depend on being able to query or manipulate
1045 distribution lists as members of other distribution lists.

1046 **4.9. Use of Usage Directives**

1047 Typical use of ID-SIS-CB involves WSC querying the Contact Book for a contact card for a particular use. The
1048 ID-SIS-CB provider MAY invoke the interaction service [[LibertyInteract11](#)] to ask the Principal to choose from many
1049 cards. The Principal will need to know the purpose of the request to make an intelligent choice. Knowing the identity
1050 of the WSC may not always be sufficient to understand probable use of the data.

1051 When querying, a WSC MUST include usage directives that indicate the reason for the request and the type of use the
1052 data will be put. If the ID-SIS-CB provider invokes the service [[LibertyInteract11](#)] service, it SHOULD display this
1053 information, usually after localization, to the Principal.

1054 The actual format of the usage directives MUST be agreed out-of-band (e.g., in business agreement between WSC and
1055 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.

1093 8. If type has other parameters, they are mapped to corresponding elements (e.g., HOME, WORK) contained within the
1094 main element.

1095 9. All elements, except the extended elements, generated by the mapping MUST be in the `cdm` namespace.

1096 **5.3. Mapping vCard 3.0**

1097 The same [vCard21] mapping is used except that type parameters are mapped differently. It should be noted that the
1098 generic to [vCard21] mapping observes all the restrictions required to generate a valid vCard 3.0 ([RFC2426])

1099 Instead of rule 5.1.7, the elements representing types are mapped to vCard 3.0 ([RFC2426]) type parameter TYPE such
1100 that the value of the parameter corresponds to the name of the element.

1101 Instead of rule 5.2.8, the vCard 3.0 [RFC2426]) type parameters whose name is TYPE are mapped to corresponding
1102 XML elements such that the value of the parameter is upper-case and used as the element name.

1103 6. Mapping vCard Jabber

1104 6.1. Pure Jabber

1105 This mapping is indicated by specifying `cb:format` XML attribute with value `"urn:liberty:cb:format:vcard-temp."`

1106 The content shall be formatted as specified in [JEP0054]. Since the contact book conceptual data model is
1107 derived from [JEP0054], the mapping is almost one-to-one (i.e., null mapping) with the only exception being that
1108 the implementation MAY remove `cb:group` and `xml:lang` XML attributes.

1109 6.2. Generic vCard

1110 This mapping is indicated by specifying `cb:format` XML attribute with value `"urn:liberty:id-sis-cb:2005-05."`

1111 This mapping is a null mapping (i.e., the conceptual data model is directly used on the wire as well). This mapping
1112 differs from the Jabber mapping in that `cb:group` and `xml:lang` XML attributes MUST NOT be removed.

1113 7. Mapping vCard RDF

- 1114 This mapping is indicated by specifying `cb:format XML` attribute with value "`http://www.w3.org/2001/vcard-rdf/3.0#`."
1115 The content shall be formatted as specified in [\[vCardRDF\]](#).
- 1116 Each generic element is mapped "as-is," except that the `vCard:group` XML attribute MAY be ignored or MAY
1117 be mapped to an RDF bag construct. Subelements of `N`, `ADR`, and `ORG` are converted to the [\[vCardRDF\]](#) naming
1118 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).

```

1125 <?xml version="1.0" encoding="UTF-8"?>
1126 <xs:schema
1127     xmlns="urn:liberty:id-sis-cb:2005-05"
1128     xmlns:cb="urn:liberty:id-sis-cb:2005-05"
1129     xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"
1130     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1131     targetNamespace="urn:liberty:id-sis-cb:2005-05"
1132     elementFormDefault="qualified">
1133   <xs:import
1134     namespace="urn:liberty:cb:conceptual-data-model:2005-05"
1135     schemaLocation="liberty-id-sis-cb-cdm-v1.0.xsd"/>
1136   <xs:include schemaLocation="liberty-idwsf-dst-v2.0.xsd"/>
1137   <xs:include schemaLocation="liberty-idwsf-dst-dt-v2.0.xsd"/>
1138   <xs:element name="Card" type="cb:cardType"/>
1139   <xs:complexType name="cardType">
1140     <xs:choice>
1141       <xs:element ref="cb:charData"/>
1142       <xs:any namespace="##other" processContents="lax"/>
1143     </xs:choice>
1144     <xs:attribute ref="cb:format" use="required"/>
1145   </xs:complexType>
1146   <xs:attribute name="format" type="xs:anyURI"/>
1147   <xs:element name="charData" type="cb:charDataType"/>
1148   <xs:simpleType name="charDataType">
1149     <xs:restriction base="xs:string"/> <!-- vCard data in specified format -->
1150   </xs:simpleType>
1151   <xs:complexType name="SelectType"> <!-- connects to DST framework -->
1152     <xs:simpleContent>
1153       <xs:extension base="xs:string"> <!-- XPath expression -->
1154         <xs:attribute ref="cb:format"/>
1155       </xs:extension>
1156     </xs:simpleContent>
1157   </xs:complexType>
1158   <xs:complexType name="SortType">
1159     <xs:sequence>
1160       <xs:element ref="By" maxOccurs="unbounded"/>
1161     </xs:sequence>
1162     <!-- connects to DST framework -->
1163   </xs:complexType>
1164   <xs:element name="By" type="cb:ByType"/>
1165   <xs:complexType name="ByType" mixed="true">
1166     <xs:attribute ref="sortAlg"/>
1167     <xs:attribute ref="sortWeight"/>
1168   </xs:complexType>
1169   <xs:attribute name="sortAlg">
1170     <xs:simpleType>
1171       <xs:restriction base="xs:string">
1172         <xs:enumeration value="asc"/>
1173         <xs:enumeration value="desc"/>
1174         <!-- Ascending order, e.g., alphabetic order -->
1175         <!-- Descending order, e.g., inverse alphabet -->
1176       </xs:restriction>
1177     </xs:simpleType>
1178   </xs:attribute>
1179   <xs:attribute name="sortWeight">

