Trade Working Group
INTERNET-DRAFT

Expires: August 2003

February 2003
Ko Fujimura
Masayuki Terada
NTT

XML Voucher: Generic Voucher Language
<draft-ietf-trade-voucher-lang-05.txt>

Status of This Document

Abstract

This document specifies rules for defining voucher properties in XML syntax. A voucher is a logical entity that represents a right to claim goods or services. A voucher can be used to transfer a wide-range of electronic-values, including coupons, tickets, loyalty points, and gift certificates, which are often necessary to process in the course of payment and/or delivery transactions.

Acknowledgements

The following persons, in alphabetic order, contributed substantially to the material herein:

Donald Eastlake 3rd
Ian Grigg
Renato Iannella
Yoshiaki Nakajima

Table of Contents

1. Introduction

This document, XML Voucher, specifies rules for defining voucher properties in XML syntax. The motivation and background of the specification are described in [GVT].

A voucher is a logical entity that represents a certain right and is logically managed by the Voucher Trading System (VTS). A voucher is generated by the issuer, traded among users, and finally collected by the collector using VTS.

This document defines the syntax and semantics of Voucher Component, which defines voucher meaning and processing rules in XML syntax [XML]. A Voucher Component define the properties that must be satisfied to allow the voucher to be processed by VTS or other trading systems, e.g., wallet or merchant system. VTS definitions and models are also defined in [GVT].

Note: This document uses "voucher" as an "instance of voucher" whose meaning is defined by Voucher Component. In other words, multiple vouchers can be issued and managed by the VTS using the same Voucher Component.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119]

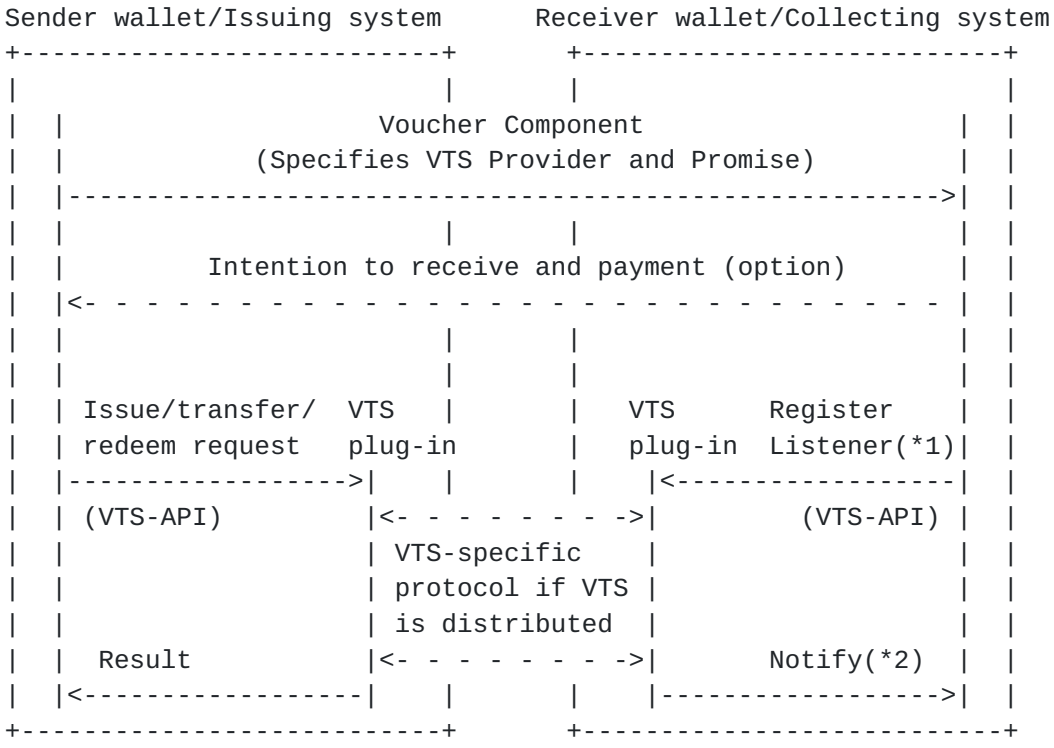
2. Processing Model

There are several ways of implementing VTS and technologies are continually changing. For discount coupons or event tickets, for example, the smart-card-based offline VTS is often preferred, whereas for bonds or securities, the centralized online VTS is preferred. It is impractical to define standard protocols for issuing, transferring, or redeeming vouchers at this moment.

