Network Working Group
Request for Comments: 4153
Category: Informational

K. Fujimura
NTT
M. Terada
NTT DoCoMo
D. Eastlake 3rd
Motorola Laboratories

September 2005

XML Voucher: Generic Voucher Language

Status of This Memo

Copyright Notice

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 often have to be processed in the course of payment and/or delivery transactions.

Table of Contents

1. Introduction

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

A voucher is a logical entity that represents a certain right and that 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 the Voucher Component, which defines voucher meaning and processing rules in XML syntax [XML]. A Voucher Component defines the properties that must be satisfied to allow the voucher to be processed by VTS or other trading systems; e.g., a wallet or merchant system. VTS definitions and models are also defined in [VTS].

Note: This document uses "voucher" as an "instance of voucher", whose meaning is defined by the Voucher Component. In other words, a Voucher Component is NOT a voucher, and 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 [RFC2119]

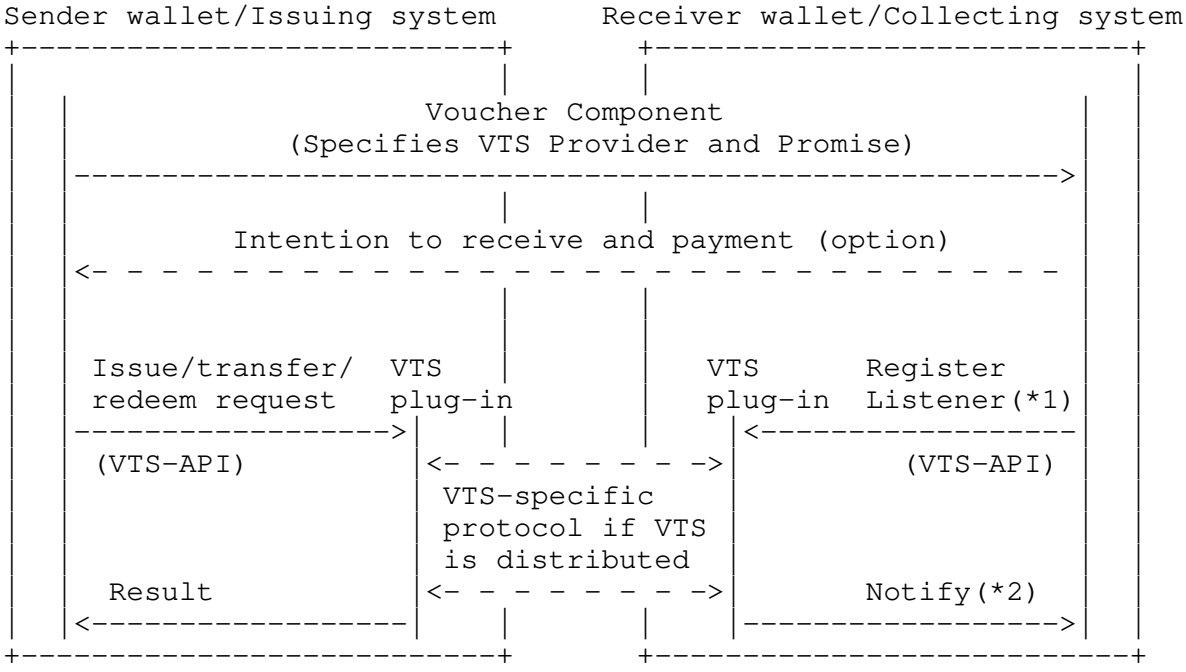
2. Processing Model

There are several ways of implementing VTS and technologies are continually changing. For discount coupons or event tickets, for example, the smartcard-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 time.

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

After the sender and receiver agree on which 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, as 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 the Issuer and VTS Provider depends on the type (or Promise) of the voucher. To provide the information needed for verification, the conditions of the 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.

When a Voucher Component is delivered from the trusted VTS Provider, Issuer, or trusted third party, a secure communication channel (e.g., [TLS], [IPSEC], or object security, such as [XMLDSIG]) should be used to prevent alteration during the delivery.

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 a transaction is conducted.

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

- how much value/items can be claimed in exchange for the voucher, and
- restrictions applied to the voucher
  - participants (VTS Provider, Issuer, Holder, and Collector),
  - objects (merchandise) to be claimed,
  - time when valid (validity period), and
  - 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 following components:

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.

Collector Component

Provides restrictions on the Collector of the voucher.

