EML Schema Descriptions

Version 4.0

OASIS Standard, 1st February 2006

Document identifier:

EML v4.0 Schema Descriptions

Editor:

e-Government Unit, Cabinet Office, UK

Contributors:

John Ross
Paul Spencer
John Borras
Farah Ahmed
Charbel Aoun
Bruce Elton
Jim O'Donnel
Roy Hill
Bernard Van Acker
Hans von Spakovsky

Abstract:

This document contains the descriptions of the schemas used in EML v4.0. 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 is an OASIS Standard.

It is updated periodically on no particular schedule. Committee members should send comments on this specification to the election@lists.oasis-open.org list. Others should subscribe to and send comments to the election-services-comment@lists.oasis-open.org. To subscribe, send an email message to election-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.
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 Election and Voter Services TC web page (http://www.oasis-open.org/committees/election/).

Table of Contents

1 Introduction

This document describes the OASIS Election Mark-up Language (EML) version 4.0 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 Background

The following is the Executive Summary of the 'EML Process & 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 election systems to interact with each other. 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'.
- 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 the Election Mark-up 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
- Voter Registration information, including eligible voter lists
- Various communications between voters and election officials, such polling information, election notices, etc.
- Logical 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

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.

1.2 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 [4].

1.3 Schema Diagrams in this Document

The schema diagrams in this document were created using XML Spy 2004. 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*.

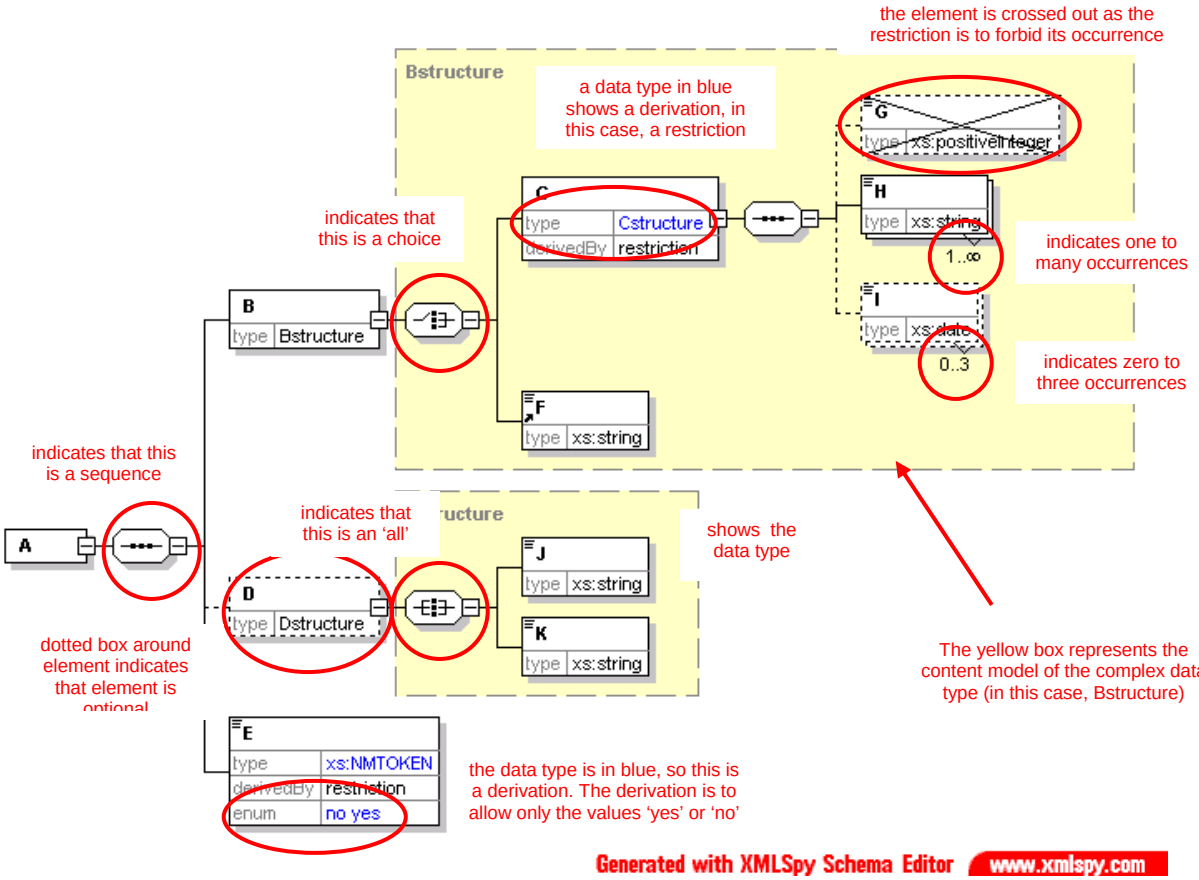
Note that the diagrams in this document do not use the default diagramming options of XML Spy, but have additional information. The additional information to be shown can be set using the menu selections Schema Design | View Config.

