Election Markup Language (EML) Version 5.0

Schema Descriptions

Committee Draft 01

14 March 2007

Specification URIs:
This Version:
http://docs.oasis-open.org/election/eml/v5.0/cd01/EML-Schema-Descriptions-v5.0.doc
http://docs.oasis-open.org/election/eml/v5.0/cd01/EML-Schema-Descriptions-v5.0.html
http://docs.oasis-open.org/election/eml/v5.0/cd01/EML-Schema-Descriptions-v5.0.pdf
http://docs.oasis-open.org/election/eml/v5.0/cd01/EML-v5.0-cd01.zip

Previous Version:
http://www.oasis-open.org/committees/download.php/18158/EML%20v4.0%20-%20OASIS%20Standard.zip

Latest Version:
http://docs.oasis-open.org/election/eml/v5.0/EML-Schema-Descriptions-v5.0.doc
http://docs.oasis-open.org/election/eml/v5.0/EML-Schema-Descriptions-v5.0.html
http://docs.oasis-open.org/election/eml/v5.0/EML-Schema-Descriptions-v5.0.pdf
http://docs.oasis-open.org/election/eml/v5.0/EML-v5.0-cd01.zip

Technical Committee:
OASIS Election and Voter Services TC

Chair:
John Borras

Editor:
John Borras

Related work:
This specification supercedes:

- Election Markup Language (EML) v4.0

See also:

- EML Process and Data Requirements
- EML Data Dictionary

Declared XML Namespace:
urn:oasis:names:tc:evs:schema:eml

Abstract:
This document contains the descriptions of the schemas used in EML v5.0. This document provides an explanation of the core schemas used throughout, definitions of the simple and complex datatypes, plus the EML schemas themselves. It also covers the conventions used in

the specification and the use of namespaces, as well as the guidance on the constraints, extendibility, and splitting of messages.

Status:

This document was last revised or approved by the Election and Voter Services Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/election/

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/election/ipr.php

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/election/.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

This document describes the OASIS Election Mark-up Language (EML) version 5.0 schemas.

The messages that form part of EML are intended for transfer between systems. It is not intended that all outputs of a registration or election system will have a corresponding schema.

This document and its accompanying set of schemas do not claim to satisfy the final requirements of a registration or election system. It is incumbent on the users of this document to identify any mistakes, inconsistencies or missing data and to propose corrections to the OASIS Election and Voter Services Technical Committee.

1.1 Terminology

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]**.

1.2 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

1.3 Non-Normative References

[MIME PART 2] Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types IETF http://www.ietf.org/rfc/rfc2046.txt

[MIME] MIME Media Types IANA http://www.iana.org/assignments/media-types/

[XMLDSig] XML-Signature Syntax and Processing W3C http://www.w3.org/TR/xmldsig-core/

[XPATH] XML Path Language (XPath) Version 1.0 W3C http://www.w3.org/TR/xpath

2 The EML Schemas

2.1 Background

The following is the Executive Summary of the “EML Process and Data Requirements”:

OASIS, the XML interoperability consortium, formed the Election and Voter Services Technical Committee in the spring of 2001 to develop standards for election and voter services information using XML. The committee’s mission statement is, in part, to:

“Develop a standard for the structured interchange among hardware, software, and service providers who engage in any aspect of providing election or voter services to public or private organizations...”

The objective is to introduce a uniform and reliable way to allow systems involved in the election process to interact. The overall effort attempts to address the challenges of developing a standard that is:

- **Multinational:** Our aim is to have these standards adopted globally.
- **Flexible**: Effective across the different voting regimes (e.g. proportional representation or 'first past the post') and voting channels (e.g. Internet, SMS, postal or traditional paper ballot).
- **Multilingual:** Flexible enough to accommodate the various languages and dialects and vocabularies.
- **Adaptable:** Resilient enough to support elections in both the private and public sectors.
- **Secure:** Able to secure the relevant data and interfaces from any attempt at corruption, as appropriate to the different requirements of varying election rules.

The primary deliverable of the committee is the Election Markup Language (EML). This is a set of data and message definitions described as XML schemas. At present EML includes specifications for:

- Candidate Nomination, Response to Nomination and Approved Candidate Lists
- Referendum Options Nomination, Response to Nomination and Approved Options Lists
- Voter Registration information, including eligible voter lists
- Various communications between voters and election officials, such as polling information, election notices, etc.
- Ballot information (races, contests, candidates, etc.)
- Voter Authentication
- Vote Casting and Vote Confirmation
- Election counts and results
- Audit information pertinent to some of the other defined data and interfaces
- EML is flexible enough to be used for elections and referendums that are primarily paper-based or that are fully e-enabled.

As an international specification, EML is generic in nature, and so needs to be tailored for specific scenarios. Some aspects of the language are indicated in EML as required for all scenarios and so can be used unchanged. Some aspects (such as the ability to identify a voter easily from their vote) are required in some scenarios but prohibited in others, so EML defines them as optional. Where they are prohibited, their use must be changed from an optional to prohibited classification, and where they are mandatory, their use must be changed from an optional to required classification.

2.2 Viewing Schemas

EML schemas are supplied as text documents. For viewing the structure of the schemas, we recommend use of one of the many schema development tools available. Many of these provide graphical displays.

The Schematron schemas are mainly short and simple to understand as text documents for those with a working knowledge of Xpath.

2.3 Schema Diagrams in this Document

The schema diagrams in this document were created using XML Spy 2007. The following is a guide to their interpretation.

In this section, terms with specific meanings in XML or XML Schema are shown in italics, e.g. sequence.

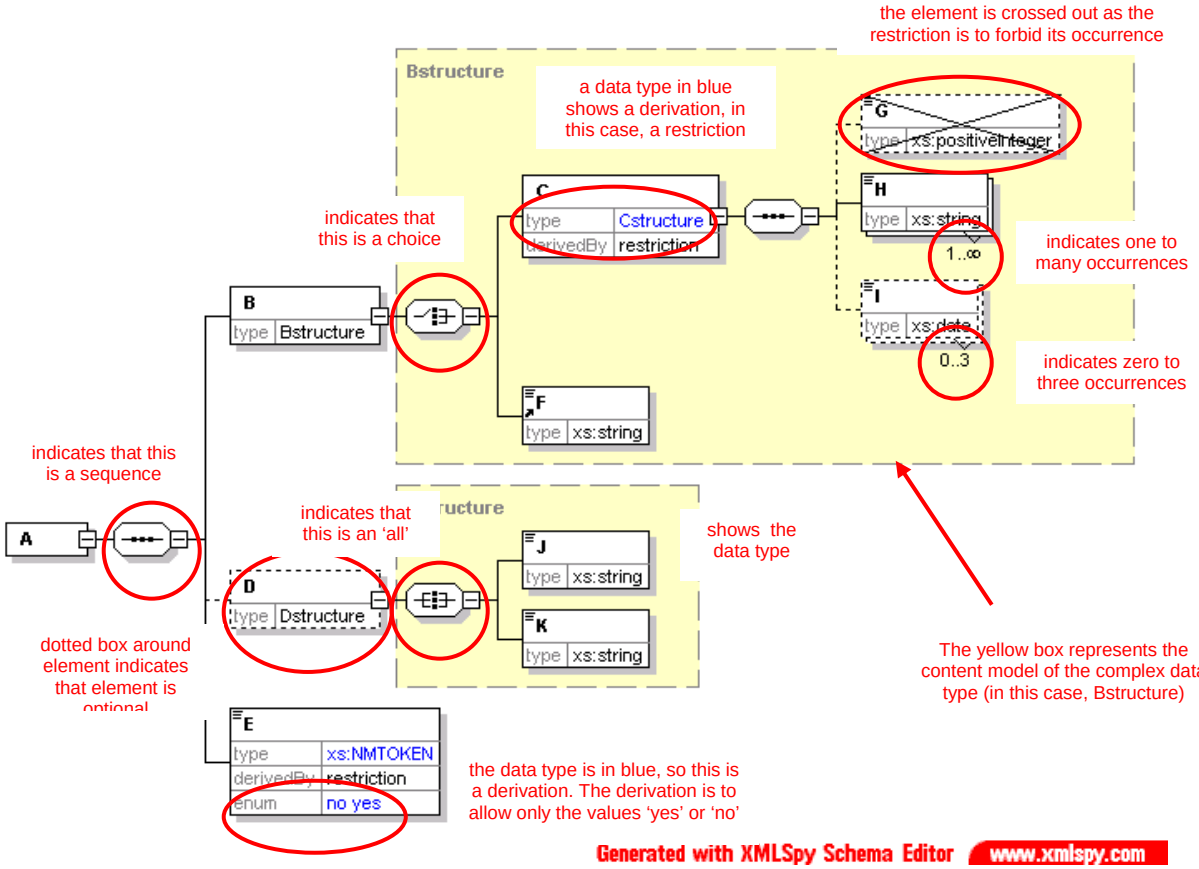
Note that the diagrams in this document do not use the default diagramming options of XML Spy, but have additional information. The additional information to be shown can be set using the menu selections Schema Design | View Config.

In this section, and throughout this document, the prefix "xs" denotes the XML schema namespace `http://www.w3.org/2001/XMLSchema`.

The diagram below represents a simple schema. The *root element* of an *instance* described by this schema is the *element* `A`. The *content model* of this element is a *sequence* of the elements `B`, `D` and `E`. The *element* `B` is of *complex data type* `Bstructure`. This contains a *choice* of either *element* `C` or *element* `F`. *Element* `C` is a *restriction* of another *complex data type* `Cstructure`. In this case, the restriction is to forbid the use of the *element* `G` (which is defined in `Cstructure` as optional). The other *elements* allowed are `H`, which can appear any number of times (but must appear at least once), and `I`, which can appear up to three times (or not at all). *Element* `D` is optional, and of *data type* `Dstructure`. This has a *content model* requiring *all* of *elements* `J` and `K`, which are both of *type* `xs:string`. Finally, *element* `E` is of *simple data type* `Etype`, which is *restricted* from the `xs:NMTOKEN` *data type* by only allowing the values 'yes' and 'no'.

It is important to remember that these diagrams do not include any *attributes*. In this document, these are shown in tables below the diagrams.