```

```

1180     <xs:simpleType>
1181       <xs:restriction base="xs:integer" />
1182     </xs:simpleType>
1183   </xs:attribute>
1184   <xs:attributeGroup name="typeAttributes">
1185     <xs:attributeGroup ref="localizedLeafAttributes" />
1186     <xs:attribute ref="group" />
1187   </xs:attributeGroup>
1188   <xs:attribute name="group">
1189     <xs:simpleType>
1190       <xs:restriction base="xs:string" />
1191     </xs:simpleType>
1192   </xs:attribute>
1193   <xs:element name="ReportUsage" type="cb:ReportUsageType" />
1194   <xs:complexType name="ReportUsageType">
1195     <xs:sequence>
1196       <xs:group ref="ResourceIDGroup" minOccurs="0" />
1197       <xs:element ref="cdm:CARDID" />
1198       <xs:element ref="cb:UsageType" minOccurs="0" maxOccurs="unbounded" />
1199     </xs:sequence>
1200     <xs:attribute name="id" type="xs:ID" />
1201   </xs:complexType>
1202   <xs:element name="UsageType" type="UsageTypeType" />
1203   <xs:complexType name="UsageTypeType">
1204     <xs:simpleContent>
1205       <xs:extension base="cb:DSTURI">
1206         <xs:attribute name="success" type="xs:boolean" default="1" />
1207       </xs:extension>
1208     </xs:simpleContent>
1209   </xs:complexType>
1210   <xs:element name="ReportUsageResponse" type="ReportUsageResponseType" />
1211   <xs:complexType name="ReportUsageResponseType">
1212     <xs:sequence>
1213       <xs:element ref="Status" />
1214     </xs:sequence>
1215     <xs:attribute name="id" type="xs:ID" />
1216   </xs:complexType>
1217   <xs:complexType name="TypeType">
1218     <xs:complexContent>
1219       <xs:restriction base="EmptyType" />
1220     </xs:complexContent>
1221   </xs:complexType>
1222   <xs:complexType name="TriggerType">
1223     <xs:complexContent>
1224       <xs:restriction base="EmptyType" />
1225     </xs:complexContent>
1226   </xs:complexType>
1227 </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].

```

1236 <?xml version="1.0" encoding="UTF-8"?>
1237 <xs:schema
1238   xmlns="urn:liberty:cb:conceptual-data-model:2005-05"
1239   xmlns:cdm="urn:liberty:cb:conceptual-data-model:2005-05"

```

```
1240     xmlns:cb="urn:liberty:id-sis-cb:2005-05"
1241     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1242     targetNamespace="urn:liberty:cb:conceptual-data-model:2005-05">
1243 <xs:import
1244     namespace="urn:liberty:id-sis-cb:2005-05"
1245     schemaLocation="liberty-id-sis-cb-proto-v1.0.xsd"/>
1246 <xs:element name="vCard">
1247   <xs:complexType>
1248     <xs:sequence>
1249       <xs:sequence>
1250         <xs:element ref="cdm:VERSION" />
1251         <xs:element ref="cdm:CARDID" />
1252         <xs:element ref="cdm:DISTRIBUTIONLIST" minOccurs="0"/>
1253         <xs:element ref="cdm:SELF" minOccurs="0"/>
1254         <xs:element ref="cdm:FAVORITE" minOccurs="0"/>
1255         <xs:element ref="cdm:FN" maxOccurs="unbounded"/>
1256         <xs:element ref="cdm:N" maxOccurs="unbounded"/>
1257         <xs:element ref="cdm:LISTMEMBER" minOccurs="0" maxOccurs="unbounded"/>
1258       </xs:sequence>
1259       <xs:sequence minOccurs="0" maxOccurs="unbounded">
1260         <xs:element ref="cdm:NICKNAME" minOccurs="0" />
1261         <xs:element ref="cdm:PHOTO" minOccurs="0" />
1262         <xs:element ref="cdm:BDAY" minOccurs="0" />
1263         <xs:element ref="cdm:ADR" minOccurs="0" />
1264         <xs:element ref="cdm:LABEL" minOccurs="0" />
1265         <xs:element ref="cdm:TEL" minOccurs="0" />
1266         <xs:element ref="cdm:EMAIL" minOccurs="0" />
1267         <xs:element ref="cdm:JABBERID" minOccurs="0" />
1268         <xs:element ref="cdm:MAILER" minOccurs="0" />
1269         <xs:element ref="cdm:TZ" minOccurs="0" />
1270         <xs:element ref="cdm:GEO" minOccurs="0" />
1271         <xs:element ref="cdm:TITLE" minOccurs="0" />
1272         <xs:element ref="cdm:ROLE" minOccurs="0" />
1273         <xs:element ref="cdm:LOGO" minOccurs="0" />
1274         <xs:element ref="cdm:AGENT" minOccurs="0" />
1275         <xs:element ref="cdm:ORG" minOccurs="0" />
1276         <xs:element ref="cdm:CATEGORIES" minOccurs="0" />
1277         <xs:element ref="cdm:NOTE" minOccurs="0" />
1278         <xs:element ref="cdm:PROPID" minOccurs="0" />
1279         <xs:element ref="cdm:REV" minOccurs="0" />
1280         <xs:element ref="cdm:SORT-STRING" minOccurs="0" />
1281         <xs:element ref="cdm:SOUND" minOccurs="0" />
1282         <xs:element ref="cdm:UID" minOccurs="0" />
1283         <xs:element ref="cdm:URL" minOccurs="0" />
1284         <xs:element ref="cdm:CLASS" minOccurs="0" />
1285         <xs:element ref="cdm:KEY" minOccurs="0" />
1286         <xs:element ref="cdm:DESC" minOccurs="0" />
1287         <xs:element ref="cdm:PHYSICALACCESS" minOccurs="0" />
1288         <xs:element ref="cdm:CALURI" minOccurs="0" />
1289         <xs:element ref="cdm:CAPURI" minOccurs="0" />
1290         <xs:element ref="cdm:CALADRURI" minOccurs="0" />
1291         <xs:element ref="cdm:FBURL" minOccurs="0" />
1292         <xs:element ref="cdm:Extension" minOccurs="0" />
1293       </xs:sequence>
1294     </xs:sequence>
1295     <xs:attributeGroup ref="cb:commonAttributes"/>
1296   </xs:complexType>
1297 </xs:element>
1298 <xs:element name="VERSION">
1299   <xs:complexType mixed="true">
1300     <xs:attributeGroup ref="cb:leafAttributes"/>
1301   </xs:complexType>
1302 </xs:element>
1303 <xs:element name="FN">
1304   <xs:complexType mixed="true">
1305     <xs:attributeGroup ref="cb:typeAttributes"/>
1306   </xs:complexType>
```

