Internet Draft (SPCI)
Expires April 1994
Filename `draft-adie-spci-00.txt`

C. Adie
Edinburgh University
12 October, 1993

SGML-based Personal Contact Information (SPCI)

STATUS OF THIS DOCUMENT

SUMMARY

This document describes how personal contact information may be exchanged in a structured format suitable for machine processing. The SGML-based Personal Contact Information (SPCI) format can be used to encode personal information of the type which might be found on a business card, in a form which is also suitable for human interpretation. Possible uses of this format include: exchange of personal information in email messages; encoding author information within a document; providing information about a mailing list; as an exchange format for "address book" databases; and providing contact information for a company. The SPCI information is encoded using SGML, and the specification is aligned with the SHAVE rules.

A MIME body part type is defined for SPCI information.

Table of Contents

1. Introduction

This document defines the SGML-based Personal Contact Information (SPCI) format. The purpose of SPCI is as a standard for exchanging personal information of the type which might be found on a business card. It is envisaged that SPCI information will find a range of applications - some potential applications are listed below.

- The most common use of SPCI will likely be to provide information about the sender of an email message. SPCI information could be supplied in a `.signature` file, automatically appended to mail messages; or it could be sent as a separate MIME or X.400 body part.
- SPCI could be used within a document (such as this) to indicate the author's contact details.
- SPCI could be used as an exchange format for "address book" databases.
- SPCI could be used to convey information about a mailing list, rather than about an individual.
- SPCI could be used to convey information about how to contact a company, rather than an individual.

In most cases, it is envisaged that the SPCI information will be typed in by hand, and will be read both by eye and by machine.

SHAVE [Adie 93] defines a set of rules for using SGML to represent hierarchical attribute/value information in SGML. This SPCI specification adheres to those rules by definition.

An SGML document type definition (DTD) for SPCI is defined in this specification.

The remainder of this document is structured as follows. Chapter 2 is an informal tutorial description of the SPCI syntax. Chapter 3 defines the semantics of the SGML elements defined in the SPCI DTD. Chapter 4 gives information specific to the MIME registration of SPCI. The SPCI DTD itself is given in Appendix I.

2. Informal description

This chapter informally describes the syntax of SPCI. It does not assume any prior knowlege of SGML on the reader's part. The syntax is described through examples, without distinguishing between the features of SGML, SHAVE and SPCI. This chapter is not definitive.

Here is our first example:

```
<spci>
  <person key="Charles Babbage">
    <name>          Charles Babbage
    </name>
    <title>         Assistant Programmer
    </title>
    <org>           Calculating Machines plc
    </org>
    <email>         babbage@calc.co.uk
    </email>
  </person>
</spci>
```

An important SGML concept is the "element" - part of a document which is delimited by a "start tag" and an "end tag", and which may contain other elements and plain text. The main element in this example has a start tag of `<spci>` and a corresponding end tag `</spci>`. A program can use these particular tags to identify where the SPCI information starts and ends.

The `spci` element contains a `person` element with its own start and end tags (we'll ignore the `key="Charles Babbage"` part for now), which delimit the information about one particular individual. You can have more than one `person` element within an `spci` element.

The person element contains `name`, `title`, `org` (for organisation) and `email` elements, each of which contains plain text data. There are other elements which could also go in here, as we'll see.

The tags in the above example have been indented to make the structure clearer. You don't have to do this, and in subsequent examples we often won't bother. However, note that a tag can't occur in the middle or at the end of a line - it must occur at the start (*ie* before any other non-blank characters).

Having to specify the end tag of every element is unnecessarily verbose - the end of an element such as title can be indicated by the start tag of the next element. In fact, we can omit all the end tags except for the last:

```
<spci>
<person key="Horatio Nelson">
<name>          Horatio, Viscount Nelson
<title>         Vice Admiral of the Red
<org>           His Majesty's Navy
<email>         flag@victory.navy.mil.uk
<email X.400>   /CN=flag/OU=HMS Victory/O=Navy/P=MIL/A= /C=GB
<work>
<phone>         +44 31 234 5678
  <>            +44 826 123456
<motto>         England expects
                that every man this day
                will do his duty.
</spci>
```

Note the two occurrances of the `email` element above. The first contains an RFC822 address. The second `email` start tag is qualified by an "SGML attribute" indicating that the contents are an X.400 email address. The `key="Horatio Nelson"` in the person tag is an example of another SGML attribute. This `key` attribute provides a convenient key which could be used for searching and indexing purposes by an address book program, for instance.