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>
```

1.4 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.

1.5 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`.

1.6 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.

1.7 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. 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.

1.8 Conventions

Within this specification, the following conventions are used throughout:

Diagrams are shown as generated by XML Spy 2004 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 Processing using Schematron

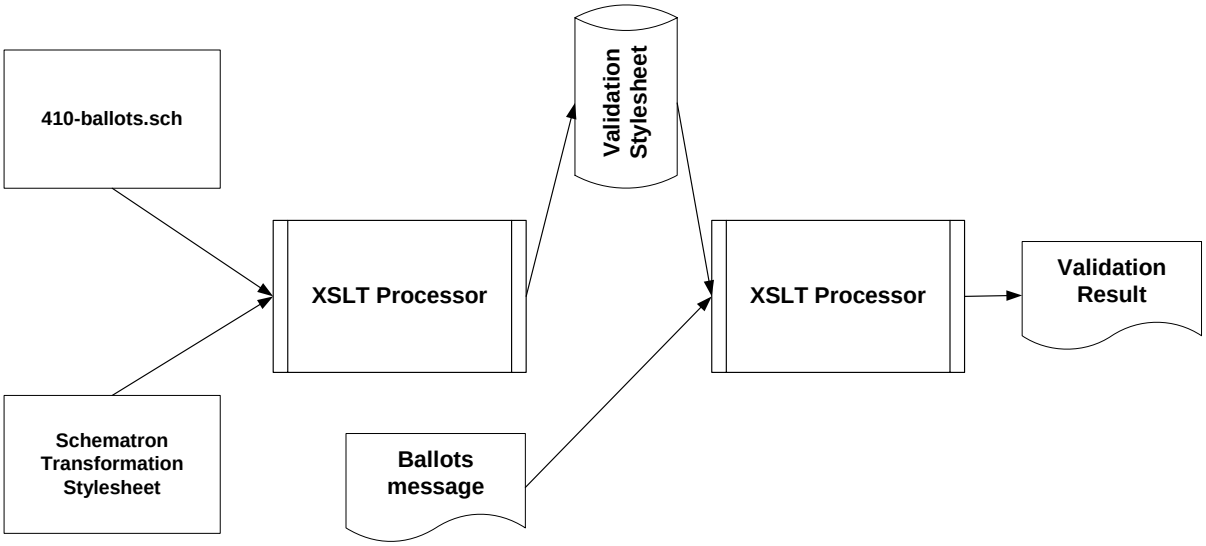
This section gives a short introduction to how validation can be achieved using Schematron schemas and an XSLT processor. Alternatively, direct validation using the Schematron schemas can be achieved using a dedicated Schematron processor.

2.1 Validation using the 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.



3 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 message. This will contain all the background information with a number of the repeated element types. 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".

4 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 localization applicable to all message types could therefore be 4000-001. One specific to the localization of schema 110 could be 4110-002.

4.1 All Schemas

4.1.1 XML well-formedness or Schema validation error

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

4.1.2 Seal Errors

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

4.1.3 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.

391

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

5 EML Core Components

The 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	CandidateIdentifierStructure	ShortCodeType
Candidate	CandidateStructure	TelephoneNumberType
CandidateIdentifier	ComplexDateRangeStructure	VotingChannelType
ContactDetails	ContactDetailsStructure	VotingMethodType
ContestIdentifier	ContestIdentifierStructure	VotingValueType
CountingAlgorithm	DocumentIdentifierStructure	YesNoType
DocumentIdentifier	ElectionGroupStructure	
ElectionIdentifier	ElectionIdentifierStructure	
ElectionStatement	EmailStructure	
EventIdentifier	EMLStructure	
EventQualifier	EventIdentifierStructure	
Gender	EventQualifierStructure	
Logo	IncomingGenericCommunicationStructure	
ManagingAuthority	InternalGenericCommunicationStructure	
MaxVotes	LogoStructure	
MessageType	ManagingAuthorityStructure	
MinVotes	MessagesStructure	
NominatingOfficer	NominatingOfficerStructure	
NumberInSequence	OutgoingGenericCommunicationStructure	
NumberOfPositions	PeriodStructure	
Period	PictureDataStructure	
PersonName	PollingDistrictStructure	
PollingDistrict	PollingPlaceStructure	

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

5.1 Simple Data Types

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

5.1.1 ConfirmationReferenceType

`xs:token`.

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

5.1.2 CountingAlgorithmType

`xs:token`

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

5.1.3 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.

5.1.4 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.

5.1.5 ErrorCodeType

`xs:token`

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

5.1.6 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).

5.1.7 LanguageType

`xs:language`

Declaration of the type of language used in the election.

5.1.8 MessageTypeType

`xs:NMTOKEN`

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

5.1.9 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.

5.1.10 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.