To provide implementation flexibility, this document assumes a modular wallet architecture that allows multiple VTS to be added as plug-ins. In this architecture, instead of specifying a standard voucher transfer protocol, two specifications, i.e., Voucher Component and VTS-API specifications, are standardized (Figure 1).

After sender and receiver agree on what vouchers are to be traded and which VTS is to be used, the issuing system or wallet system requests the corresponding VTS plug-in to permit the issue, transfer, or redeem transactions to be performed via the VTS API. The VTS then rewrites the ownership of the vouchers using the VTS-specific protocol. Finally, a completion event is sent to the wallet systems or issuing/collecting systems.

This document describes the Voucher Component specification. The VTS-API specification is defined in [VTS-API].



(*1) Registration is optional. Note also that the VTS plug-ins are usually pre-registered when the wallet or collecting system is started.
(*2) If a listener is registered.

Figure 1. Wallet architecture with VTS plug-ins

3. Trust Model

A voucher is trusted if the Issuer and VTS Provider are trusted, since the Issuer is responsible for the contents of the voucher and the VTS Provider is responsible for preventing ownership from being assigned to multiple users.

The trust level required for Issuer and VTS Provider depends on the type (or Promise) of the voucher. To provide the information needed for verification, the conditions of Issuer and VTS Provider are specified in the Voucher Component, and given as input to the verifier, e.g., wallet system or other software. The trust of a voucher is thus verified through the Voucher Component. This model enables trading partners to verify their trust in the voucher regardless of their trust in the partners.

This document assumes that the Voucher Component is the root of trust. If a malicious user could alter the Voucher Component, a forged voucher, could be verified as valid.

The Voucher Component is usually delivered from the trusted VTS

Provider, Issuer or trusted third party using a secure communication channel, such as [XMLDSIG], [TLS], or [IPSEC]. Delivery of the Voucher Component is beyond the scope of this document.

Note: The Voucher Component does not have to be sent from the sender of the voucher. Note also that a set of trusted Voucher Components can be downloaded before conducting a transaction.

4. Component Structure

The Voucher Component provides the information needed to identify the monetary value or merchandize rendered when the voucher is redeemed. It includes:

o How much value/items can be claimed in exchange for the voucher

o Restrictions applied to the voucher
- Participants (VTS Provider, Issuer, Holder, and Collector)
- Objects (merchandise) to be claimed
- Time when valid (validity period)
- Others

The Voucher Component also provides common properties useful for displaying and manipulating the contents of wallet systems. It includes the title and description of each voucher.

The Voucher Component contains the Title Component, Description Component, Provider Component, Issuer Component, Holder Component, Collector Component, Value Component, Merchandise Component, ValidPeriod Component, and Condition Component as follows:

Title Component

Provides the title of the voucher. This is mainly for listing the entities stored in a wallet system.

Description Component

Provides a short description of the voucher. This is mainly for listing the entities stored in a wallet system.

Provider Component

Provides restrictions on which VTS Provider (or VTS plug-in) can be used for trading the voucher.

Issuer Component

Provides restrictions on the Issuer of the voucher.

Holder Component

Provides restrictions on the Holder of the voucher.

Provides restrictions on the Collector of the voucher.

Value Component

Provides the value of each voucher. There are two types of values, i.e., fixed and ratio values. For a fixed value, the currency and the figure is specified. For a ratio value, the discount ratio of the corresponding merchandize is specified.

The Value Component also indicates the number of vouchers to be redeemed for claiming the merchandise or monetary value specified in Merchandise Component or Value Component. If "n"(>0) is specified, the merchandize or monetary value can be claimed in exchange for "n sheets of" vouchers. If "0" is specified, it can be used repeatedly.