The `motto` element shows how the value of a parameter may have multiple lines.

The `work` element contains a `phone` element. The `phone` element in this case represents a parameter with two values, which are separated by an empty tag <>.

Here's another example, showing some more elements, including an experimental element `X`, which in this case contains the name of the individual's Personal Assistant:

```
<spci>
 <person key="Olivia Peterson">
  <name>                          Olivia Peterson
  <email>                         op@clip.stormford.edu
  <title>                         Chairman &amp; Chief Executive
  <org>                           CLIP Inc
  <note>                          Out of the office &lt;16th
  <work>
   <phone>                        +1 415 123 4567
   <fax>                          +1 415 987 6543
   <postal>                       456 San Sebastian Avenue, Suite 4629
      <>                          Los Angeles, CA, US
      <>                          94040
   <X type="Personal Assistant">  John Smith
   </X>
  <home>
   <phone>                        +1 415 234 5678  <!-- After 7pm -->
   <fax>                          +1 415 765 4321
   <instruct>           Operator - please notify fax arrival to 234 5678.
  <mobile>
   <phone>                        +1 123 456 7890
</spci>
```

Comments are delimited by `<!--` and `-->`, and may occur anywhere on the line (outside of a tag). To represent the less than character "<" in the value of a parameter, you use the four-character string `&lt;`, and to represent the ampersand "&" you use `&amp;`. No other characters need escaping in SPCI.

Here is a highly artificial example, which shows almost all the tags and almost all the features of the SPCI format.

```
<spci>
<person key="John Jones">

<name>             Dr John J. Jones
<title>            Consulting Physicist
<dept>             External contracts department
<org>              High Energy Physics Clearances Inc
<email>            jjj@boson.hepi.com
<email X.400>      /I=J/S=Jones/O=HEPC/P=Fargle/A= /C=US
<note>             Ignore any error messages for the X.400 address
<X.500>            J. Jones, High Energy Physics Clearances Inc, US

<motto>            Let us clear those old, unwanted Feynman diagrams
                   out of your office.  Green's functions also accepted.

<work>
<phone>            +1 123 4567 8901
  <>               +1 123 4567 8902 <!-- Secretary -->
<fax>              +1 123 4567 8955
  <>               +1 123 4567 8956
<room>             2309
<building>         Higgs Boson Building
<street>           321 Main St
<city>             Los Angeles
<state>            CA
<postcode>         12345
<country>          USA
<telex>            456789012

<home>
<phone>            +44 81 765 4321
<fax>
<faxno>            +44 81 765 4321 <!-- Automatic fax/voice switch -->
<faxno>            +44 81 765 6789
<instruct>         Operator - please call 765 4321 and notify fax arrival
<postal>           12 Leafy Avenue
  <>               Croyden
  <>               LONDON
                   SW9 9XX
                   Great Britain
<courier>
<carrier>          TNT
<phone>            +44 81 765 4321

<mobile>
<phone>            +1 898 1234 5678
<pager>
<phone>            +1 123 7654 0987
```

```
<instruct>        Ask for page number 3456789
<fax>             +1 898 1234 5000 <!-- In the car -->
</spci>
```

Here is an example of a mailing list represented in SPCI:

```
<spci>
<person key="MMIS list">
<name>                  Multimedia Information Systems mailing list
<email list>            mmis@mailbase.ac.uk
<email listownr>        mmis-request@mailbase.ac.uk
<note>                  This is the mailing list of the RARE Multimedia
                        Information Systems Task Force.
</spci>
```

3. SPCI Semantics

The DTD for SPCI is given in Appendix I. The remainder of this chapter describes the semantics of the elements defined in the DTD. Extracts from the DTD are included in the text for ease of reference, but in the event of a disagreement between the text and the DTD in Appendix I, the latter takes priority.

The SPCI DTD conforms to the SHAVE rules for DTDs specified in [Adie 93]. SPCI documents must comply with the SHAVE rules for document instances also defined therein.