```
1307     </xs:element>
1308     <xs:element name="N">
1309       <xs:complexType>
1310         <xs:sequence>
1311           <xs:element ref="cdm:FAMILY" minOccurs="0" />
1312           <xs:element ref="cdm:GIVEN" minOccurs="0" />
1313           <xs:element ref="cdm:MIDDLE" minOccurs="0" />
1314           <xs:element ref="cdm:PREFIX" minOccurs="0" />
1315           <xs:element ref="cdm:SUFFIX" minOccurs="0" />
1316         </xs:sequence>
1317         <xs:attributeGroup ref="cb:commonAttributes" />
1318       </xs:complexType>
1319     </xs:element>
1320     <xs:element name="FAMILY">
1321       <xs:complexType mixed="true">
1322         <xs:attributeGroup ref="cb:typeAttributes" />
1323       </xs:complexType>
1324     </xs:element>
1325     <xs:element name="GIVEN">
1326       <xs:complexType mixed="true">
1327         <xs:attributeGroup ref="cb:typeAttributes" />
1328       </xs:complexType>
1329     </xs:element>
1330     <xs:element name="MIDDLE">
1331       <xs:complexType mixed="true">
1332         <xs:attributeGroup ref="cb:typeAttributes" />
1333       </xs:complexType>
1334     </xs:element>
1335     <xs:element name="PREFIX">
1336       <xs:complexType mixed="true">
1337         <xs:attributeGroup ref="cb:typeAttributes" />
1338       </xs:complexType>
1339     </xs:element>
1340     <xs:element name="SUFFIX">
1341       <xs:complexType mixed="true">
1342         <xs:attributeGroup ref="cb:typeAttributes" />
1343       </xs:complexType>
1344     </xs:element>
1345     <xs:element name="NICKNAME">
1346       <xs:complexType mixed="true">
1347         <xs:attributeGroup ref="cb:typeAttributes" />
1348       </xs:complexType>
1349     </xs:element>
1350     <xs:element name="PHOTO">
1351       <xs:complexType>
1352         <xs:choice>
1353           <xs:sequence>
1354             <xs:element ref="cdm:TYPE" />
1355             <xs:element ref="cdm:BINVAL" />
1356           </xs:sequence>
1357           <xs:element ref="cdm:EXTVAL" />
1358         </xs:choice>
1359         <xs:attributeGroup ref="cb:commonAttributes" />
1360       </xs:complexType>
1361     </xs:element>
1362     <xs:element name="BDAY">
1363       <xs:complexType mixed="true">
1364         <xs:attributeGroup ref="cb:leafAttributes" />
1365       </xs:complexType>
1366     </xs:element>
1367     <xs:element name="ADR">
1368       <xs:complexType>
1369         <xs:sequence>
1370           <xs:element ref="cdm:HOME" minOccurs="0" />
1371           <xs:element ref="cdm:WORK" minOccurs="0" />
1372           <xs:element ref="cdm:POSTAL" minOccurs="0" />
1373           <xs:element ref="cdm:PARCEL" minOccurs="0" />

```