The full schema is shown below the diagram.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Paul Spencer
(Boynings Consulting) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="A">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="B" type="Bstructure"/>
        <xs:element name="D" type="Dstructure" minOccurs="0"/>
        <xs:element name="E">
          <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="no"/>
              <xs:enumeration value="yes"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Bstructure">
    <xs:choice>
      <xs:element name="C">
        <xs:complexType>
          <xs:complexContent>
            <xs:restriction base="Cstructure">
              <xs:sequence>
                <xs:element name="G" type="xs:positiveInteger" minOccurs="0"
maxOccurs="0"/>
                <xs:element name="H" type="xs:string" maxOccurs="unbounded"/>
                <xs:element name="I" type="xs:date" minOccurs="0"
maxOccurs="3"/>
              </xs:sequence>
            </xs:restriction>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element ref="F"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="Cstructure">
    <xs:sequence>
      <xs:element name="G" type="xs:positiveInteger" minOccurs="0"/>
      <xs:element name="H" type="xs:string" maxOccurs="unbounded"/>
      <xs:element name="I" type="xs:date" minOccurs="0" maxOccurs="3"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Dstructure">
    <xs:all>
      <xs:element name="J" type="xs:string"/>
      <xs:element name="K" type="xs:string"/>
    </xs:all>
  </xs:complexType>
  <xs:element name="F" type="xs:string"/>
</xs:schema>
```

2.4 EML Message Validation

It is up to each specific system implementation whether it uses these schemas for validation of EML messages for either testing or live use. The recommended approach is to validate incoming messages

against the EML schemas (with the application-specific EML externals schema), then further validate against the relevant Schematron schema. The first stage requires the use of an XML processor (parser) that conforms to W3C XML Schema. The second stage requires either an XSLT processor or a dedicated Schematron processor.

However, an implementation may choose to:

- modify the EML schemas to incorporate those application-specific constraints that can be represented in W3C XML Schema;
- not validate the rules that are encoded as Schematron schemas;
- not perform any validation; or
- develop some alternative validation.

2.5 Namespaces

The message schemas and the core schema are associated with the namespace `urn:oasis:names:tc:evs:schema:eml`. This is defined using the prefix `eml`. The XML Schema namespace `http://www.w3c.org/2001/XMLSchema` is identified by the prefix `xs` and the XML Schema Instance namespace `http://www.w3.org/2001/XMLSchema-instance` by the prefix `xsi`.

Use is also made of namespaces for the Extensible Name and Address Language (xNAL). The Extensible Name Language namespace `urn:oasis:tc:ciq:xsdschema:xNL:2.0` is identified by the prefix `xNL`, and the Extensible Language namespace `urn:oasis:names:tc:ciq:xsdschema:xAL:2.0` by the prefix `xAL`.

2.6 Extensibility

Various elements allow extensibility through the use of the `xs:any` element. This is used both for display information (for example, allowing the sending of HTML in a message) and for local extensibility. Note that careless use of this extensibility mechanism could reduce interoperability.

2.7 Additional Constraints

The EML schemas provide a set of constraints common to most types of elections worldwide. Each specific election type will require additional constraints, for example, to enforce the use of a seal or to ensure that a cast vote is anonymous. It is recommended that these additional constraints be expressed using the Schematron language although other validators, eg OASIS CAM, can be used. This allows additional constraints to be described without altering or interacting with the EML schemas. Any document that is valid to a localization expressed in Schematron must also be a valid EML document.

2.8 Conventions

Within this specification, the following conventions are used throughout:

- Diagrams are shown as generated by XML Spy 2007 which was also used to generate the schemas and samples. These diagrams show element content, but not attributes
- Elements and attributes in schemas are identified by partial XPath expressions. Enough of a path is used to identify the item without putting in a full path.

2.9 Metadata

Some messages need information relating to the issuing of them, such as the issue date, who issued them etc. This is most likely to be a requirement for the 330 message but is equally applicable to 120, 130, 230, 350a and several others. For that reason, it is useful to make this optional information available in the header. The information usually consists of: managing authority, date of issue, start of list period (used for changes to the list to indicate the start of the period for which changes are being shown), end of list period (i.e. the date of the snapshot of the list).

3 Processing using Schematron

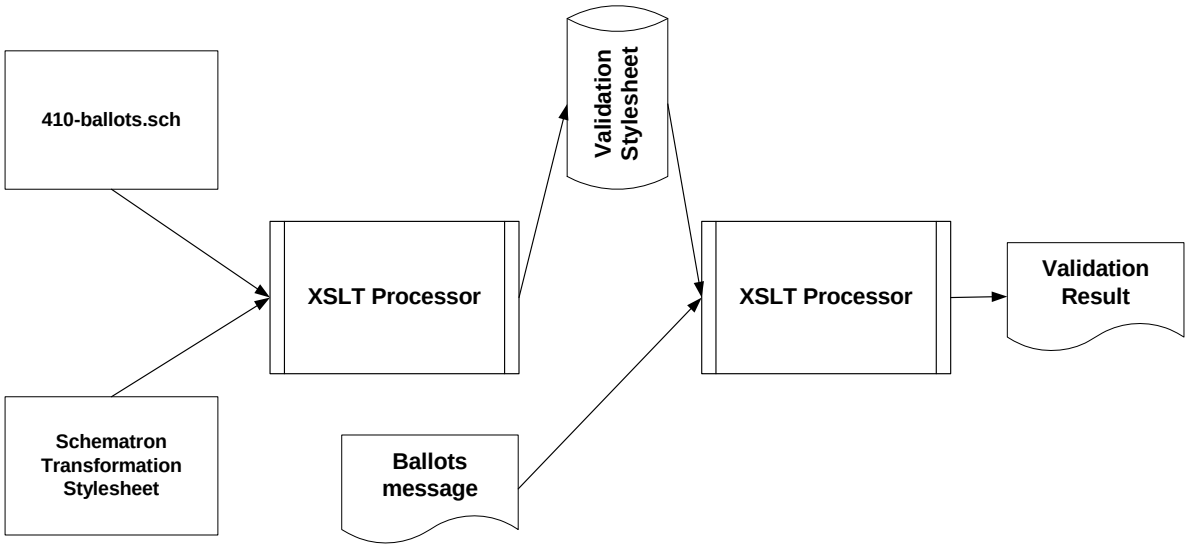
This section gives a short introduction to how validation can be achieved using Schematron schemas and an XSLT processor. Alternatively, direct validation using the Schematron schemas can be achieved using a dedicated Schematron processor.

3.1 Validation using Schematron Schemas

A Schematron schema is an XML document that can be converted to XSLT using an XSLT stylesheet. There is a published stylesheet (skeleton1-5.xslt) that can be used to achieve this. This produces an HTML output from the validation. A separate stylesheet can be produced that will create an output to the specification below. This stylesheet can import the skeleton and just over-ride those aspects where changes are required.

This stylesheet can be used once on each Schematron schema to produce the XSLT file that will be used for validating a specific message type. This stylesheet is then used to transform the incoming EML message into an error report based on the additional constraints.

The process is shown in the diagram below.



4 Splitting of Messages

There is sometimes a need to split long messages into several parts. By their nature, each of these messages will contain a small amount of background information and a single element type that is repeated many times. For example, the 330-electionlist message can have many VoterDetails elements.

When a message is split, each part must be a complete, valid EML document. This will contain all the elements required by EML and the specific application. Those parts outside the repeated element that relate to the message as a whole, such as the TransactionId, must have the same values in each part message. The values of those elements and attributes that relate to an individual part message, such as the SequenceNumber, may vary between the individual part messages. Information in the EML element indicates the sequence number of the message and the number of messages in the sequence. Each message in the sequence must contain the same TransactionId, and must indicate the repeated element according to the table below. Only the messages shown in the table may be split in this way.\

Message	Repeated Element
330-electionlist	VoterDetails
340-pollinginformation	Polling
410-ballots	Ballot
460-votes	CastVote
470-vtokenlog	VTokens
480-auditlog	LoggedSeal

For ease of implementation, a message that can be split may contain the elements used for splitting even if the entire message is sent in one piece. In this case, the values of SequenceNumber and NumberInSequence will both be "1".

5 Error Messages

The 130 schema is used to define a message for reporting errors in EML messages.

Error messages are given codes. These fall into one of five series:

1000	XML well-formedness or Schema validation error
2000	Seal error
3000	EML rule error
4000	Localization rule error
5000	System specific error

If the error type is not message-specific (or is a general rule applying to several schemas), the series reference above is used. If it is message-specific, the last three digits of the error series (and possibly a final alpha character) reflect the message type. A three digit error code is appended to the series code, separated by a hyphen.

An error code relating to a localisation applicable to all message types could therefore be 4000-001. One specific to the localization of schema 110 could be 4110-002.

5.1 All Schemas

5.1.1 XML well-formedness or Schema validation error

Error code	Error Description
1000-001	Message is not well-formed
1000-002	Message is not valid

5.1.2 Seal Errors

Error code	Error Description
2000-001	The Seal does not match the data

5.1.3 EML Additional Rules

The following rules apply to messages regardless of localization. One of the two rules on splitting will apply to each message type as described in the table below.

Error Code	Error Description
3000-001	If there are processing units in the `AuditInformation`, one must have the role of sender
3000-002	If there are processing units in the `AuditInformation`, one must have the role of receiver
3000-003	This message must not contain the elements used for splitting
3000-004	The value of the `Id` attribute of the EML element is incorrect
3000-005	The message type must match the `Id` attribute of the `EML` element

3000-006	All messages that are split must include the correct sequenced element name.

236

	3000-003	3000-006
110	✓	
120	✓	
130	✓	
210	✓	
220	✓	
230	✓	
310	✓	
330		✓
340		✓
350 a	✓	
350 b	✓	
350 c	✓	
360 a	✓	
360 b	✓	
410		✓
420	✓	
430	✓	
440	✓	
445	✓	
450		✓
460		✓
470		✓
480		✓
510	✓	
520	✓	
610	✓	
620	✓	
630	✓	

237

6 EML Core Components

The EML Core schema contains elements and data types that are used throughout the e-voting schemas.

To help message schema diagrams fit on the page, these elements and data types are not expanded each time they appear in other diagrams.

The following schema components are defined in the EML Core:

Elements	Complex Data Types	Simple Data Types
Accepted	AffiliationIdentifierStructure	ConfirmationReferenceType
Affiliation	AffiliationStructure	CountingAlgorithmType
AffiliationIdentifier	AgentIdentifierStructure	DateType
Agent	AgentStructure	EmailType
AgentIdentifier	AreaStructure	ErrorCodeType
Area	AuditInformationStructure	GenderType
AuditInformation	AuthorityIdentifierStructure	LanguageType
AuthorityIdentifier	BallotIdentifierRangeStructure	MessageTypeType
BallotIdentifier	BallotIdentifierStructure	SealUsageType
BallotIdentifierRange	BinaryStructure	ShortCodeType
Candidate	CandidateIdentifierStructure	TelephoneNumberType
CandidateIdentifier	CandidateStructure	VotingChannelType
ContactDetails	ChannelStructure	VotingMethodType
ContestIdentifier	ComplexDateRangeStructure	VotingValueType
CountQualifier	ContactDetailsStructure	WriteInType
CountingAlgorithm	ContestIdentifierStructure	YesNoType
DocumentIdentifier	CountQualifierStructure	
ElectionIdentifier	DocumentIdentifierStructure	
ElectionStatement	ElectionGroupStructure	
EventIdentifier	ElectionIdentifierStructure	
EventQualifier	EmailStructure	
Gender	EMLStructure	
Logo	Endorsement	
ManagingAuthority	EventIdentifierStructure	
MaxVotes	EventQualifierStructure	
MessageType	IncomingGenericCommunicationStructure	
MinVotes	InternalGenericCommunicationStructure	
NominatingOfficer	LogoStructure	
NumberInSequence	ManagingAuthorityStructure	
NumberOfPositions	MessagesStructure	
Period	NominatingOfficerStructure	
PersonName	OutgoingGenericCommunicationStructure	
PollingDistrict	PartyStructure	
PollingPlace	PartyIdentiferStructure	

Elements	Complex Data Types	Simple Data Types
Position	PeriodStructure	
PreferredChannel	PictureDataStructure	
PreviousElectoralAddress	PollingDistrictStructure	
Profile	PollingPlaceStructure	
Proposal	PositionStructure	
ProposalIdentifier	ProcessingUnitStructure	
Proposer	ProposalIdentifierStructure	
Proxy	ProposalStructure	
ReferendumOptionIdentifier	ProposerStructure	
ReportingUnitIdentifier	ProxyStructure	
ResponsibleOfficer	ReferendumOptionIdentifierStructure	
ScrutinyRequirement	ReportingUnitIdentifierStructure	
Seal	ResponsibleOfficerStructure	
SequenceNumber	ScrutinyRequirementStructure	
TransactionId	SealStructure	
VoterId	SimpleDateRangeStructure	
VoterName	TelephoneStructure	
VotingChannel	VoterIdentificationStructure	
VotingMethod	VoterInformationStructure	
VToken	VTokenStructure	
VTokenQualified	VTokenQualifiedStructure	

6.1 Simple Data Types

The simple data types are included here with their base data types and any restrictions applied.

6.1.1 ConfirmationReferenceType

xs:token.

The reference generated once the confirmation of a vote has been completed.

6.1.2 CountingAlgorithmType

xs:token

The method of counting used for more complex forms of election.

6.1.3 DateType

Union of xs:date and xs:dateTime

There are several possible dates associated with an election. Some of these can be either just a date or have a time associated with them. These can use this data type.

6.1.4 EmailType

xs:token with restrictions.

Restrictions: `xs:maxLength`: 129

`xs:pattern`: [^@]+@[^@]+

This type is a simple definition of an email address, pending a more complete description that is widely accepted in industry and government. It allows any characters except the @ symbol, followed by an @ symbol and another set of characters excluding this symbol.

6.1.5 ErrorCodeType

xs:token

One of a pre-defined set of error codes as described in the section "Error Messages".

6.1.6 GenderType

xs:token with restrictions.

Restrictions: `xs:enumeration`: male, female, unknown

The gender of a voter or candidate. Options are male, female or unknown (unknown is not allowed in all contexts).

6.1.7 LanguageType

xs:language

Declaration of the type of language used in the election.

6.1.8 MessageTypeType

xs:NMTOKEN

This is the alphanumeric type of the message (e.g. 440 or 350a). This may be required for audit purposes.

6.1.9 SealUsageType

xs:NMTOKEN with restrictions.

Restrictions: `xs:enumeration`: receiver, sender

Indicates whether a device logging a seal was the sender or receiver of the seal.

6.1.10 ShortCodeType

xs:NMTOKEN

This identifies an aspect of the election (such as a contest or candidate) when voting using SMS or other voting mechanisms where a short identifier is required.

6.1.11 TelephoneNumberType

xs:token with restrictions.

Restrictions: `xs:maxLength`: 35

`xs:minLength`: 1

`xs:pattern`: \+?[0-9\(\)\-\s]{1,35}