3.1 SPCI

```
<!ELEMENT spci    - -   (person*) >
```

All SPCI data is enclosed within an `spci` element, which must have both a start tag and an end tag. An SPCI element may contain zero, one or more `person` elements.

The purpose of the `spci` element is to delimit SPCI information within some larger text object such as a mail message or document. Note that if SPCI information is sent as a body part in a MIME message, the MIME message structure could be regarded as delimiting the SPCI information - however the `spci` start and end tags must still be present.

There may be more than one `spci` element within a text object such as a file or mail message.

3.2 Person

```
<!ELEMENT person  - O   (name? & email* & title? & org? & dept? &
                         motto? & note? & X.500? & work? &  home? &
                         mobile? & support? & picture?) +(X) >
```

The basic element of SPCI is the `person` element. A `person` element may contain zero or one of each of the following types of element, in any order: `name`, `X.500`, `title`, `org`, `dept`, `motto`, `note`, `work`, `home`, `mobile`, `support`, `picture`. It may contain any number of `email` elements which must all be adjacent. Experimental `X` elements may be present anywhere within a `person` element and its included elements.

The start tag of a `person` element is mandatory, but the end tag is not. The end of a `person` element is also indicated by a start tag of the next `person`, or by an `spci` end tag.

```
<!ATTLIST person key    CDATA #REQUIRED>
```

A `person` element has one (mandatory) SGML attribute, `key`. The key takes a value (enclosed in double quotes) which may be used for indexing purposes. The key should be the person's name, but omitting any prefixes (eg Dr, Professor) and suffixes (eg Jr, BSc), in order to facilitate sorting and searching. Note that an application such as an address book program may modify or ignore the key it reads from a received document. One reason for doing this might be that the application already has an entry in its database which uses that key.

The elements which may be contained in a `person` element are as follows:

`name`	Personal name of an individual, with prefixes and suffixes if desired.
`title`	The role of the individual within the organisation for which s/he works.
`org`	The name of the organisation for which the individual works.
`dept`	The department (or other sub-unit) of the organisation within which the individual works.
`motto`	Descriptive text associated with the individual, workgroup or organisation.
`note`	Information which does not fit elsewhere.
`email`	Electronic mail address of the individual. There may be more than one such element, but they must all be located together. See the following section.
`work`	See the section on Location-dependent Contact Information below.
`home`	See the section on Location-dependent Contact Information below.
`mobile`	See the section on Location-dependent Contact Information below.
`support`	A list of the supported MIME types, each type in the standard MIME format of the type/subtype, and each such pair separated by a list separator <>. Note that the list of MIME types may not be relevant for non-RFC822 email addresses.
`X.500`	The Directory Name of the individual, in UFN form (see RFCxxxx). If the program which reads the SPCI is capable of looking up the Directory, any attributes found therein override their SPCI equivalents.
`picture`	A photograph of an individual may make it easier to establish contact - eg when meeting someone at an airport. The contents of this element is a URL which points to an image of the individual. The element has a mandatory attribute `encoding`, which contains the MIME type and subtype of the image file. The MIME type will usually be "image".

3.3 EMail

```
<!ELEMENT email   - O   (#PCDATA) >
<!ATTLIST email
            mailsys     (X.400|RFC822)          RFC822
            mailtype    (person|list|listownr)  person
>
```

The `email` element can occur more than once in a `person` element. It contains an electronic mail address of some kind. The `email` element has two SGML attributes, which may both be defaulted.

The `mailsys` SGML attribute can take one of the following values: `X.400`, `RFC822`. If no value is specified, `RFC822` is assumed. If the value of `mailsys` is `X.400`, the contents of the element is an X.400 O/R Address, expressed in textual form according to CCITT recommendation F.4xx. If the value of `mailsys` is `RFC822`, the contents of the element is an email address as specified in RFC822.

The `mailtype` attribute can take one of the following values: `person`, `list`, `listownr`. If no value is specified, `person` is assumed. The meaning of these values is as follows:

`person`	The email address is the address of an individual.
`list`	The email address is the submission address of a mailing list.
`listownr`	The email address is the administration address for a mailing list (for subscription requests, address changes etc).

The latter two are intended for use when the enclosing `person` element describes a mailing list rather than an individual.