5.1.11 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.

5.1.12 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.

5.1.13 VotingMethodType

`xs:token` with restrictions.

Restrictions: `xs:enumeration`: AMS, FPP, OPV, SPV, STV, approval, block, partylist, 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
- `OPV` - Optional Preferential Voting
- SPV - Single Preferential Vote
- STV - Single Transferable Vote

5.1.14 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.

5.1.15 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.

5.2 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.

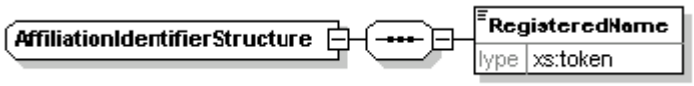
When defining a complete mark-up language, limiting the use of elements and types can restrict further development of the language. For that reason, both data types and elements are defined in EML. Only where an element is an example of a primitive or derived data type defined in XML Schema part 2 is 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.

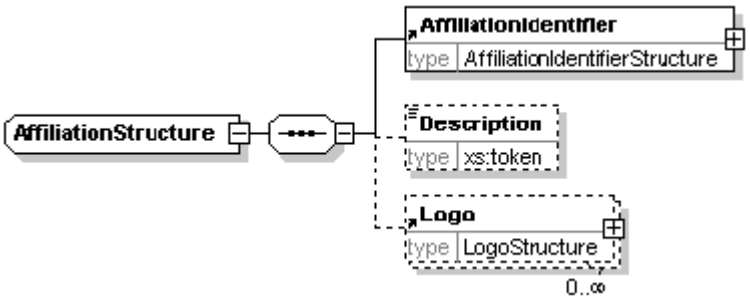
5.2.1 AffiliationIdentifierStructure



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

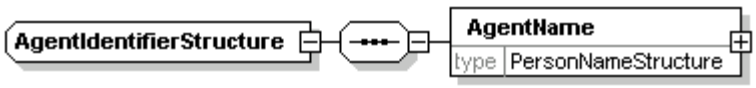
This data type is used to identify an affiliation, such as a political party. The identifier indicates the official name and ID of the organization. It supports use of a short code for voting systems such as SMS, and an expected confirmation reference for security systems that require this.

5.2.2 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 rôle. For example, one rôle 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.

5.2.3 AgentIdentifierStructure

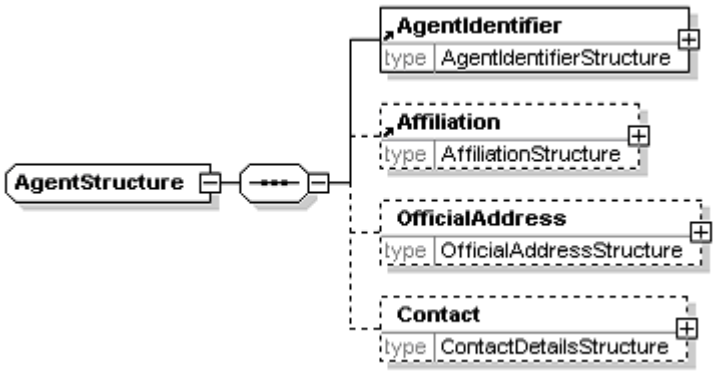


Element	Attribute	Type	Use	Comment
AgentIdenttifierStructure	Id	xs:NMTOKEN	optional	

	DisplayOrder	xs:positiveInteger	optional	

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

514 ### 5.2.4 AgentStructure



515

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

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

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

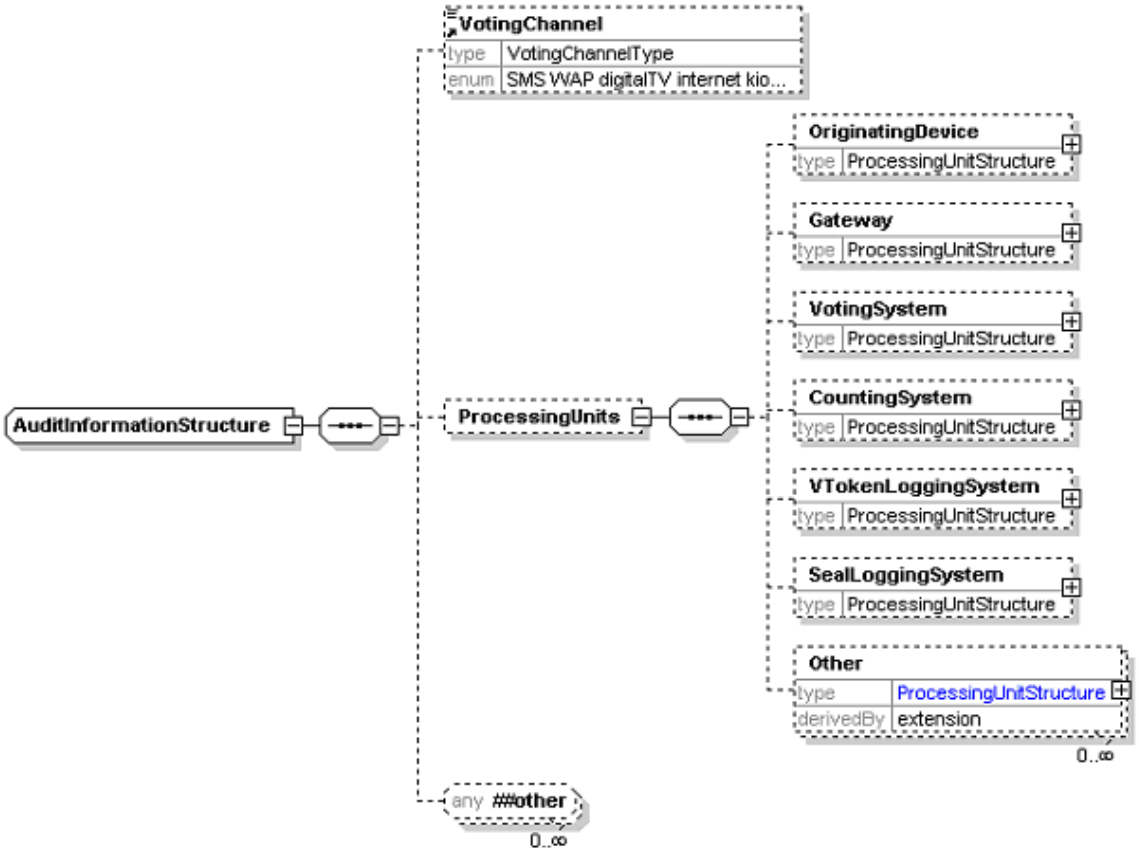
521 ### 5.2.5 AreaStructure

522 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	

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

525 ## 5.2.6 AuditInformationStructure



526

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

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

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

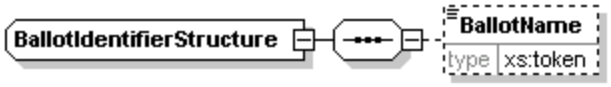
533 ## 5.2.7 AuthorityIdentifierStructure

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

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

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

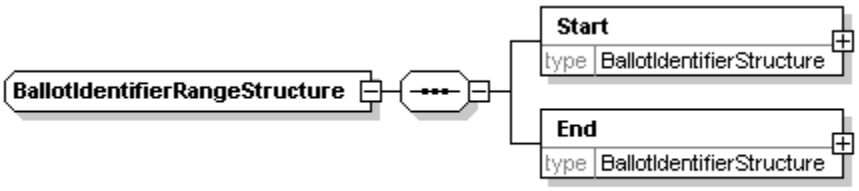
5.2.8 BallotIdentifierStructure



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.

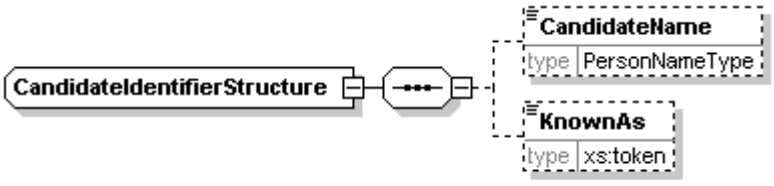
5.2.9 BallotIdentifierRangeStructure



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.

5.2.10 CandidateIdentifierRangeStructure

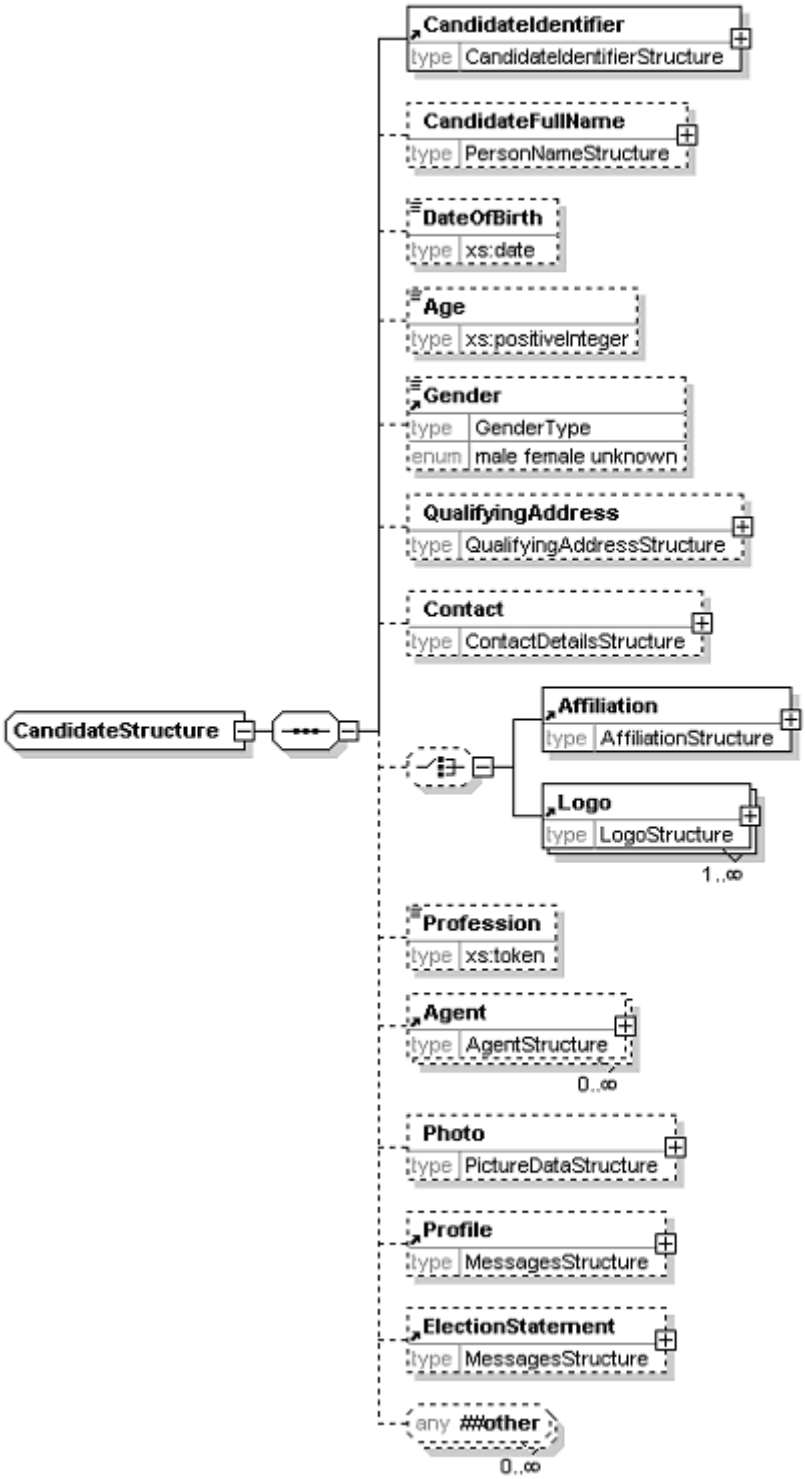


Element	Attribute	Type	Use	Comment
CandidateIdentifierStructure	Id	xs:NMTOKEN	required	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

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

568

5.2.11 CandidateStructure



569

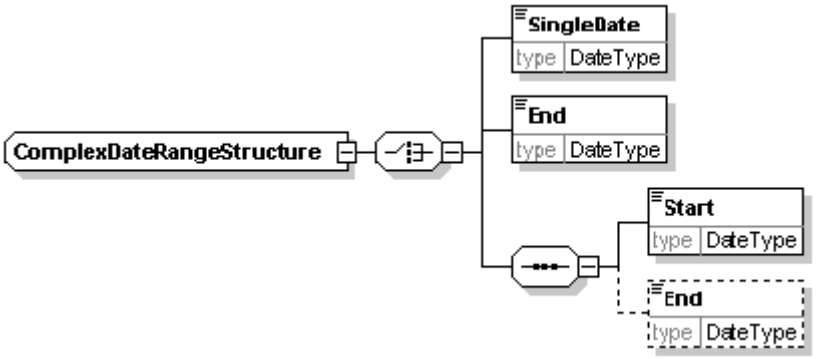
Element	Attribute	Type	Use	Comment
CandidateStructure	Independent	YesNoType	optional	
	DisplayOrder	xs:positiveInteger	optional	

570 The candidate description includes all the information required about the candidate. In different
571 messages, the amount of information is reduced, either by restricting the information in EML or as