Merchandise Component

Provides restrictions on the object to be claimed. Domain-specific meaning of the voucher, e.g., reference number of the merchandize or seat number for an event ticket, is specified to identify the merchandize rendered when the voucher is redeemed.

ValidPeriod Component

Provides restrictions on the validity period of the voucher, i.e., start date and end date.

Condition Component

Provides any other applicable restrictions. This is intended to cover contracts between the issuer and holder of the voucher in natural language form.

Using the above Components, semantics for diverse types of vouchers can be defined as shown in Table 1.

	Value			Restrictions
Examples	Ratio	Fixed		Merchandise
		Amount	Currency	Number Needed for redemption	
Gift certificate		25	USD	1	(Not specified)
Loyalty point		1	AUD	10	(Not specified)
Member card	20%			0	(Not specified)
Coupon	30%			1	Beef 500g
Event ticket	100%			1	Hall A, S ,K23

Exchange ticket	100%			1	ISBN:0071355014

Table 1. Examples of vouchers and their properties

5. Syntax Overview and Examples

This section provides an overview and examples of Voucher Component. The formal syntax and semantics are found in Sections 6 and 7.

Voucher Components are represented by the <Voucher> element which has the following structure (where "?" denotes zero or one occurrence):

```
<Voucher>
  (Title)
  (Description)?
  (Provider)
  (Issuer)?
  (Holder)?
  (Collector)?
  (Value)
  (Merchandise)?
  (ValidPeriod)?
  (Conditions)?
</Voucher>
```

An example of a Voucher Component is described below. This is an example of a five dollar discount coupon for specific merchandize, a book with ISBN number 0071355014. The coupon is valid from April 1st in 2001 to March 31st in 2002. To claim this offer, one voucher must be spent.

```
<?xml version="1.0"?>
<Voucher xmlns="urn:ietf:params:xml:schema:vts-lang"
         xmlns:vts="http://www.example.com/vts">
  <Title>IOTP Book Coupon</Title>
  <Description>$5 off IOTP Book</Description>
  <Provider name="Voucher Exchanger 2002">
    <vts:Version>VE2.31</vts:Version>
  </Provider>
  <Issuer name="Alice Book Center, Ltd.">
    <vts:KeyInfo>
      1DA8DFCF95521014BBB7171B95545E8D61AE803F
    </vts:KeyInfo>
  </Issuer>
  <Collector name="Alice Book Center, Ltd."/>
  <Value type="discount" spend="1">
    <Fixed amount="5" currency="USD"/>
  </Value>
  <Merchandise>
    <bk:Book xmlns:bk="http://www.example.com/bk"
             bk:isbn="0071355014"/>
```

```
    </Merchandise>
    <ValidPeriod start="2002-04-01" end="2003-03-31"/>
```

```
  <Conditions>
    The value of this coupon is subject to tax.
  </Conditions>
</Voucher>
```

6. Syntax and Semantics

The general structure of an XML voucher is described in Component Structure (section 4). This section details the Voucher Component features. Features described in this section MUST be implemented unless otherwise indicated. The syntax is defined via [XML-Schema-1] [XML-Schema-2]. For clarity, unqualified elements in schema definitions are in the XML schema namespace:

```
xmlns="http://www.w3.org/2001/XMLSchema"
```

References to XML Voucher schema defined herein use the prefix "gvl" and are in the namespace:

```
xmlns:gvl="urn:ietf:params:xml:schema:vts-lang"
```

This namespace URI for elements defined by this document is a URN [URN], using the namespace identifier 'ietf' defined by [URN-NS-IETF] and extended by [XML-Registry].

This namespace is also used for unqualified elements in voucher examples.

6.1 <Voucher>

The <Voucher> element contains <Title>, <Provider>, and <Value> elements and optionally contains <Description>, <Issuer>, <Holder>, <Collector>, <ValidPeriod>, and <Condition> elements. These sub-elements are defined in the following sections.