Since this must allow for various styles of international telephone number, the pattern has been kept simple. This allows an optional plus sign, then between 1 and 35 characters with a combination of digits, brackets, the dash symbol and white space. If a more complete definition becomes widely accepted in industry and government, this will be adopted.

6.1.12 VotingChannelType

xs:token with restrictions.

Restrictions: `xs:enumeration`: SMS, WAP, digitalTV, internet, kiosk, polling, postal, telephone, other

This type exists to hold the possible enumerations for the channel through which a vote is cast.

SMS is the Short Message Service (text message). WAP is the Wireless Access Protocol.

If other is used, it is assumed that those managing the election will have a common understanding of the channel in use.

6.1.13 VotingMethodType

xs:token with restrictions.

Restrictions: `xs:enumeration`: AMS, FPP, OPV, SPV, STV, NOR, cumulative, approval, block, partylist, partisan, supplementaryvote, other

The VotingMethod type holds the enumerated values for the type of election (such as first past the post or single transferable vote). The meanings of the acronyms are:

AMS – Additional Member System

FPP - First Past the Post

NOR – Norwegian Voting

OPV - Optional Preferential Voting

SPV - Single Preferential Vote

STV - Single Transferable Vote

6.1.14 VotingValueType

xs:positiveInteger.

Indicates a value assigned when voting for a candidate or referendum option. This might be a weight or preference order depending on the election type.

6.1.15 WriteInType

xs:token

Indicates the type of Write-ins allowed, eg allowed, strikeout, none.

6.1.16 YesNoType

xs:token with restrictions.

Restrictions: `xs:enumeration`: no, yes

This is a simple enumeration of yes and no and is used for elements and attributes that can only take these binary values.

6.2 Complex Data Types

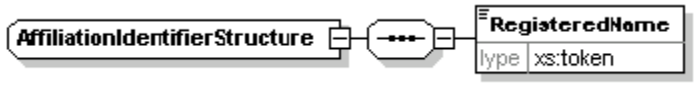
The choice between defining an element or a data type for a reusable message component is a significant design issue. It is widely accepted as good practice to use element declarations when there is good reason to always refer to an element by the same name and there is no expectation of a need to derive new definitions. In all other cases, data type declarations are preferable. The term schema component is used to refer to elements and data types collectively.

When defining a complete mark-up language, limiting the use of elements and types can restrict further development of the language. For that reason, both data types and elements are defined in EML. Only where an element is an example of a primitive or derived data type defined in XML Schema Descriptions is no explicit data type defined within EML.

In use, it is expected that, for example:

- A voting token will always have an element name VToken and so will use the element name.
- A logo or a map have similar definitions, so both use the PictureDataStructure. There is no PictureData element.
- Within voter identification, some elements will usually need to be made mandatory and so a schema will specify a new element based on the VoterIdentificationStructure data type.

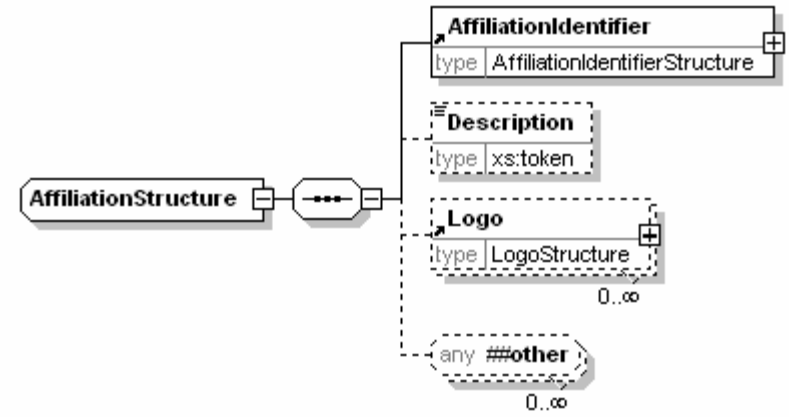
6.2.1 AffiliationIdentifierStructure



Element	Attribute	Type	Use	Comment
AffiliationIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

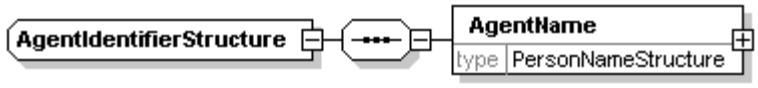
This data type is used to identify an affiliation, such as a political party. The identifier indicates the official name and ID of the organization. It supports use of a short code for voting systems such as SMS, and an expected confirmation reference for security systems that require this.

6.2.2 AffiliationStructure



AffiliationStructure data type indicates membership of some organization such as a political party. The description will normally be used to indicate the name usually associated with the organization, and so is the value that will usually be shown on a ballot. An organization may indicate several logos, each with a role. For example, one role might indicate that the logo should be used on a ballot paper. Each logo can be identified by a URL or sent as a Base64 encoded binary value. In the latter case, the format of the logo (BMP, TIFF, PNG, GIF or JPEG) must be indicated.

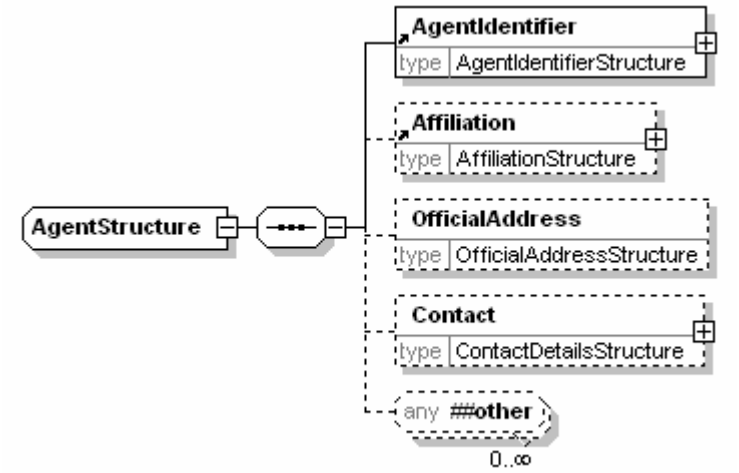
6.2.3 AgentIdentifierStructure



Element	Attribute	Type	Use	Comment
AgentIdenttifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

363 The agent identifier contains a name and ID. The data type for the name is localized using the EML
364 externals schema.

365 6.2.4 AgentStructure



366

Element	Attribute	Type	Use	Comment
AgentStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	Role	xs:token	optional	

367 A candidate in an election can have one or more agents, each agent having a specific role, identified by
368 the Role attribute. For example, an agent may be allowed access to the count, but not to amend details of
369 the candidate.

370 The agent has an identifier, comprising a name and ID, and an affiliation. He or she also has an official
371 address and a standard set of contact details.

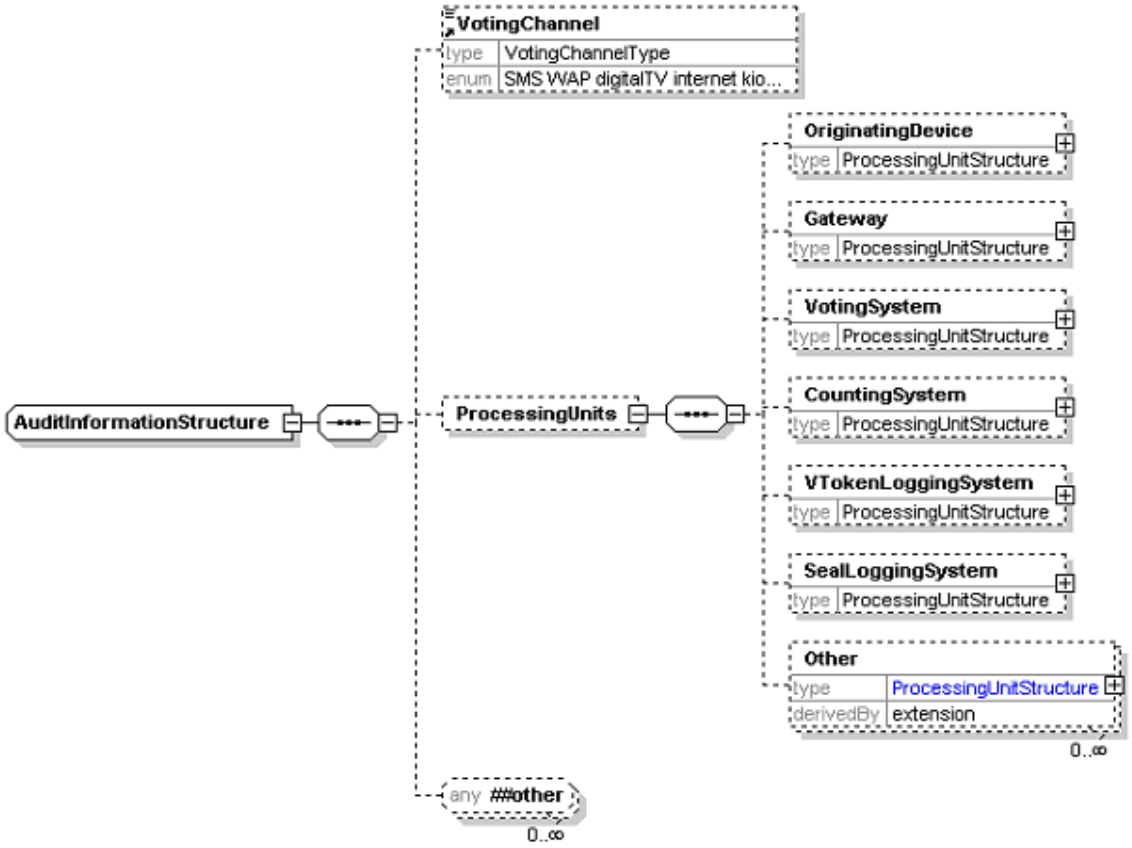
372 6.2.5 AreaStructure

373 The AreaStructure is an extension of xs:token to add the following attributes:

Element	Attribute	Type	Use	Comment
AreaStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	Type	xs:token	optional	

374 This data type is used to define elements defining the geographical area covered by a contest. The Type
375 attribute is used to indicate the type of area, such as "county".

6.2.6 AuditInformationStructure



Element	Attribute	Type	Use	Comment
Other	Role	xs:token (restricted)	required	Standard attribute for a ProcessingUnitStructure
	Type	xs:token	required	Additional attribute for this element

The AuditInformationStructure is used to define an element to provide information for audit purposes. It allows the voting channel in use to be described, with the identities of those devices that have participated in the message being sent. Each device has an attribute to describe its role (see ProcessingUnitStructure).

Where a device does not fit any of the categories here, it can be described as Other with the addition of a Type attribute.

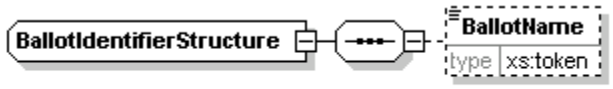
6.2.7 AuthorityIdentifierStructure

The AuthorityIdentifierStructure is an extension of xs:token to add the following attributes:

Element	Attribute	Type	Use	Comment
AuthorityIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

This data type defines information to identify an election authority. This may include a system ID and text description.

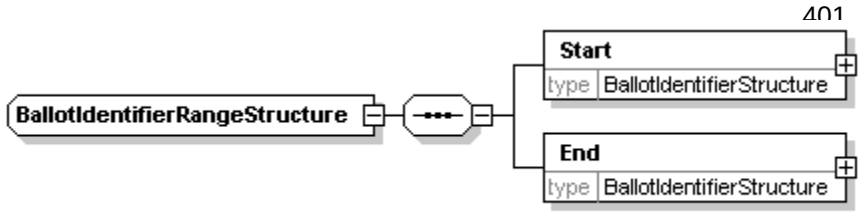
6.2.8 BallotIdentifierStructure



Element	Attribute	Type	Use	Comment
BallotIdentifierStructure	Id	xs:NMTOKEN	required	
	DisplayOrder	xs:positiveInteger	optional	

This data type is used to define an element that is an identifier for a ballot. This will usually use the Id attribute as the identifier, but might use a name to indicate a set of identical ballots. Elements using this data type will usually only be used for paper ballots.

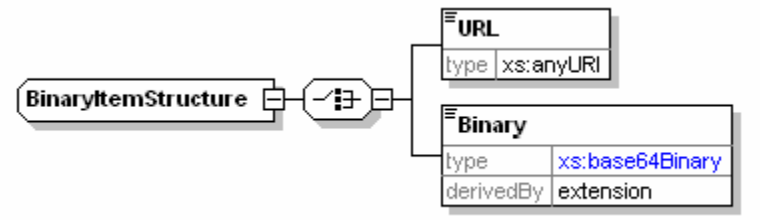
6.2.9 BallotIdentifierRangeStructure



Element	Attribute	Type	Use	Comment
BallotIdentifierRangeStructure	Colour	xs:token	optional	

This data type is used to define an element that identifies a range of ballots. This might be used, for example, to assign ranges of ballot identifiers to different reporting units for a contest. It is unlikely that the ballot name would be used when defining range, the Id attribute being used instead. Elements using this data type will usually only be used for paper ballots.

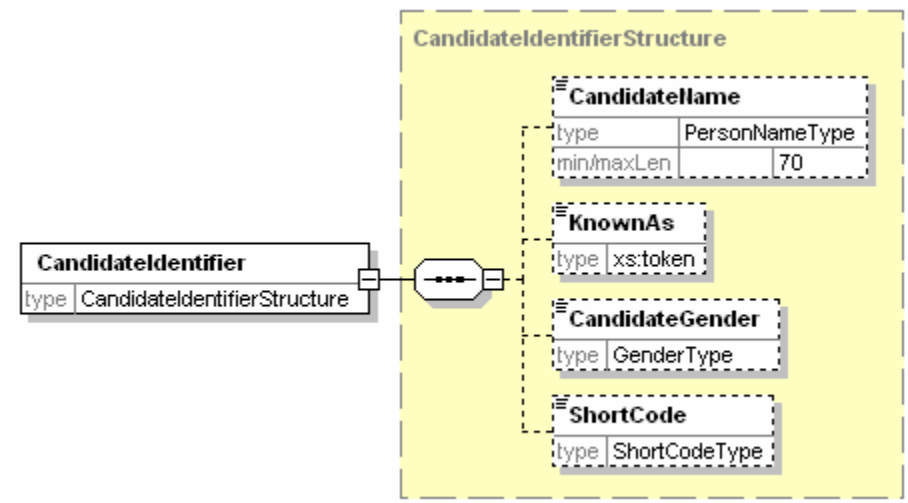
6.2.10 BinaryItemStructure



Element	Attribute	Type	Use	Comment
BinaryItemStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	ItemType	Xs:token	optional	
	Verified	YesNoType	optional	
	Problem	YesNoType	optional	
	Notes	Xs:string	optional	
	Role	Xs:token		
Binary	Format	xs:NMTOKEN (restricted)	required	

Where a binary (fingerprint, logo, map, photo,) is provided, it may be given as either a link or as Base64 encoded binary data. In the latter case, the format of the binary (bmp, gif, jpeg, png or tiff) must be indicated using the Format attribute of the Binary element.

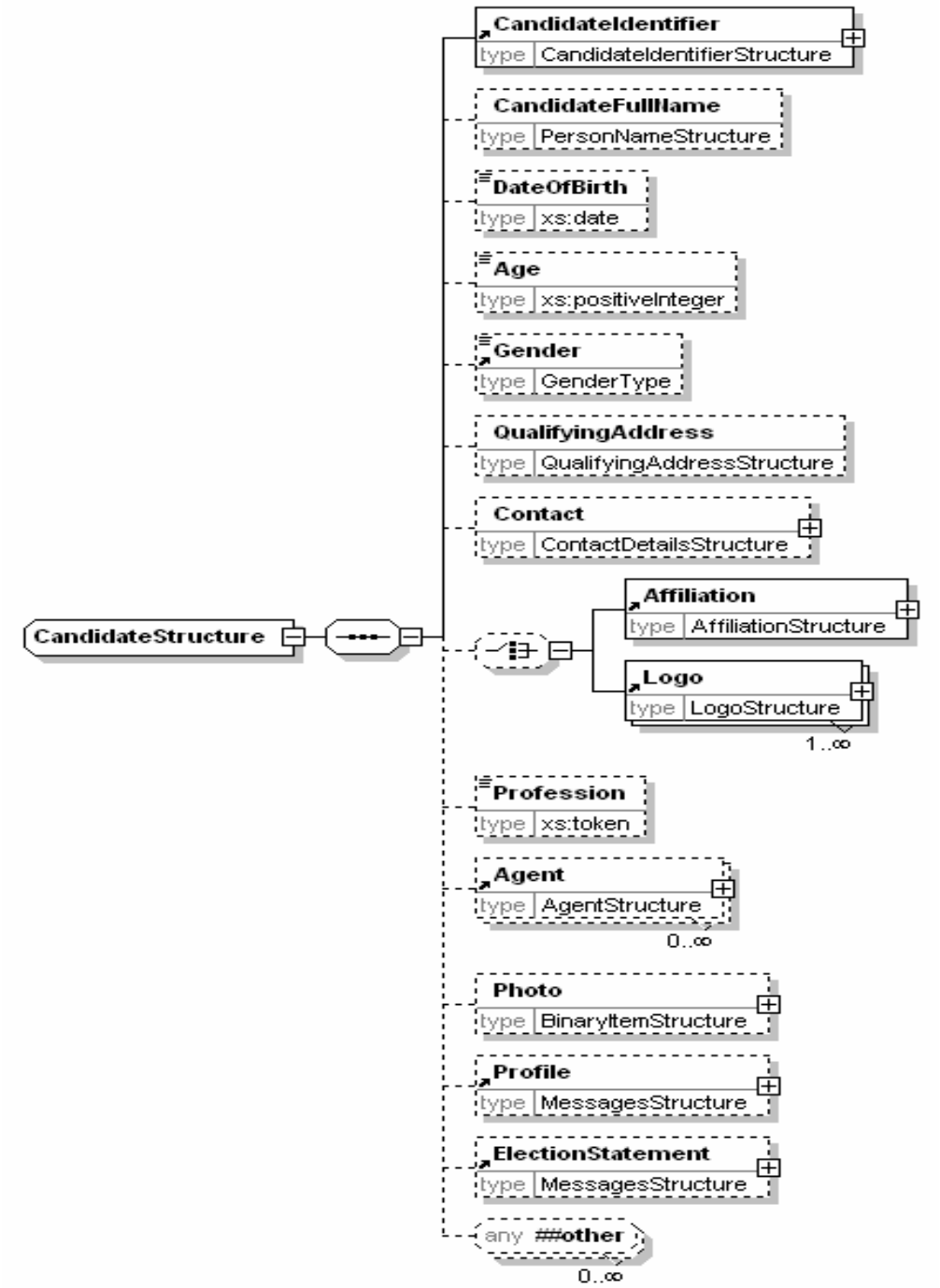
6.2.11 CandidateIdentifierStructure



Element	Attribute	Type	Use	Comment
CandidateIdentifierStructure	Id	xs:NMTOKEN	required	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

The candidate identifier indicates a system ID for the candidate and the candidate's name as it will appear in a ballot. Sometimes an additional line is required on the ballot to help identify the candidate. This will use the `KnownAs` element of the candidate identifier. A short code can also be included, either for SMS voting or where the security mechanism in place requires it. An `ExpectedConfirmationReference` attribute also allows for security mechanisms where the confirmation reference may be different for each combination of voter and candidate.

425 ## 6.2.12 CandidateStructure



426

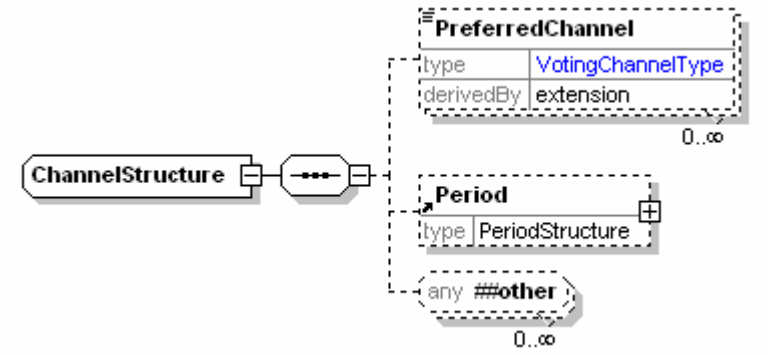
Element	Attribute	Type	Use	Comment
CandidateStructure	Independent	YesNoType	optional	
	DisplayOrder	xs:positiveInteger	optional	

427 The candidate description includes all the information required about the candidate. In different
428 messages, the amount of information is reduced, either by restricting the information in EML or as part of
429 a localisation.

The candidate has an identifier. The full name of the candidate may also be provided, and whether the candidate is an independent. This is supplied as an attribute rather than affiliation as certain election types treat independents differently from other candidates, even though they may define an affiliation.

The candidate profile describes the candidate. The election statement describes the opinions of the candidate. Optionally, a photo may be included, either as a link or as Base64 encoded binary.

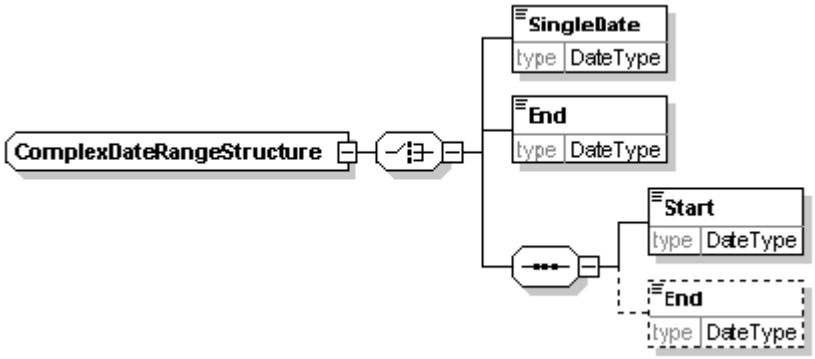
6.2.13 ChannelStructure



Element	Attribute	Type	Use	Comment
PreferredChannel	Fixed	Yes/NoType	optional	

This data type is used to describe the voter's preferred channel for casting of the vote and the period for which that preference is valid.

6.2.14 ComplexDateRangeStructure

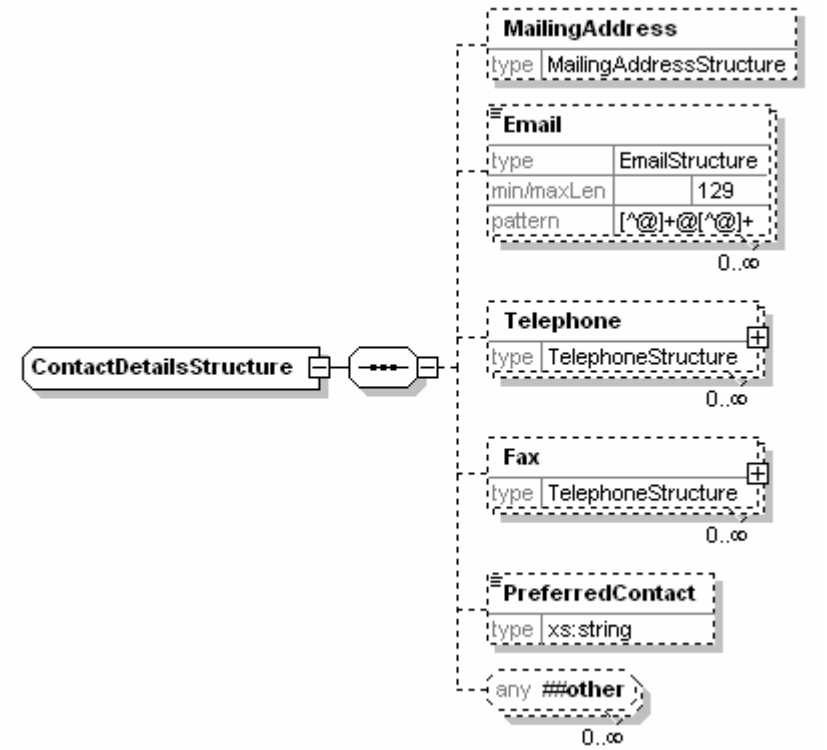


Element	Attribute	Type	Use	Comment
ComplexDateRangeStructure	Type	xs:token	required	

This data type is used to describe ranges of dates or dates and times. Each date can be a single date, a start date, an end date or include both start and end dates.

The Type attribute is used to indicate the purpose of the date (e.g. "deadline for nominations").

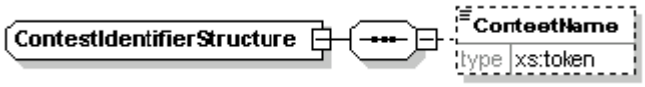
6.2.15 ContactDetailsStructure



Element	Attribute	Type	Use	Comment
ContactDetailsStructure	DisplayOrder	xs:positiveInteger	optional	

This data type is used in many places throughout the EML schemas. The mailing address uses whatever format is defined in the EML externals schema document. Where several addresses or numbers can be given (for example, email addresses), there is a facility to indicate whichever is preferred. The overall preferred method of contact can also be provided by placing an XPath to the preferred method in the `PreferredContact` element.

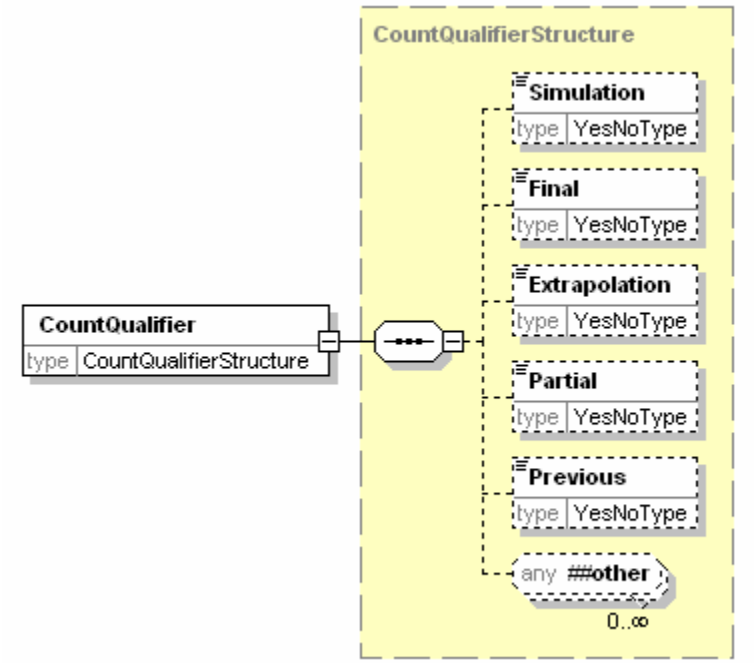
6.2.16 ContestIdentifierStructure



Element	Attribute	Type	Use	Comment
ContestIdentifierStructure	Id	xs:NMTOKEN	required	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	

This data type is used to define an element that is an identifier for a contest. It holds a name and ID. A short code can also be included, for example, for SMS voting.

6.2.17 CountQualifierStructure



This allows for an indication of whether the count is final or not, and for the count to be either simulated or extrapolated.

6.2.18 DocumentIdentifierStructure

The DocumentIdentifierStructure is an extension of xs:token to add the following attribute:

Element	Attribute	Type	Use	Comment
DocumentIdentifierStructure	Href	xs:anyURI	required	

This allows identification of external documents relating to an event, election or contest. The document can have a name and URL.

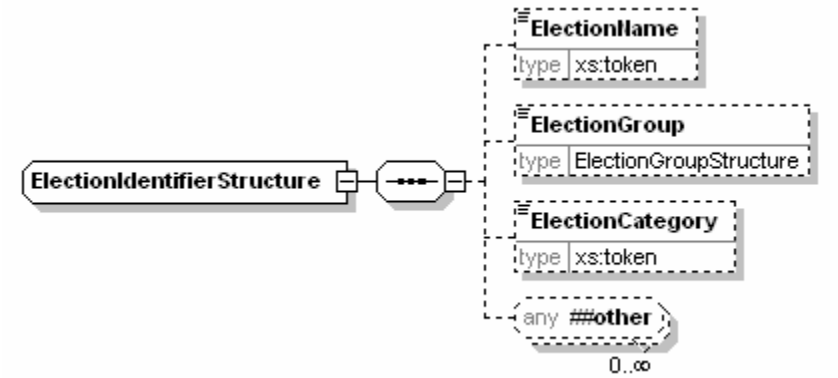
6.2.19 ElectionGroupStructure

The ElectionGroupStructure is an extension of xs:token to add the following attribute:

Element	Attribute	Type	Use	Comment
DocumentIdentifierStructure	Id	xs:token	required	

The election group is used to group a number of elections together. This could be required, for example, under the additional member system, where two elections are held, the result of one influencing the result of the other. It could also be used at a company AGM, where proposals might be grouped for display purposes.

6.2.20 ElectionIdentifierStructure



Element	Attribute	Type	Use	Comment
ElectionIdentifierStructure	Id	xs:NMTOKEN	required	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	

The election identifier is used wherever the election needs to be specified. There is an `Id` attribute, which can often be used on its own to identify the election. In other cases, particularly where the content of a message is to be displayed, the election name can also be provided. The election group is used to group a number of elections together as described above.

The election category is used in messages where several elections are included in the message, but may be treated differently under localisation rules. Each election that requires different treatment will be given a category unique within that election event, allowing a Schematron processor to distinguish between the elections.

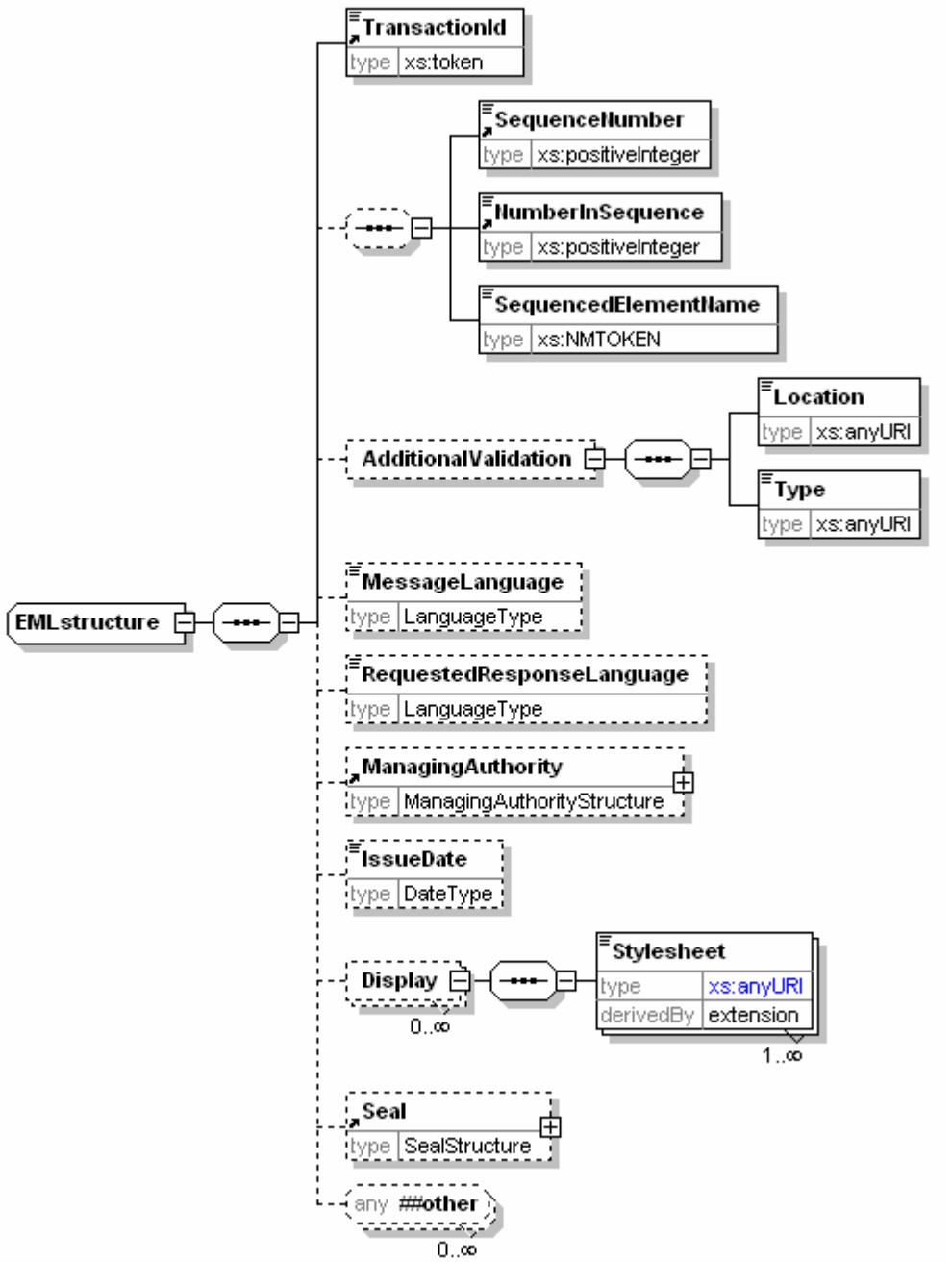
6.2.21 EmailStructure

The EmailStructure is an extension of the EmailType to add the following attribute:

Element	Attribute	Type	Use	Comment
EmailStructure	Preferred	YesNoType	optional	

The Preferred attribute is used to distinguish which of several email addresses to use.

6.2.22 EMLstructure

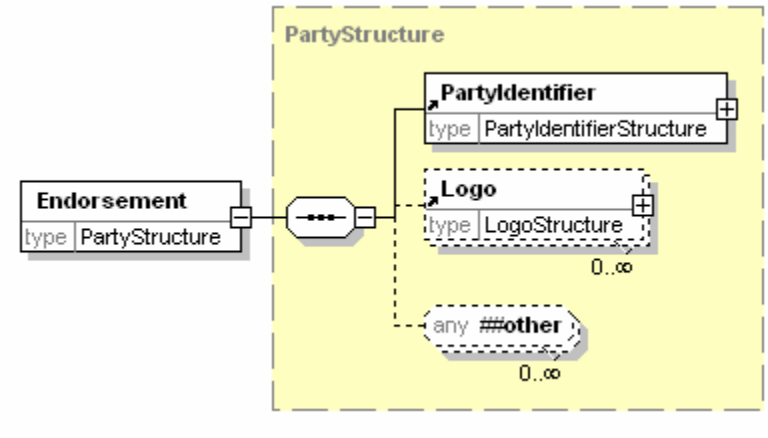


Element	Attribute	Type	Use	Comment
EMLstructure	Id	MessageTypeType	required	
	SchemaVersion	xs:NMTOKEN	requried	
Stylesheet	Type	xs:token	required	

The EML element defined by this data type forms the root element of all EML documents. The transaction ID is used to group messages together, for example, when they are split using the message splitting mechanism. This mechanism is implemented using the next three elements. The optional message language indicates the language of the message using ISO 639 three letter language codes, while the requested response language can be used to indicate the preferred language for a response. This element is used in messages from the voter or candidate to the election organizers.