572 part of a localization.

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

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

579 ### 5.2.12 ComplexDateRangeStructure



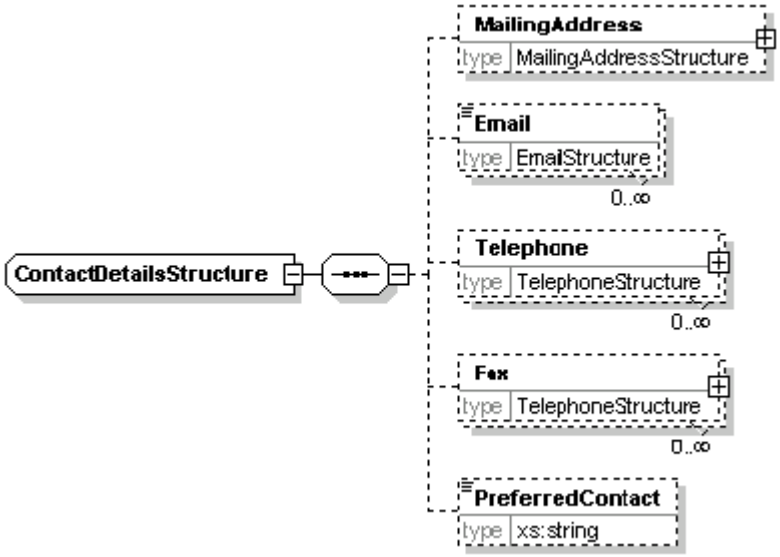
580

Element	Attribute	Type	Use	Comment
ComplexDateRangeStructure	Type	xs:token	required	

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

583 The Type attribute is used to indicate the purpose of the date (e.g. "deadline for nominations"). It
584 is likely that this will be removed before release of EML version 4 and applied to elements instead
585 as an extension of this data type.

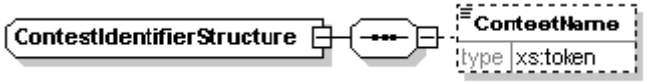
5.2.13 ContactDetailsStructure



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.

5.2.14 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.

5.2.15 DocumentIdentifierStructure

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

Element	Attribute	Type	Use	Comment
DocumentIdentifierStructure	Href	xs:anyURI	required	

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

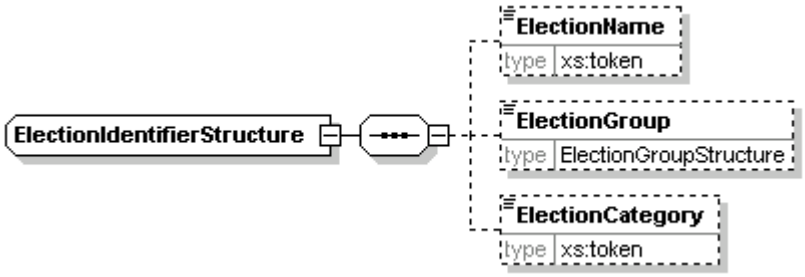
603 5.2.16 ElectionGroupStructure

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

Element	Attribute	Type	Use	Comment
DocumentIdentifierStructure	Id	xs:token	required	

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

609 5.2.17 ElectionIdentifierStructure



610

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

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

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

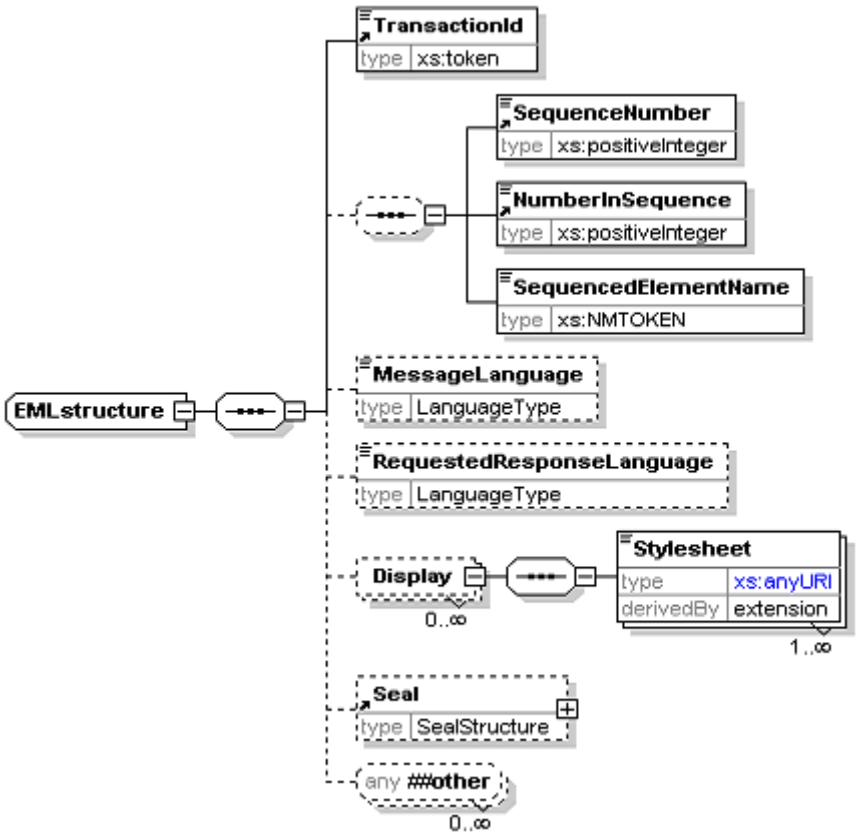
5.2.18 EmailStructure

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.

5.2.19 EMLstructure



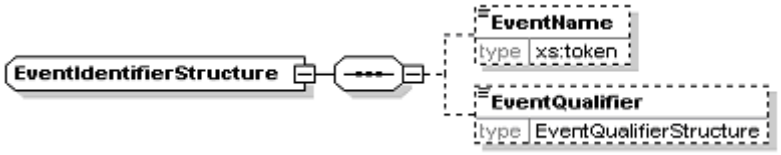
Element	Attribute	Type	Use	Comment
EMLstructure	Id	MessageTypeType	required	
	SchemaVersion	xs:NMTOKEN	requried	
	ShortCode	ShortCodeType	optional	
Stylesheet	Type	xs:token	required	

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 [2], for example, text/xsl. The final element defined is the seal, which is used to seal the complete message.

5.2.20 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.

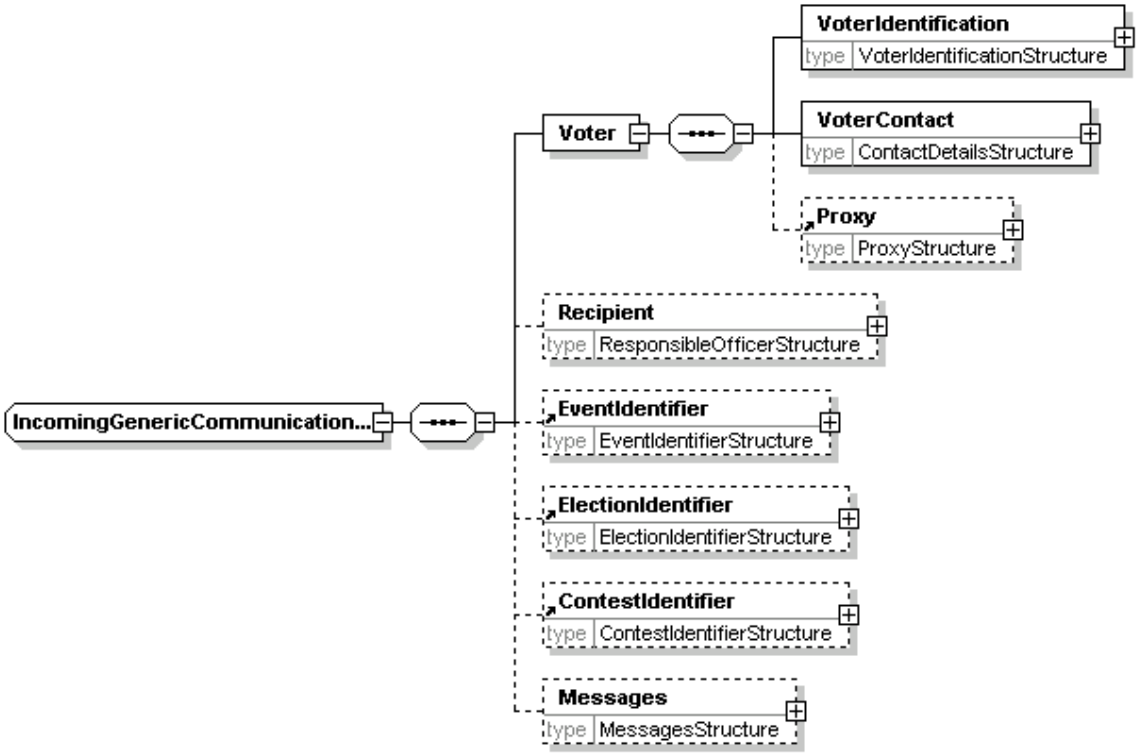
5.2.21 EventQualifierStructure

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

Element	Attribute	Type	Use	Comment
EventQualifierStructure	Id	xs:NMTOKEN	optional	

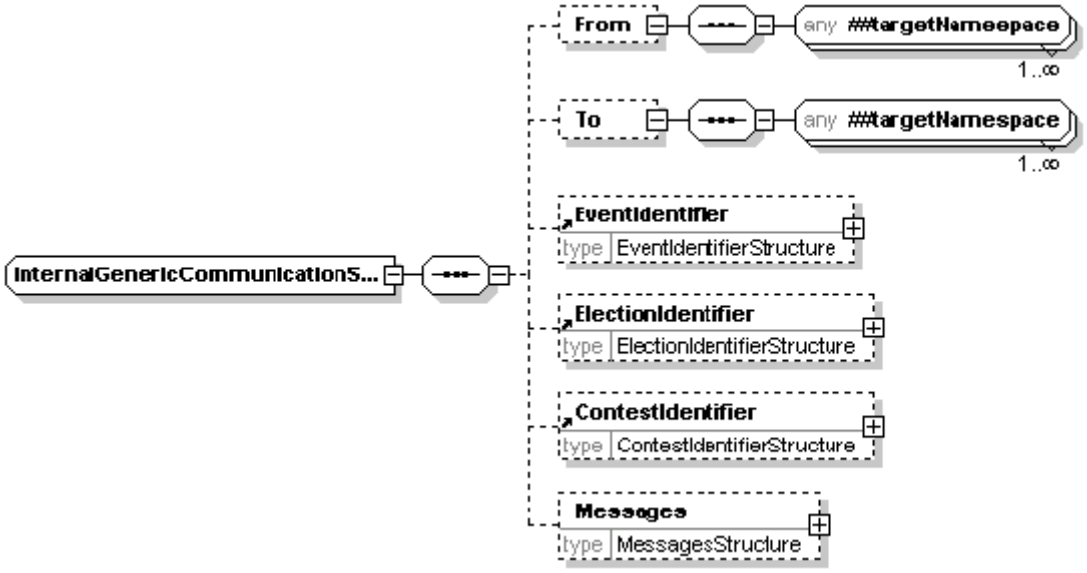
The event qualifier is used to further identify the event. For example, there might be "County Elections" covering an entire country, but the events are organized at a county level, so the event qualifier would identify the county.

5.2.22 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.

5.2.23 InternalGenericCommunicationStructure



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

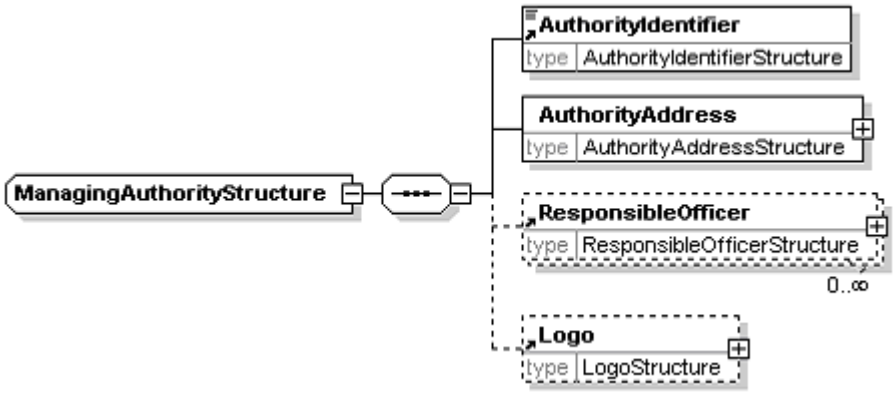
5.2.24 LogoStructure

The `LogoStructure` is an extension of the `PictureDataStructure` to add one attribute:

Element	Attribute	Type	Use	Comment
LogoStructure	Id	xs:NMTOKEN	optional	Standard attribute for a PictureDataStructure
	DisplayOrder	xs:positiveInteger	optional	Standard attribute for a PictureDataStructure
	Role	xs:token	optional	Additional attribute for this element

This element extends the picture data structure by adding an attribute to define the rôle of the logo. This can be used to indicate the purpose of the logo (for example, it is to appear on a ballot).

663 ## 5.2.25 ManagingAuthorityStructure



664

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

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

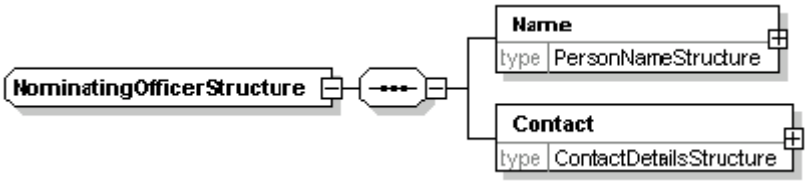
674 ## 5.2.26 MessageStructure