The <Voucher> element is defined by the following schema:

```
 <element name="Voucher" type="gvl:VoucherType"/>
 <complexType name="VoucherType">
  <sequence>
   <element ref="gvl:Title"/>
   <element ref="gvl:Description" minOccurs="0"/>
   <element ref="gvl:Provider"/>
   <element ref="gvl:Issuer" minOccurs="0"/>
   <element ref="gvl:Holder" minOccurs="0"/>
   <element ref="gvl:Collector" minOccurs="0"/>
   <element ref="gvl:Value"/>
   <element ref="gvl:Merchandise" minOccurs="0"/>
   <element ref="gvl:ValidPeriod" minOccurs="0"/>
   <element ref="gvl:Conditions"  minOccurs="0"/>
```

```
     </sequence>
    </complexType>
```

6.2 <Title>

The <Title> element contains a simpletext title of the voucher. This is mainly for listing the entities stored in a wallet system.

The <Title> element has no attribute.

The <Title> element is defined by the following schema:

```
<element name="Title" type="string"/>
```

6.3 <Description>

The <Description> element contains a simpletext description of the voucher. This is mainly for listing the entities stored in a wallet system.

The <Description> element has no attribute.

The <Description> element is defined by the following schema:

```
<element name="Description" type="string"/>
```

6.4 <Provider>

The <Provider> element may contain any elements that is used to specify or restrict the VTS Provider of the voucher. The sub-elements contained in this element depend on the implementation of the VTS.

An implementation of a wallet system may use this information to identify and/or authenticate the VTS Provider when the VTS plug-in is registered (See Section 7 of [VTS-API]). These implementation-specific elements of the VTS can be extended using [XML-ns]. An example of such schema definition is described in Section 8.

The <Provider> element has a string-type "name" attribute that is used to specify the name of the VTS Provider.

The <Provider> element is defined by the following schema:

```
<element name="Provider" type="gvl:RoleType"/>
<complexType name="RoleType" mixed="true">
 <sequence>
  <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
 <attribute name="name" type="string"/>
</complexType>
```

6.5 <Issuer>

The <Issuer> element may contain any element that is used to specify or restrict the Issuer of the voucher.

The Issuer of the voucher is generally managed by the VTS [VTS-API]. There is no need to specify the Issuer of the voucher using this element if there are no restrictions on the Issuer.
An implementation of a VTS may use this element to describe the authentication data and/or qualification information of the Issuer. This implementation-specific information can be extended as sub-elements using [XML-ns]. An example of such schema definition is described in Section 8.

The <Issuer> element has a string-type "name" attribute that is used to specify the name of the Issuer.

The <Issuer> element is defined by the following schema:

<element name="Issuer" type="gvl:RoleType"/>
The <RoleType> element type is defined in Section 6.4.

If the <Issuer> element is omitted, it MUST be interpreted that there are no restrictions on the Issuer.

6.6 <Holder>

The <Holder> element may contain any elements that is used to specify or restrict the Holder of the voucher.

The Holder of the voucher is generally managed by the VTS [VTS-API]. There is no need to specify the Holder of the voucher using this element if there are no restrictions on the Holder.
An implementation of a VTS may use this element to describe the authentication data and/or qualification information of the Holder. This implementation-specific information can be extended as sub-elements using [XML-ns].

The <Holder> element has a string-type "name" attribute that is used to specify the name of the Holder.

The <Holder> element is defined by the following schema:

<element name="Holder" type="gvl:RoleType"/>
The <RoleType> element type is defined in Section 6.4.

If the <Holder> element is omitted, it MUST be interpreted that there are no restrictions on the Holder.

6.7 <Collector>

The <Collector> element may contain any elements that is used to

specify or restrict the Collector of the voucher.

There is no need to specify the Collector of the voucher using this

element if there are no restrictions on the Collector.
An implementation of a VTS may use this element to describe the authentication data and/or qualification information of the Collector. This implementation-specific information can be extended as sub-elements using [XML-ns].

The <Collector> element has a string-type "name" attribute that is used to specify the name of the Collector.

The <Collector> element is defined by the following schema:

<element name="Collector" type="gvl:RoleType"/>
The <RoleType> element type is defined in Section 6.4.