```
1374         <xs:choice minOccurs="0">
1375             <xs:element ref="cdm:DOM" />
1376             <xs:element ref="cdm:INTL" />
1377         </xs:choice>
1378         <xs:element ref="cdm:PREF" minOccurs="0" />
1379         <xs:element ref="cdm:POBOX" minOccurs="0" />
1380         <xs:element ref="cdm:EXTADR" minOccurs="0" />
1381         <xs:element ref="cdm:STREET" minOccurs="0" />
1382         <xs:element ref="cdm:LOCALITY" minOccurs="0" />
1383         <xs:element ref="cdm:REGION" minOccurs="0" />
1384         <xs:element ref="cdm:PCODE" minOccurs="0" />
1385         <xs:element ref="cdm:CTRY" minOccurs="0" />
1386     </xs:sequence>
1387     <xs:attributeGroup ref="cb:commonAttributes" />
1388 </xs:complexType>
1389 </xs:element>
1390 <xs:element name="POBOX">
1391     <xs:complexType mixed="true">
1392         <xs:attributeGroup ref="cb:typeAttributes" />
1393     </xs:complexType>
1394 </xs:element>
1395 <xs:element name="EXTADR">
1396     <xs:complexType mixed="true">
1397         <xs:attributeGroup ref="cb:typeAttributes" />
1398     </xs:complexType>
1399 </xs:element>
1400 <xs:element name="STREET">
1401     <xs:complexType mixed="true">
1402         <xs:attributeGroup ref="cb:typeAttributes" />
1403     </xs:complexType>
1404 </xs:element>
1405 <xs:element name="LOCALITY">
1406     <xs:complexType mixed="true">
1407         <xs:attributeGroup ref="cb:typeAttributes" />
1408     </xs:complexType>
1409 </xs:element>
1410 <xs:element name="REGION">
1411     <xs:complexType mixed="true">
1412         <xs:attributeGroup ref="cb:typeAttributes" />
1413     </xs:complexType>
1414 </xs:element>
1415 <xs:element name="PCODE">
1416     <xs:complexType mixed="true">
1417         <xs:attributeGroup ref="cb:typeAttributes" />
1418     </xs:complexType>
1419 </xs:element>
1420 <xs:element name="CTRY">
1421     <xs:complexType mixed="true">
1422         <xs:attributeGroup ref="cb:typeAttributes" />
1423     </xs:complexType>
1424 </xs:element>
1425 <xs:element name="LABEL">
1426     <xs:complexType>
1427         <xs:sequence>
1428             <xs:element ref="cdm:HOME" minOccurs="0" />
1429             <xs:element ref="cdm:WORK" minOccurs="0" />
1430             <xs:element ref="cdm:POSTAL" minOccurs="0" />
1431             <xs:element ref="cdm:PARCEL" minOccurs="0" />
1432             <xs:choice minOccurs="0">
1433                 <xs:element ref="cdm:DOM" />
1434                 <xs:element ref="cdm:INTL" />
1435             </xs:choice>
1436             <xs:element ref="cdm:PREF" minOccurs="0" />
1437             <xs:element ref="cdm:LINE" maxOccurs="unbounded" />
1438         </xs:sequence>
1439         <xs:attributeGroup ref="cb:commonAttributes" />
1440     </xs:complexType>
```



```
1441     </xs:element>
1442     <xs:element name="LINE">
1443       <xs:complexType mixed="true">
1444         <xs:attributeGroup ref="cb:typeAttributes"/>
1445       </xs:complexType>
1446     </xs:element>
1447     <xs:element name="TEL">
1448       <xs:complexType>
1449         <xs:sequence>
1450           <xs:element ref="cdm:HOME" minOccurs="0"/>
1451           <xs:element ref="cdm:WORK" minOccurs="0"/>
1452           <xs:element ref="cdm:VOICE" minOccurs="0"/>
1453           <xs:element ref="cdm:FAX" minOccurs="0"/>
1454           <xs:element ref="cdm:PAGER" minOccurs="0"/>
1455           <xs:element ref="cdm:MSG" minOccurs="0"/>
1456           <xs:element ref="cdm:CELL" minOccurs="0"/>
1457           <xs:element ref="cdm:VIDEO" minOccurs="0"/>
1458           <xs:element ref="cdm:BBS" minOccurs="0"/>
1459           <xs:element ref="cdm:MODEM" minOccurs="0"/>
1460           <xs:element ref="cdm:ISDN" minOccurs="0"/>
1461           <xs:element ref="cdm:PCS" minOccurs="0"/>
1462           <xs:element ref="cdm:PREF" minOccurs="0"/>
1463           <xs:element ref="cdm:NUMBER"/>
1464         </xs:sequence>
1465         <xs:attributeGroup ref="cb:commonAttributes"/>
1466       </xs:complexType>
1467     </xs:element>
1468     <xs:element name="NUMBER">
1469       <xs:complexType mixed="true">
1470         <xs:attributeGroup ref="cb:leafAttributes"/>
1471       </xs:complexType>
1472     </xs:element>
1473     <xs:element name="EMAIL">
1474       <xs:complexType>
1475         <xs:sequence>
1476           <xs:element ref="cdm:HOME" minOccurs="0"/>
1477           <xs:element ref="cdm:WORK" minOccurs="0"/>
1478           <xs:element ref="cdm:INTERNET" minOccurs="0"/>
1479           <xs:element ref="cdm:PREF" minOccurs="0"/>
1480           <xs:element ref="cdm:X400" minOccurs="0"/>
1481           <xs:element ref="cdm:USERID"/>
1482         </xs:sequence>
1483         <xs:attributeGroup ref="cb:commonAttributes"/>
1484       </xs:complexType>
1485     </xs:element>
1486     <xs:element name="USERID">
1487       <xs:complexType mixed="true">
1488         <xs:attributeGroup ref="cb:leafAttributes"/>
1489       </xs:complexType>
1490     </xs:element>
1491     <xs:element name="JABBERID">
1492       <xs:complexType mixed="true">
1493         <xs:attributeGroup ref="cb:leafAttributes"/>
1494       </xs:complexType>
1495     </xs:element>
1496     <xs:element name="MAILER">
1497       <xs:complexType mixed="true">
1498         <xs:attributeGroup ref="cb:leafAttributes"/>
1499       </xs:complexType>
1500     </xs:element>
1501     <xs:element name="TZ">
1502       <xs:complexType mixed="true">
1503         <xs:attributeGroup ref="cb:leafAttributes"/>
1504       </xs:complexType>
1505     </xs:element>
1506     <xs:element name="GEO">
1507       <xs:complexType>
```