675

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

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

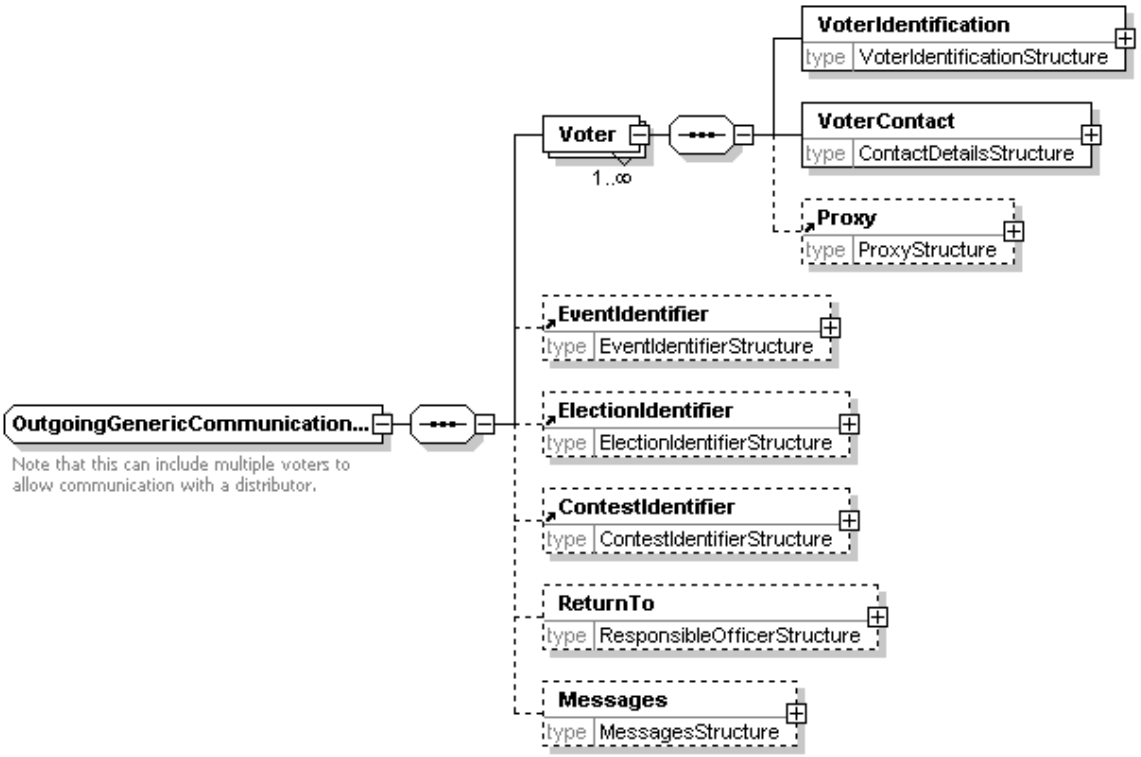
682 ## 5.2.27 NominatingOfficerStructure



683

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

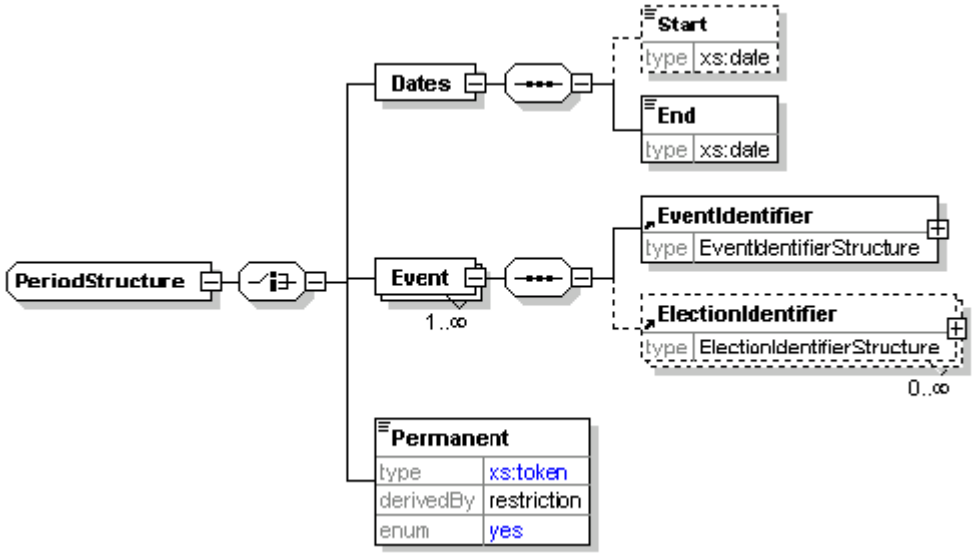
686 ## 5.2.28 OutgoingGenericCommunicationStructure



687

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

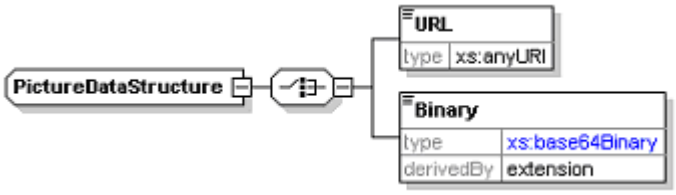
692 ## 5.2.29 PeriodStructure



693

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

697 ## 5.2.30 PictureDataStructure

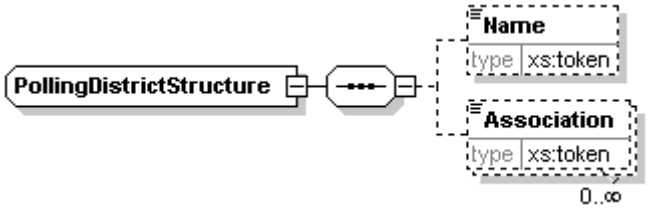


698

Element	Attribute	Type	Use	Comment
PictureDataStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
Binary	Format	xs:NMTOKEN (restricted)	required	

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

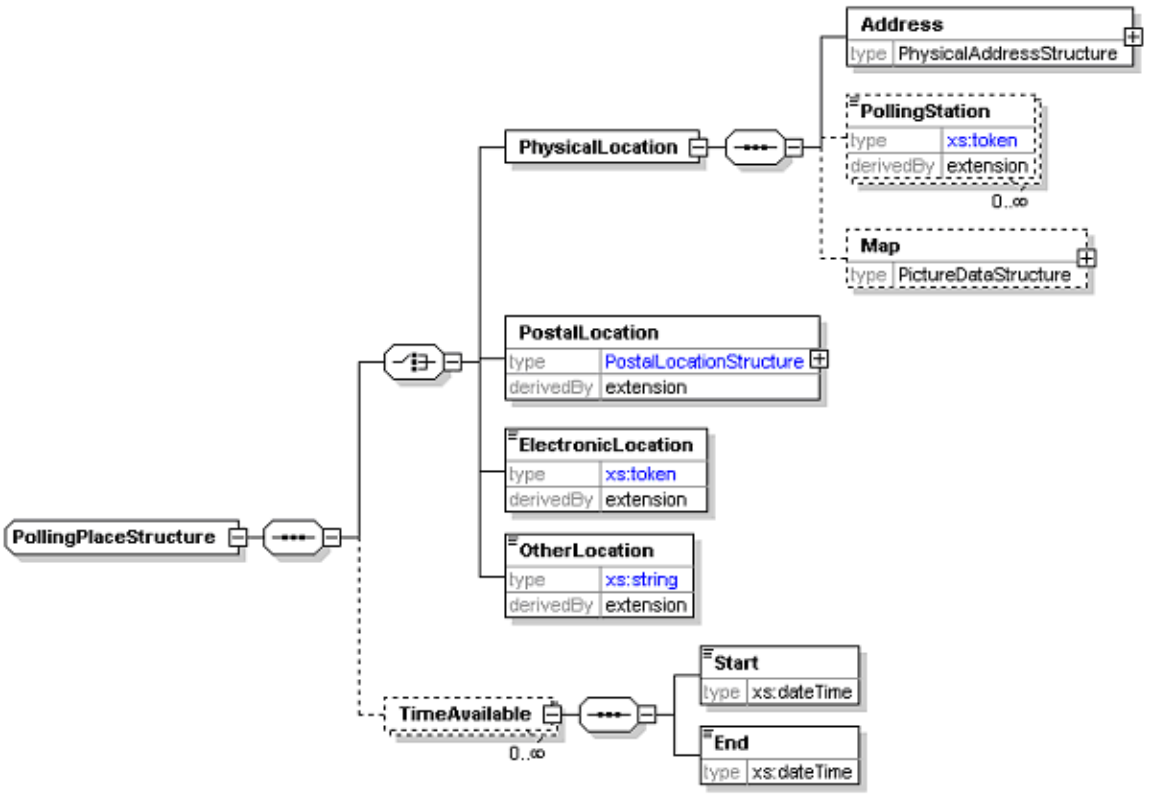
5.2.31 PollingDistrictStructure



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

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

5.2.32 PollingPlaceStructure



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.

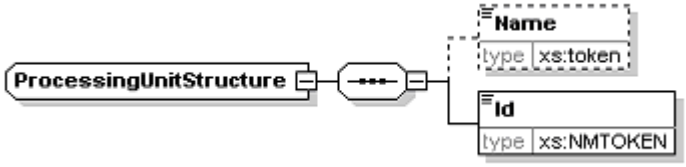
5.2.33 PositionStructure

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.

5.2.34 ProcessingUnitStructure

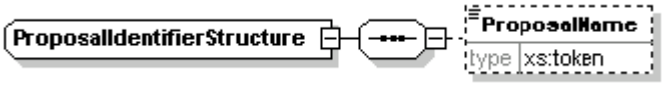


Element	Attribute	Type	Use	Comment
ProcessingUnitStructure	Role	xs:token (restricted)	required	

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) and optional name.

Each processing unit has an attribute to describe its rôle. The rôle 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.

5.2.35 ProposalIdentifierStructure

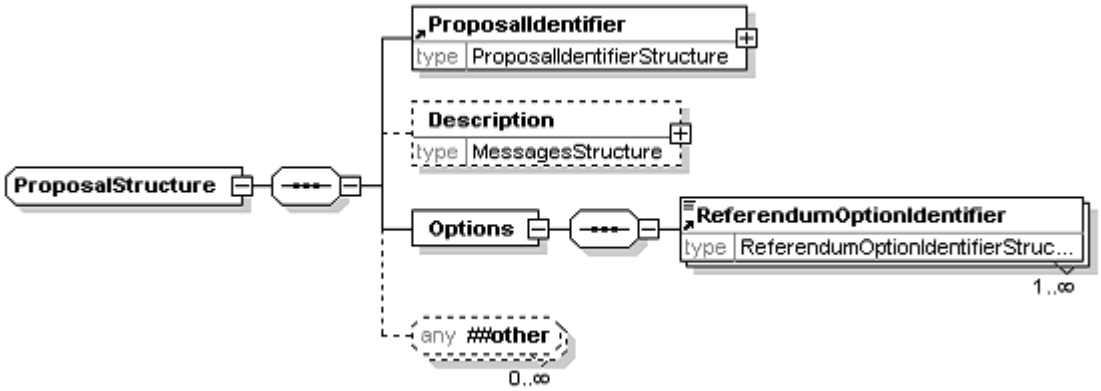


Element	Attribute	Type	Use	Comment
ProposalIdentifierStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
	ShortCode	ShortCodeType	optional	
	ExpectedConfirmationReference	ConfirmationReferenceType	optional	

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

The proposal identifier indicates a system ID for the proposal. 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.

5.2.36 ProposalStructure

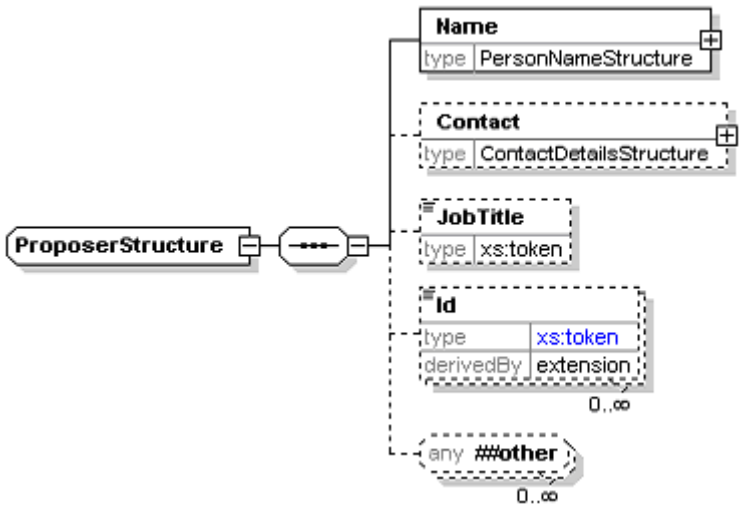


Element	Attribute	Type	Use	Comment
ProposalStructure	Type	xs:token	optional	

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

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

750 5.2.37 ProposerStructure



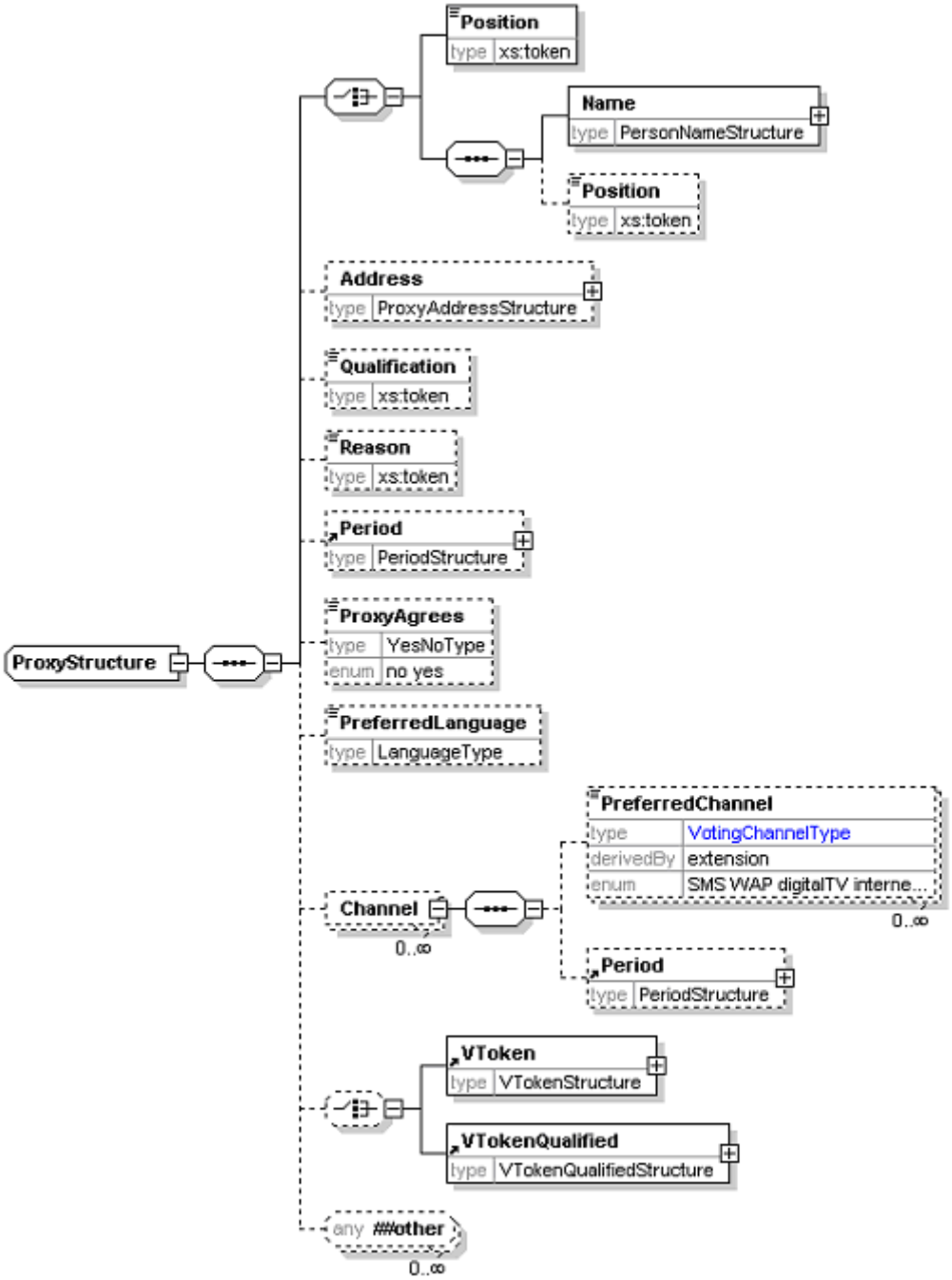
751

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

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

755

5.2.38 ProxyStructure



756

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

757 In many elections, a voter may appoint a proxy to vote on his or her behalf. That proxy may be
758 identified by position (for example, appointing the chairman as proxy at a company AGM), or by
759 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.

5.2.39 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.

5.2.40 ReportingUnitIdentifierStructure

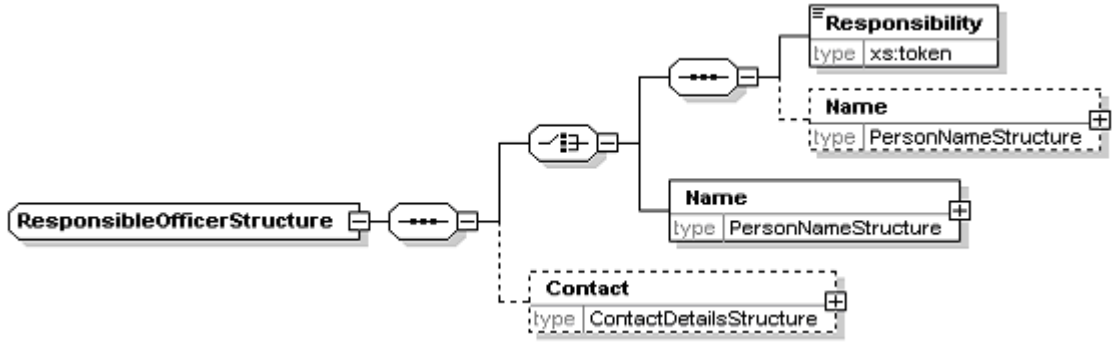
The `ReportingUnitIdentifierStructure` is an extension of `xs:token` to add the following attributes:

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`.

5.2.41 ResponsibleOfficerStructure



Element	Attribute	Type	Use	Comment
ResponsibleOfficerStructure	Id	xs:NMTOKEN	optional	

A responsible officer is someone who has some sort of rôle 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.

5.2.42 ScrutinyRequirementStructure

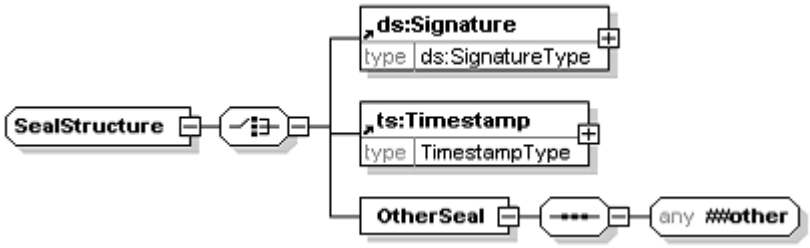
The `ScrutinyRequirementStructure` is an extension of `xs:token` to add the following attribute:

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>
```