3.4 Location-dependant Contact Information

An email address is generally independant of the actual physical location at any particular instant of the individual being addressed - you could for instance read your email either from home or at work. However, this is not generally true of other forms of communication such as telephone, fax or postal services. Therefore, within a `person` element, there can be any combination of a `work` element, a `home` element and a `mobile` element, specifying contact information specific to an individual's location.

```
<!ELEMENT work    - O   (phone? & fax? & telex? & ((%physadr;),courier?)?) >
<!ELEMENT home    - O   (phone? & fax? & telex? & ((%physadr;),courier?)?) >
<!ELEMENT mobile  - O   (phone? & fax? & pager?) >
```

A `work` element may contain (a list of) phone numbers, (a list of) fax numbers and associated details, a telex address, and a physical address suitable for use by a postal service. Any combination of these components may be present, in any order. A `home` element may contain the same elements as `work`.. A `mobile` element may contain phone and fax information, and pager details as well.

```
<!ELEMENT phone   - O   (item+)>
```

The `phone` element may thus occur in several places. Its content is one or more voice phone numbers, conforming to CCITT recommendation F.103. Phone numbers are separated by the usual list separator characters <>, per the SHAVE rules. The order of phone numbers in the list is significant - the first number in the list should be tried first, and if it is engaged or there is no answer, subsequent numbers should be tried in order.

```
<!ELEMENT fax      - O  (faxno)+ >
<!ELEMENT faxno    O O  (#PCDATA,(grain? & instruct? & faxgate?)) >
<!ELEMENT instruct - O  (#PCDATA) >
<!ELEMENT grain    - O  (#PCDATA) >
<!ELEMENT faxgate  - O  (#PCDATA) >
```

The `fax` element is a bit more complex. It is a list of one or more `faxno` elements, each of which consists of `#PCDATA` (representing the fax number itself, confirming to CCITT F.103), followed by any combination of the following three elements:

`grain`	The grain of the fax machine - may have the values `regular` or `fine` only.
`instruct`	Handling instructions for facsimile operator.
`faxgate`	Internet address of gateway to the facsimile service.

The order of fax numbers in the list is significant - the first number in the list should be tried first, and if it is engaged or there is no answer, subsequent numbers should be tried in order.

```
<!ELEMENT telex   - O   (#PCDATA)>
```

A `telex` element contains the telex number to contact.

```
<!ELEMENT pager   - O   (phone, instruct?)>
```

A `pager` element must contain a `phone` element with the number to call to arrange for the individual to be paged. It may also contain an `instruct` element, containing further details of how to page.

3.5 Physical Address

```
<!ENTITY % physadr "postal|(room?,building?,street?,restante?,pobox?,
                         city?,ponumber?,state?,postcode?,country?)" >
```

A physical address could be used for delivery of a physical object such as a letter or parcel, by a postal or courier service. There are two ways in which a physical address can be encoded in SPCI. One method or the other must be selected - they cannot be mixed.

The first way uses the `postal` element. The contents of this is a list of address items, separated by the normal list separator <>. Each item in the list represents one line of address information which might appear on an envelope. The order of items in the list is significant - the first item in the list appears at the top of the address, and the last appears at the bottom. The last line will often (but not always) be the country name.

The second way of encoding a physical address uses individual elements for each component of the address. Each element may occur zero or once, and the elements must be in the following order:

`room`	The room within a building for delivery.

`building`	The name of the building.
`street`	Street address (name and number).
`restante`	Post restante address.
`pobox`	Post office box number.
`city`	Town or city.
`ponumber`	Post Office number.
`state`	District, region, province or state.
`postcode`	Postal or ZIP code.
`country`	Country name (in full - ISO3166 abbreviations are not normally acceptable except where they are in very common use - *eg* "USA").

Following a physical address (in either form), there may be a `courier` element, specifying a preferred courier service for delivery to the address.

```
<!ELEMENT courier - O   (carrier, account?, delivery?, phone?) >
```

The `courier` element may contain the following elements, in the order given below.

`carrier`	Name of the courier organisation. Mandatory in a `courier` element.
`account`	Account reference number.
`delivery`	Copied from PCI. What is this for?
`phone`	Phone number of recipient (many courier companies require this information).