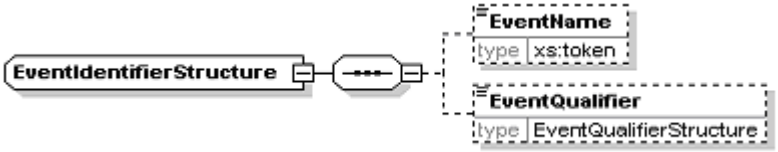
The display element allows the definition of stylesheets to display the message. Multiple stylesheets can be declared. When displaying on the web, the first is likely to be an XSLT stylesheet, while the second might describe a CSS stylesheet to be incorporated as well. The `Type` attribute of the `Stylesheet` element should contain a media types as defined in RFC 2046 Pt 2 [1] using the list of media types defined by IANA, for example, text/xsl. The final element defined is the seal, which is used to seal the complete message.

6.2.23 Endorsement



The endorsement element is used to show which political party specifically endorses a candidate. This can be different from the party(ies) that the candidate says he/she is affiliated to.

6.2.24 EventIdentifierStructure



Element	Attribute	Type	Use	Comment
EventIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

The event identifier is used wherever the election event needs to be specified. There is an Id attribute, which can often be used on its own to identify the event. In other cases, particularly where the content of a message is to be displayed, the event name can also be provided. The event qualifier is used to further identify the event.

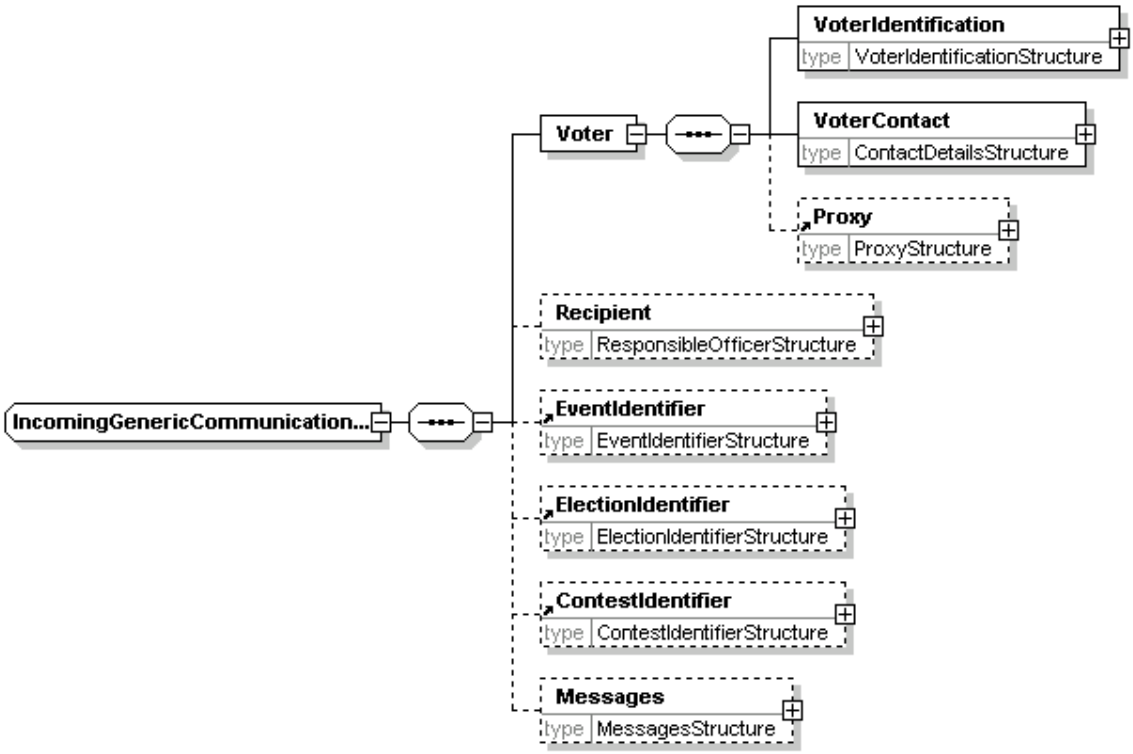
6.2.25 EventQualifierStructure

The EventQualifierStructure is an extension of `xs:token` to add the following attribute:

Element	Attribute	Type	Use	Comment
EventQualifierStructure	Id	xs:NMTOKEN	optional	

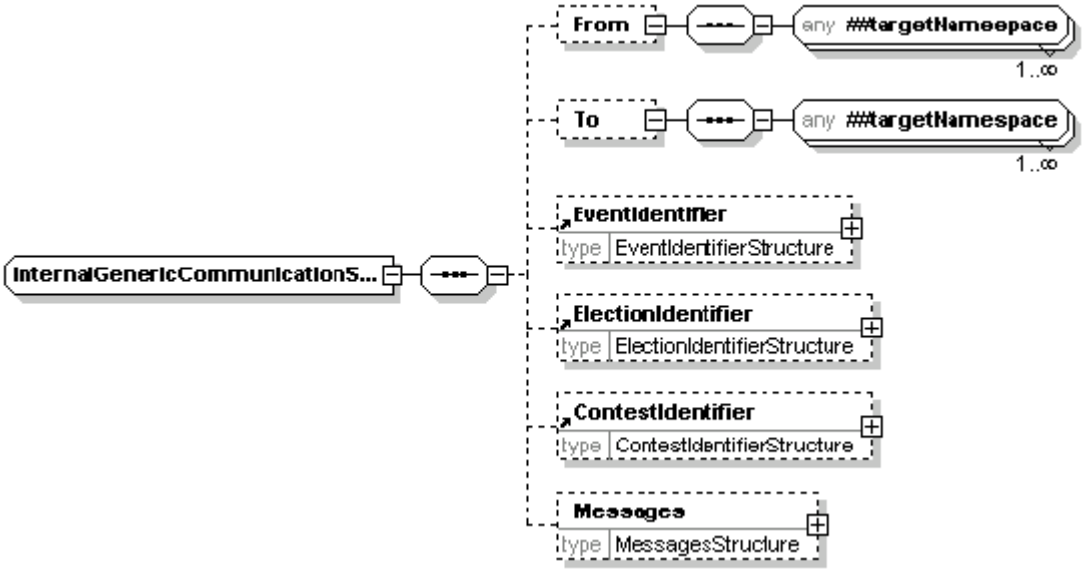
The event qualifier is used to further identify the event. For example, there might be "County Elections" covering an entire country, but the events are organized at a county level, so the event qualifier would identify the county.

6.2.26 IncomingGenericCommunicationStructure



This data type provides a common structure for incoming communications. Individual message types, such as that used for selecting a preferred voting channel (schema 360b) are based on extensions of this type.

6.2.27 InternalGenericCommunicationStructure



This data type provides a common structure for communications between entities involved in the organisation of an election. Individual message types are based on extensions of this type. The sender and recipient can use any elements defined within EML.

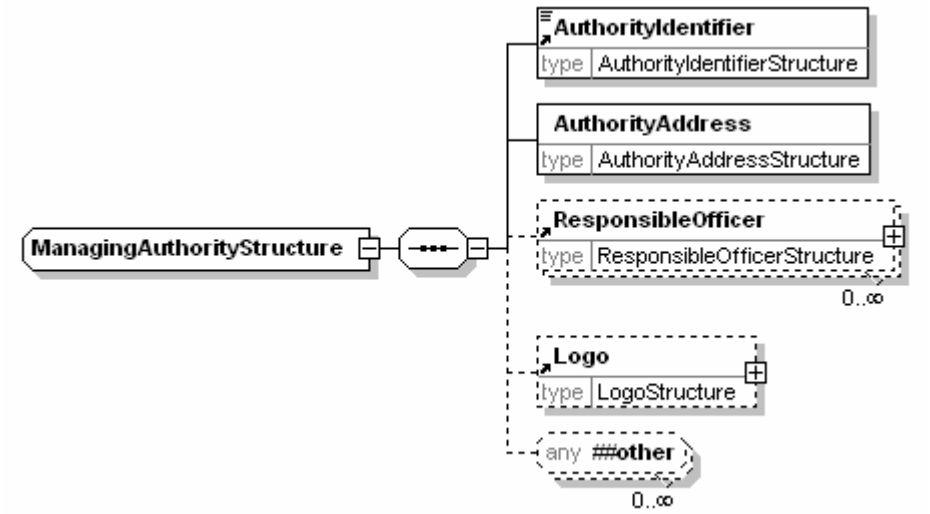
6.2.28 LogoStructure

The LogoStructure is an extension of the BinaryItemStructure to add attributes:

Element	Attribute	Type	Use	Comment
LogoStructure	Id	xs:NMTOKEN	optional	Standard attribute for a BinaryItemStructure
	DisplayOrder	xs:positiveInteger	optional	Standard attribute for a BinaryItemStructure
	ItemType	xs:token	optional	
	Verified	YesNoType	optional	
	Problem	YesNoType	optional	
	Notes	Xs:string	optional	
	Role	xs:token	optional	Additional attribute for this element

This element extends the binary item structure by adding attributes to define the type and role of the logo. This can be used to indicate the purpose of the logo (for example, it is to appear on a ballot).

6.2.29 ManagingAuthorityStructure



The managing authority is the body responsible for an election event, election, contest or reporting unit. In most cases, not all of these will be required, but sometimes more than one is necessary. For example, an election using the additional member system might be organized on a regional basis, whilst local authorities organise their local election events. In this case, the region becomes the managing authority for the contest, whilst the local authority is the managing authority for the event. There will also be an authority responsible for the overall conduct of the election, although this information might not be required.

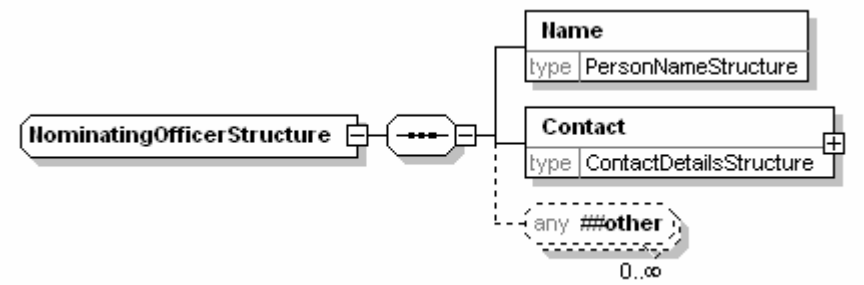
The managing authority indicates the authority name, address, Id, any logo that might be required for display during the election and a list of responsible officers.

6.2.30 MessageStructure

Element	Attribute	Type	Use	Comment
MessagesStructure	DisplayOrder	xs:positiveInteger	optional	
Message	Format	xs:topken	optional	
	Type	xs:token	optional	
	Lang	LanguageType	optional	

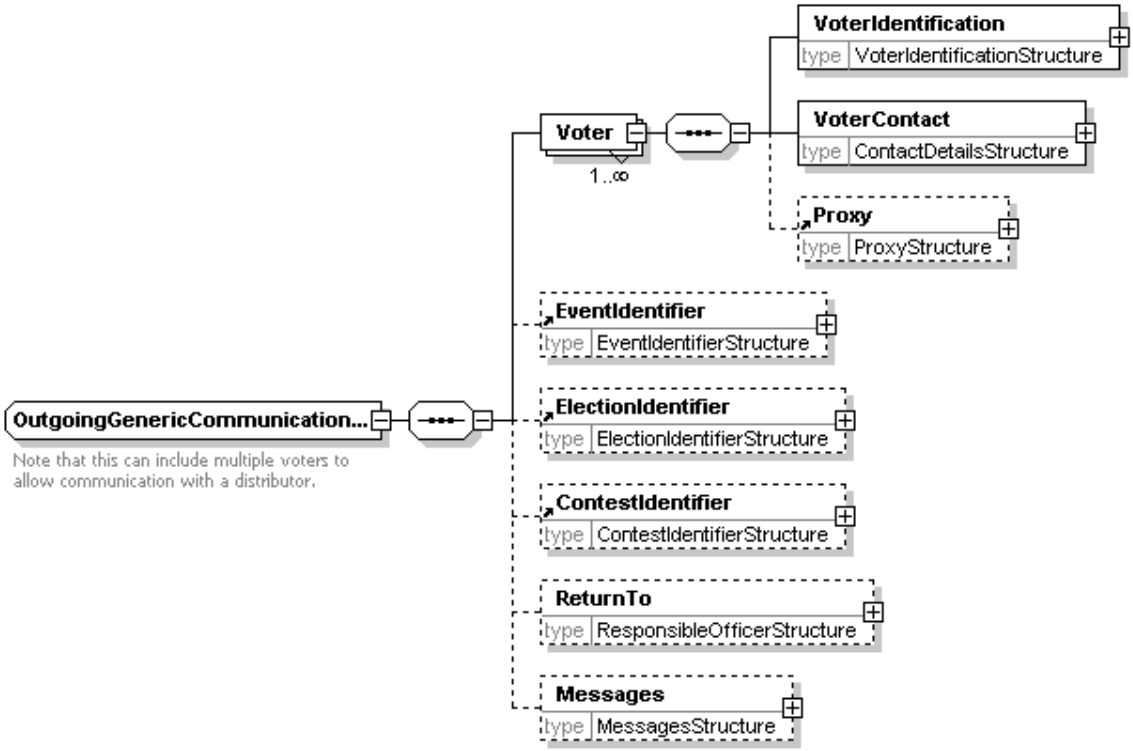
The Message element is of 'mixed' type, so can have both text and element content. The intention is that it should have one or the other. The Message element has three attributes: Lang is used to indicate the language of the message using ISO 639 three letter language codes, Format indicates the format of element content using the media types definition from RFC 2046 Pt 2 [1] and the list of media types defined by IANA, for example, text/html, and Type indicates the purpose of the message.

6.2.31 NominatingOfficerStructure



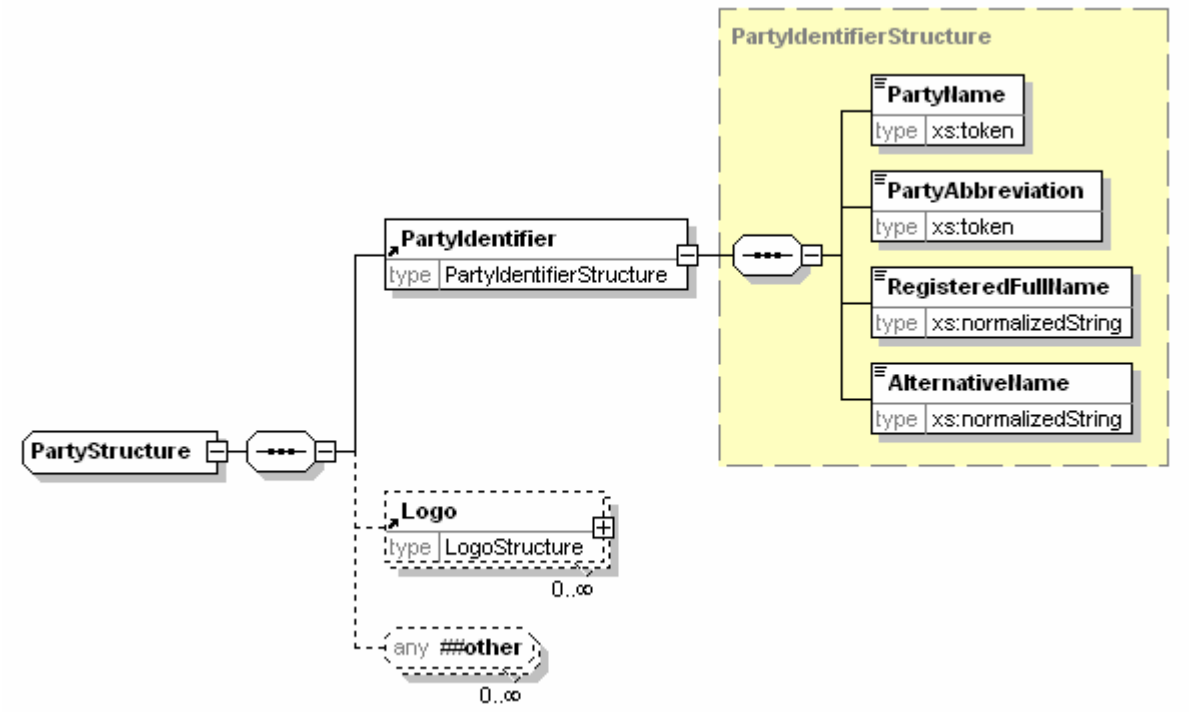
The nominating officer is the person nominating a party in an election run under, for example, the party list system. The data type includes a name and contact information.

6.2.32 OutgoingGenericCommunicationStructure



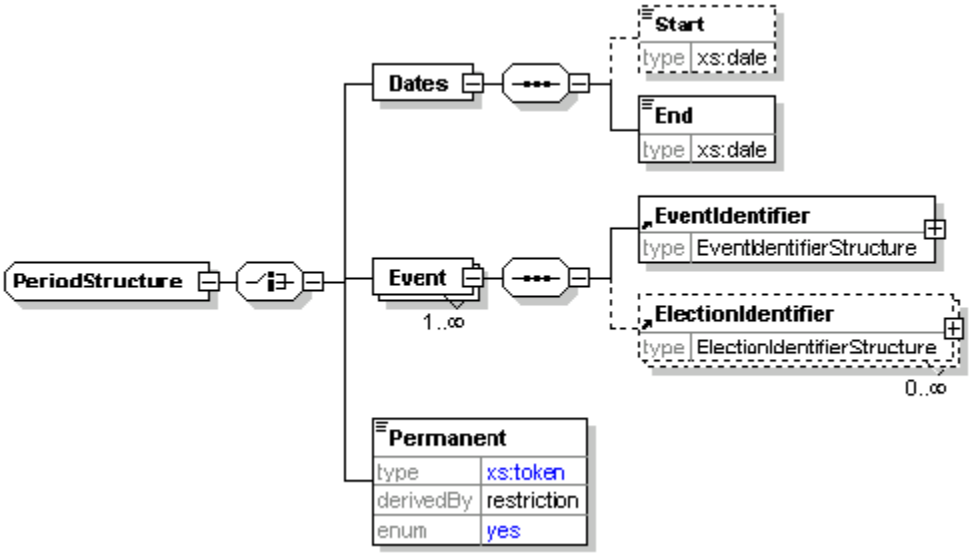
This data type provides a common structure for communications from electoral service organisers to voters. Multiple voters can be identified to allow printing of messages. Individual message types, such as that used for offering voting channel options (360a) are based on extensions of this type.

6.2.33 PartyStructure



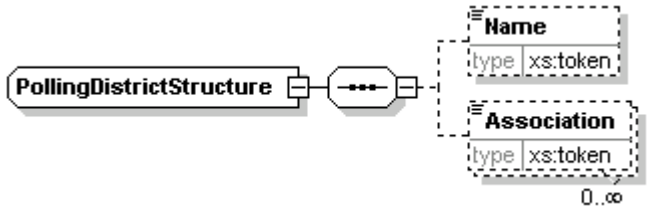
This element records information about a political party that specifically endorses a candidate as opposed to the parties that candidates say they are affiliated to.

6.2.34 PeriodStructure



This element can be used when appointing a proxy or registering to vote using a specific channel (e.g. postal). It allows this registration to be for a period of time, for specific election events (and possibly elections within those events) or permanently.

562 ## 6.2.35 PollingDistrictStructure

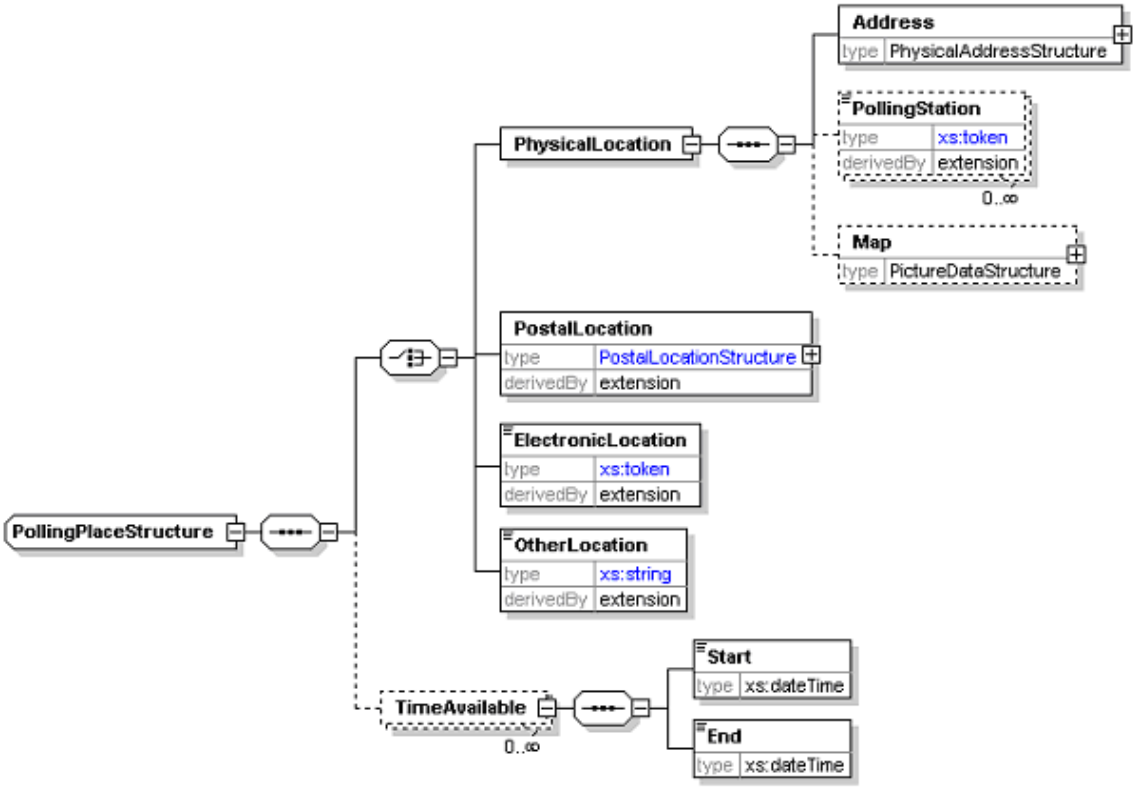


563

Element	Attribute	Type	Use	Comment
PollingDistrictStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

564 The polling district indicates where a voter is registered to vote. The polling district can have a name and
565 an Id attribute. It can also be associated with other terms such as a constituency. This is done through the
566 `Association` element, which has `Type` attribute and may have an `Id` attribute as well as a text value.

567 ## 6.2.36 PollingPlaceStructure



568

Element	Attribute	Type	Use	Comment
PollingPlaceStructure	Channel	VotingChannelType	required	
	DisplayOrder	xs:positiveInteger	optional	
PhysicalLocation	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
PostalLocation	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
ElectronicLocation	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
OtherLocation	Id	xs:NMTOKEN	optional	

	DisplayOrder	xs:positiveInteger	optional	
PollingStation	Id	xs:NMTOKEN	optional	

In general, a polling place will be either a physical location (for paper or kiosk voting), a postal address (for postal votes) or an electronic location (for Internet, SMS, telephone and other electronic means of voting). However, it is possible that none of these types will meet every need, and so an `OtherLocation` element has been included. Each of these locations must indicate the channel for which it is to be used. If a single location supports multiple channels, it must be included multiple times.

A physical location has an address. Sometimes, several polling stations will be at the same address, so a polling station can be defined by name and/or Id within the address. Access to an external map can also be provided as a URI or Base64 encoded binary data.

An electronic location must indicate its address (e.g. phone number, URL).

An optional `TimeAvailable` element is also provided. In most cases, this is not required as the time a location is available is the same as the time the channel is available. However, there are circumstances, such as the use of mobile polling stations, where this is not the case.

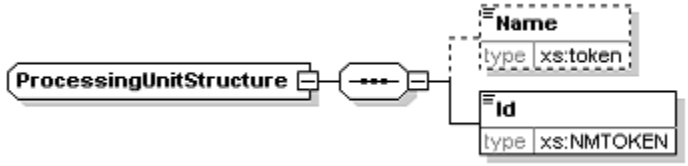
6.2.37 PositionStructure

The PositionStructure is an extension of xs:token to add the following attributes:

Element	Attribute	Type	Use	Comment
PositionStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

The element defined by this type indicates the position (e.g. President) for which an election is being held. It has a text description and an optional ID.

6.2.38 ProcessingUnitStructure

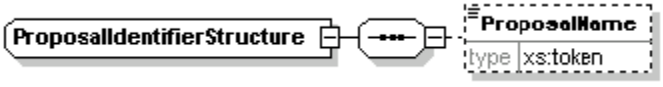


Element	Attribute	Type	Use	Comment
ProcessingUnitStructure	Role	xs:token (restricted)	required	

A processing unit is a physical system used in the election process. It is identified as part of audit information by its ID (which might be an IP address or an URI) and optional name.

Each processing unit has an attribute to describe its role. The role can be "sender", "receiver", "previous sender" or "next receiver". The latter two are used when there is a gateway involved. For example, a 440 (cast vote) message might have an `OriginatingDevice` as its original sender, a gateway as sender and voting system as receiver.

6.2.39 ProposalIdentifierStructure

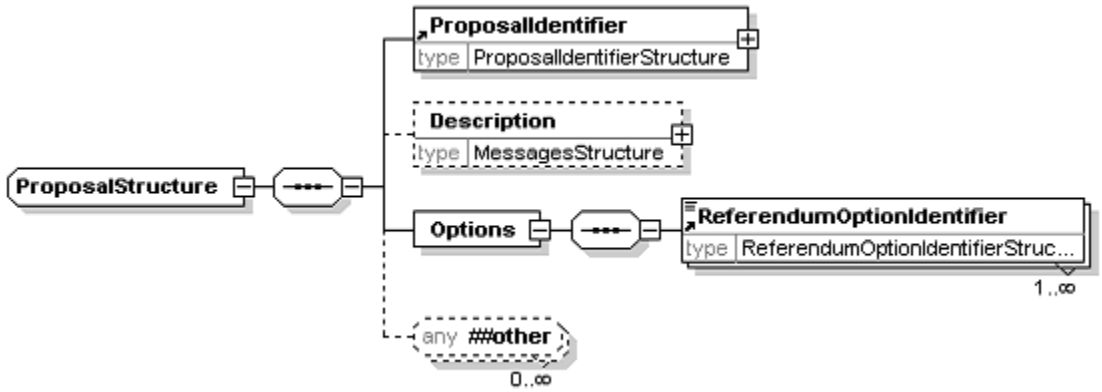


Element	Attribute	Type	Use	Comment
ProposalIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

595 A proposal is used in a referendum. At a basic level, it is a piece of text with the options ('yes' and 'no',
596 'for' and 'against' etc) to be voted on.

597 The proposal identifier indicates a system ID for the proposal. A short code can also be included, either
598 for SMS voting or where the security mechanism in place requires it. An
599 `ExpectedConfirmationReference` attribute also allows for security mechanisms where the
600 confirmation reference may be different for each combination of voter and candidate.

601 ## 6.2.40 ProposalStructure



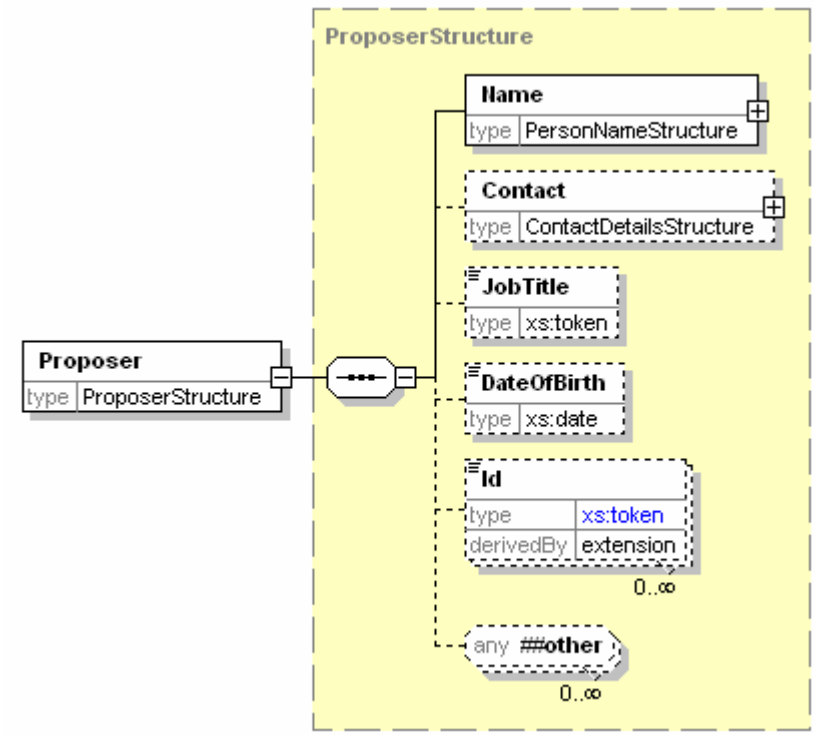
602

Element	Attribute	Type	Use	Comment
ProposalStructure	Type	xs:token	optional	

603 The proposal identifier provides a name and ID. The description is used to provide the information that
604 will be displayed to the voter to indicate the aim of the proposal. The options are then used to indicate
605 how the voter may vote.

606 The `Type` attribute allows for referenda where there are different kinds of proposal, for example,
607 'initiative' or 'referendum'.

608 ## 6.2.41 ProposerStructure

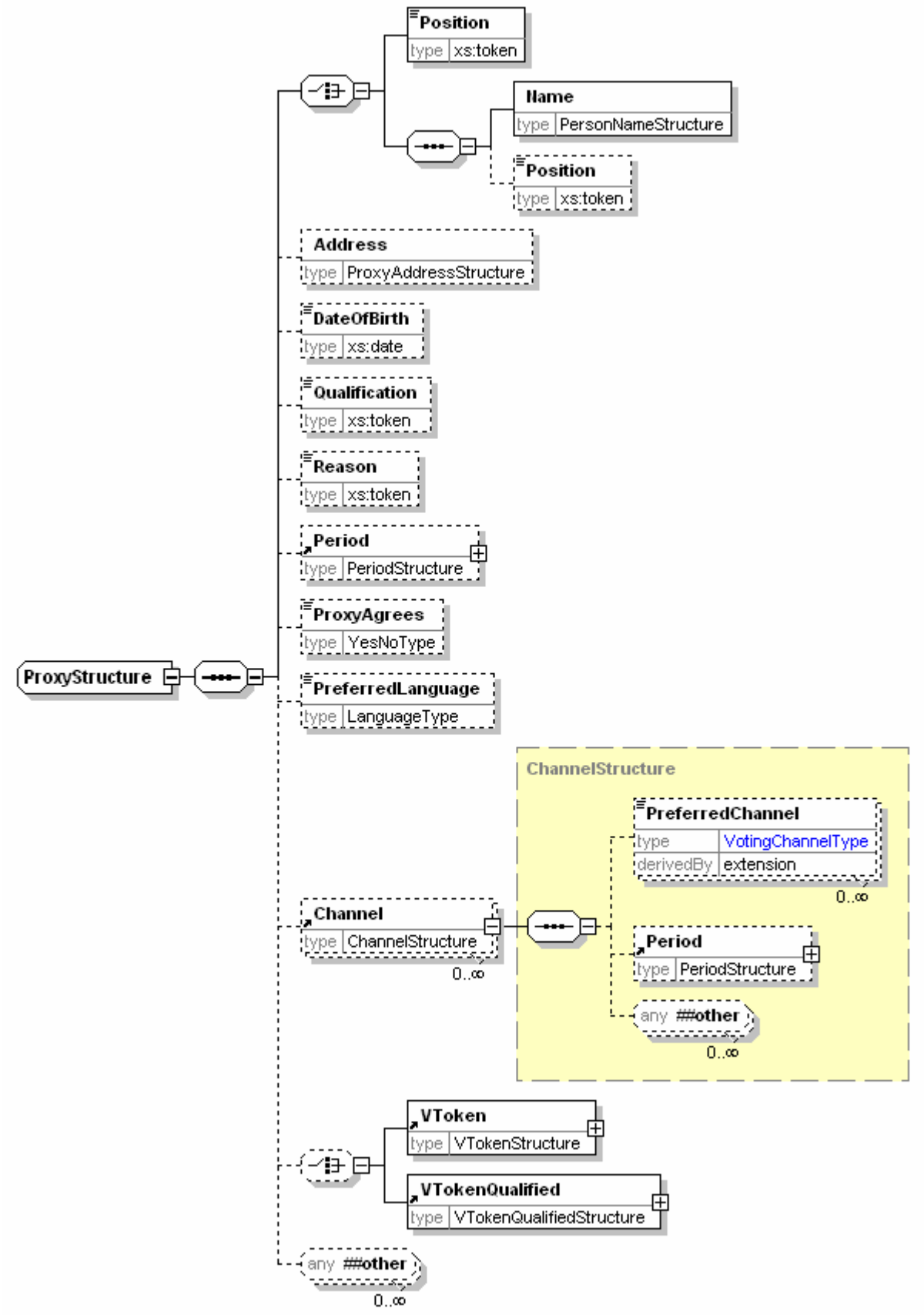


609

Element	Attribute	Type	Use	Comment
ProposerStructure	Category	xs:token (restricted)	optional	

610 A proposer proposes, seconds or endorses a candidate or referendum proposal. A proposer can have a
611 category, which indicates one of "primary", "secondary" or "other". A name is always required, and
612 additional information might be needed.

613 ## 6.2.42 ProxyStructure



614

Element	Attribute	Type	Use	Comment
ProxyStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
PreferredChannel	Fixed	YesNoType	optional	

In many elections, a voter may appoint a proxy to vote on his or her behalf. That proxy may be identified by position (for example, appointing the chairman as proxy at a company AGM), or by name (for example, appointing your spouse as proxy for a public election), or both.

In some elections, the proxy must, for example, be a family member. This is indicated using the `Qualification` element, while a reason for appointing a proxy can be indicated using the Reason element.

A proxy can be permanent (i.e. appointed until revoked), appointed for one or more election events (and individual elections within each event) or for a period of time. A proxy can also list his or her preferred voting channels. These are listed in order of preference for a given period (which may be specific election events, a date range or permanent), so that information can be sent regarding the most appropriate voting channel at any election. The channel may be fixed, for example, if registering to vote by a specific channel prevents voting by other means.

A proxy may also have a voting token, indicating the right to vote, or a qualified voting token, indicating that there is a question over their right to vote.

6.2.43 ReferendumOptionIdentifierStructure

The ReferendumOptionIdentifierStructure is an extension of xs:token to add the following attributes:

Element	Attribute	Type	Use	Comment
ReferendumOptionIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

A referendum option is used to indicate the possible answers to a referendum question, such as "yes" and "no" or "for" and "against".

The referendum option identifier has a text description and can have a system ID. A short code can also be included, either for SMS voting or where the security mechanism in place requires it. An ExpectedConfirmationReference attribute also allows for security mechanisms where the confirmation reference may be different for each combination of voter and option.

6.2.44 ReportingUnitIdentifierStructure

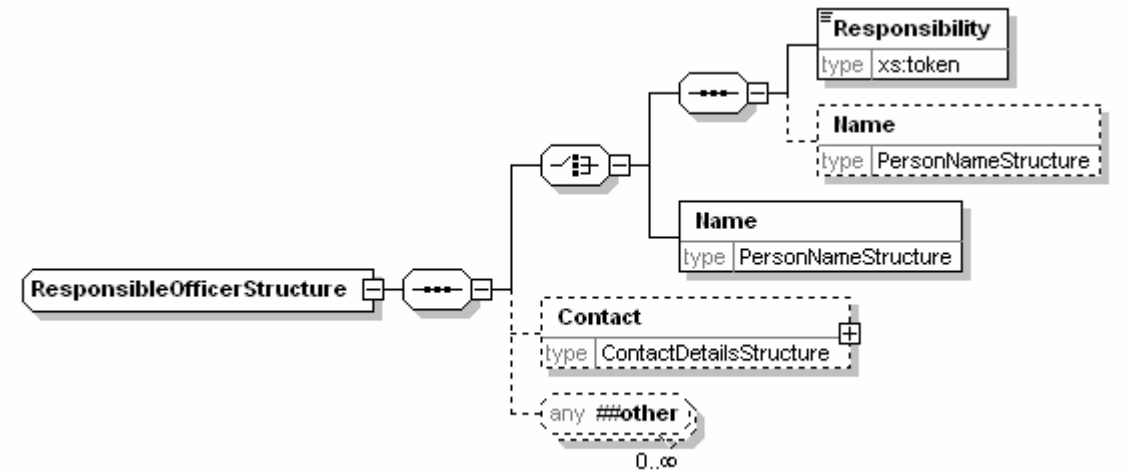
The ReportingUnitIdentifierStructure is an extension of xs:token to add the following attributes:

Element	Attribute	Type	Use	Comment
ReportingUnitIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

A reporting unit is an entity that reports partial information relating to a contest (votes or the results of a count) without having the full set of information required to generate a result. This will happen when votes from several independently managed areas must be amalgamated to produce a result.

The reporting unit identifier structure defines a string with an optional Id.

6.2.45 ResponsibleOfficerStructure



Element	Attribute	Type	Use	Comment
ResponsibleOfficerStructure	Id	xs:NMTOKEN	optional	

A responsible officer is someone who has some sort of role to play in the organization of an election. Each responsible officer has a name and/or responsibility (such as 'returning officer') and optional contact information. Local rules will usually indicate the values allowed in the `Responsibility` element.

6.2.46 ScrutinyRequirementStructure

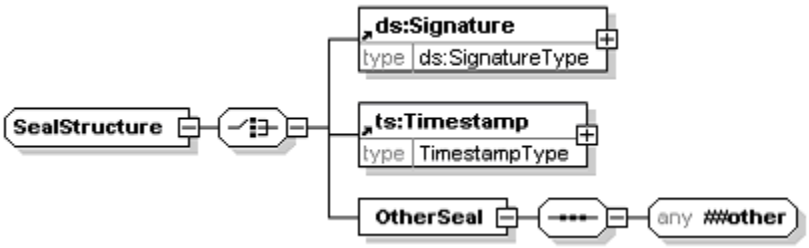
The ScrutinyRequirementStructure is an extension of xs:token to add the following attribute:

Element	Attribute	Type	Use	Comment
ScrutinyRequirementStructure	Type	xs:token	required	

A scrutiny requirement has two parts, a Type attribute and a text value. The Type specifies a condition that a candidate must meet, such as an age or membership requirement or the payment of a fee. The text describes how that condition has been met. For example:

```
<ScrutinyRequirement Type="dateofbirth">8 June 1955</ScrutinyRequirement>
```

6.2.47 SealStructure



Element	Attribute	Type	Use	Comment
OtherSeal	Type	xs:token	required	

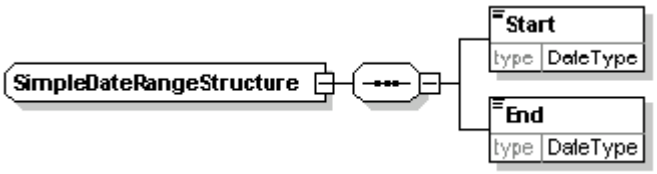
The seal is used to protect information such as a vote, voting token or complete message. The seal provides the means of proving that no alterations have been made to a message or individual parts of a message such as a vote or collection of votes, from when they were originally created by the voter. The seal may also be used to authenticate the identity of the system that collected a vote, and provide proof of the time at which the vote was cast.

If a message is to be divided, each part must be separately sealed to protect the integrity of the data. For example, if votes in several elections are entered on a single ballot, and these votes are being counted in separate locations, each vote must be separately sealed.

A seal may be any structure which provides the required integrity characteristics, including an XML signature [1] or a time-stamp.

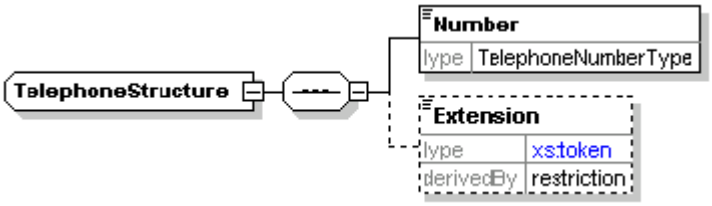
The XML signature created by the voting system provides integrity and authentication of the identity of the system that collected the vote. The time-stamp provides integrity of the vote and proof of the time that the vote was cast.

6.2.48 SimpleDateRangeStructure



This data type is used to describe ranges of dates or dates and times.

6.2.49 TelephoneStructure



Element	Attribute	Type	Use	Comment
TelephoneStructure	Preferred	YesNoType	optional	
	Mobile	YesNoType	optional	

This is an extension of the `TelephoneType` and adds an `Extension` element and the two attributes `Preferred` and `Mobile` of `YesNoType`. The Preferred attribute indicates which of several phone numbers or fax numbers is preferred.

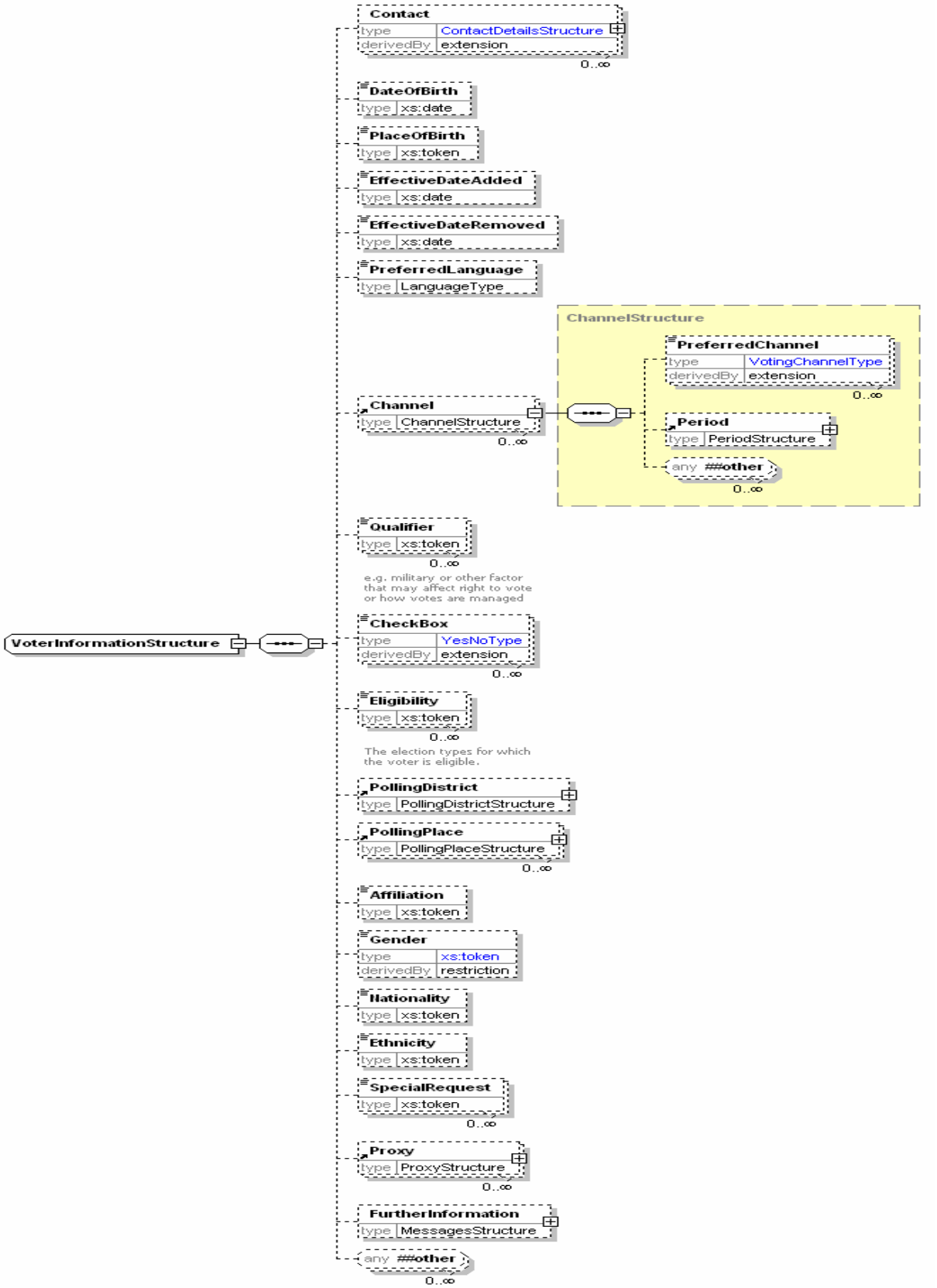
677 ## 6.2.50 VoterIdentificationStructure

678

Element	Attribute	Type	Use	Comment
VoterIdentificationStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	

679 An element defined by this data type is used wherever identification of a voter is required. It contains the
680 voter's name and electoral address (the address that gives them the right to vote in a specific contest),
681 the voting token (either normal or qualified) and a number of identifiers (such as an electoral registration
682 number or the scanned signature of the voter). It may also include a previous electoral address if this is
683 required (for example, because a voter has not been at his or her current address for more than a
684 predefined period).

6.2.51 VoterInformationStructure



Element	Attribute	Type	Use	Comment
VoterInformationStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
ContactDetailsStructure	DisplayOrder	xs:positiveInteger	optional	standard attribute for this data type
	ElectionId	xs:NMTOKEN	optional	additional attribute
PreferredChannel	Fixed	YesNoType	optional	
Checkbox	Type	xs:token	required	

687 This contains more information about the voter. It contains all the information that would typically be
688 included on an electoral register other than that used for identification of the voter. In many cases, it will
689 be restricted to only include the information required in a specific message type.

690 A voter can list his or her preferred voting channels. These are listed in order of preference for a given
691 period (which may be specific election events, a date range or permanent), so that information can be
692 sent regarding the most appropriate voting channel at any election. The channel may be fixed, for
693 example, if registering to vote by a specific channel prevents voting by other means.

694 The `Qualifier` element is used to hold information that might affect a voter's right to vote or how the
695 voting process is managed. Suitable enumerations for this are likely to be added as part of localisation.
696 The CheckBox element with its Type attribute allows binary information such as whether the voter's entry
697 on the electoral register can be sold, or whether the voter wants to participate in the count. The eligibility
698 indicates what election types a voter is eligible to participate in.

699 Special requests are requests from the voter, for example, for wheelchair access to a polling station.

700 6.2.52 VTokenStructure

701

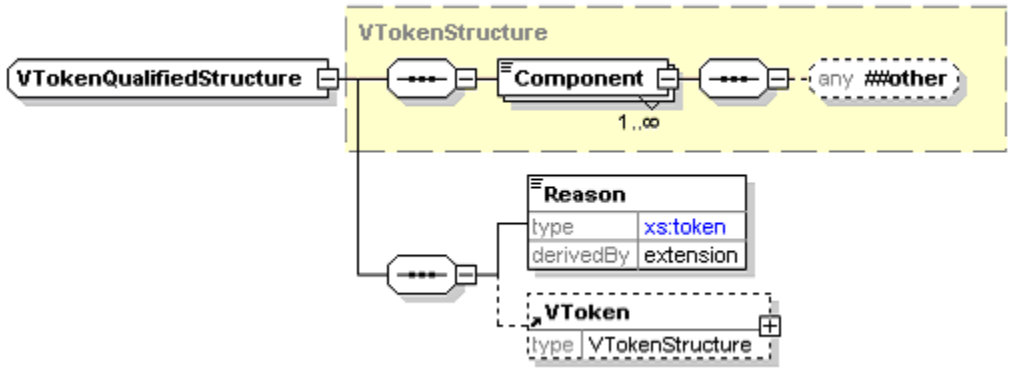
Element	Attribute	Type	Use	Comment
Component	Type	xs:NMTOKEN	required	

702 The voting token contains the information required to authenticate the voter's right to vote in a specific
703 election or contest. A voting token can consist of a continuous string of encoded or encrypted data,
704 alternatively it may be constructed from several data components that a user may input at various stages
705 during the voting process (such as PIN, password and other coded data elements). The totality of the
706 voting token data proves that a person with the right to vote in the specific election has cast the vote.

707 Depending on the type of election, the voter may need to cast their votes anonymously, thus not providing
708 a link to the voter's true identity. In this case the voting token data will not identify the actual person
709 casting the vote; it just proves that the vote was cast by a person with the right to do so. Election rules
710 may require a link to be maintained between a vote and a voter, in which case a link is maintained
711 between the voting token data and the voter's identity.

712 The components of the voting token are identified by a Type attribute and may contain text or markup
713 from any namespace depending on the token type. The content could be defined further in separate
714 schemas for specific types of token.

6.2.53 VTokenQualifiedStructure



Element	Attribute	Type	Use	Comment
Reason	Type	xs:token	required	

There are occasions when a normal voting token cannot be used. For example, if a voter is challenged, or an election officer claims the voter has already voted. In these circumstances a qualified voting token can be used and treated appropriately by the election system according to the election rules. For example, challenged votes might be ignored unless there were sufficient to alter the result of the election, in which case each vote would be investigated and counted if deemed correct to do so.

The `VTokenQualifiedStructure` is therefore an extension of the `VTokenStructure` to add the additional information required. This additional information comprises a reason for qualification (as a Reason element with a Type attribute and textual description) and possibly an original VToken.

7 Elements

The following elements are simply specified by their similarly-named data type and are not described further here:

Affiliation, AffiliationIdentifier, Agent, AgentIdentifier, Area, AuditInformation, AuthorityIdentifier, BallotIdentifier, BallotIdentifierRange, Candidate, CandidateIdentifier, ContactDetails, ContestIdentifier, CountingAlgorithm, DocumentIdentifier, ElectionIdentifier, EventIdentifier, EventQualifier, Gender , Logo, ManagingAuthority, MessageType, NominatingOfficer, NumberOfPositions, Period, PollingDistrict, PollingPlace, Position, PreferredChannel, Proposal, ProposalIdentifier, Proposer, Proxy, ReferendumOptionIdentifier, ReportingUnitIdentifier, ResponsibleOfficer, ScrutinyRequirement, Seal, VoterId, VToken, VTokenQualified

7.1 Accepted

YesNoType

This element indicates that a candidate, referendum proposal or vote has been accepted.

7.2 Election Statement

MessagesStructure

This is the candidate's message to voters.

7.3 MaxVotes

xs:positiveInteger

The maximum number of votes allowed (also known as the vote limit). This defaults to the value of "1".

7.4 MinVotes

xs:nonNegativeInteger

The minimum number of votes allowed. This defaults to the value of "0".

7.5 NumberInSequence

xs:positiveInteger

The number of partial messages when a message is split. See “Spitting of Messages”

7.6 NumberOfSequence

This element represents the number of identical positions that will be elected as the result of a contest. For example, in a contest for a Town Council, three councillors might be elected as the result of the contest in one part of the town. The element is an `xs:positiveInteger` and defaults to a value of "1".

7.7 PersonName

This element uses the PersonNameStructure defined in the EML externals schema.

7.8 Profile

MessagesStructure

This is the candidate's profile statement.

7.9 SequenceNumber

xs:positiveInteger

The sequence number of a partial message when a message is split. See “Splitting of Messages” (Section 4).

7.10 TransactionId

xs:token

A reference code for a specific transaction, which may comprise several messages.

7.11 VoterName

PersonNameStructure

The name of a voter.

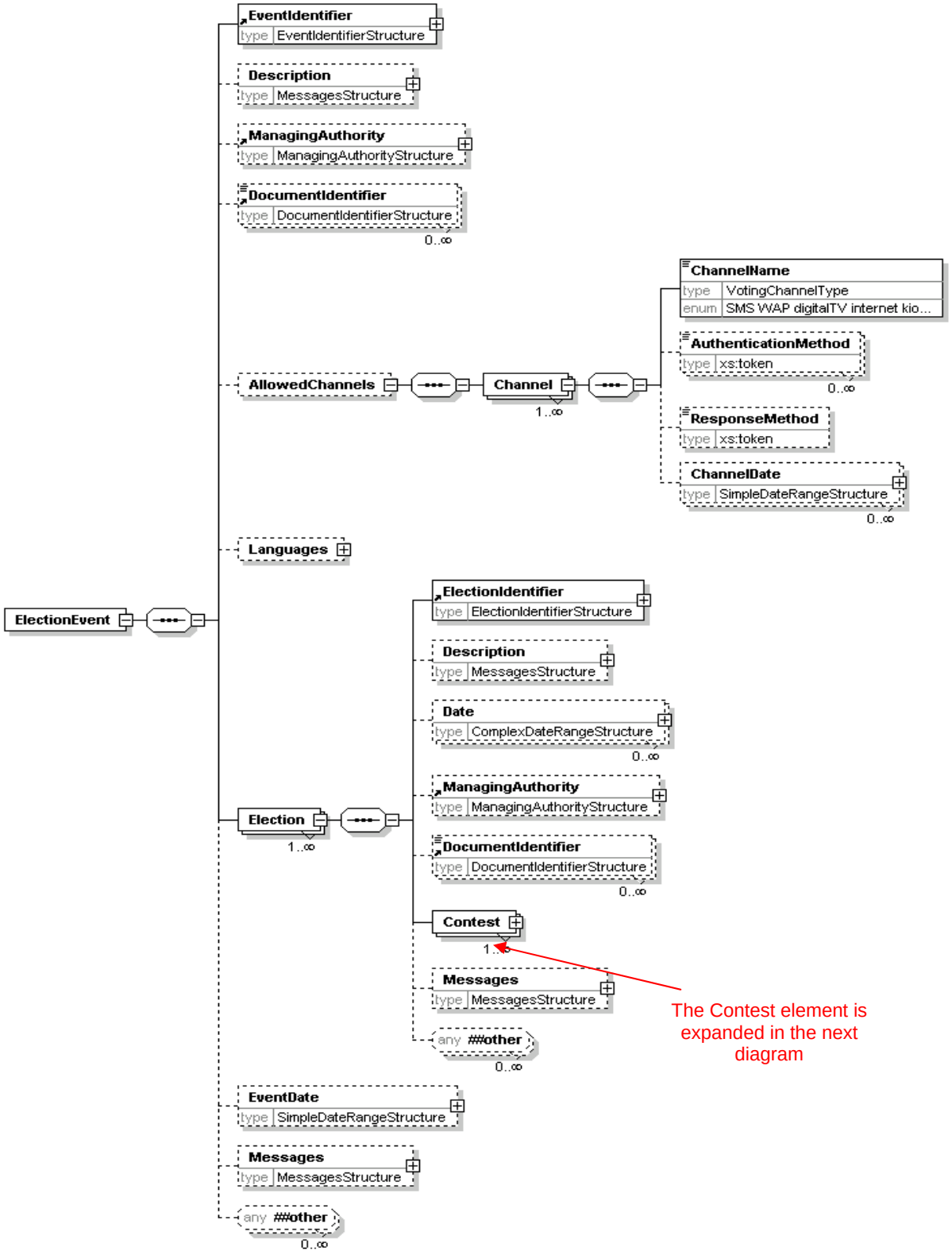
8 EML Message Schemas

This section describes the EML messages and how the message specifications change for this application. It uses the element and attribute names from the schemas.

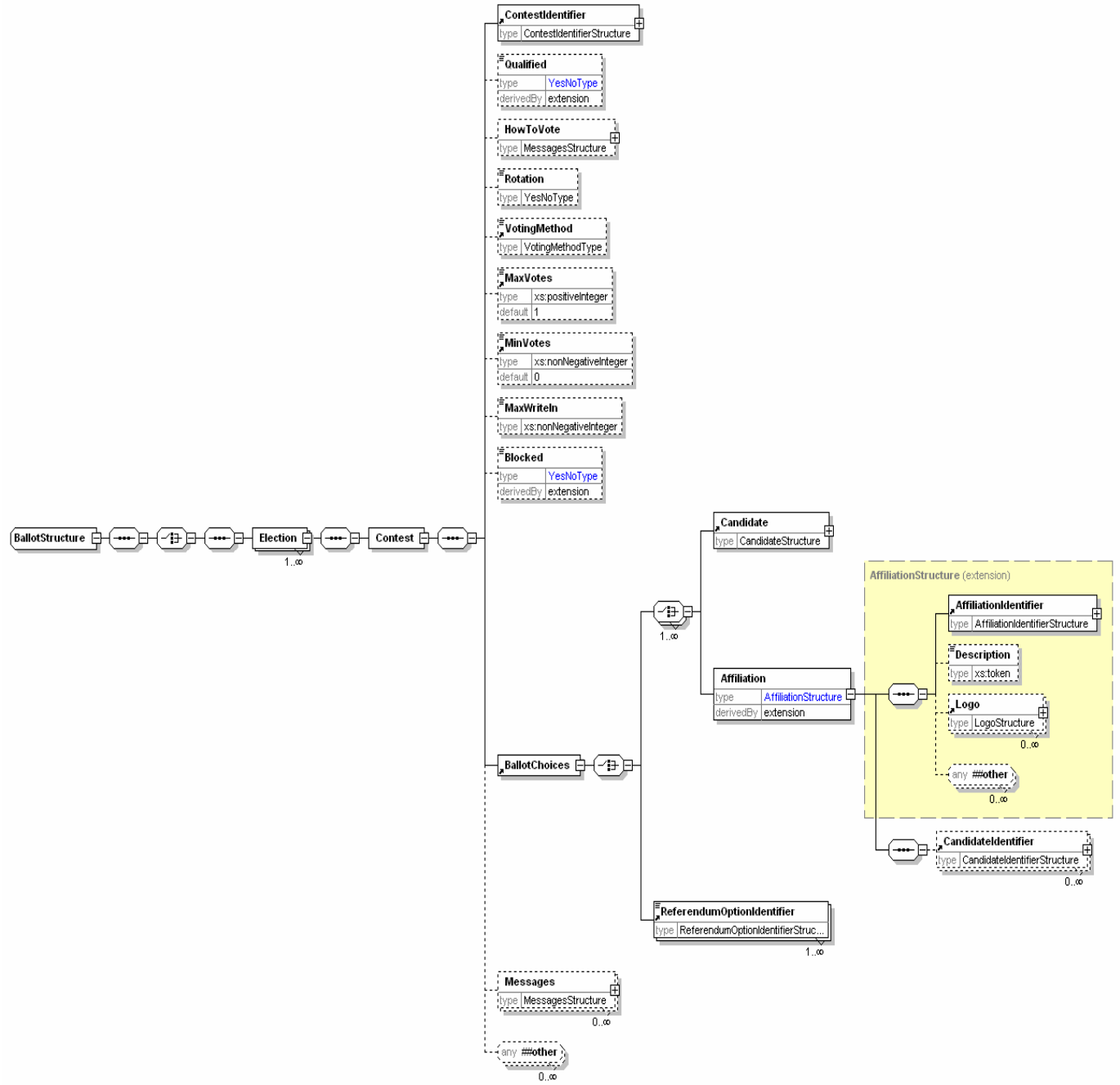
Attributes are shown where they are not the standard attributes of data types already described.

773

8.1 Election Event (110)



774



775

Element	Attribute	Type	Use	Comment
AllowedChannels	DisplayOrder	xs:positiveInteger	optional	
Contest	DisplayOrder	xs:positiveInteger	optional	

776 ## 8.1.1 Description of Schema

777 This schema is used for messages providing information about an election or set of elections. It is usually
778 used to communicate information from the election organisers to those providing the election service.

779 The message therefore provides information about the election event, all elections within that event and
780 all contests for each election.

781 For the election event, the information includes the ID and name of the event, possibly with a qualifier on
782 the event. This qualifier is used when an event has several local organisers. For example, for a UK

general election, each constituency organises its own contests. The election event is therefore the general election, whilst the qualifier would indicate the constituency. Other information regarding an election event comprises the languages to be used, the start and end dates of the event, potentially a list of external documents that are applicable (such as the rules governing the election), a description and information about the managing authority.

The managing authority can be indicated for the event, each election, each contest within the election and each reporting unit.

An election can have a number of dates associated with it. For example, there is likely to be a period allowed for nomination of candidates and a date when the list of eligible voters is fixed. Each date can be expressed as a single date when something happens, a start date, an end date, or both start and end dates. These dates can be either just a date or both a date and time using the subset of the ISO 8601 format supported by XML Schema.