If the <Collector> element is omitted, it MUST be interpreted that there are no restrictions on the Collector.

6.8 <Value>

The <Value> element optionally contains a <Fixed> or a <Ratio> element but not both. These sub-elements are defined in the following sections.

The <Value> element has a "type" attribute that is used to specify the value process type. This attribute is provided to calculate the amount paid when the vouchers are redeemed at Merchant site, etc.

The following identifiers are defined for the "type" attribute.

Exchange: Items specified in the <Merchandise> element can be claimed in exchange for the voucher. If this type is selected, neither <Ratio> nor <Fixed> element MUST be specified. Note that this value process type has the same meaning as:
<Value type="discount"><Ratio percentage="100"/></Value>

Discount: Items specified in the <Merchandise> element can be purchased at the discount price calculated by the <Ratio> or <Fixed> element.

Monetary: Items specified in the <Merchandise> element can be purchased using the value of the voucher. (Note: if the <Merchandise> element is not specified, the voucher can be used for any purchase.) If this type is selected, the <Fixed> element MUST be specified.

The <Value> element also has a "spend" attribute that is used to specify the number of vouchers to be redeemed for claiming the goods, services, or monetary value specified. For example, if "n" (>0) is specified, goods etc. can be claimed in exchange for "n sheets of" vouchers. (Note: Multiple vouchers for the same Voucher

Component must exist in this case.) If "0" is specified, it can be used repeatedly.

If the "spend" attribute is omitted or the whole element is omitted, it MUST be interpreted that "1" is specified for the "spend" attribute.

The <Value> element is defined by the following schema:

```
 <element name="Value" type="gvl:ValueType"/>
 <complexType name="ValueType">
  <sequence minOccurs="0">
   <choice>
    <element name="Ratio" type="gvl:RatioValueType"/>
    <element name="Fixed" type="gvl:FixedValueType"/>
   </choice>
  </sequence>
  <attribute name="type" type="gvl:ValueProcessType"
             use="required"/>
  <attribute name="spend" type="nonNegativeInteger"
             default="1"/>
 </complexType>
```

The <ValueProcessType> element type is defined by the following schema:

```
 <simpleType name="ValueProcessType">
  <restriction base="string">
   <enumeration value="exchange"/>
   <enumeration value="discount"/>
   <enumeration value="monetary"/>
  </restriction>
 </simpleType>
```

6.8.1 <Ratio>

The <Ratio> element does not contain any contents.

The <Ratio> element has a "percentage" attribute that is used to specify the discount ratio of the price of the corresponding merchandize in percentage.

The <RatioValueType> element type is defined by the schema:

```
 <complexType name="RatioValueType">
  <attribute name="percentage" use="required">
   <simpleType>
    <restriction base="float">
     <maxInclusive value="100"/>
    </restriction>
   </simpleType>
  </attribute>
 </complexType>
```

6.8.2 <Fixed>

The <Fixed> element does not contain any contents.

The <Fixed> element has "currency" and "amount" attributes and optionally a "decimalPower" attribute as follows:

Currency: Provides the unit of the monetary value in the three letter ISO currency code [ISO4217]. For example, for US dollars it is "USD".

Amount: Provides the amount of the monetary value per voucher.

DecimalPower: Provides the number of decimal digits from the decimal point applied to the base for the "amount" attribute above. If the "decimalPower" attribute is omitted, it MUST be interpreted that "0" is specified for the "decimalPower" attribute.

For example, with a dollar currency denominated in cents, "1" is specifed for the "amount" attribute, and "-2" is specified for the "decimalPower" attribute. Alternately, "0.01" is specified for the "amount" attribute and the "decimalPower" attribute is omitted.

The <FixedValueType> type is defined by the following definition:

```
 <complexType name="FixedValueType">
  <attribute name="currency" type="string" use="required"/>
  <attribute name="amount" type="float" use="required"/>
  <attribute name="decimalPower" type="short" default="0"/>
 </complexType>
```

6.9 <Merchandise>

The <Merchandise> element may contain any elements used to specify or restrict the goods or services rendered when the voucher is redeemed. The sub-elements contained in this element are depending on the application of the voucher and are left to the other domain-specific specifications. Domain-specific elements can be extended as sub-elements using [XML-ns].