```

1508         <xs:sequence>
1509             <xs:element ref="cdm:LAT"/>
1510             <xs:element ref="cdm:LON"/>
1511         </xs:sequence>
1512         <xs:attributeGroup ref="cb:commonAttributes"/>
1513     </xs:complexType>
1514 </xs:element>
1515 <xs:element name="LAT">
1516     <xs:complexType mixed="true">
1517         <xs:attributeGroup ref="cb:leafAttributes"/>
1518     </xs:complexType>
1519 </xs:element>
1520 <xs:element name="LON">
1521     <xs:complexType mixed="true">
1522         <xs:attributeGroup ref="cb:leafAttributes"/>
1523     </xs:complexType>
1524 </xs:element>
1525 <xs:element name="TITLE">
1526     <xs:complexType mixed="true">
1527         <xs:attributeGroup ref="cb:typeAttributes"/>
1528     </xs:complexType>
1529 </xs:element>
1530 <xs:element name="ROLE">
1531     <xs:complexType mixed="true">
1532         <xs:attributeGroup ref="cb:typeAttributes"/>
1533     </xs:complexType>
1534 </xs:element>
1535 <xs:element name="LOGO">
1536     <xs:complexType>
1537         <xs:choice>
1538             <xs:sequence>
1539                 <xs:element ref="cdm:TYPE"/>
1540                 <xs:element ref="cdm:BINVAL"/>
1541             </xs:sequence>
1542             <xs:element ref="cdm:EXTVAL"/>
1543         </xs:choice>
1544         <xs:attributeGroup ref="cb:commonAttributes"/>
1545     </xs:complexType>
1546 </xs:element>
1547 <xs:element name="AGENT">
1548     <xs:complexType>
1549         <xs:choice>
1550             <xs:element ref="cdm:vCard"/>
1551             <xs:element ref="cdm:EXTVAL"/>
1552         </xs:choice>
1553         <xs:attributeGroup ref="cb:commonAttributes"/>
1554     </xs:complexType>
1555 </xs:element>
1556 <xs:element name="ORG">
1557     <xs:complexType>
1558         <xs:sequence>
1559             <xs:element ref="cdm:ORGNAME"/>
1560             <xs:element ref="cdm:ORGUNIT" minOccurs="0" maxOccurs="unbounded"/>
1561         </xs:sequence>
1562         <xs:attributeGroup ref="cb:commonAttributes"/>
1563     </xs:complexType>
1564 </xs:element>
1565 <xs:element name="ORGNAME">
1566     <xs:complexType mixed="true">
1567         <xs:attributeGroup ref="cb:typeAttributes"/>
1568     </xs:complexType>
1569 </xs:element>
1570 <xs:element name="ORGUNIT">
1571     <xs:complexType mixed="true">
1572         <xs:attributeGroup ref="cb:typeAttributes"/>
1573     </xs:complexType>
1574 </xs:element>

```

```
1575 <xs:element name="CATEGORIES">
1576   <xs:complexType>
1577     <xs:sequence>
1578       <xs:element ref="cdm:KEYWORD" maxOccurs="unbounded" />
1579     </xs:sequence>
1580     <xs:attributeGroup ref="cb:commonAttributes" />
1581   </xs:complexType>
1582 </xs:element>
1583 <xs:element name="KEYWORD">
1584   <xs:complexType mixed="true">
1585     <xs:attributeGroup ref="cb:leafAttributes" />
1586   </xs:complexType>
1587 </xs:element>
1588 <xs:element name="NOTE">
1589   <xs:complexType mixed="true">
1590     <xs:attributeGroup ref="cb:typeAttributes" />
1591   </xs:complexType>
1592 </xs:element>
1593 <xs:element name="PRODID">
1594   <xs:complexType mixed="true">
1595     <xs:attributeGroup ref="cb:leafAttributes" />
1596   </xs:complexType>
1597 </xs:element>
1598 <xs:element name="REV">
1599   <xs:complexType mixed="true">
1600     <xs:attributeGroup ref="cb:leafAttributes" />
1601   </xs:complexType>
1602 </xs:element>
1603 <xs:element name="SORT-STRING">
1604   <xs:complexType mixed="true">
1605     <xs:attributeGroup ref="cb:typeAttributes" />
1606   </xs:complexType>
1607 </xs:element>
1608 <xs:element name="SOUND">
1609   <xs:complexType>
1610     <xs:choice>
1611       <xs:element ref="cdm:PHONETIC" />
1612       <xs:element ref="cdm:BINVAL" />
1613       <xs:element ref="cdm:EXTVAL" />
1614     </xs:choice>
1615     <xs:attributeGroup ref="cb:commonAttributes" />
1616   </xs:complexType>
1617 </xs:element>
1618 <xs:element name="PHONETIC">
1619   <xs:complexType mixed="true">
1620     <xs:attributeGroup ref="cb:typeAttributes" />
1621   </xs:complexType>
1622 </xs:element>
1623 <xs:element name="UID">
1624   <xs:complexType mixed="true">
1625     <xs:attributeGroup ref="cb:leafAttributes" />
1626   </xs:complexType>
1627 </xs:element>
1628 <xs:element name="URL">
1629   <xs:complexType mixed="true">
1630     <xs:attributeGroup ref="cb:leafAttributes" />
1631   </xs:complexType>
1632 </xs:element>
1633 <xs:element name="DESC">
1634   <xs:complexType mixed="true">
1635     <xs:attributeGroup ref="cb:typeAttributes" />
1636   </xs:complexType>
1637 </xs:element>
1638 <xs:element name="CLASS">
1639   <xs:complexType>
1640     <xs:choice>
1641       <xs:element ref="cdm:PUBLIC" />
```