Like the event, an election can have both a managing authority and referenced documents. Finally, there is a `Messages` element for additional information.

A contest has a name and ID. It can also have reporting unit identifiers. A contest may need to specify its geographical area independently from its name, for which purpose the Area element is provided. Each contest can specify the voting channels allowed. In general, the list of possible channels will be further restricted as part of a local customisation. Each channel can specify several methods for authenticating the voter, such as PIN and password, and a response method, indicating the type of response to be given to a cast vote. Finally, facilities are provided to indicate the dates and times when the channel will be available to the voter.

As described previously, a contest can indicate its managing authority. It may also indicate the position (such as 'President') for which votes are being cast. The Description allows for additional text describing the contest. Each contest indicates the voting method being used, whilst the `CountingAlgorithm` indicates the method of counting (such as the d'Hondt or Meeks method) that will be used. The minimum and maximum number of votes to be cast by each voter can also be indicated.

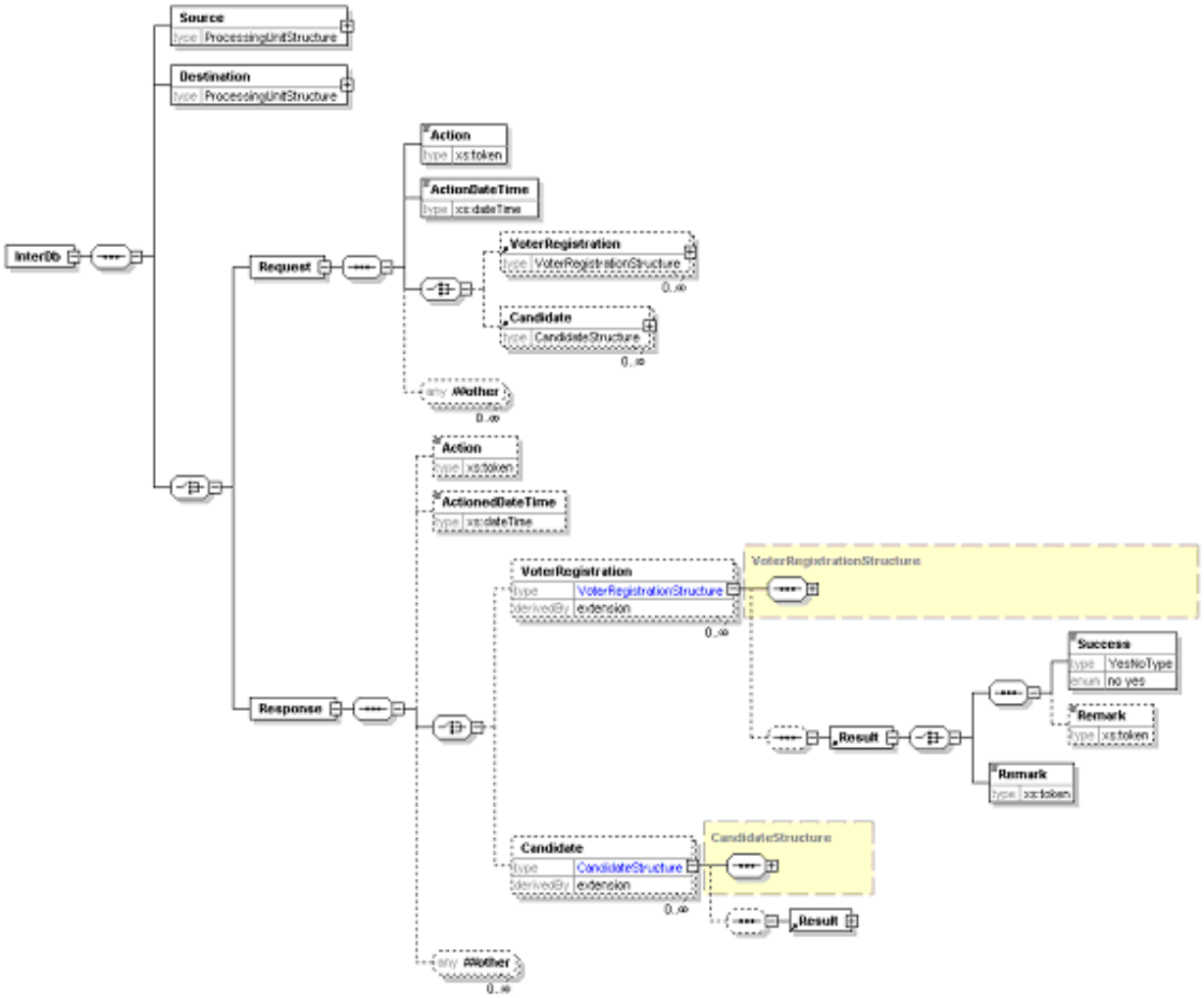
A list of polling places can be provided. These can be either physical locations for people to go to vote, postal addresses for postal votes or electronic locations. An 'other location' is also allowed for cases where these do not meet the requirements. A location can also say when it will be available. This is intended for mobile polling stations that will only be available at a given address for a part of the voting period.

Finally, a `Messages` element allows for additional information that might be communicated to the voter later through other messages.

8.1.2 EML Additional Rules

Error Code	Error Description
3110-001	The allowed channels must not be declared at both the election event level and the contest level.

8.2 Inter Database (120)



8.2.1 Description of Schema

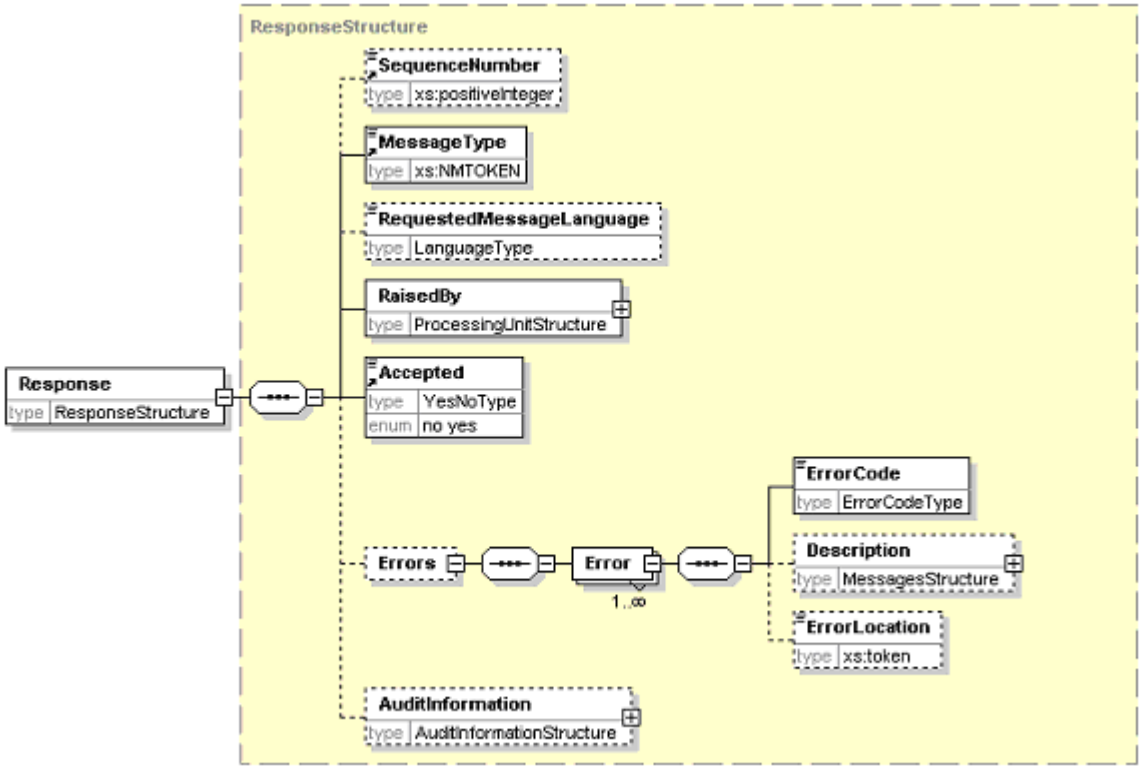
This schema is used for messages requesting services from other electoral registers or candidate databases. This can, for example, be used to de-dupe databases, check that a candidate in an election is only standing in one contest or confirm that the proposers of a candidate are included on a relevant electoral register. The schema is in two parts, so a message will be either a request or a response.

Both request and response start by identifying the source and destination as processing units.

A request has an `Action` code to identify the request being made. Possible actions include, but are not limited to, ‘add’, ‘delete’, ‘replace’, ‘confirm’ and ‘return’. The code ‘confirm’ returns success if the person indicated is included in the database. The code ‘return’ causes the receiving the database to return the full information for the person identified. The `ActionDateTime` is used to specify when the action should be carried out, and then there is an optional list of voters or candidates.

A response has a similar structure. It could be that the Action code is no longer required, so this is now optional. The `TransactionID` must match that given in the request. The Result is either a binary Success flag or a remark or both. Again, there is a date and time, but in this case it is the date and time at which the action took place.

8.3 Response (130)



8.3.1 Description of Schema

Some messages have a defined response message that provides useful information. However, there is a need for a more general response, either to indicate that a message has been accepted, or to indicate the reasons for rejection.

The message includes information to identify the message to which the response applies (by using the same transaction id in the EML element and, if necessary, including the sequence number of the message to which the response applies in the Response element), with information on the entity raising the message, whether the message was accepted and information about the errors if it was not. The desired language for a display message can also be included to allow a downstream processor to substitute a language-specific error message if required.

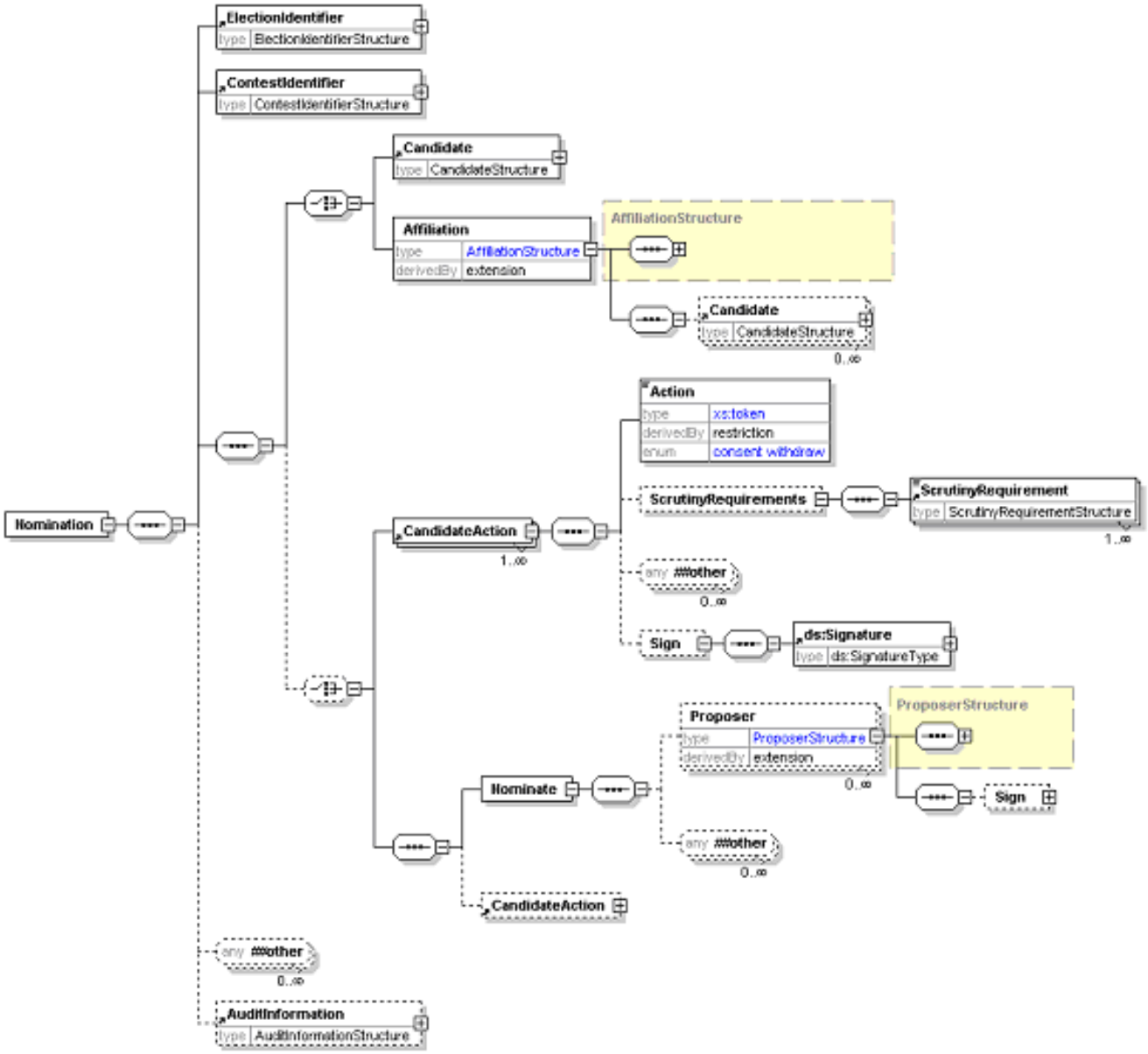
If the message is reporting an error, the location of the error within the message can be indicated. Usually, this will be an XPath to the location of the error. However, errors detected by an XML parser may be in a different format, such as a line number.

Note that a single response can be raised for a series of sub-messages with the same transaction ID. This allows indication, for example, that a sub-message was missing.

8.3.2 Additional EML Rules

Error Code	Error Description
3130-001	If the message is not accepted, there must be an Errors element

8.4 Candidate Nomination (210)



8.4.1 Description of Schema

Messages conforming to this schema are used for four purposes:

1. nominating candidates in an election;
2. nominating parties in an election;
3. consenting to be nominated; or
4. withdrawing a nomination.

Candidate consent can be combined in a single message with a nomination of the candidate or party or sent separately.

Note that the message does not cover nomination for referendums.

The election and contest must be specified. When a candidate is being nominated, there must be information about the candidate and one or more proposers. The candidate must supply a name.

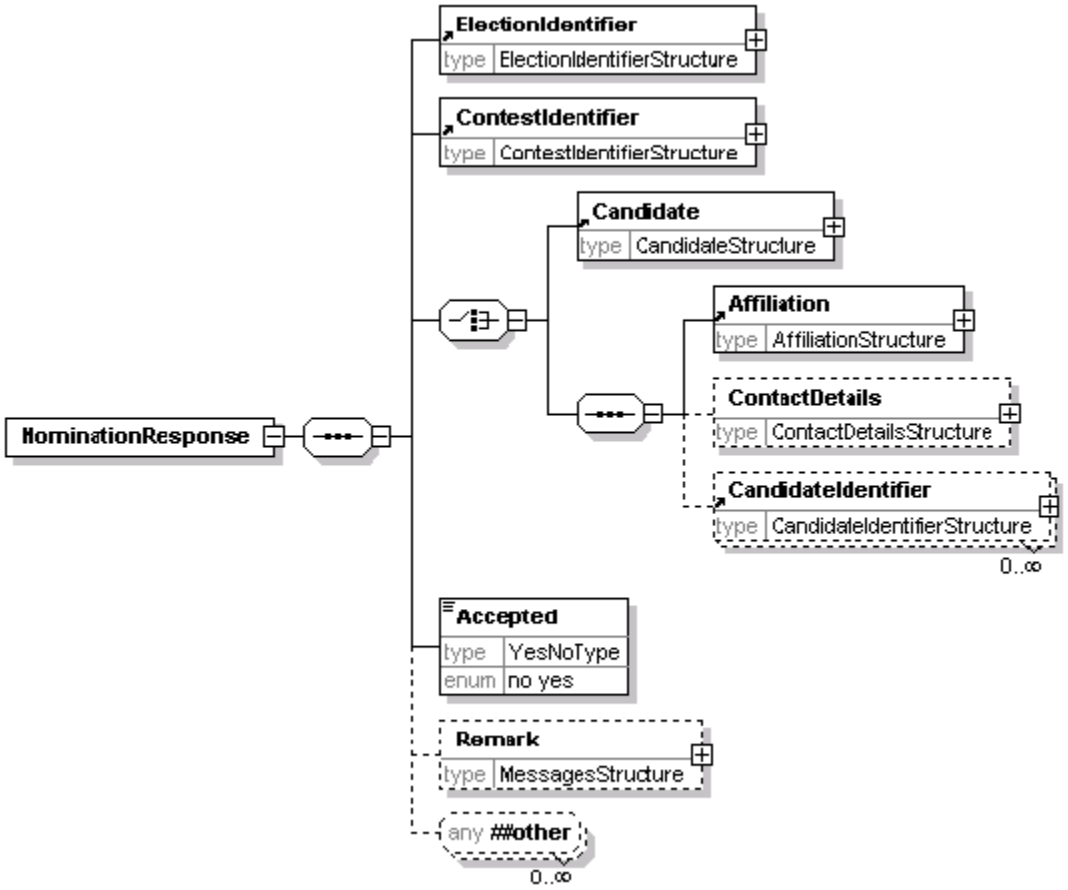
865 Optionally, the candidate can provide contact information, an affiliation (e.g. a political party) and textual
866 profiles and election statements. These two items use the `MessagesStructure` to allow text in multiple
867 languages. There is also scope to add additional information defined by the election organiser.

868 The proposers use the standard proposer declaration with a mandatory name and optional contact
869 information and job title. Again, additional information can be required.

870 If a party is being nominated, the primary proposer will be the contact. Information on candidates in a
871 party list can also be provided.

872 Candidates, either individuals or on a party list, must define the action being taken and may provide
873 scrutiny information. The scrutiny requirements indicate how the candidate has met any conditions for
874 standing in this election. This could include indicating that a deposit has been paid or providing a
875 reference to prove that he or she lives in the appropriate area. This information can be signed
876 independently of the complete message.

877 8.5 Response to Nomination (220)



878

879 8.5.1 Description of Schema

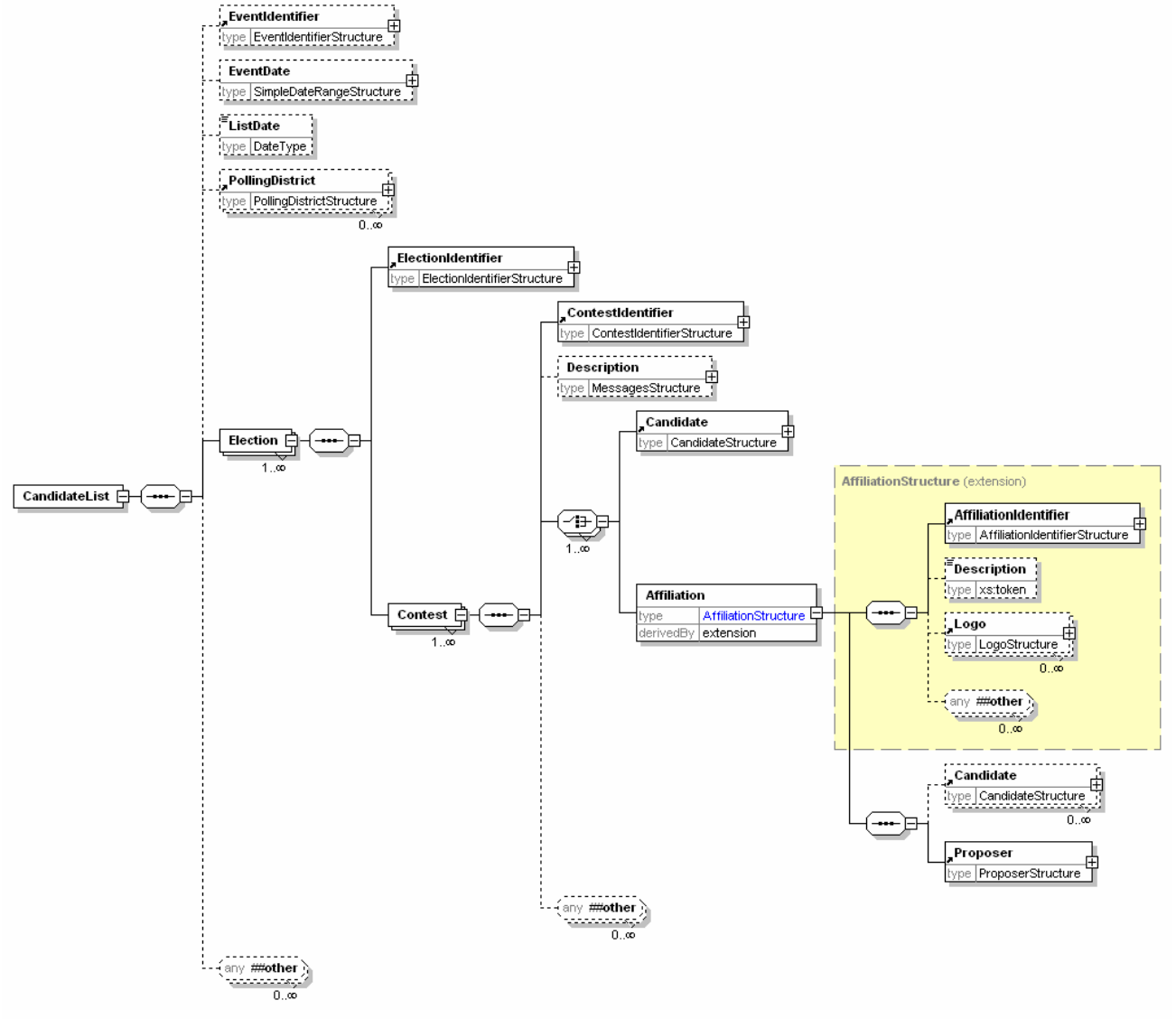
880 This message is sent from the election organiser to the candidate or nomination authority for a party to
881 say whether the nomination has been accepted. Along with the acceptance information and the basic
882 information of election, contest and party and candidate names, the candidate's contact details and
883 affiliation can be included and a remark explaining the decision.

884

8.5.2 EML Additional Rules

Error Code	Error Description
3220-001	If the nomination has not been accepted, a reason for rejection is required in the Remark element

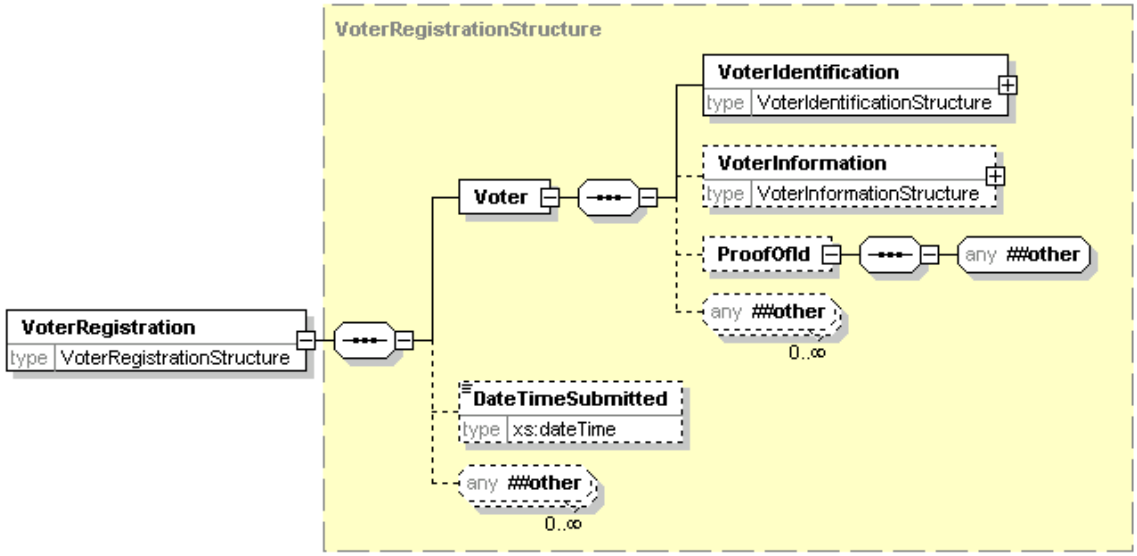
8.6 Candidate List (230)



8.6.1 Description of Schema

This schema is used for messages transferring candidate lists for specified contests. It has the election event, election and contest identifiers, and optionally the event dates and a contest description. The list itself can be either a list of candidates, each with a name, address, optional affiliation and other useful data, or a list of parties. In the latter case, contact information and a list of candidates under a party list system can also be included.

8.7 Voter Registration (310)



8.7.1 Description of Schema

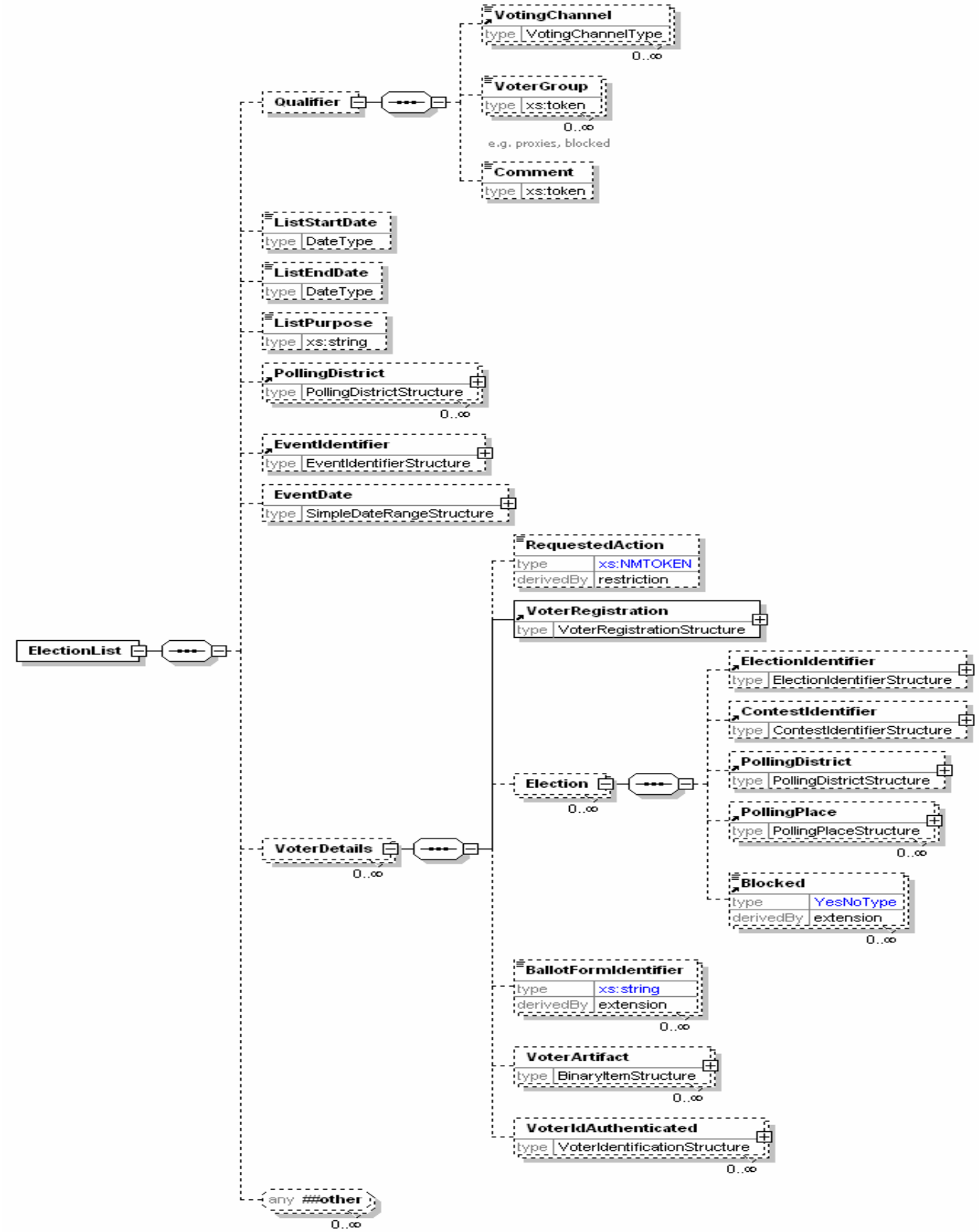
This schema is used for messages registering voters. It uses the `VoterIdentificationStructure`. The `VoterInformationStructure` is used unchanged. Proof of ID can be provided.

There is the facility for the transmission channel (for example a trusted web site) to add the time of transmission.

8.7.2 EML Additional Rules

Error Code	Error Description
3310-001	The Proxy must not have a VToken or VTokenQualified

902 ## 8.8 Election List (330)



903
904

Element	Attribute	Type	Use	Comment

Blocked	Reason	xs:token	optional	
	Channel	VotingChannelType	optional	

8.8.1 Description of Schema

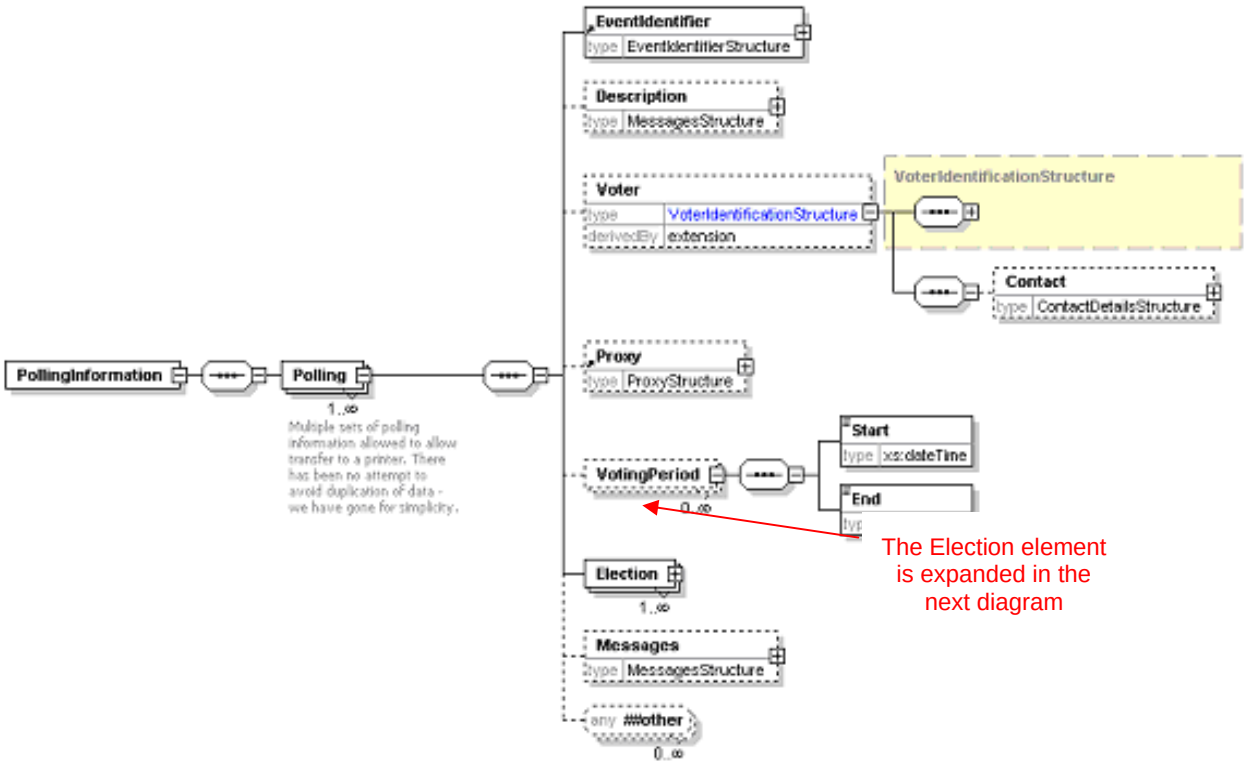
This schema is primarily used for messages communicating the list of eligible voters for an election or set of elections. It can also be used for any other purpose that involves the transfer of voter information where the 120-interDB message is not appropriate. Partial lists are allowed through the use of the Qualifier, Blocked and VoterGroup elements. So, for example, a list of postal voters or a list of proxies can be produced. The schema can also be used for filtered lists such as a list of postal proxies. These lists sometimes do not contain any names meeting the filter so empty lists are allowed.

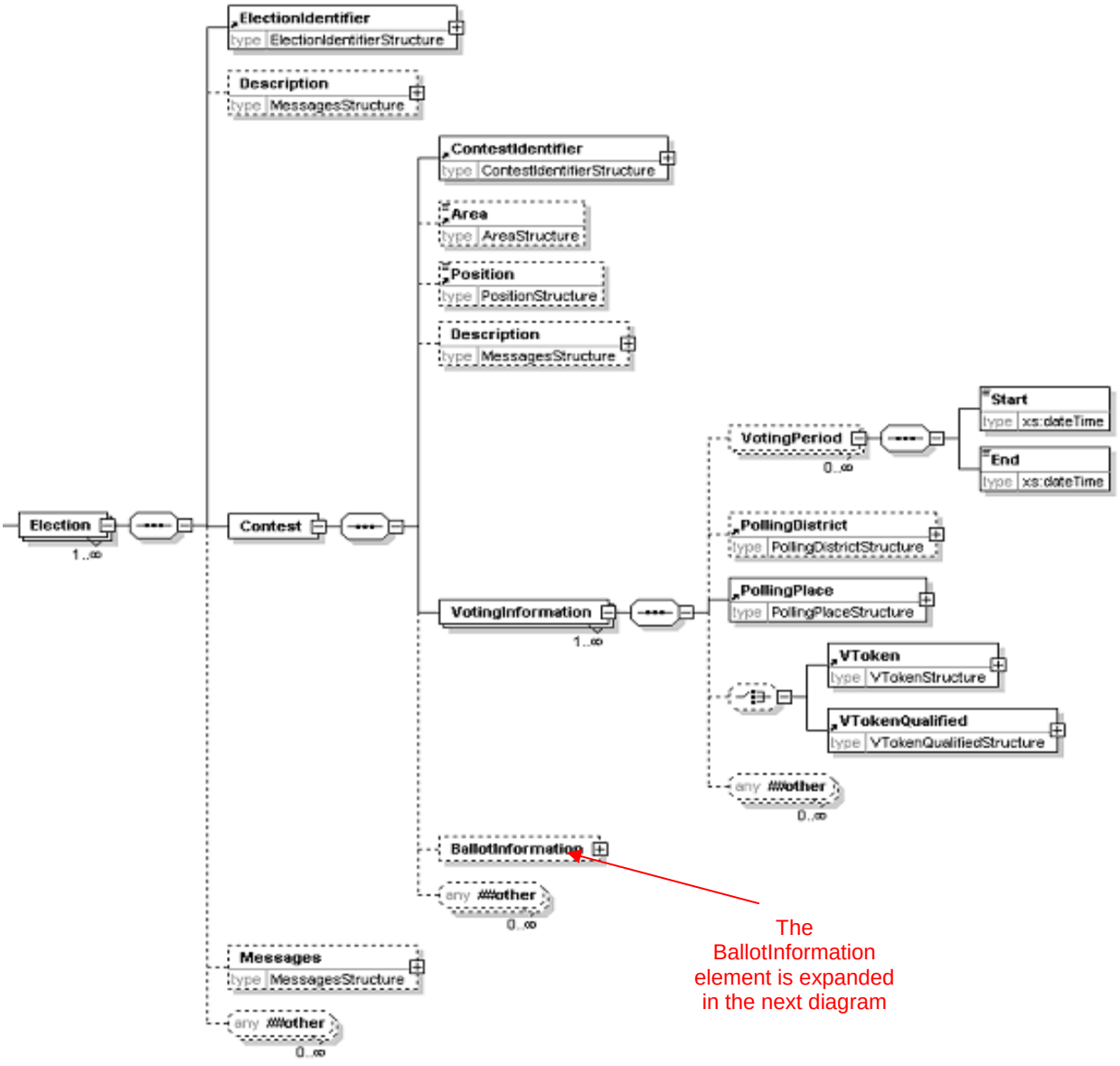
For each voter, information is provided about the voter himself or herself, and optionally about the elections and contests in which the voter can participate. The information about the voter is the same as that defined in the 310-voterregistration schema. Added to this can be a list of elections, each identifying the election and the contest in which this voter is eligible to vote, and the polling places available. Any voter can have a Blocked element set against them with an optional Reason and Channel. This allows a list to be produced for a polling place indicating those that have already voted by another means or who have registered for a postal vote. It can also be used if the complete electoral register must be transmitted (perhaps as a fraud prevention measure) but some people on the register are no longer eligible to vote.

8.8.2 EML Additional Rules

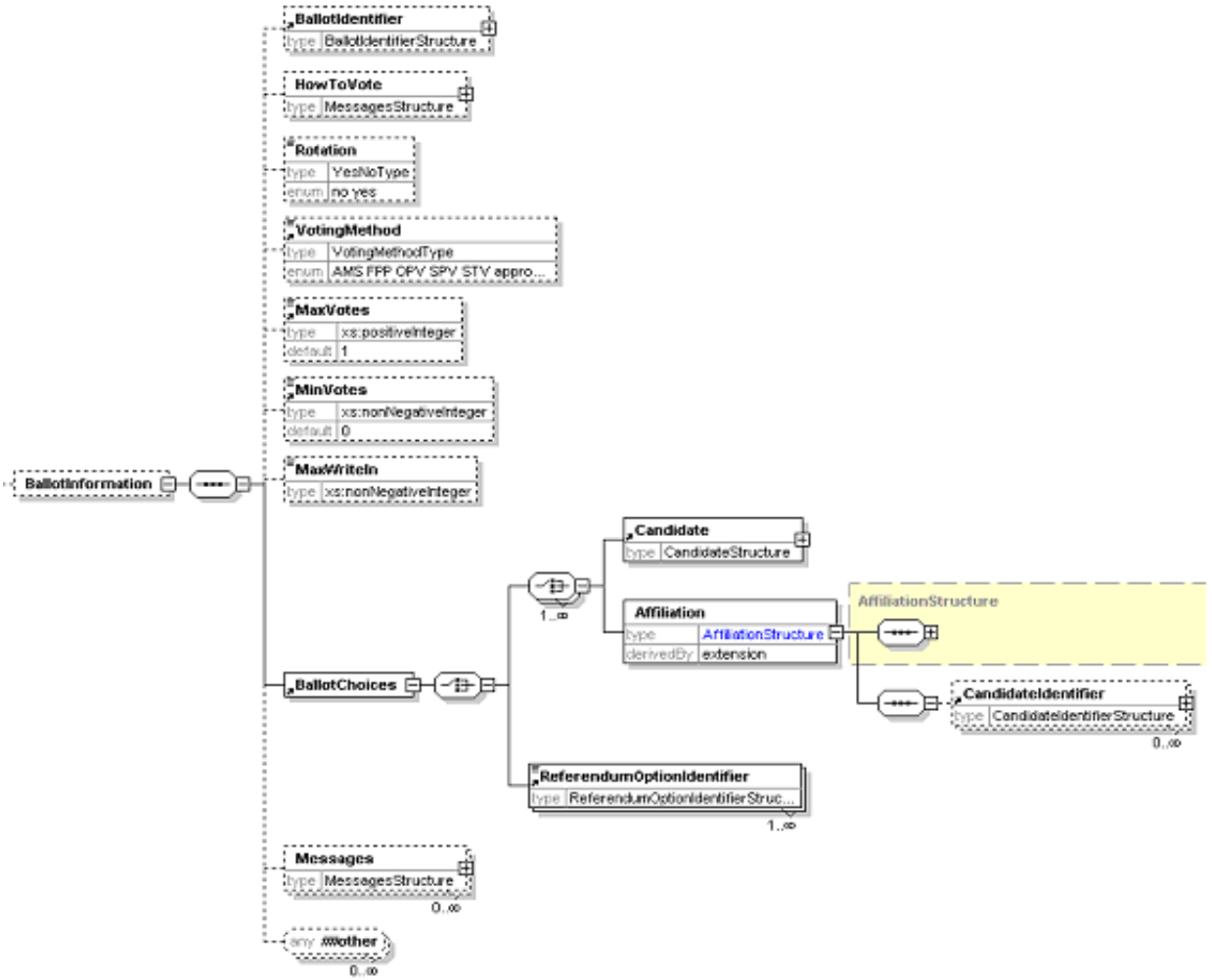
Error Code	Error Description
3330-002	The polling district can only be included for either the voter or the election.
3330-003	The polling place can only be included for either the voter or the election.

8.9 Polling Information (340)





923



924

Element	Attribute	Type	Use	Comment
BallotChoices	Contested	YesNoType	optional	
VotingPeriod	DisplayOrder	xs:positiveInteger		
VotingInformation	DisplayOrder	xs:positiveInteger	optional	
	Channel	VotingChannelType	optional	

925 ## 8.9.1 Description of Schema

926 The polling information message defined by this schema is sent to a voter to provide details of how to
927 vote. It can also be sent to a distributor, so multiple sets of information are allowed. In the case of SMS