4. MIME Registration

The MIME registration for SPCI will be as follows:

MIME type name:	TEXT
MIME subtype name:	SPCI
Required parameters:	none
Optional parameters:	none
Encoding considerations:	Some characters may not be sent directly through some transport services. Where possible, it is recommended that quoted-printable encoding be used, to preserve readability.
Security considerations:	See separate section.
Published specification:	A TEXT/SPCI body part will comprise exactly one `spci` element as defined in this specification.
Rationale:	Provides human-readable and machine-processable method of encoding personal contact information. This will permit automated processing, such as for maintaining personal files of contact information.
Contact information:	See separate section.

5. References

[Adie 93]	*SGML-based Hierarchical Attribute/Value Encoding (SHAVE)*, C. Adie, September 1993. Internet Draft (work in progress).
[Crocker 93a]	*Structured Text Interchange Format (STIF)*, D. Crocker, June 1993. Internet Draft (work in progress).
[Crocker 93b]	*Encoding for Personal Contact Information (PCI)*, D. Crocker, June 1993. Internet Draft (work in progress).

6. Security Considerations

This document specifies a format for encoding personal contact information. It does not address issues relating to the accuracy or authenticity of the information so encoded.

Individuals who supply contact information are invited to consider whether the disclosure of such information (*eg* home address) might lead to undesired consequences.

7. Acknowlegements

SPCI was inspired by Dave Crocker's work on STIF [Crocker 93a] and PCI [Crocker 93b], and the semantics of the SPCI elements are based on those specifications.

If you provide constructive comments, you could find your name appearing here.

8. Contact

```
<spci>
 <person key="Chris Adie">
  <name>          Chris Adie
  <dept>          Computing Service
  <org>           Edinburgh University
  <email>         C.J.Adie@edinburgh.ac.uk
  <work>
   <phone>        +44 31 650 3363
   <fax>          +44 31 662 4809
   <building>     University Library Building
   <street>       George Square
   <city>         Edinburgh
   <postcode>     EH8 9LJ
   <country>      Great Britain
</spci>
```