The <Merchandise> element does not have any attribute.

The <Merchandise> element is defined by the following schema:

```
 <element name="Merchandise" type="gvl:MerchandiseType"/>
 <complexType name="MerchandiseType" mixed="true">
  <sequence>
   <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
```

If the <Merchandise> element is omitted, it will be generally interpreted that there is no restriction on the merchandise when using the voucher.

6.10 <ValidPeriod>

The <VaridPeriod> element does not contain any contents.

The <ValidPeriod> element has dateTime-type "start" and "end" attributes that are used to place limits on the validity of the voucher.

The <ValidPeriod> element is defined by the following schema:

```
<element name="ValidPeriod" type="gvl:ValidPeriodType"/>
<complexType name="ValidPeriodType">
  <attribute name="start" type="dateTime"/>
  <attribute name="end" type="dateTime"/>
</complexType>
```

If the "start" attribute is omitted, it MUST be interpreted that the voucher is valid on any date up to the date (inclusive) specified by the end attribute. If the "end" attribute is omitted, it MUST be interpreted that the voucher is valid from the start attribute with no expiry. If neither attribute is specified or the whole element is omitted, it MUST be interpreted that the voucher is valid at any time.

6.10 <Conditions>

The <Conditions> element may contain any element used to specify any other restrictions or conditions applied that limit the use of the voucher. The sub-elements contained in this element are depending on the application of the voucher and are left to the other domain-specific specifications. Domain-specific elements can be extended as sub-elements using [XML-ns].

The <Conditions> element does not have any attribute.

The <Conditions> element is defined by the following schema:

```
<element name="Conditions" type="gvl:ConditionsType"/>
<complexType name="ConditionsType" mixed="true">
 <sequence>
  <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

If the <Conditions> element is omitted, it will be generally interpreted that there are no other restrictions or conditions on using the voucher.

7. IANA Considerations

This document calls for IANA to register a new XML schema with the URN. The registation form [XML-Registry] for this is below:

URI
urn:ietf:params:xml:schema:vts-lang

Registrant Contact
See the "Author's Address" section of this document.

XML
BEGIN

```
<?xml version="1.0"?>

<schema
  targetNamespace="urn:ietf:params:xml:schema:vts-lang"
  xmlns:gvl="urn:ietf:params:xml:schema:vts-lang"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

<element name="Voucher" type="gvl:VoucherType"/>
<complexType name="VoucherType">
 <sequence>
  <element ref="gvl:Title"/>
  <element ref="gvl:Description" minOccurs="0"/>
  <element ref="gvl:Provider"/>
  <element ref="gvl:Issuer" minOccurs="0"/>
  <element ref="gvl:Holder" minOccurs="0"/>
  <element ref="gvl:Collector" minOccurs="0"/>
  <element ref="gvl:Value"/>
  <element ref="gvl:Merchandise" minOccurs="0"/>
  <element ref="gvl:ValidPeriod" minOccurs="0"/>
  <element ref="gvl:Conditions"  minOccurs="0"/>
 </sequence>
</complexType>

<element name="Title" type="string"/>

<element name="Description" type="string"/>

<element name="Provider" type="gvl:RoleType"/>
<element name="Issuer"   type="gvl:RoleType"/>
<element name="Holder"   type="gvl:RoleType"/>
<element name="Collector"   type="gvl:RoleType"/>
<complexType name="RoleType" mixed="true">
 <sequence>
  <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
 <attribute name="name" type="string"/>
</complexType>

<element name="Value" type="gvl:ValueType"/>
```

```
      <complexType name="ValueType">
       <sequence minOccurs="0">
        <choice>
```

```
        <element name="Ratio" type="gvl:RatioValueType"/>
        <element name="Fixed" type="gvl:FixedValueType"/>
       </choice>
      </sequence>
      <attribute name="type" type="gvl:ValueProcessType"
                 use="required"/>
      <attribute name="spend" type="nonNegativeInteger"
                 default="1"/>
    </complexType>