928 voting, ballot information may also be required, so this can be included. Either one or several sets of
929 polling information may be sent to each voter for any election event.

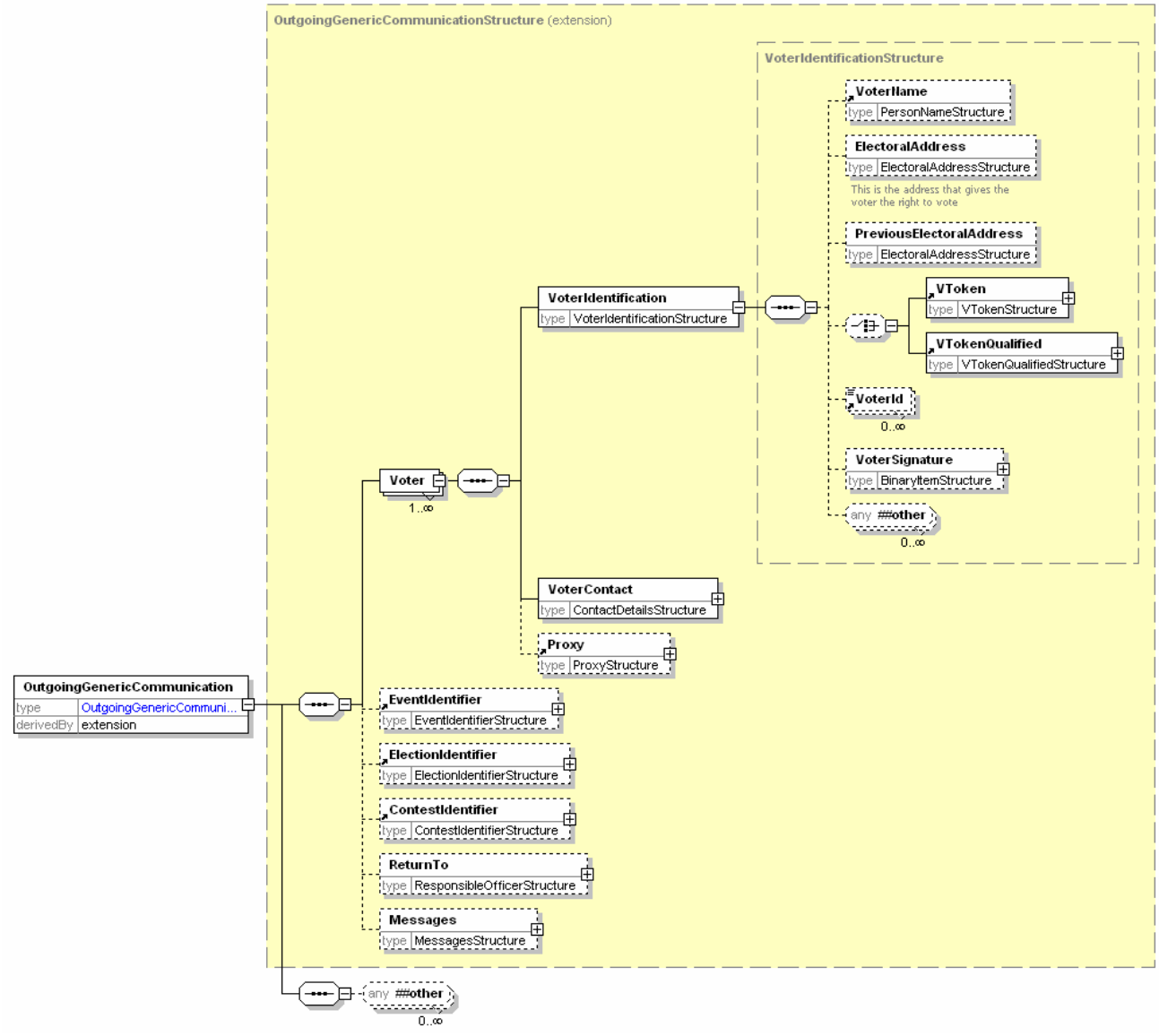
930 Some information about the voter and any proxy may be included, for example to print on a polling card.
931 This can also include a mailing address for a distributor to use.

932 Information about the elections and contests is included for the benefit of the voter. For each voting
933 channel, this includes where to vote (which could be a polling station, address for postal voting, URL for
934 Internet voting, phone number for SMS voting etc) and the times that votes can be placed. Use of the
935 DisplayOrder attribute on these allows the display or printing of information to be tailored from within the
936 XML message.

937 Ballot information may be included if required. This is a subset of the information defined in the 410-
938 ballots schema. In this case, it is likely that the short code for a candidate will be used for SMS voting. It is

possible that an expected response code will be provided as well. Both the short code and expected response code may be tailored to the individual voter as part of a security mechanism.

8.10 Outgoing Generic Communication (350a)



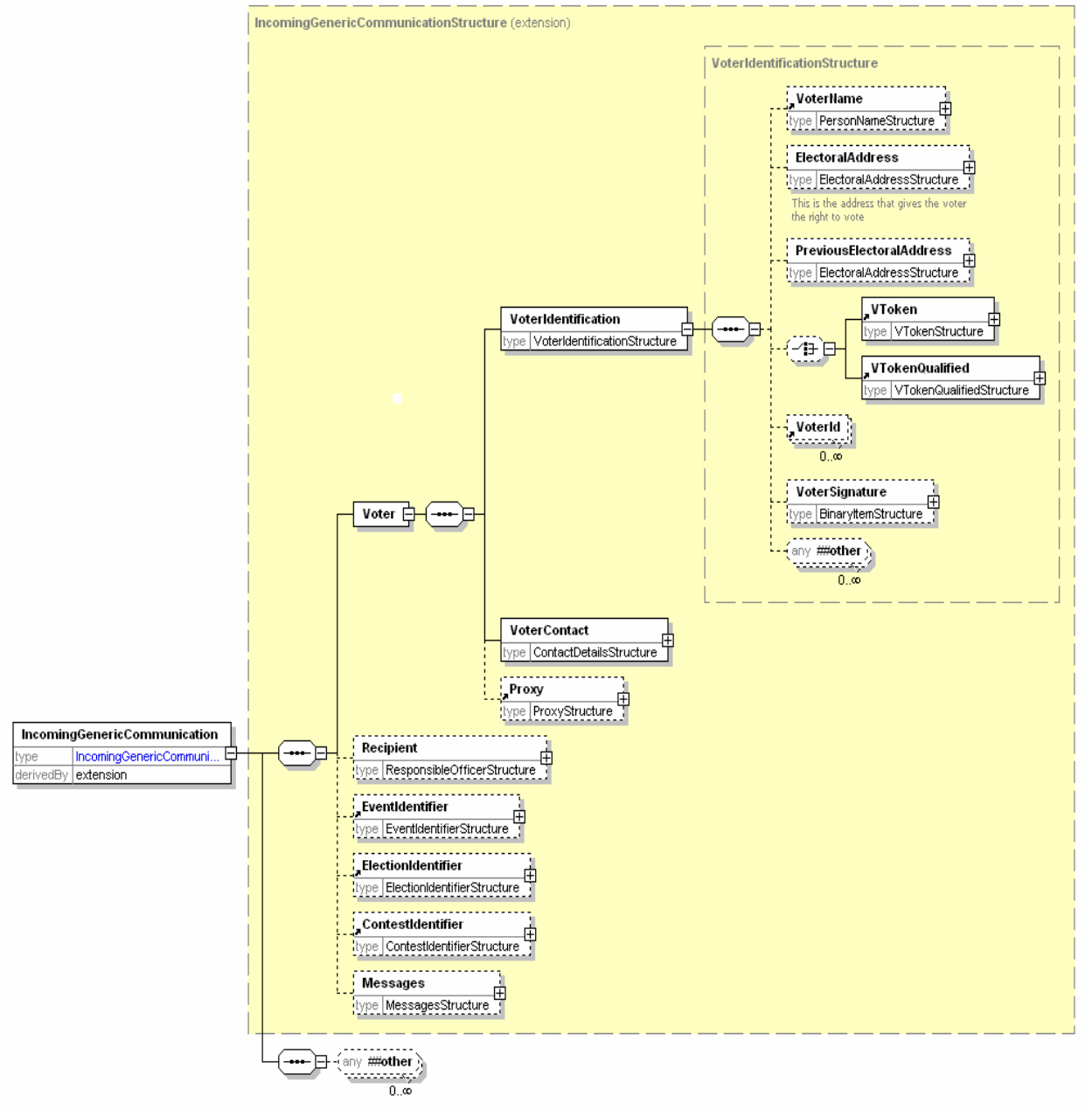
8.10.1 Description of Schema

This schema provides a common structure for communications to the voter. Individual message types can be designed based on extensions of this schema.

The voter must always provide a name and might provide one or more identifiers. These are shown as a restriction of the VoterIdentificationStructure, the restriction being to leave out the VToken and VTokenQualified. Contact details are also required, and it is expected that at least one of the allowed contact methods will be included. Inclusion of proxy information is optional.

The identifiers for the election event, election and contest are optional. There is then an element in which a message can be placed in any of several different formats according to the channel being used.

8.11 Incoming Generic Communication (350b)



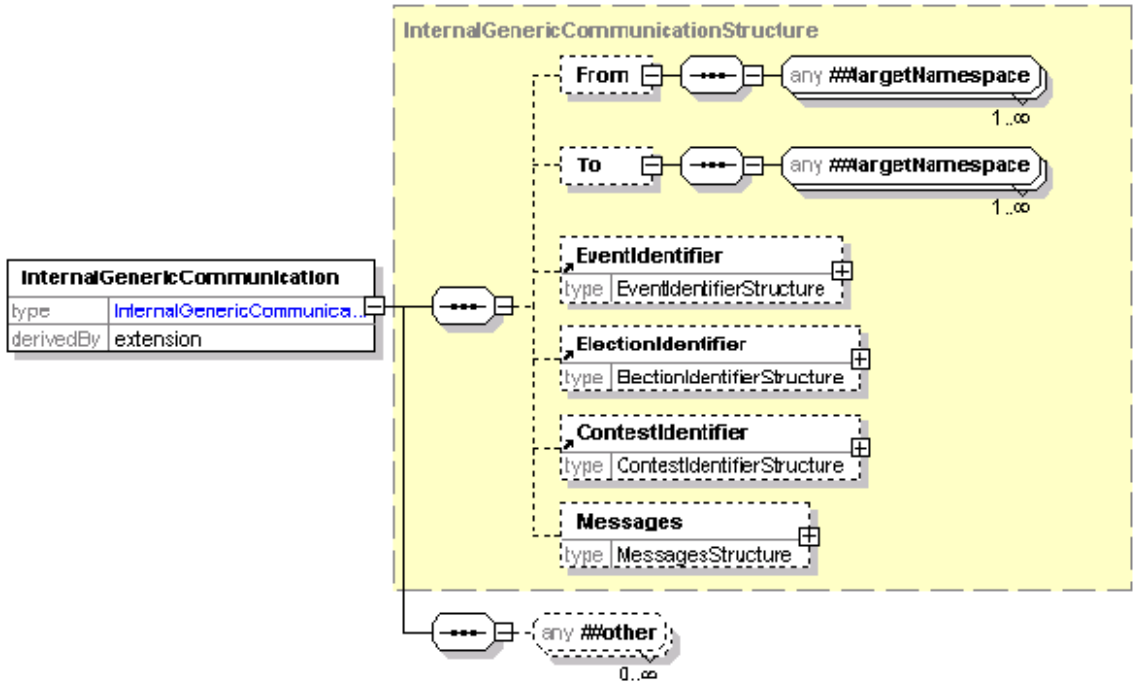
8.11.1 Description of Schema

This schema provides a common structure for communications from the voter. Individual message types can be designed based on extensions of this schema.

The voter's name must be provided and there can be one or more identifiers. These are shown as a restriction of the `VoterIdentificationStructure`, the restriction being to leave out the `VToken` and `VTokenQualified`. Contact details are also required, and it is expected that at least one of the allowed contact methods will be included. Inclusion of proxy information is optional.

The identifiers for the election event, election and contest are optional. There is then an element in which a message can be placed in any of several different formats according to the channel being used.

963 ## 8.12 Internal Generic (350c)



964

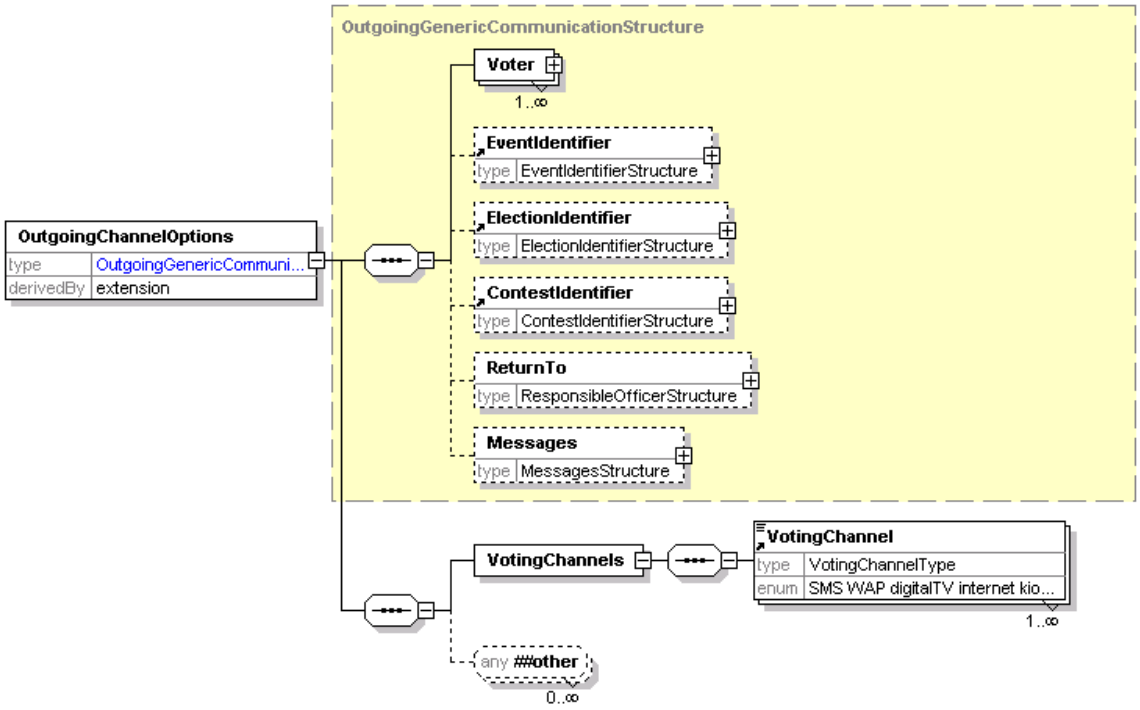
965 ### 8.12.1 Description of Schema

966 This schema provides a common structure for communications between those involved in organizing an
967 election. Individual message types can be designed based on extensions of this schema.

968 There are optional `To` and `From` elements, which can contain any EML elements. It is expected that
969 these will usually be a responsible officer or a person's name and contact information.

970 The identifiers for the election event, election and contest are optional. There is then an element in which
971 a message can be placed in any of several different formats according to the channel being used.

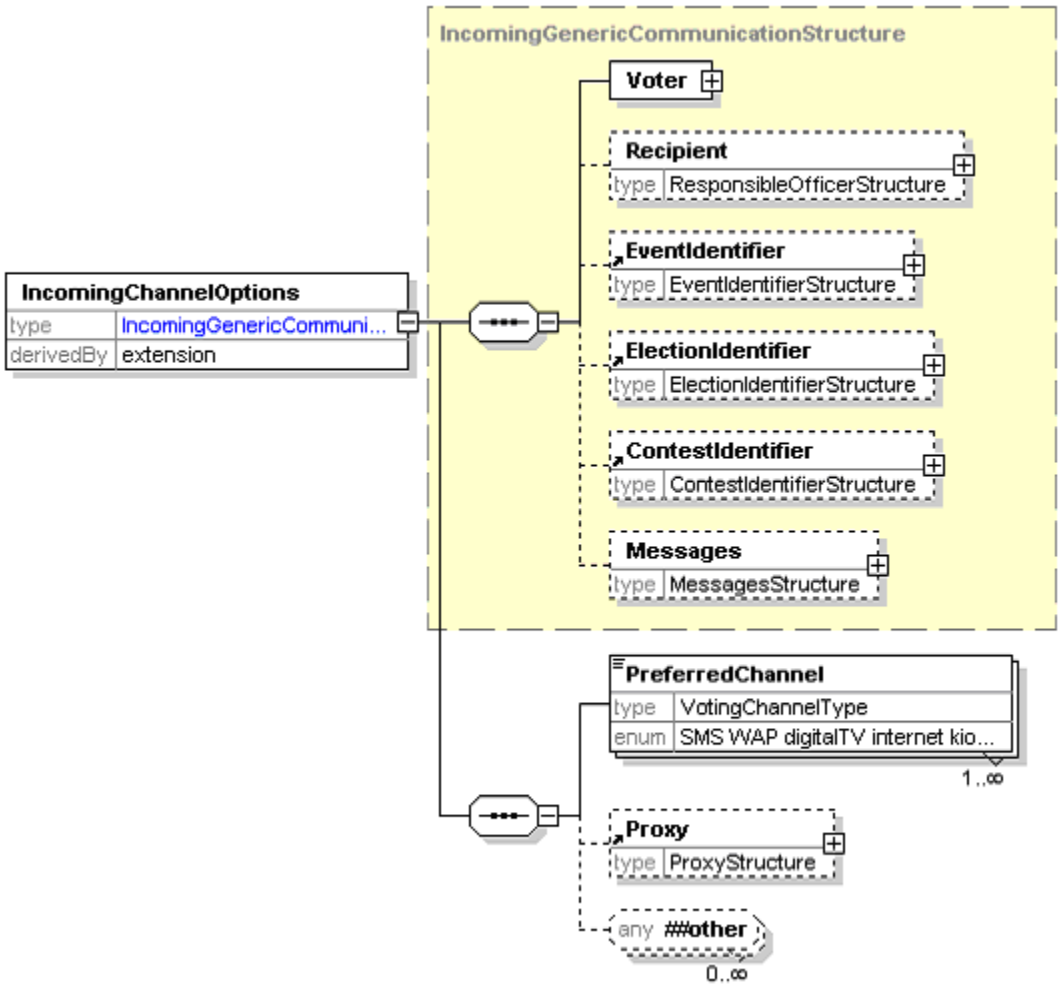
8.13 Outgoing Channel Options (360a)



8.13.1 Description of Schema

This schema is used for messages offering a set of voting channels to the voter. It is an extension of schema 350a. A message conforming to this schema will include a list of allowed channels, either to request general preferences or for a specific election event or election within the event.

978 ## 8.14 Incoming Channel Options (360b)



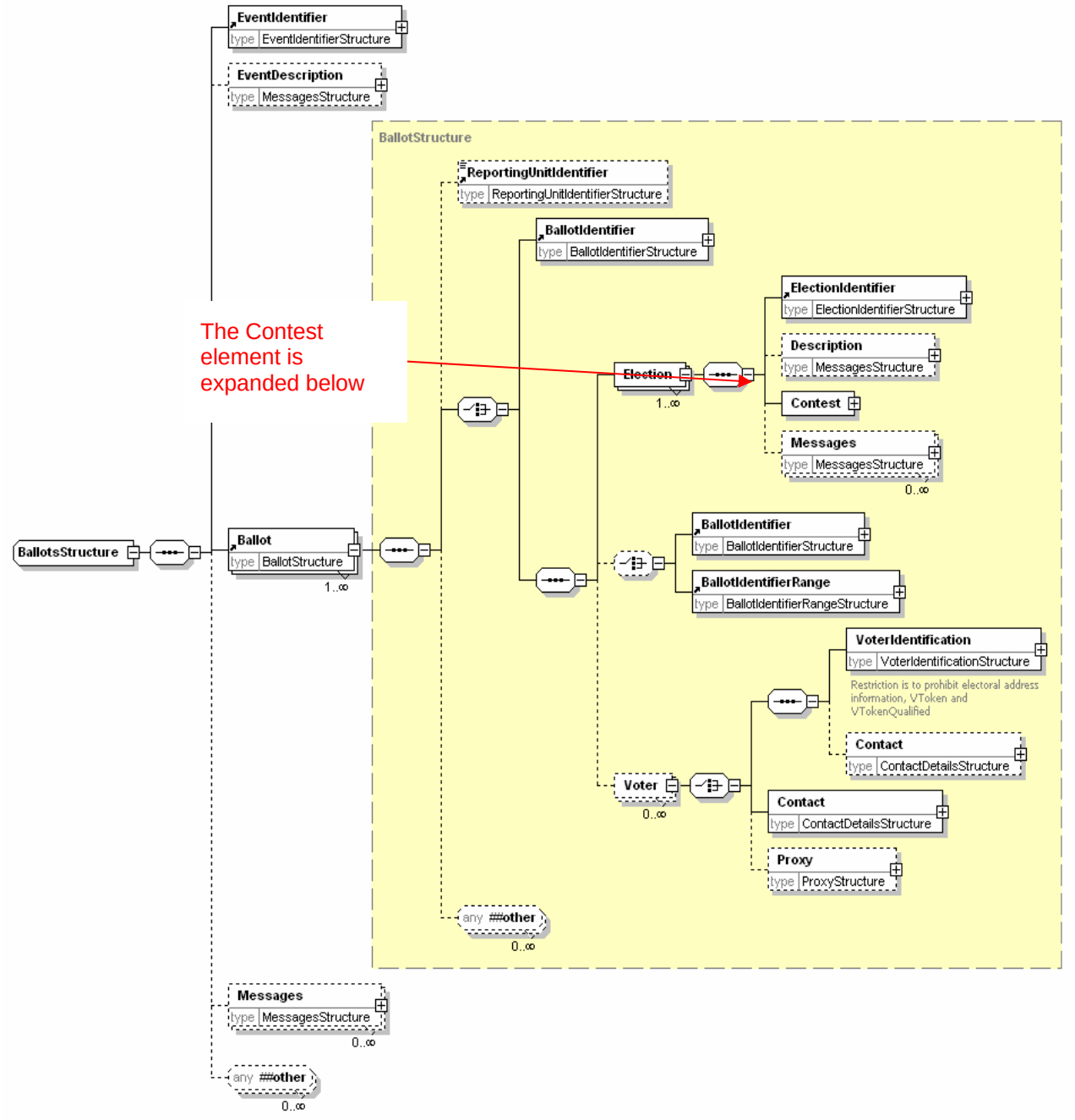
979

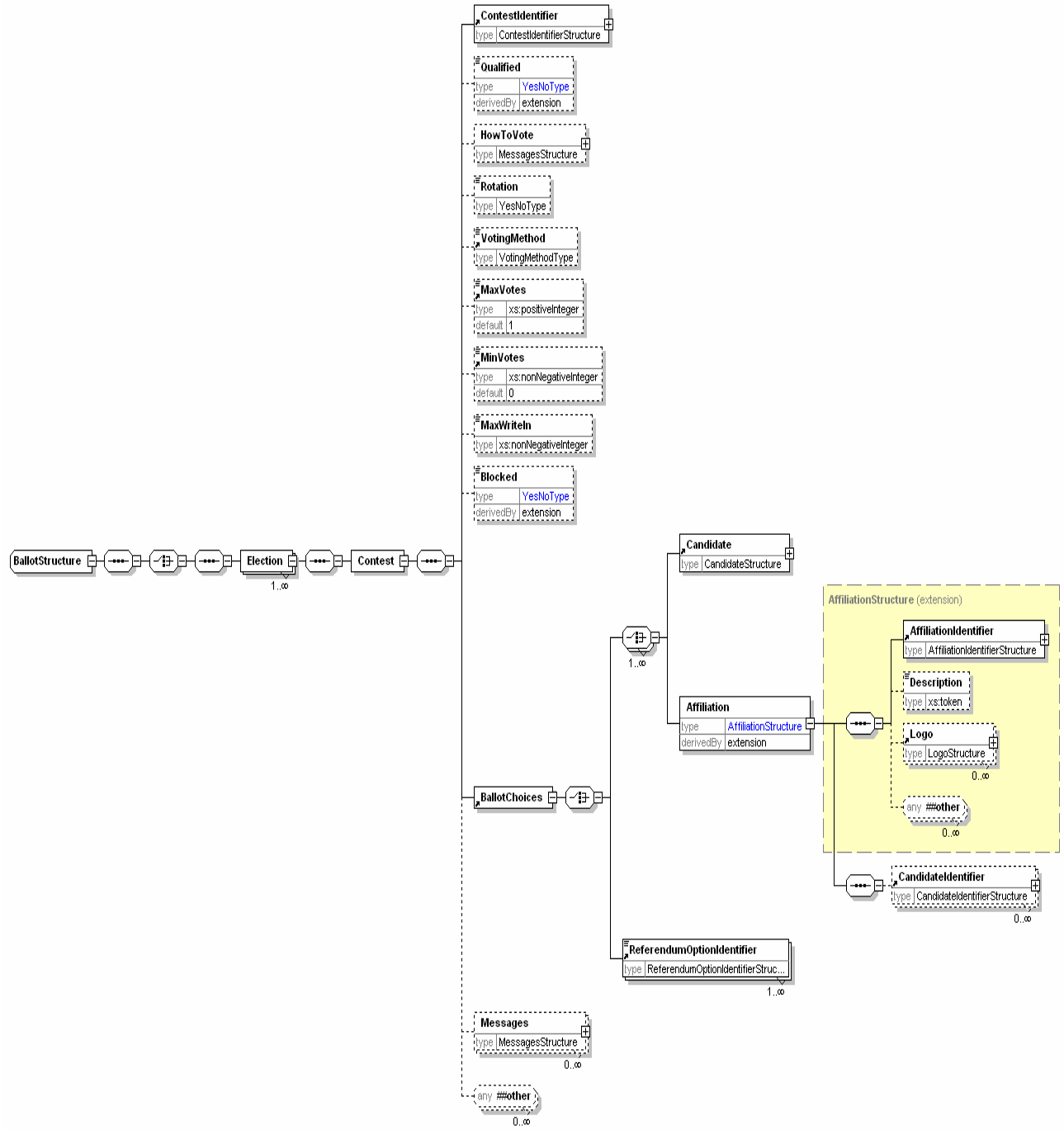
980 ### 8.14.1 Description of Schema

981 This schema is used for messages indicating one or more preferred voting channels. It may be sent in
982 response to 360a or as an unsolicited message if this is supported within the relevant jurisdiction.

983 It is an extension of schema 350b, and indicates preferred voting channels in order of preference.

8.15 Ballots (410)





986

Element	Attribute	Type	Use	Comment
Contest	DisplayOrder	xs:positiveInteger	optional	
	Completed	YesNoType	optional	
Qualified	Reason	xs:token	required	
Blocked	Reason	xs:token	optional	
	Channel	VotingChannelType	optional	
BallotChoices	Contested	YesNoType	optional	

8.15.1 Description of Schema

This schema is used for messages presenting the ballot to the voter or providing a distributor with the information required to print or display multiple ballots.

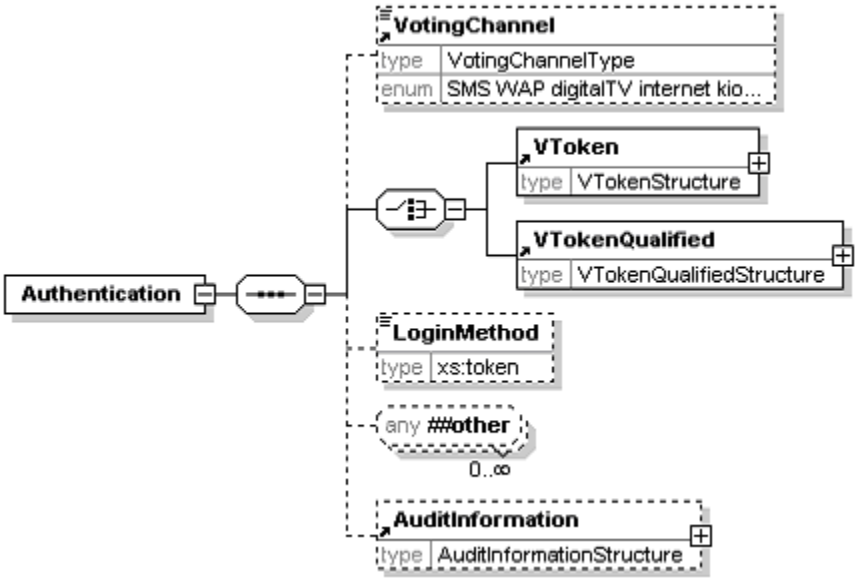
In the simplest case, a distributor can be sent information about the election event and a ballot ID to indicate the ballot to print.

In other cases, the full information about the elections will be sent with either an election rule ID to identify the voters to whom that election applies or a set of voter names and contact information. If the ballot is being sent directly to the voter, this information is not required. Since printed ballot papers are likely to require a unique identifier printed on them, the range to be used for each ballot type can be defined.

The election information starts with the election identifier and description. This is followed by information related to the contest and any other messages and information required. Note that each voter can only vote in a single contest per election, so only a single iteration of the Contest element is required.

A contest must have its identifier and a list of choices for which the voter can vote. A voter can vote for a candidate, an affiliation (possibly with a list of candidates) or a referendum proposal. There is also a set of optional information that will be required in some circumstances. Some of this is for display to the voter (`HowToVote` and `Messages`) and some controls the ballot and voting process (`Rotation, VotingMethod, MaxVotes, MinVotes, MaxWriteIn`).

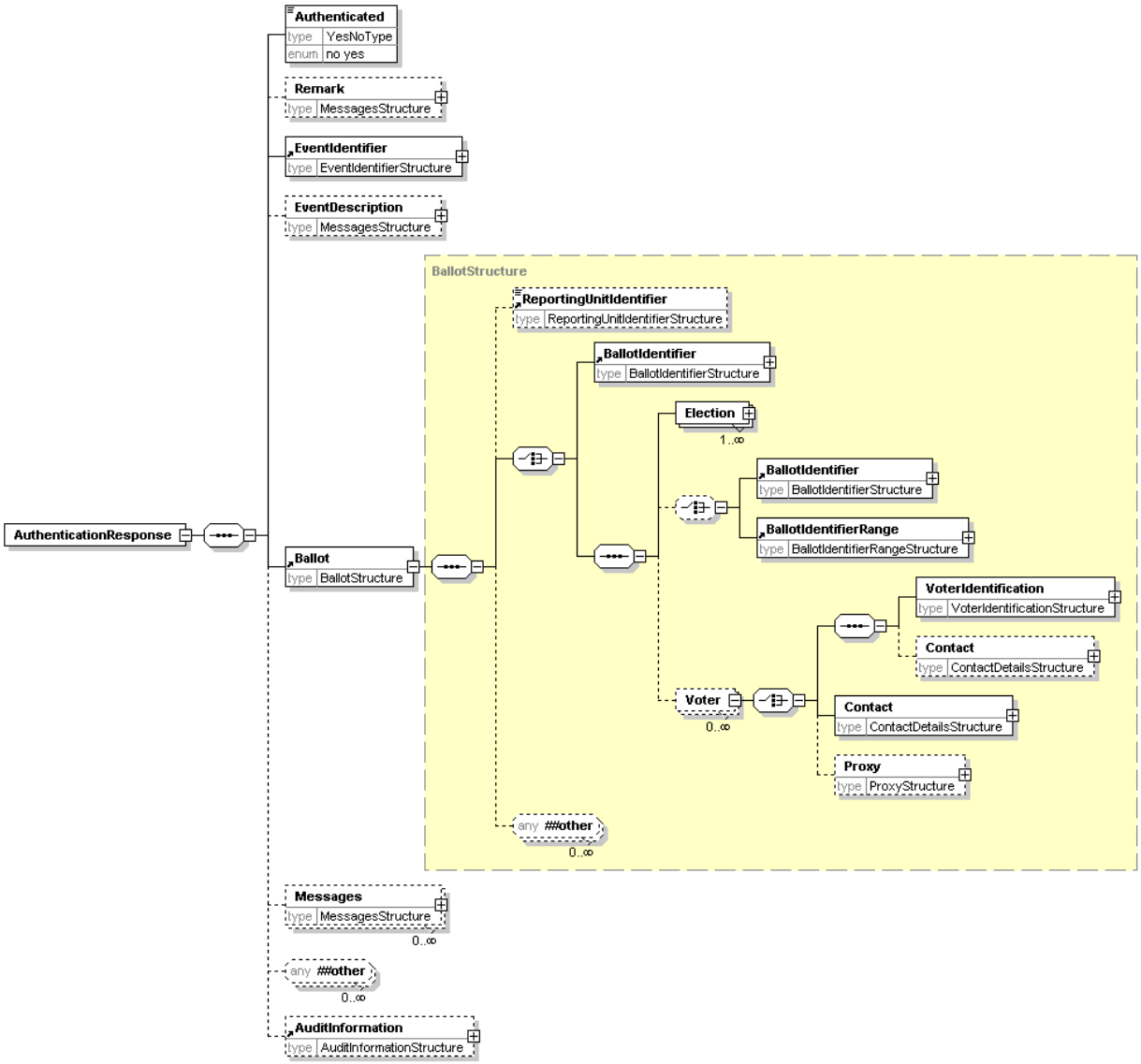
8.16 Authentication (420)



8.16.1 Description of Schema

The authentication message defined by this schema may be used to authenticate a user during the voting process. Depending on the type of election, a voter's authentication may be required. The precise mechanism used may be channel and implementation specific, and can be indicated using the `LoginMethod` element. In some public elections the voter must be anonymous, in which case the prime method used for authentication is the voting token. The voting token can contain the information required to authenticate the voter's right to vote in a specific election or contest, without revealing the identity of the person voting. Either the `VToken` or the `VTokenQualified` must always be present in an authenticated message. The `VotingChannel` identifies the channel by which the voter has been authenticated.

8.17 Authentication Response (430)

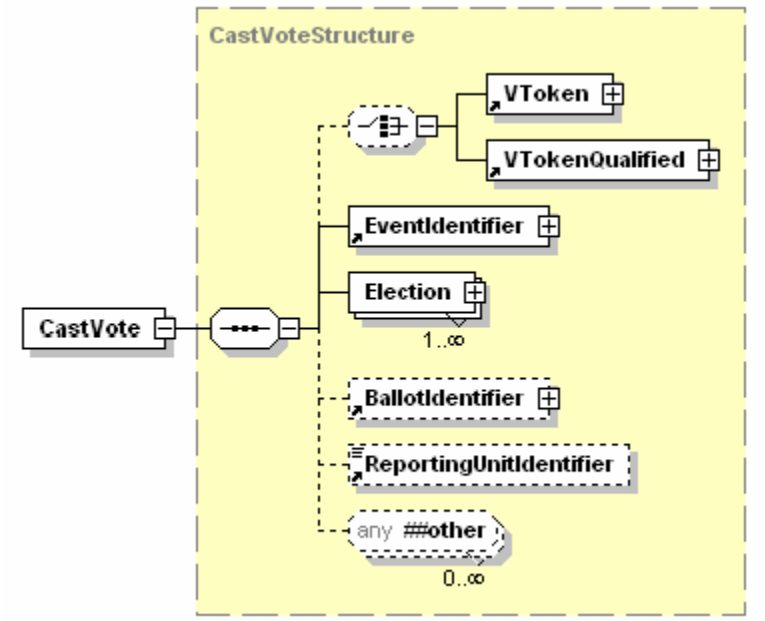


Element	Attribute	Type	Use	Comment
Contest	DisplayOrder	xs:positiveInteger	optional	
	Completed	YesNoType	optional	
Qualified	Reason	xs:token	required	
Blocked	Reason	xs:token	optional	
	Channel	VotingChannelType	optional	
BallotChoices	Contested	YesNoType	optional	

8.17.1 Description of Schema

The authentication response is a response to message 420. It indicates whether authentication succeeded using the Authenticated element, and might also present the ballot to the user. This is a restriction of the Ballots element to allow only a single ballot per reply.

8.18 Cast Vote (440)



Element	Attribute	Type	Use	Comment
CastVote	Spoilt	xs:token	optional	
Contest	Spoilt	xs:token	optional	
Selection	Value	VotingValueType	optional	
	ShortCode	ShortCodeType	optional	
Candidate	Value	VotingValueType	optional	

8.18.1 Description of Schema

This message represents a cast vote, which comprises an optional voting token (which may be qualified) to ensure that the vote is being cast by an authorized voter, information about the election event, each election within the event and the vote or votes being cast in each election, an optional reference to the ballot used, the identifier of the reporting unit if applicable and a set of optional audit information.

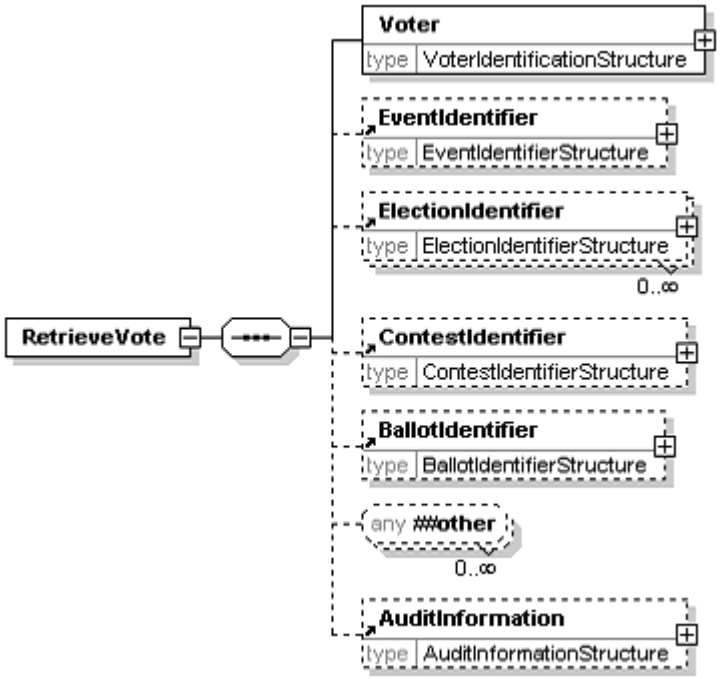
For each election, the contest is identified, with a set of, possibly sealed, votes. The votes are sealed at this level if there is a chance that the message will be divided, for example so that votes in different elections can be counted in different locations.

The selection of candidates, affiliations or a referendum option uses the `Selection` element. If an election requires preferences to be expressed between candidates, multiple `Selection` elements will be used, each of these having a suitable `Value` attribute. Some elections allow write-in candidates, and these are handled in a similar way. Preferences can also be expressed between parties, using the `Affiliation` element. The `PersonalIdentifier` is used in elections where each voter is given an individual list of codes to indicate their selection.

A more complex election might request the voter to vote for a party, then express a preferences of candidates within the party. In this case, the `Affiliation` element is used to indicate the party selected, and multiple `CandidateIdentifier` elements, each with a `Value` attribute are used to express candidate preferences.

Preferences in a referendum are handled in the same way as they are for candidates and parties, using the `ReferendumOptionIdentifier.`

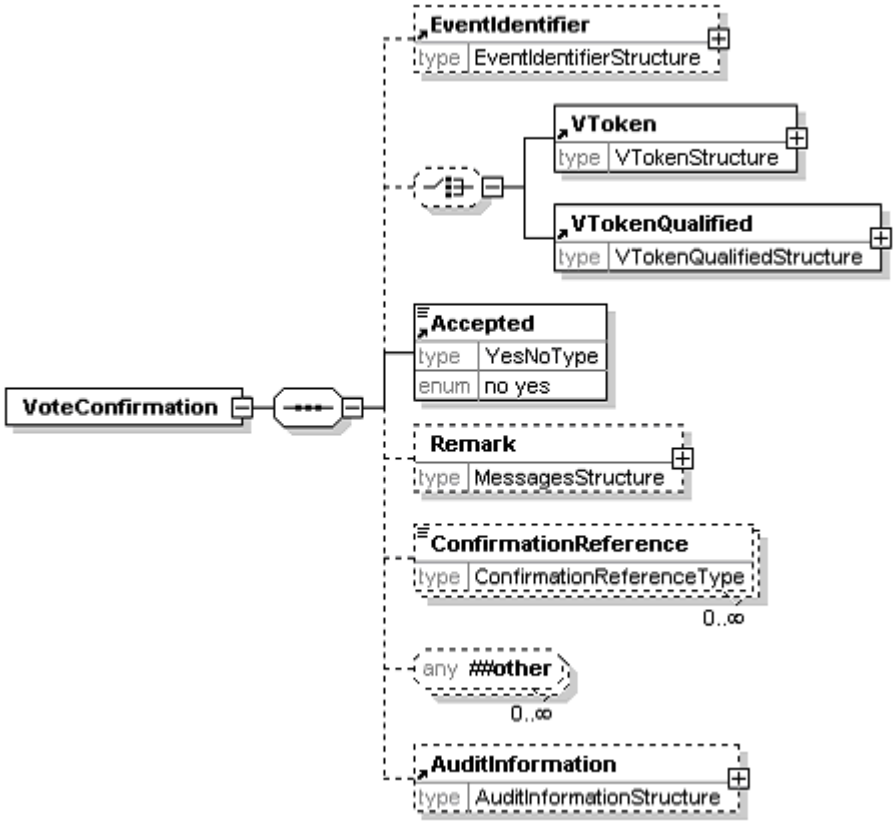
8.19 Retrieve Vote (445)



8.19.1 Description of Schema

This message is used for voting systems that include a pre-ballot box from which votes can be retrieved and amended before being counted. When a vote is retrieved, it should be deleted from the pre-ballot box.

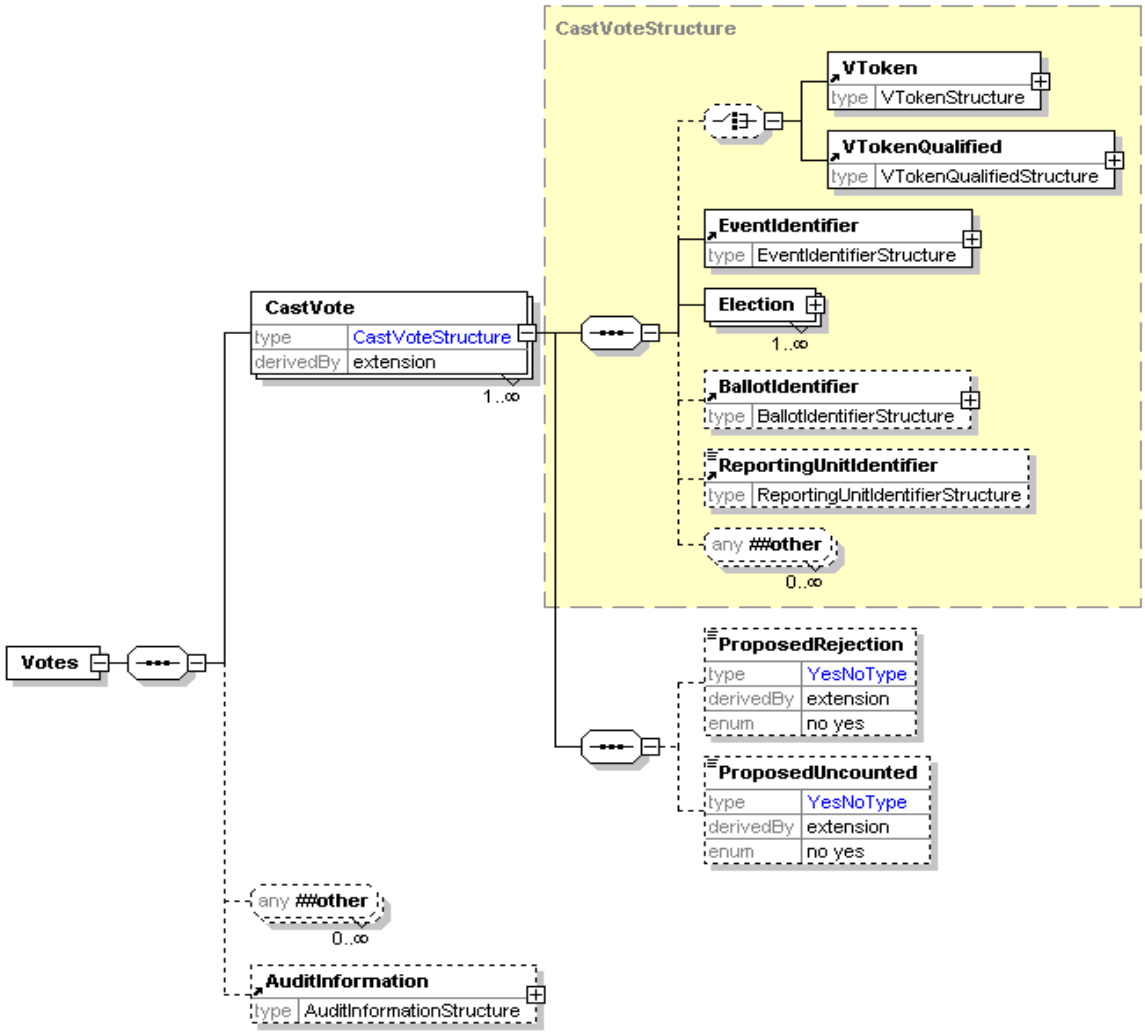
8.20 Vote Confirmation (450)



8.20.1 Description of Schema

The vote confirmation message can be used to show whether a vote has been accepted and provide a reference number in case of future queries. Some voting mechanisms require multiple ConfirmationReference elements. If the vote is rejected, the Remark element can be used to show a reason.

8.21 Votes (460)



See 440-CastVote for the detail of the `CastVoteStructure`.

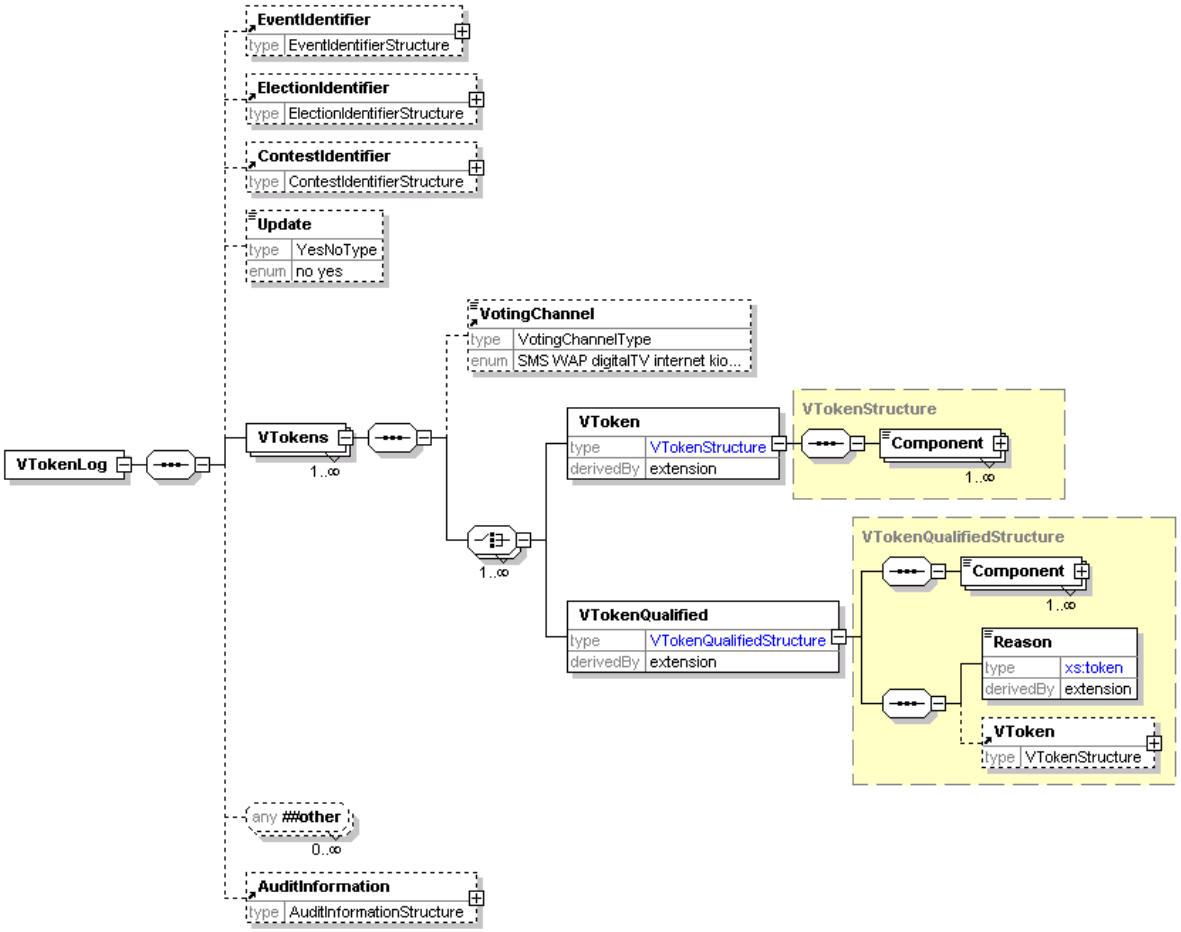
Element	Attribute	Type	Use	Comment
CastVote	Spoilt	xs:token	optional	
Contest	Spoilt	xs:token	optional	
Selection	Value	VotingValueType	optional	