```
1642         <xs:element ref="cdm:PRIVATE" />
1643         <xs:element ref="cdm:CONFIDENTIAL" />
1644     </xs:choice>
1645     <xs:attributeGroup ref="cb:leafAttributes" />
1646 </xs:complexType>
1647 </xs:element>
1648 <xs:element name="PUBLIC">
1649     <xs:complexType />
1650 </xs:element>
1651 <xs:element name="PRIVATE">
1652     <xs:complexType />
1653 </xs:element>
1654 <xs:element name="CONFIDENTIAL">
1655     <xs:complexType />
1656 </xs:element>
1657 <xs:element name="KEY">
1658     <xs:complexType>
1659         <xs:sequence>
1660             <xs:element ref="cdm:TYPE" minOccurs="0" />
1661             <xs:element ref="cdm:CRED" />
1662         </xs:sequence>
1663     </xs:complexType>
1664 </xs:element>
1665 <xs:element name="CRED">
1666     <xs:complexType mixed="true">
1667         <xs:attributeGroup ref="cb:leafAttributes" />
1668     </xs:complexType>
1669 </xs:element>
1670 <xs:element name="HOME">
1671     <xs:complexType />
1672 </xs:element>
1673 <xs:element name="WORK">
1674     <xs:complexType />
1675 </xs:element>
1676 <xs:element name="POSTAL">
1677     <xs:complexType />
1678 </xs:element>
1679 <xs:element name="PARCEL">
1680     <xs:complexType />
1681 </xs:element>
1682 <xs:element name="DOM">
1683     <xs:complexType />
1684 </xs:element>
1685 <xs:element name="INTL">
1686     <xs:complexType />
1687 </xs:element>
1688 <xs:element name="PREF">
1689     <xs:complexType />
1690 </xs:element>
1691 <xs:element name="VOICE">
1692     <xs:complexType />
1693 </xs:element>
1694 <xs:element name="FAX">
1695     <xs:complexType />
1696 </xs:element>
1697 <xs:element name="PAGER">
1698     <xs:complexType />
1699 </xs:element>
1700 <xs:element name="MSG">
1701     <xs:complexType />
1702 </xs:element>
1703 <xs:element name="CELL">
1704     <xs:complexType />
1705 </xs:element>
1706 <xs:element name="VIDEO">
1707     <xs:complexType />
1708 </xs:element>
```

```
1709 <xs:element name="BBS">
1710 <xs:complexType/>
1711 </xs:element>
1712 <xs:element name="MODEM">
1713 <xs:complexType/>
1714 </xs:element>
1715 <xs:element name="ISDN">
1716 <xs:complexType/>
1717 </xs:element>
1718 <xs:element name="PCS">
1719 <xs:complexType/>
1720 </xs:element>
1721 <xs:element name="INTERNET">
1722 <xs:complexType/>
1723 </xs:element>
1724 <xs:element name="X400">
1725 <xs:complexType/>
1726 </xs:element>
1727 <xs:element name="TYPE">
1728 <xs:complexType mixed="true">
1729 <xs:attributeGroup ref="cb:leafAttributes"/>
1730 </xs:complexType>
1731 </xs:element>
1732 <xs:element name="BINVAL">
1733 <xs:complexType mixed="true">
1734 <xs:attributeGroup ref="cb:leafAttributes"/>
1735 </xs:complexType>
1736 </xs:element>
1737 <xs:element name="EXTVAL">
1738 <xs:complexType mixed="true">
1739 <xs:attributeGroup ref="cb:leafAttributes"/>
1740 </xs:complexType>
1741 </xs:element>
1742 <xs:element name="PHYSICALACCESS">
1743 <xs:complexType mixed="true">
1744 <xs:attributeGroup ref="cb:typeAttributes"/>
1745 </xs:complexType>
1746 </xs:element>
1747 <xs:element name="SELF">
1748 <xs:complexType/>
1749 </xs:element>
1750 <xs:element name="FAVORITE">
1751 <xs:complexType/>
1752 </xs:element>
1753 <xs:element name="DISTRIBUTIONLIST">
1754 <xs:complexType/>
1755 </xs:element>
1756 <xs:element name="CALURI" type="cdm:QualifiedURIType"/>
1757 <xs:element name="CAPURI" type="cdm:QualifiedURIType"/>
1758 <xs:element name="CALADRURI" type="cdm:QualifiedURIType"/>
1759 <xs:element name="URI" type="cb:DSTURI"/>
1760 <xs:element name="FBURL" type="cdm:QualifiedURIType"/>
1761 <xs:complexType name="QualifiedURIType" mixed="true">
1762 <xs:sequence>
1763 <xs:element ref="cdm:PREF" minOccurs="0"/>
1764 <xs:element ref="cdm:URI"/>
1765 </xs:sequence>
1766 <xs:attributeGroup ref="cb:leafAttributes"/>
1767 </xs:complexType>
1768 <xs:element name="CARDID" type="xs:anyURI"/>
1769 <xs:element name="LISTMEMBER" type="cb:DSTURI"/>
1770 <xs:element name="Extension" type="cb:extensionType"/>
1771 </xs:schema>
```

9. Abstract WSDL for ID-SIS-CB

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