    <simpleType name="ValueProcessType">
     <restriction base="string">
      <enumeration value="exchange"/>
      <enumeration value="discount"/>
      <enumeration value="monetary"/>
     </restriction>
    </simpleType>

    <complexType name="RatioValueType">
     <attribute name="percentage" use="required">
      <simpleType>
       <restriction base="float">
        <maxInclusive value="100"/>
       </restriction>
      </simpleType>
     </attribute>
    </complexType>

    <complexType name="FixedValueType">
     <attribute name="currency" type="string" use="required"/>
     <attribute name="amount" type="float" use="required"/>
     <attribute name="decimalPower" type="short" default="0"/>
    </complexType>

    <element name="Merchandise" type="gvl:MerchandiseType"/>
    <complexType name="MerchandiseType" mixed="true">
     <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
    </complexType>

    <element name="ValidPeriod" type="gvl:ValidPeriodType"/>
    <complexType name="ValidPeriodType">
      <attribute name="start" type="dateTime"/>
      <attribute name="end" type="dateTime"/>
    </complexType>

    <element name="Conditions" type="gvl:ConditionsType"/>
    <complexType name="ConditionsType" mixed="true">
```

```
        <sequence>
         <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
       </complexType>
```

```
  </schema>
  END
```

8. VTS Schema Example

An example of the schema definition for a VTS implementation is described below:

```
<?xml version="1.0"?>

<schema
 targetNamespace="http://www.example.com/vts"
 xmlns:vts="http://www.example.com/vts"
 xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <element name="Version" type="string"/>
 <element name="KeyInfo" type="hexBinary"/>
</schema>
```

Using this schema definition, the <vts:Version> can be used for specifying the VTS version number and the <vts:KeyInfo> element can be used for specifying the Issuer in the Voucher Component as shown in Section 5.

9. Security Considerations

Security issues for delivering Voucher Components are discussed in Section 3.

10. Normative References

[ISO4217] "Codes for the representation of currencies and funds", ISO 4217, 1995.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[URN] R. Moats, "URN Syntax", RFC2141, May 1997.

[URN-NS-IETF] R. Moats, "A URN Namespace for IETF Documents", RFC2648, August 1999.

[XML] "Extensible Mark Up Language (XML) 1.0 (Second Edition)", A W3C Recommendation, <http://www.w3.org/TR/REC-xml>, October 2000.

[XML-ns] "Namespaces in XML", A W3C Recommendation, <http://www.w3.org/TR/REC-xml-names>, January 1999.

[XML-Registry] M. Mealing, "The IETF XML Registry",

draft-mealling-iana-xmlns-registry-04, June 2002. In RFC Editor queue.

[XML-Schema-1] H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, "XML Schema Part 1: Structures W3C Recommendation.", <http://www.w3.org/TR/xmlschema-1/>, May 2001.

[XML-Schema-2] P. Biron and A Malhotra, "XML Schema Part 2: Datatypes W3C Recommendation.", <http://www.w3.org/TR/xmlschema-2/>, May 2001.

11. Informational References

[GVT] K. Fujimura, "Requirements and Design for Voucher Trading System (VTS)", draft-ietf-trade-drt-requirements-04.txt, July 2002. In RFC Editor queue.

[IPSEC] R. Thayer, N. Doraswamy, and R. Glenn, "IP Security Document Roadmap", RFC2411, November 1998

[TLS] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC2246, January 1999.

[VTS-API] M. Terada and K. Fujimura, "Voucher Trading System Application Programming Interface (VTS-API)", draft-ietf-trade-voucher-vtsapi-05.txt, February 2003.

[XMLDSIG] D. Eastlake, J. Reagle, and D. Solo, "XML-Signature Syntax and Processing", RFC3275, March 2002.

12. Author's Address

Ko Fujimura and Masayuki Terada
NTT Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa, 239-0847 JAPAN
Phone: +81-(0)46-859-3814
Fax: +81-(0)46-859-8329
Email: fujimura@isl.ntt.co.jp, terada@isl.ntt.co.jp

Full Copyright Statement