	ShortCode	ShortCodeType	optional	
Candidate	Value	VotingValueType	optional	
ProposedRejection	Reason	xs:token	optional	
	ReasonCode	xs:token	required	
	Objection	YesNoType	optional	
ProposedUncounted	Reason	xs:token	optional	
	ReasonCode	xs:token	required	
	Objection	YesNoType	optional	

8.21.1 Description of Schema

This schema is used to define a message comprising a set of votes being transferred for counting. It is a set of `CastVote` elements from schema 440 with the addition of the `ProposedRejection` and `ProposedUncounted` elements and audit information for the voting system. If a vote is rejected, for example, because a voter has chosen to spoil a ballot paper, many authorities will want to count that vote

as having been cast. The `UncountedVotes` element is reserved for those cases where that record is not required, for example when the result is thought to be fraudulent. A `ProposedRejection` or `ProposedUncounted` element must have a `ReasonCode` attribute, and may have a `Reason` attribute to describe the code. They may also have an `Objection` attribute. This indicates that someone has objected to this vote being rejected or the proposal that it should not be counted.

8.22 VToken Log (470)



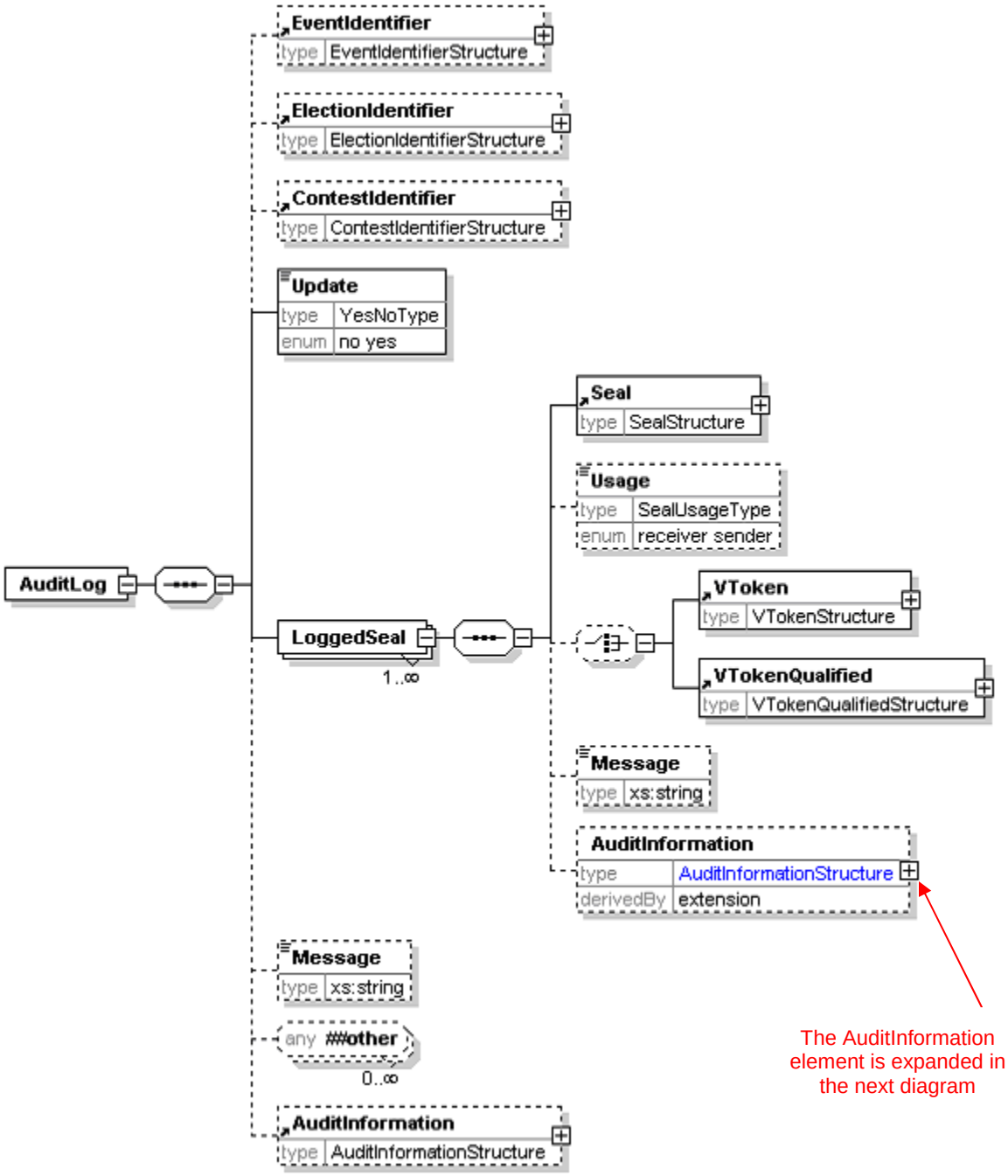
Element	Attribute	Type	Use	Comment
VToken	Status	xs:token (restricted)	required	
VTokenQualified	Status	xs:token (restricted)	required	

8.22.1 Description of Schema

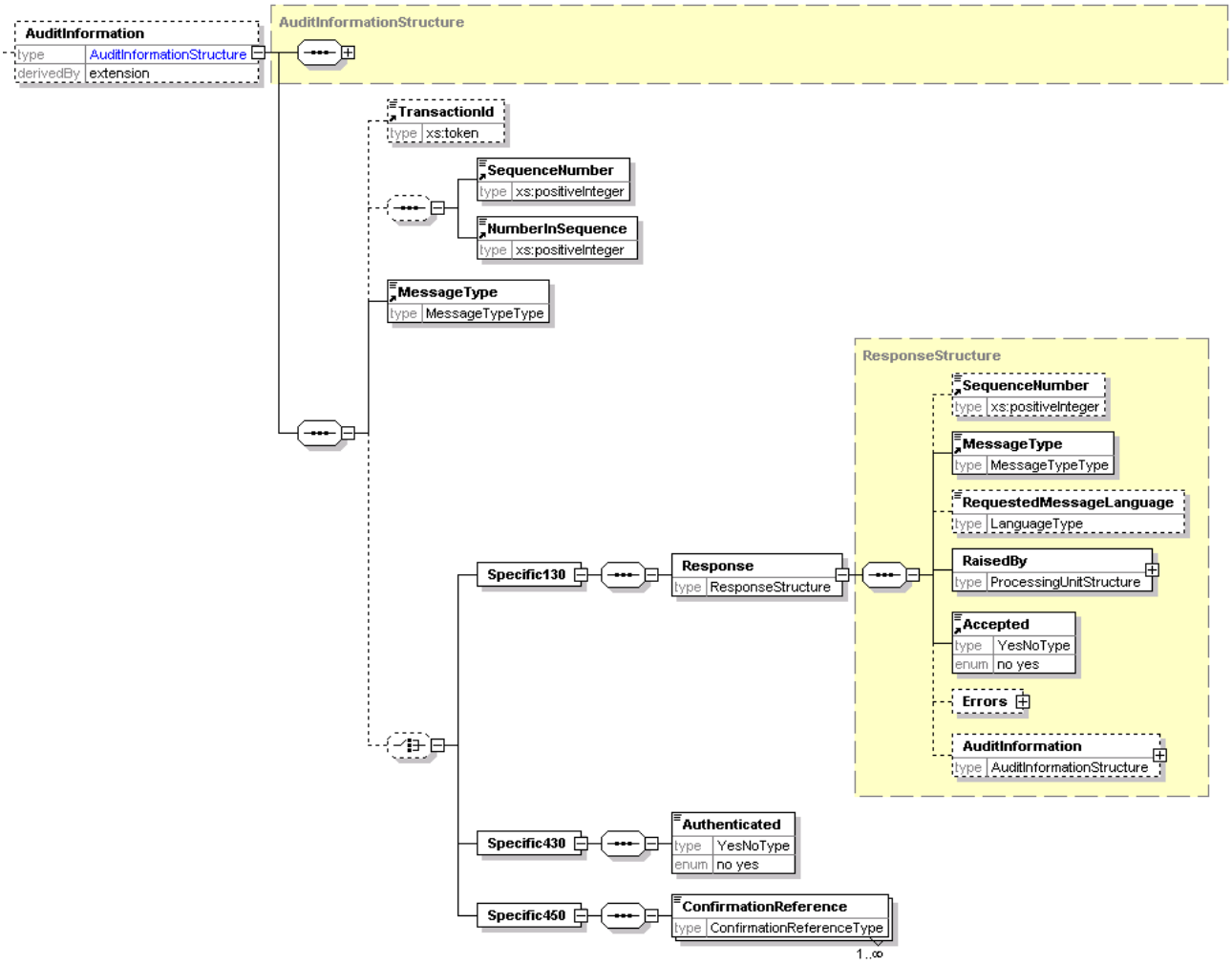
The message defined by this schema is used to add voting tokens (which may be qualified) to an audit log. The `VToken` or `VTokenQualified` is extended by the addition of a `Status` attribute with a value of voted or unvoted for the `VToken` and voted, unvoted and withdrawn for the `VTokenQualified`. In addition to sending single tokens as they are used, the schema can be used to validate a message sending multiple tokens optionally grouped by voting channel. This might be used instead of sending tokens as they used or, for example, to send the unused tokens at the end of an election. The `Update` element can be used to indicate that an existing log is being updated rather than the message containing a complete new log. The logging system can also be identified for audit purposes.

1080

8.23 Audit Log (480)



1081



8.23.1 Description of Schema

The message defined by this schema is used to log the use of each seal with associated information for audit purposes.

An audit log message can be transmitted individually as the message causing the log entry is sent or received, or the logs can be stored, and several seals logged at once. Ideally, every device that can create or consume a message will create a log entry so that pairs of entries can be matched. The most important messages to log are those associated with the voting process itself, and these are shown below.

When used in this message, the `Response` element will not have an `AuditInformation` child.

	Originating Device	*Gateway*	*Voting System*	*Counting System*	*Vtoken Logging System*	*Seal Logging System*	*Other*	*Notes*
130								4
410	next receiver	receiver	sender					
420	previous sender	sender	receiver					
430	next receiver	receiver	sender				sender / receiver	3
440	previous sender	sender	receiver					
445	previous sender	sender	receiver					
450	next receiver	receiver	sender					
460			sender	receiver				
470			sender	sender	receiver		sender	
480	sender	sender	sender	sender	sender	receiver	sender	2
510				sender			receiver	
520				sender			sender / receiver	

Notes:

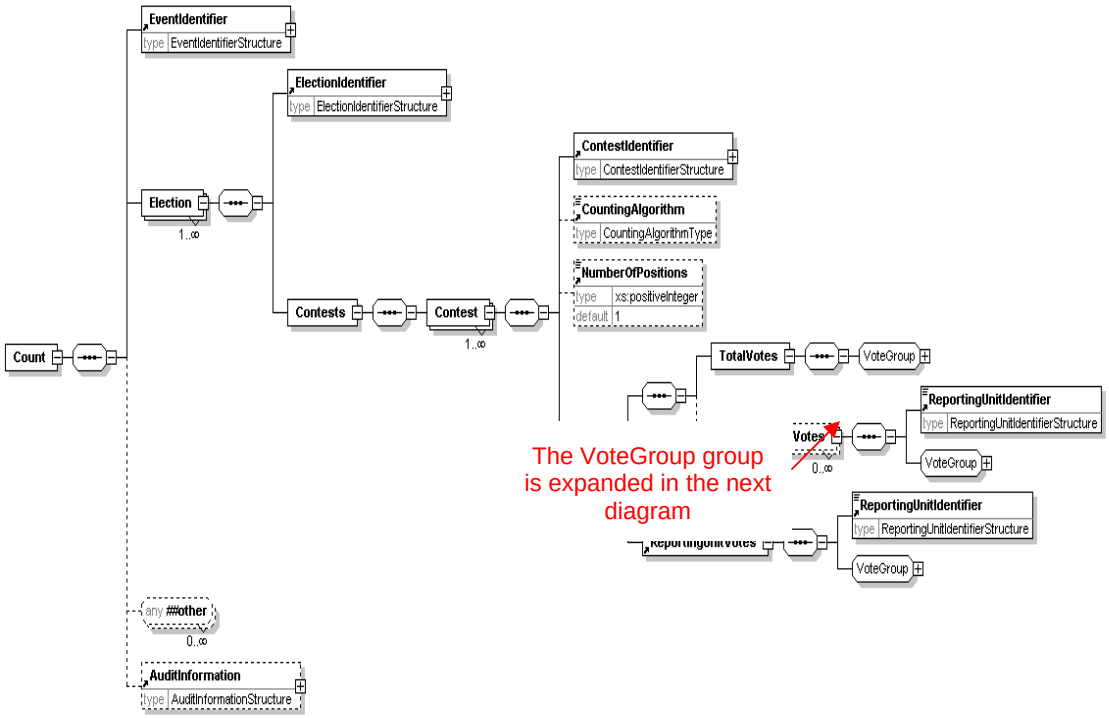
1. In some cases (e.g. a kiosk) there may be no gateway involved. In this case, the values in the Gateway column apply to the Originating Device.
2. Creators and receivers of 480 (audit log) messages may not be required to log the seals. In particlar, if an adit log message is sent per seal created or received, the seal on the 480 message must not be logged.
3. "Other" may be the sender when the message is sent to a printer. In this case, the receiver will also be an "Other".
4. An audit log should only be created when the message is used to communicate an error. Most devices can send or receive 130 messages.

1092

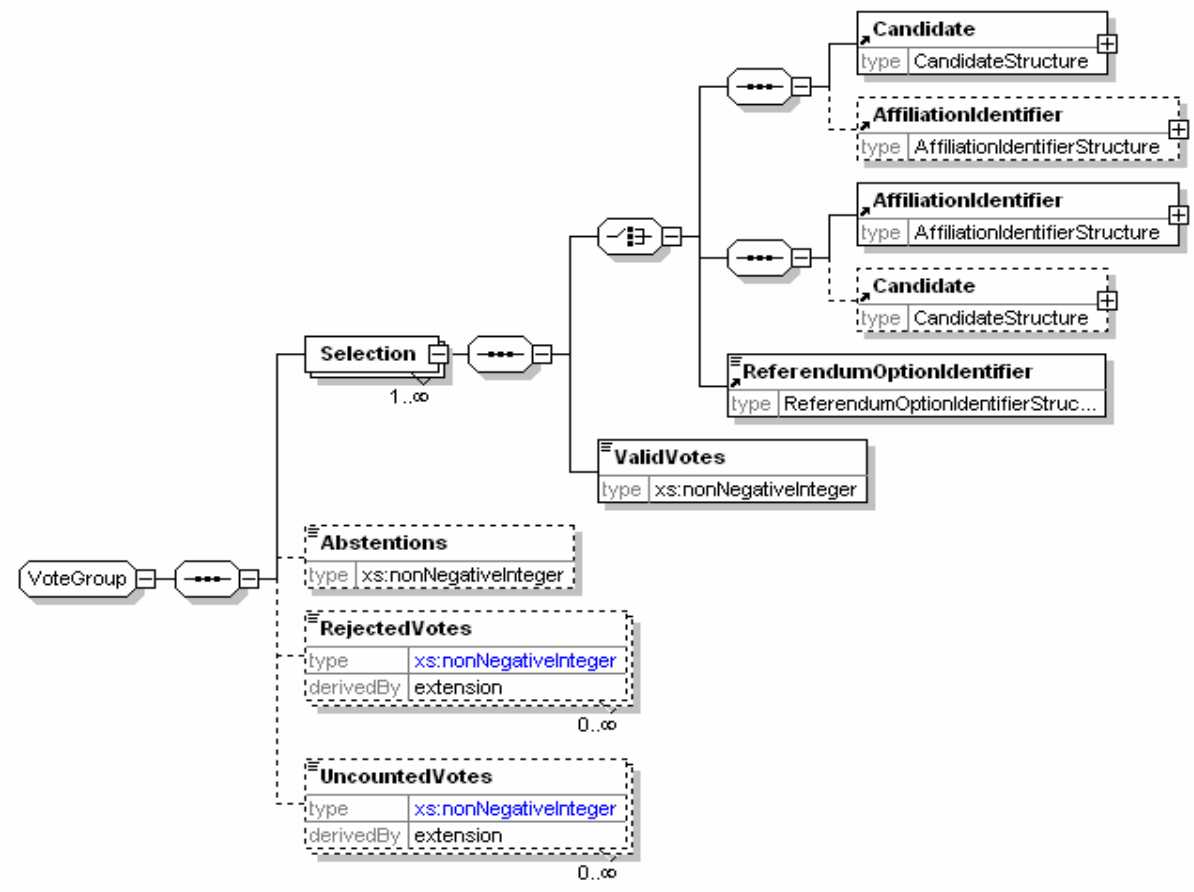
1093 The message may contain the name and ID of the event, election and contest. It can also indicate
1094 whether this is an update to an existing log or a new log. Following the logged seals, a text message can
1095 be added as well as audit information for the audit logging message itself.

1096 Each seal being logged must indicate whether the device sending the log was the sender or receiver of
1097 the sealed message. It may be accompanied by the voting token associated with the seal and possibly
1098 additional audit information. This will be the audit information from the message being logged with
1099 additional information about the message. Most of this is common to all message types, but some
1100 message types require specific audit information. One of these is the 130-response message. When this
1101 is used to convey an error, almost the complete message payload (the `Response` element and its
1102 contents apart from the audit information) is logged with the usual message-independent data.

1103 # 8.24 Count (510)



1104



1105

Element	Attribute	Type	Use	Comment
Selection	Value	VotingValueType	optional	
RejectedVotes	Reason	xs:token	optional	
	ReasonCode	xs:token	required	
UncountedVotes	Reason	xs:token	optional	
	ReasonCode	xs:token	required	

8.24.1 Description of Schema

The count message defined by this schema is used to communicate the results of one or more contests that make up one or more elections within an election event. It may also be used to communicate the count of a single reporting unit for amalgamation into a complete count.

The message includes the election event identifier, and for each election, the election identifier, an optional reference to the election rule being used and information concerning the set of contests.

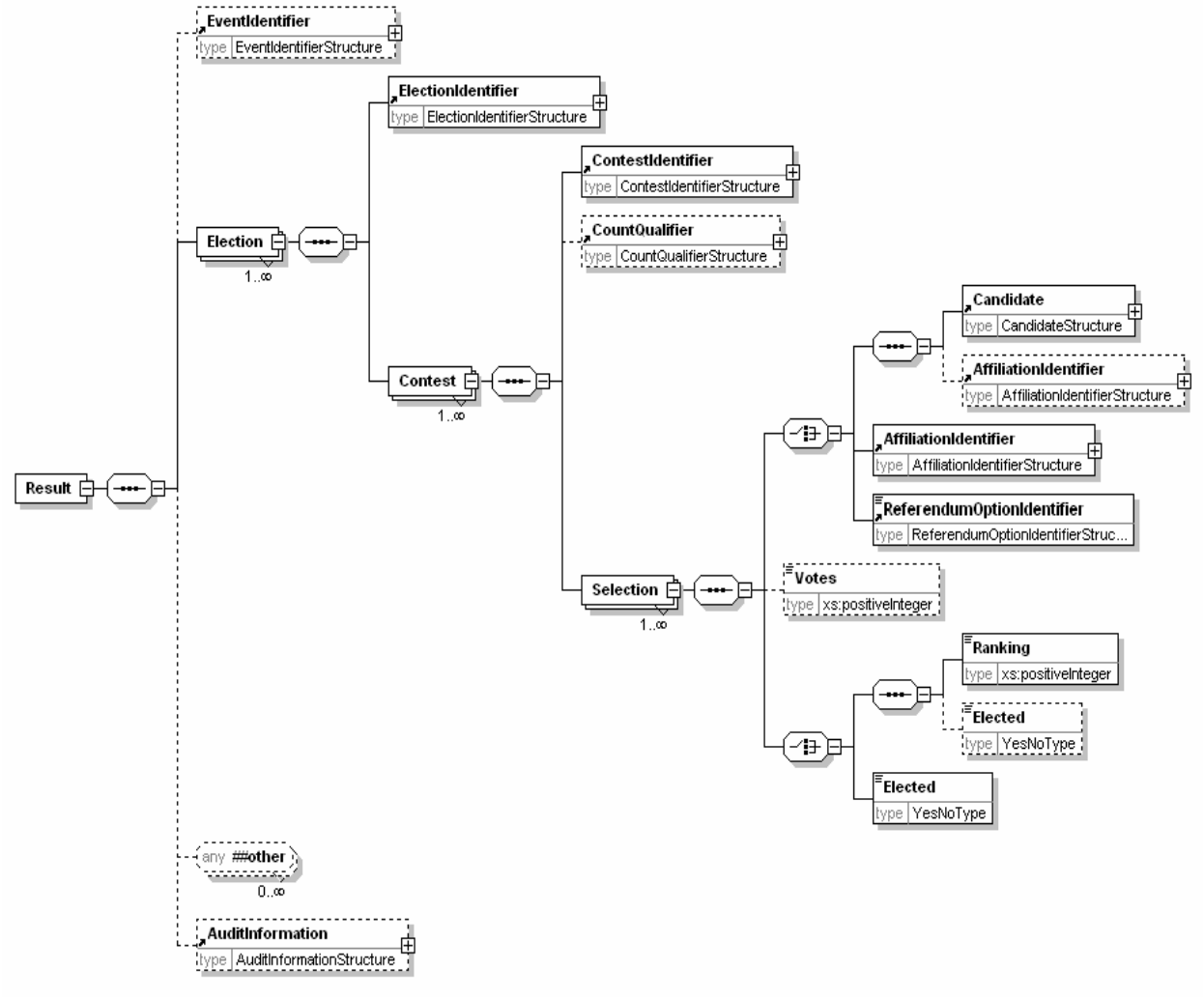
In some cases, reporting for a contest may be required at a lower level (for example, for each county in a state). For this reason, reporting may be done at the level of the reporting unit, the total votes, or for a total vote and the breakdown according to the multiple reporting units.

Each contest indicates its identifier, and optionally the counting system and the maximum number of votes that each voter could cast. The key information is that about the votes cast for each of the choices available and the numbers of abstentions and rejected and uncounted votes. If a vote is rejected, for example, because a voter has chosen to spoil a ballot paper, many authorities will want to count that vote as having been cast. The `UncountedVotes` element is reserved for those cases where that record is not required, for example when the result is thought to be fraudulent. Both the `UncountedVotes` and `RejectedVotes` elements have `Reason` (optional) and `ReasonCode` (mandatory) attributes to indicate why the votes were treated as they have been. The former is a textual description, and the latter a code.

For each choice available to the voter, the identifier and number of valid votes are mandatory. The other information provided depends on the type of election. For example, the `Value` attribute of the `Selection` element can be used to indicate whether a candidate was a first or second choice in an election run under the single transferable vote system. In the simplest cases, the identifier for the candidate (perhaps with the party), the party or the referendum option is given. If the voter was able to vote for a party and provide a preference for candidates within the party, the `AffiliationIdentifier` element is used, and multiple `CandidateIdentifier` elements may be used, each with a `Count` attribute. This count is the result of whatever algorithm has been used to calculate the ranking of the candidates.

This schema allows for Simulation and Extrapolation of Counts and subsequently Results. Simulation being the facility to forecast the result of a contest based on the result of another contest. Extrapolation is the facility to forecast the final result of a contest based on the count so far.

8.25 Result (520)



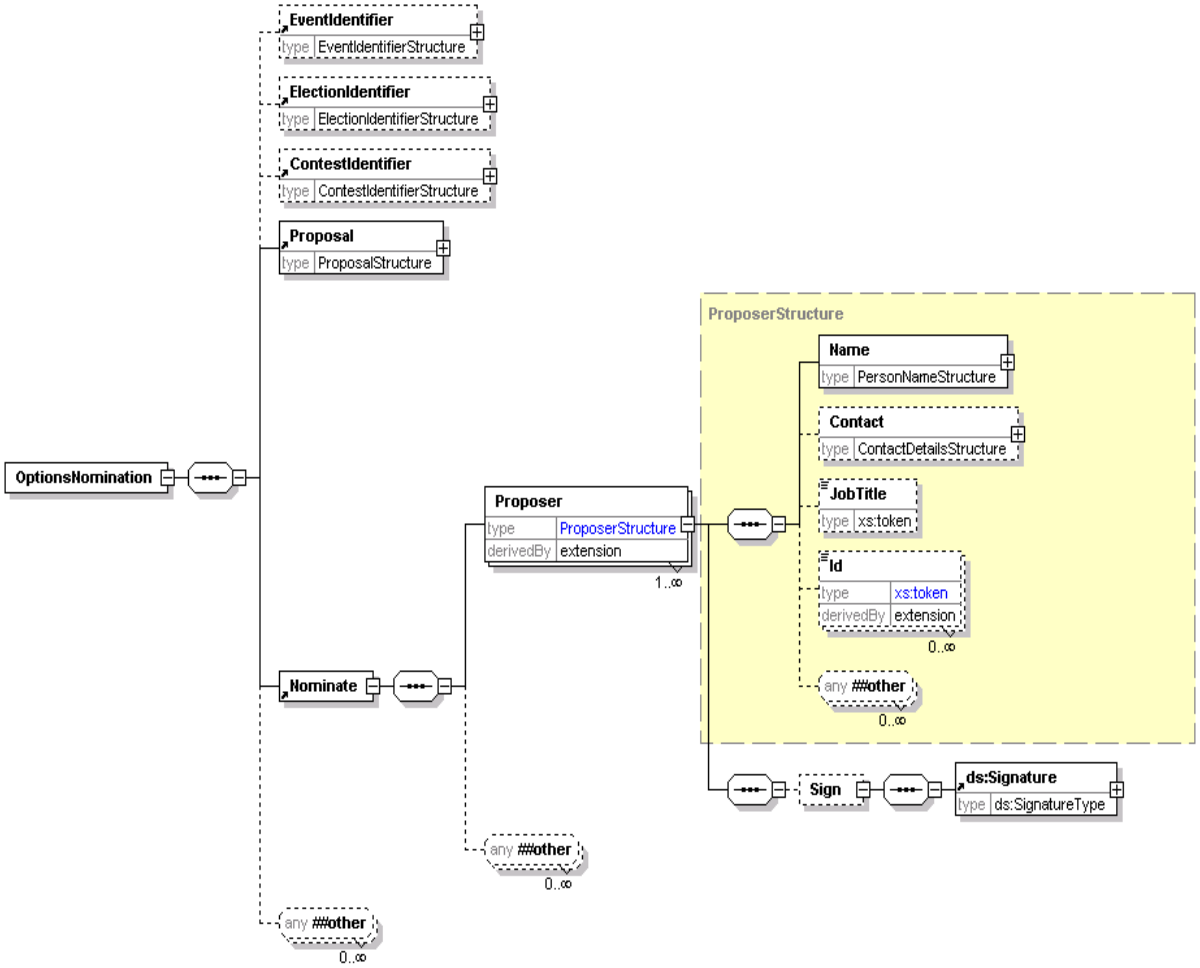
8.25.1 Description of Schema

Messages described by this schema can be used to communicate the results of simple election types. One specific use is to provide an input into the calculation algorithm for elections using the additional member system.

The main part of the schema is held within the `Selection` element. This allows a choice of candidate, affiliation or referendum option identifiers to be defined with the position that choice achieved (first, second etc). Optionally, the number of votes can be shown. A candidate can be associated with his or her affiliation if required. Write in candidates will be shown in the same way as other candidates, although they will only have an `Id` attribute if this is assigned in the election system after the votes are cast.

This schema allows for Simulation and Extrapolation of Results using data from Counts. Simulation being the facility to forecast the result of a contest based on the result of another contest. Extrapolation is the facility to forecast the final result of a contest based on the count so far.

1148 # 8.26 Options Nomination (610)

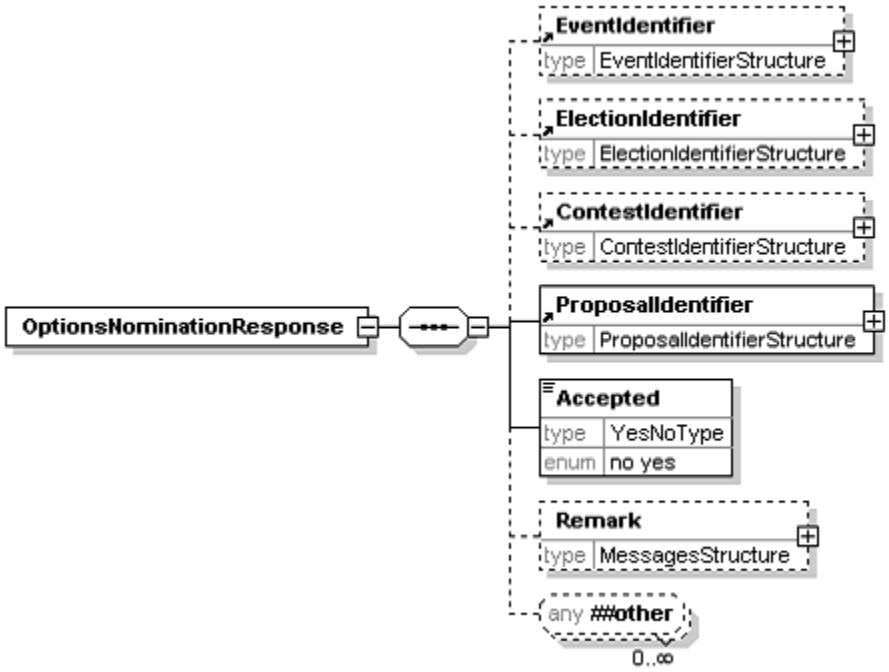


1149

1150 ## 8.26.1 Description of Schema

1151 This schema is used to submit proposals, for example for a referendum or company AGM. It uses the
1152 generic Proposal element to define the proposal itself. One of more proposers can be named and may
1153 sign the nomination.

8.27 Options Nomination Response (620)



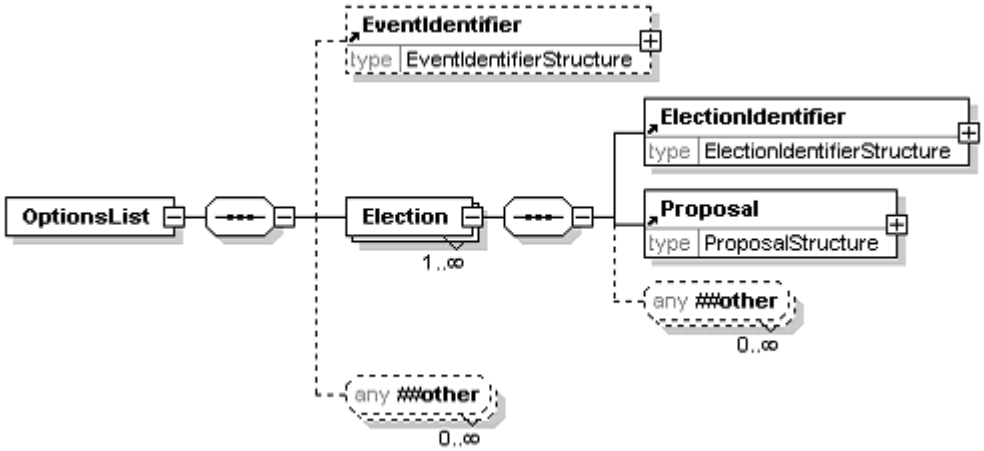
8.27.1 Description of Schema

This message is sent from the election organiser to the proposer to say whether the nomination has been accepted. Along with the acceptance information and the basic information of election, contest and identifier for the proposal, a remark can be made explaining the decision.

8.27.2 EML Additional Rules

Error Code	Error Description
3620-001	If the nomination has not been accepted, a reason for rejection is required in the `Remark` element

8.28 Options List (630)



8.28.1 Description of Schema

This schema is used for messages transferring lists of proposals for a referendum. It may identify the election event, and provides details about the election. Each proposal in a referendum counts as an election, so each election identified will hold a single proposal.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Charbel Aoun, Accenture
Siobhan Donaghy, OPT2VOTE Ltd
Bruce Elton, Oracle Corporation
Joseph Hall, University of California, Berkeley
Roy Hill, OPT2VOTE Ltd
John Ross, Associate
Paul Spencer, Associate
Johan Terryn, EDS
Bernard Van Acker, IBM
David Webber, Individual
Peter Zelechoski, Associate