```
1772
1773
1774 <?xml version="1.0" encoding="UTF-8"?>
1775 <definitions
1776     xmlns:typens="urn:liberty:id-sis-cb:2005-05:wSDL:interface"
1777     xmlns="http://schemas.xmlsoap.org/wSDL/"
1778     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1779     xmlns:cb="urn:liberty:id-sis-cb:2005-05"
1780     targetNamespace="urn:liberty:id-sis-cb:2005-05:wSDL:interface"
1781     name="id-sis-cb_2005-05_interface">
1782   <types>
1783     <xsd:schema>
1784       <xsd:import
1785         namespace="urn:liberty:id-sis-cb:2005-05"
1786         schemaLocation="liberty-id-sis-cb-PROTO-v1.0.xsd"/>
1787     </xsd:schema>
1788   </types>
1789   <message name="Query">
1790     <part name="body" element="cb:Query"/>
1791   </message>
1792   <message name="QueryResponse">
1793     <part name="body" element="cb:QueryResponse"/>
1794   </message>
1795   <message name="Modify">
1796     <part name="body" element="cb:Modify"/>
1797   </message>
1798   <message name="ModifyResponse">
1799     <part name="body" element="cb:ModifyResponse"/>
1800   </message>
1801   <message name="Subscribe">
1802     <part name="body" element="cb:Subscribe"/>
1803   </message>
1804   <message name="SubscribeResponse">
1805     <part name="body" element="cb:SubscribeResponse"/>
1806   </message>
1807   <message name="QuerySubscriptions">
1808     <part name="body" element="cb:QuerySubscriptions"/>
1809   </message>
1810   <message name="Subscriptions">
1811     <part name="body" element="cb:Subscriptions"/>
1812   </message>
1813   <portType name="ContactbookPort">
1814     <operation name="ContactbookQuery">
1815       <input message="typens:Query"/>
1816       <output message="typens:QueryResponse"/>
1817     </operation>
1818     <operation name="ContactbookModify">
1819       <input message="typens:Modify"/>
1820       <output message="typens:ModifyResponse"/>
1821     </operation>
1822     <operation name="ContactbookSubscribe">
1823       <input message="typens:Subscribe"/>
1824       <output message="typens:SubscribeResponse"/>
1825     </operation>
1826     <operation name="ContactbookQuerySubscriptions">
1827       <input message="typens:QuerySubscriptions"/>
1828       <output message="typens:Subscriptions"/>
1829     </operation>
1830   </portType>
1831 </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

<i>vCard Attribute</i>	<i>PP attribute</i>	<i>Notes</i>
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)

<i>vCard Attribute</i>	<i>PP attribute</i>	<i>Notes</i>
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		
PROPID	?	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.

1851 The result of the conversion MUST conform to ID-SIS-PP [LibertyIDPP]. This, among other requirements, means
1852 that:

- 1853 • Any line-folding or escaping permitted or specified by vCard MUST be removed and XML conventions for
1854 escaping MUST be applied.
- 1855 • All values must be represented using the UTF-8 character set

1856 10.2. Mapping a Personal Profile to a vCard

1857 When mapping a PP ([LibertyIDPP]) to a vCard (either version), the correspondence specified in the table "Mapping
1858 vCard to PP" should be used. For those containers and elements that do not appear above, the extended vCard types,
1859 possibly qualified by parameters, specified in the table "Mapping PP to vCard Extensions," below, SHOULD be used.
1860 The mapping MAY also choose to omit the containers and elements that do not have correspondence in the vCard
1861 standard.

1862 **Table 16. Mapping PP to vCard Extensions**

<i>vCard Attribute</i>	<i>PP attribute</i>	<i>Notes</i>
?	Age	Can be calculated from BDay or DOB
?	Agegroup	Can be calculated from BDay or DOB
?	Birthday	Can be calculated from BDay or DOB

1863 Any extensions of the Personal Profile ([LibertyIDPP]) SHOULD be mapped to nonstandard vCard types whose names
1864 start by "x-" as described in [RFC2426], Section 3.8, unless the creator of the extension has registered an appropriate
1865 vCard type that has same semantics.

1866 Despite the destination vCard format possibly allowing free choice of character set, the vCard MUST be produced in
1867 the UTF-8 character set if it is to appear as a `cb:Card` element returned by an ID-SIS-CB service.

1868 The result of the conversion MUST be a valid vCard according to the version of vCard that is to be produced. Among
1869 other requirements, this means

- 1870 • any values MUST be escaped and line-folded according to the requirements of the vCard version that is to be
1871 produced, cf. [RFC2426] Sections 2.5 and 2.6.

1872 10.3. MsgType, MsgMethod, and MsgTechnology Mappings

1873 vCard EMAIL and TEL types are represented by PP ([LibertyIDPP]) `MsgContact` containers. vCard allows the
1874 specification of TYPE parameters to further qualify the contact. In the `MsgContact` container, the `MsgType`,
1875 `MsgMethod`, and `MsgTechnology` perform similar roles. In order to establish an interoperable basis, the following
1876 mappings MUST be used. If a mapping is not specified here, then the implementation MAY

- 1877 1. coerce to one of the defined mappings,
- 1878 2. register an extension mapping, or
- 1879 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**

<i>vCard Type</i>	<i>MsgType</i>	<i>MsgMethod</i>	<i>MsgTechnology</i>	<i>Notes</i>
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

<i>Ordinal</i>	<i>CB XML</i>	<i>PP XML</i>	<i>Notes</i>
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

<i>TYPE</i>	<i>AddrType</i>	<i>Notes</i>
(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.