Value Component

Provides the value of each voucher. There are two types of values: 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 the 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. The 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		Number needed for redemption	Merchandise
		Amount	Currency		
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 Components. 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 1, 2001, to March 31, 2002. To claim this offer, one voucher must be spent.

```
<?xml version="1.0" encoding="UTF-8"?>
<Voucher xmlns="urn:ietf:params:xml:ns: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 Component is described in 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:ns:vts-lang"
```

This namespace URI for elements defined by this document is a URN [URN] that uses 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 element 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 a 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 through sub-elements using [XML-ns]. An example of such a 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 element 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 through 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 element 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 through 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 <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 the <Ratio> nor the <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 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 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 following 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, US dollars 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 specified 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 follows:

```
<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 element used to specify or restrict the goods or services rendered when the voucher is redeemed. The sub-elements contained in this element depend 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].

This element is intended to be interpreted by a collecting system. An implementation of a wallet system does not have to use this element. Any restrictions applied should also be described in the <Description> element or the <Conditions> elements in natural language form to enable users to check the restrictions.

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>
```

6.10. <ValidPeriod>

The <ValidPeriod> 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 that specified by the end attribute (inclusive). 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.

6.11. <Conditions>

The <Conditions> element contains any other restrictions or conditions applied. This is intended to cover contracts between the issuer and the holder of the voucher in natural language form.

An implementation of a wallet system SHOULD provide a means of displaying the text in this element.

The <Conditions> element has no attribute.

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

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

7. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemas conforming to a registry mechanism described in [XML-Registry]. IANA has registered two URI assignments.

Registration request for the vts-lang namespace:

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

Registrant Contact: See the "Authors' Addresses" section of this document.

XML: None. Namespace URIs do not represent an XML specification.

Registration request for the vts-lang XML schema:

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

Registrant Contact: See the "Authors' Addresses" section of this document.

XML:

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>

<schema
  targetNamespace="urn:ietf:params:xml:ns:vts-lang"
  xmlns:gvl="urn:ietf:params:xml:ns: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="string"/>
     </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

The VTS must provide a means to prevent forgery, alteration, duplicate-redemption, reproduction of a voucher, and non-repudiation of transactions, as described in Section 3.2 of [VTS]. This will commonly require the presence of a unique serial number or the like in each Voucher instance, usually outside the Voucher Component. These security requirements, however, mainly follow the VTS plug-ins and their protocols. This document assumes that the VTS plug-ins are trusted and are installed by some means; e.g., manually checked as are other download applications.

The Voucher Component, however, defines restrictions on the VTS Provider (or VTS plug-in), and, if this information is altered, incorrect VTS plug-ins not accepted by the issuer could be used, allowing a forged voucher to be verified as if it were valid. To prevent this situation, the Voucher Component should be stored and

acquired securely; e.g., downloaded from a trusted party using a secure communication channel, such as [TLS] or [IPSEC], or secured by the digital signature of a trusted party [XMLDSIG].

10. Acknowledgements

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

Ian Grigg
Renato Iannella
Yoshiaki Nakajima

11. Normative References

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

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

[URN] Moats, R., "URN Syntax", RFC 2141, May 1997.

[URN-NS-IETF] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, 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] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

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

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

12. Informative References

[VTS] Fujimura, K. and D. Eastlake, "Requirements and Design for Voucher Trading System (VTS)", RFC 3506, March 2003.

[IPSEC] Thayer, R., Doraswamy, N., and R. Glenn, "IP Security Document Roadmap", RFC 2411, November 1998.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[VTS-API] Terada, M. and K. Fujimura, "Voucher Trading System Application Programming Interface (VTS-API)", RFC 4154, September 2005.

[XMLDSIG] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

Authors' Addresses

Ko Fujimura
NTT Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa, 239-0847 JAPAN

Phone: +81-(0)46-859-3053
Fax: +81-(0)46-855-1730
EMail: fujimura.ko@lab.ntt.co.jp

Masayuki Terada
NTT DoCoMo, Inc.
3-5 Hikari-no-oka, Yokosuka-shi, Kanagawa, 239-8536 JAPAN

Phone: +81-(0)46-840-3809
Fax: +81-(0)46-840-3705
EMail: te@rex.yrp.nttdocomo.co.jp

Donald E. Eastlake 3rd
Motorola Laboratories
155 Beaver Street
Milford, MA 01757 USA

Phone: 1-508-786-7554 (work)
1-508-634-2066 (home)
EMail: Donald.Eastlake@motorola.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.