Appendix I - SPCI Document Type Definition

```
<!DOCTYPE spci [

<!-- Entity declarations -->
<!ENTITY amp         "&" >
<!ENTITY lt          "<" >
<!ENTITY % INHERIT   "#IMPLIED" >
<!ENTITY % aglobal   "cset CDATA %INHERIT;" >

<!-- This is used in the content models of elements which are lists. -->
<!ELEMENT item       O O  (#PCDATA) >
<!ATTLIST item       %aglobal; >

<!-- The notation used for external references by this specification. -->
<!NOTATION url       SYSTEM>

<!-- For experimental use. -->
<!ELEMENT X          - O  RCDATA >
<!ATTLIST X
          %aglobal;
          type       CDATA     #REQUIRED
>

<!-- All SPCI data must be inside an spci element -->
<!ELEMENT spci       - -  (person*) >
<!ATTLIST spci       %aglobal; >

<!-- Top-level element is a person -->
<!ELEMENT person     - O  (name? & email* & title? & org? & dept? &
                          motto? & note? & X.500? & work? & home? &
                          mobile? & support? & picture?) +(X) >
<!ATTLIST person
          %aglobal;
          -- for indexing purposes --
          key        CDATA     #REQUIRED
>

<!ELEMENT name       - O  (#PCDATA) -- Personal name -->
<!ATTLIST name       %aglobal; >

<!ELEMENT org        - O  (#PCDATA) -- Organisation -->
<!ATTLIST org        %aglobal; >

<!ELEMENT title      - O  (#PCDATA) -- Role within org -->
<!ATTLIST title      %aglobal; >

<!ELEMENT dept       - O  (#PCDATA) -- Department within org -->
<!ATTLIST dept       %aglobal; >

<!ELEMENT motto      - O  (#PCDATA) -- Your text goes here -->
<!ATTLIST motto      %aglobal; >

<!ELEMENT X.500      - O  (#PCDATA) -- Directory Name in UFN form -->
```

```
<!ATTLIST X.500      %aglobal; >

<!ELEMENT support    - O  (item+)   -- List of supported MIME types -->
<!ATTLIST support    %aglobal; >

<!ELEMENT note       - O  (#PCDATA) -- Info which doesn't fit elsewhere -->
<!ATTLIST note       %aglobal; >

<!ELEMENT picture    - O  (#PCDATA) -- Photo of the individual -->
<!ATTLIST picture
         %aglobal;
         type        NOTATION  (url) url -- Content points to image --
         encoding    CDATA     #REQUIRED -- MIME type/sybtype --
>

<!ELEMENT email      - O  (#PCDATA) -- Electronic mail address -->
<!ATTLIST email
         %aglobal;
         mailsys     (X.400|RFC822) RFC822 -- any others? --
         mailtype    (person|list|listownr) person
>

<!-- A physical address can either be an (unformatted) postal address,
      or a sequence of attributes. -->
<!ENTITY % physadr "postal|(room?,building?,street?,restante?,pobox?,
city?,ponumber?,state?,postcode?,country?)" >

<!-- These containers delimit contact info for different environments -->
<!ELEMENT work       - O  (phone? & fax? & ((%physadr;),courier?)? & telex?) >
<!ATTLIST work       %aglobal; >

<!ELEMENT home       - O  (phone? & fax? & ((%physadr;),courier?)? & telex?) >
<!ATTLIST home       %aglobal; >

<!ELEMENT mobile     - O  (phone? & fax? & pager?) >
<!ATTLIST mobile     %aglobal; >

<!ELEMENT phone      - O  (item+)  -- may be more than one phone number -->
<!ATTLIST phone      %aglobal; >

<!ELEMENT pager      - O  (phone, instruct?) >
<!ATTLIST pager      %aglobal; >

<!ELEMENT telex      - O  (#PCDATA) >
<!ATTLIST telex      %aglobal; >

<!-- Fax information is provided in several fax elements -->
<!ELEMENT fax        - O  (faxno)+ >
<!ATTLIST fax        %aglobal; >

<!ELEMENT faxno      O O  (#PCDATA,(grain? & instruct? & faxgate?)) >
<!ATTLIST faxno      %aglobal; >

<!ELEMENT instruct   - O  (#PCDATA) >
<!ATTLIST instruct   %aglobal; >

<!ELEMENT grain      - O  (#PCDATA) >
```

```
<!ATTLIST grain      %aglobal; >

<!ELEMENT faxgate    - O  (#PCDATA) >
<!ATTLIST faxgate    %aglobal; >

<!-- A physical address as a (list of) undistinguished text items. -->
<!ELEMENT postal     - O  (item+) >
<!ATTLIST postal     %aglobal; >

<!-- A physical address separated into its individual elements -->
<!ELEMENT room       - O  (#PCDATA) -- Room within a building -->
<!ATTLIST room       %aglobal; >

<!ELEMENT building   - O  (#PCDATA) -- Name of building -->
<!ATTLIST building   %aglobal; >

<!ELEMENT street     - O  (#PCDATA) -- Street address -->
<!ATTLIST street     %aglobal; >

<!ELEMENT restante   - O  (#PCDATA) -- Post restante address -->
<!ATTLIST restante   %aglobal; >

<!ELEMENT pobox      - O  (#PCDATA) -- Post Office Box number -->
<!ATTLIST pobox      %aglobal; >

<!ELEMENT city       - O  (#PCDATA) -- Town or city -->
<!ATTLIST city       %aglobal; >

<!ELEMENT ponumber   - O  (#PCDATA) -- Post Office number -->
<!ATTLIST ponumber   %aglobal; >

<!ELEMENT state      - O  (#PCDATA) -- District, region or state -->
<!ATTLIST state      %aglobal; >

<!ELEMENT postcode   - O  (#PCDATA) -- Postal or zip code -->
<!ATTLIST postcode   %aglobal; >

<!ELEMENT country    - O  (#PCDATA) -- Country -->
<!ATTLIST country    %aglobal; >

<!-- Courier delivery information -->
<!ELEMENT courier    - O  (carrier, account?, delivery?, phone?) >
<!ATTLIST courier    %aglobal; >

<!ELEMENT carrier    - O  (#PCDATA) >
<!ATTLIST carrier    %aglobal; >

<!ELEMENT account    - O  (#PCDATA) >
<!ATTLIST account    %aglobal; >

<!ELEMENT delivery   - O  (#PCDATA) >
<!ATTLIST delivery   %aglobal; >

]>
```