5.2.43 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

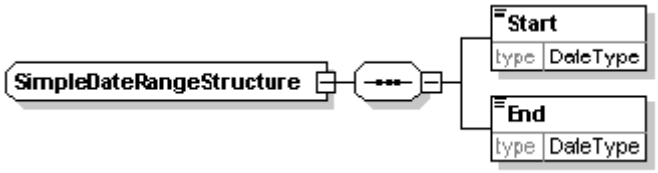
created by the voter. The seal may also be used to authenticate the identity of the system that collected a vote, and provide proof of the time at which the vote was cast.

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.

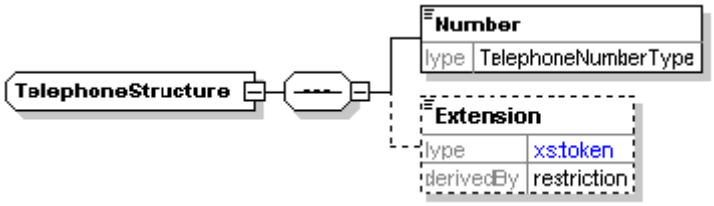
The XML signature created by the voting system provides integrity and authentication of the identity of the system that collected the vote. The time-stamp provides integrity of the vote and proof of the time that the vote was cast.

5.2.44 SimpleDateRangeStructure



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

5.2.45 TelephoneStructure

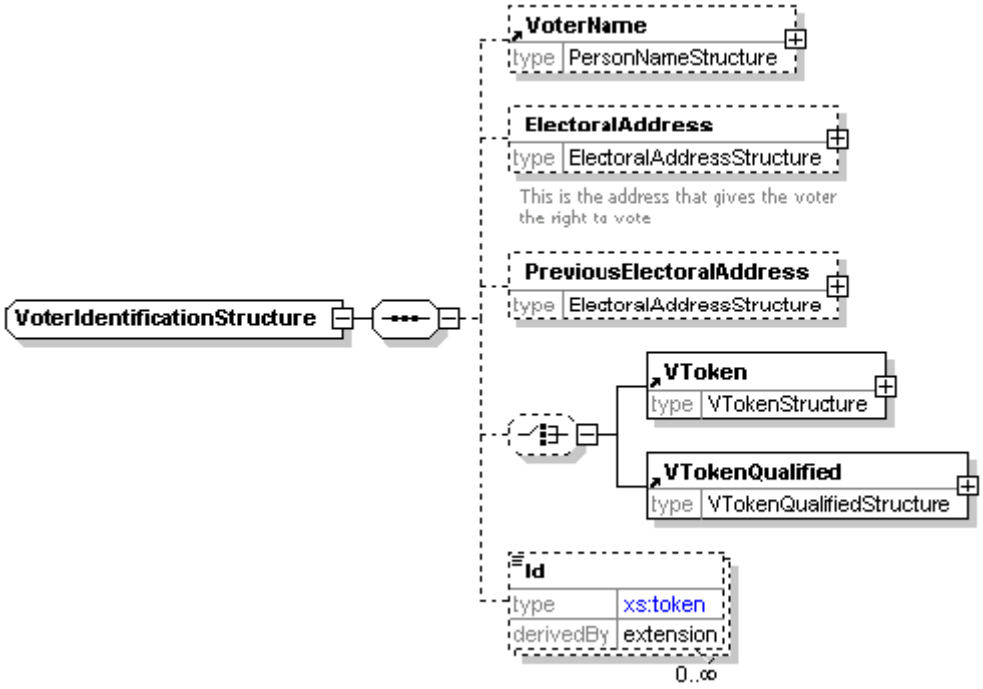


Element	Attribute	Type	Use	Comment
TelephoneStructure	Preferred	YesNoType	optional	
	Mobile	YesNoType	optional	

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.