1929 The group construct defined in [RFC2425], Section 5.8.2–penultimate paragraph, and in [vCard21], Section 2.1.4.2,
1930 SHOULD be used to group together types derived from the same AddressCard container. The group label MUST
1931 be picked such that it is unique across multiple AddressCard containers that may be present in one Personal Profile
1932 ([LibertyIDPP]). When mapping from vCard to Personal Profile ([LibertyIDPP]), the types with the same group
1933 MUST be mapped to elements within the same AddressCard container.

References

Normative

- 1934
- 1936 [DAWSON] Dawson, F., Hoffman, P., eds. (15 October, 1998). "The vCard v3.0 XML DTD,"
1937 expired draft-dawson-vcard-xml-dtd-01.txt, Internet Engineering Task Force/Watersprings.org
1938 <http://www.watersprings.org/pub/id/draft-dawson-vcard-xml-dtd-01.txt>
- 1939 [JEP0054] Saint-Andre, Peter, eds. (26 March, 2003). "Jabber Enhancement Proposal 0054: vcard-temp," v1.1, Jabber
1940 Software Foundation <http://www.jabber.org/jeps/jep-0054.html>
- 1941 [LibertyDisco12] Sergeant, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty
1942 Alliance Project (12 December 2004). <http://www.projectliberty.org/specs/>
- 1943 [LibertyDST20] Kainulainen, Jukka, Ranganathan, Aravindan, eds. "Liberty ID-WSF Data Services Template
1944 Specification," Version 2.0, Liberty Alliance Project (23 March, 2005). <http://www.projectliberty.org/specs>
- 1945 [LibertyIDEP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Employee Profile Service Specification,"
1946 Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 1947 [LibertyIDPP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Specification,"
1948 Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 1949 [LibertyInteract11] Aarts, Robert, eds. "Liberty ID-WSF Interaction Service Specification," Version 1.1, Liberty
1950 Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1951 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
1952 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1953 [LibertyReg] Kemp, John, eds. "Liberty Enumeration Registry Governance," Version 1.1, Liberty Alliance Project (14
1954 December, 2004). <http://www.projectliberty.org/specs>
- 1955 [LibertySOAPBinding12] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. "Liberty ID-WSF SOAP Binding Specifica-
1956 tion," Version 1.2, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs/>
- 1957 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
1958 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 1959 [RFC2425] Howes, T., Smith, M., Dawson, F., eds. (September 1998). "A MIME Content-Type for Directory
1960 Information," RFC 2425, The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2425.txt>
- 1961 [RFC2426] Dawson, F., Howes, T., eds. (September 1998). "vCard MIME Directory Profile," RFC 2426, The Internet
1962 Engineering Task Force <http://www.ietf.org/rfc/rfc2426.txt>
- 1963 [RFC2739] Small, T., Hennessy, D., Dawson, F., eds. (January 2000). "Calendar Attributes for vCard and LDAP,"
1964 RFC 2739, The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2739.txt>
- 1965 [Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May
1966 2001). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium
1967 <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>
- 1968 [vCard21] Consortium, Versit, eds. (18 September, 1996). "vCard, The Electronic Business Card," Version 2.1,
1969 Internet Mail Consortium <http://www.imc.org/pdi/vcard-21.txt>
- 1970 [vCardRDF] Iannella, Renato, eds. (22 February, 2001). "Representing vCard Objects in RDF/XML," W3C NOTE-
1971 vcard-rdf-20010222, World Wide Web Consortium <http://www.w3.org/TR/vcard-rdf>

1972 [XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve, Yergeau, Francois, eds. (04 February 2004).
1973 "Extensible Markup Language (XML) 1.0 (Third Edition)," Recommendation, World Wide Web Consortium
1974 <http://www.w3.org/TR/2004/REC-xml-20040204>

1975 [XPath] Clark, J., DeRose, S., eds. (16 November 1999). "XML Path Language (XPath) Version 1.0,"
1976 Recommendation, W3C <http://www.w3.org/TR/xpath> [August 2003].

1977 Informative

1978 [JReg] Software Foundation, Jabber, eds. (2004). "Jabber :: Registrar," , Jabber Software Foundation
1979 <http://www.jabber.org/registrar/>

1980 [LibertyCBGuide] Lambert, Guillaume, eds. "Liberty ID-SIS Contact Book Service Implementation Guidelines,"
1981 Version 1.0, Liberty Alliance Project (11 November, 2006). <http://www.projectliberty.org/specs>

1982 [LibertyIDEPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Employee Profile Ser-
1983 vice Implementation Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005).
1984 <http://www.projectliberty.org/specs>

1985 [LibertyIDPPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Implementation
1986 Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>

1987 [LibertyIDWSFGuide10] Weitzel, David, eds. (22 May 2005). "Liberty ID-WSF Implementation Guideline," Draft
1988 v1.0-12, Liberty Alliance Project <http://www.projectliberty.org/specs/>

1989 [LibertyIDWSFOverview11] Tourzan, Jonathan, Koga, Yuzo, eds. "Liberty ID-WSF Web Services Framework
1990 Overview," Version 1.1, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>