827

5.2.46 VoterIdentificationStructure



828

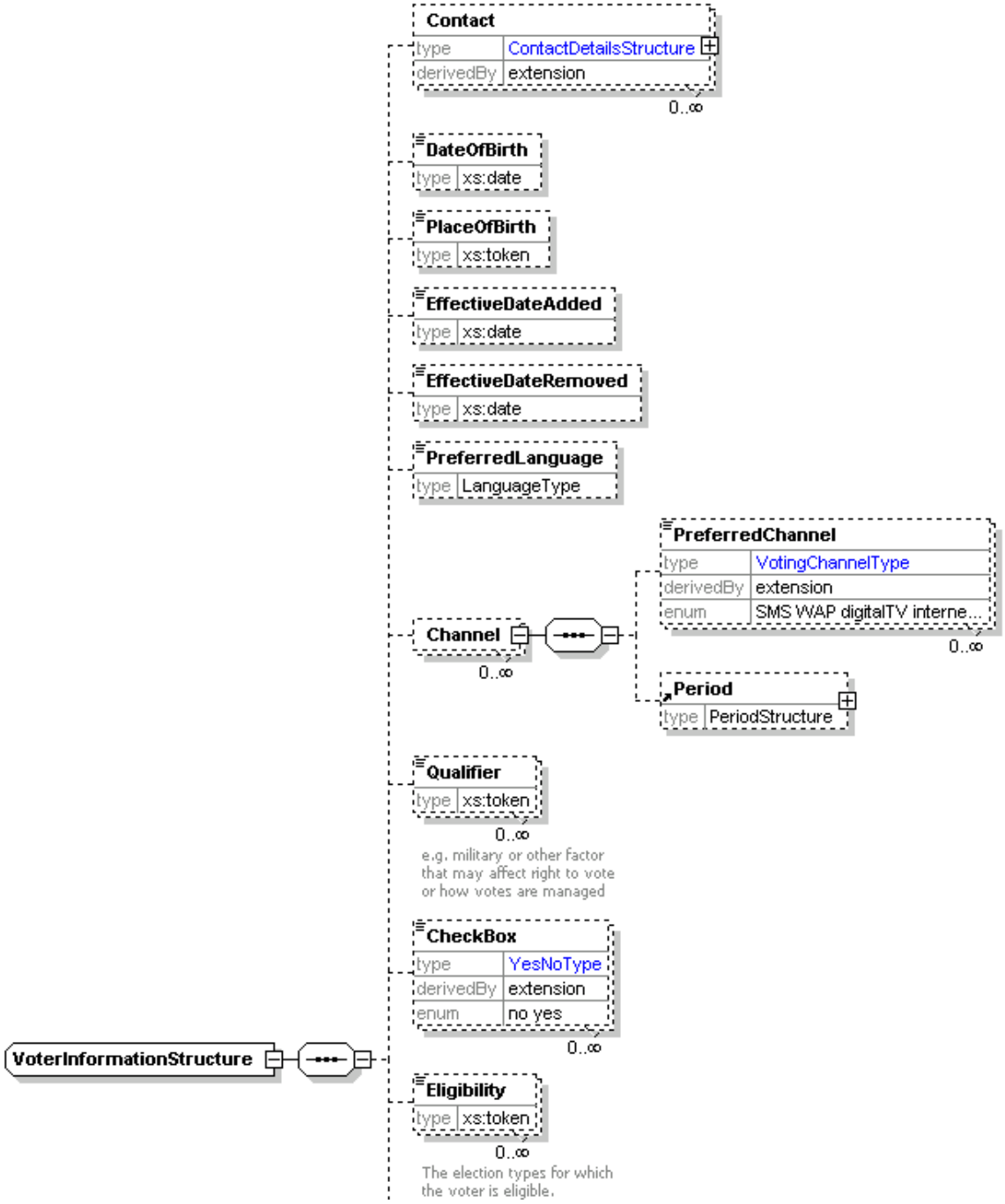
Element	Attribute	Type	Use	Comment
VoterIdentificationStructure	Id	xs:NMTOKEN	optional	
	DisplayOrder	xs:positiveInteger	optional	
Id	Type	xs:token	required	

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

835

836

5.2.47 VoterInformationStructure



837

838 Generated with XMLSpy Schema Editor www.xmlspy.com

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	

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

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

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

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

5.2.48 VTokenStructure

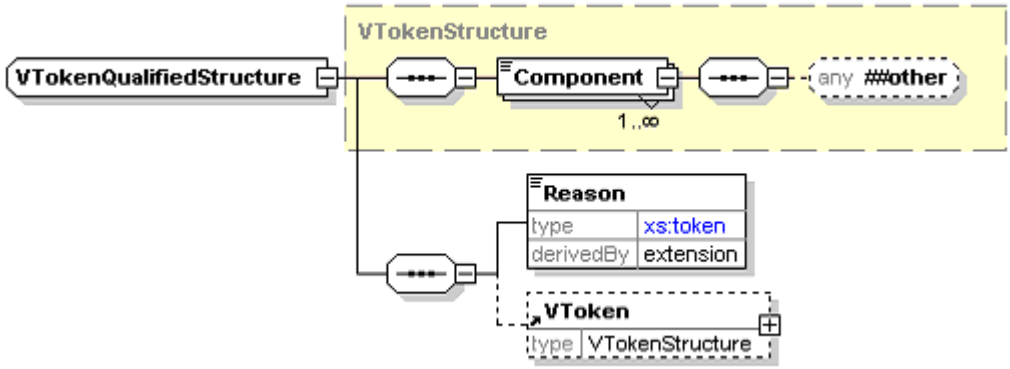
Element	Attribute	Type	Use	Comment
Component	Type	xs:NMTOKEN	required	

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

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

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

5.2.49 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`.

5.3 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, Proposal, ProposalIdentifier, Proposer, Proxy, ReferendumOptionIdentifier, ReportingUnitIdentifier, ResponsibleOfficer, ScrutinyRequirement, Seal, VToken, VTokenQualified`

5.3.1 Accepted

`YesNoType`

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

5.3.2 Election Statement

`MessagesStructure`

This is the candidate's message to voters.

5.3.3 MaxVotes

`xs:positiveInteger`

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

5.3.4 MinVotes

`xs:nonNegativeInteger`

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

5.3.5 NumberInSequence

`xs:positiveInteger`

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

5.3.6 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".

5.3.7 PersonName

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

5.3.8 Profile

`MessagesStructure`

921 This is the candidate's profile statement.

922 5.3.9 SequenceNumber

923 `xs:positiveInteger`

924 The sequence number of a partial message when a message is split. See “Splitting of
925 Messages”.

926 5.3.10 TransactionId

927 `xs:token`

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

929 5.3.11 VoterName

930 `PersonNameStructure`

931 The name of a voter.

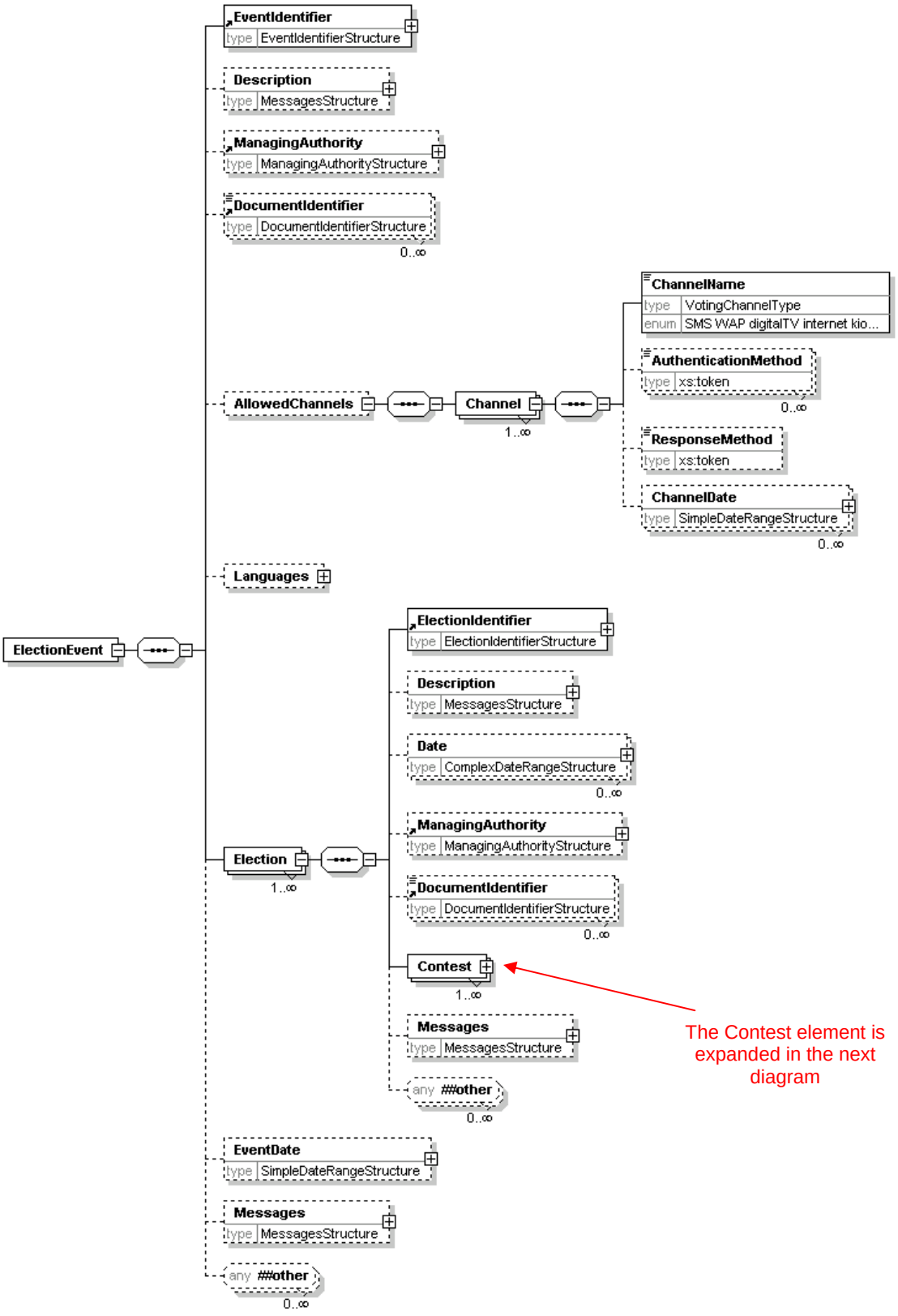
6 The EML Message Schemas

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.

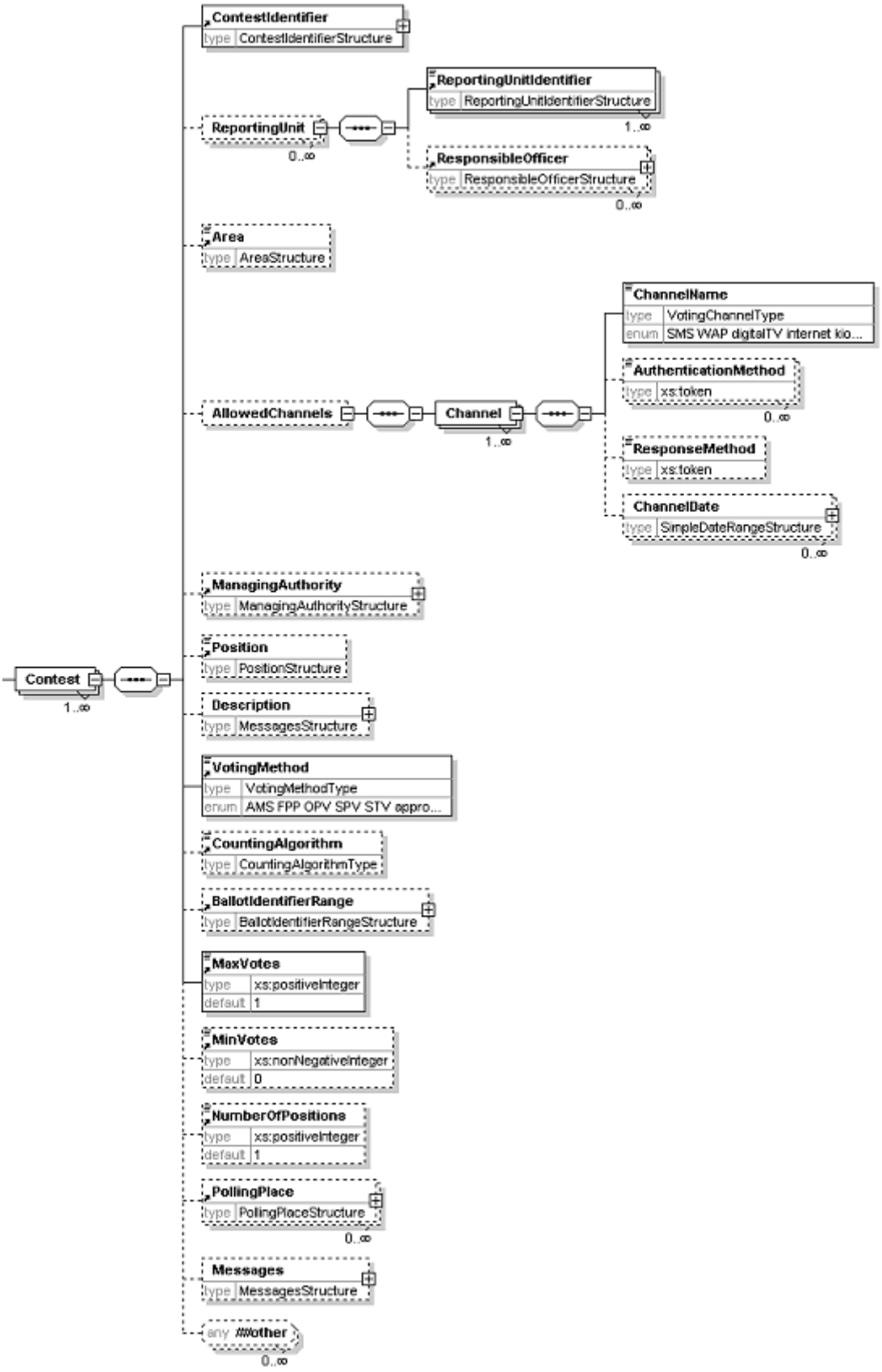
Attributes are shown where they are not the standard attributes of data types already described.

936

6.1 Election Event (110)



937



938

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

6.1.1 Description of Schema

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

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

For the election event, the information includes the ID and name of the event, possibly with a qualifier on 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.

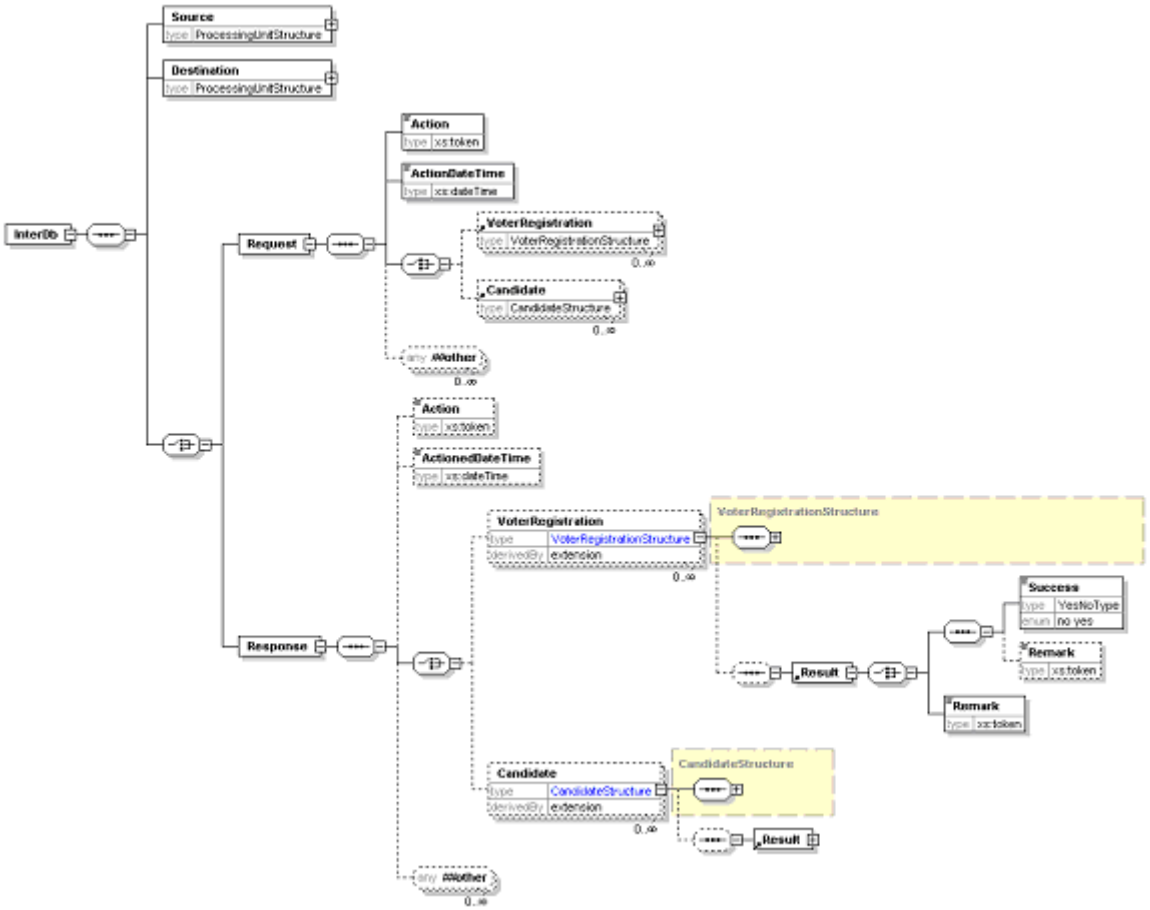
A list of polling places can be provided. These can be either physical locations for people to go to vote, postal addresses for postal votes or electronic locations. An ‘other location’ is also allowed for cases where these do not meet the requirements. A location can also say when it will be available. 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.

6.1.2 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.

6.2 Inter Database (120)



6.2.1 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.

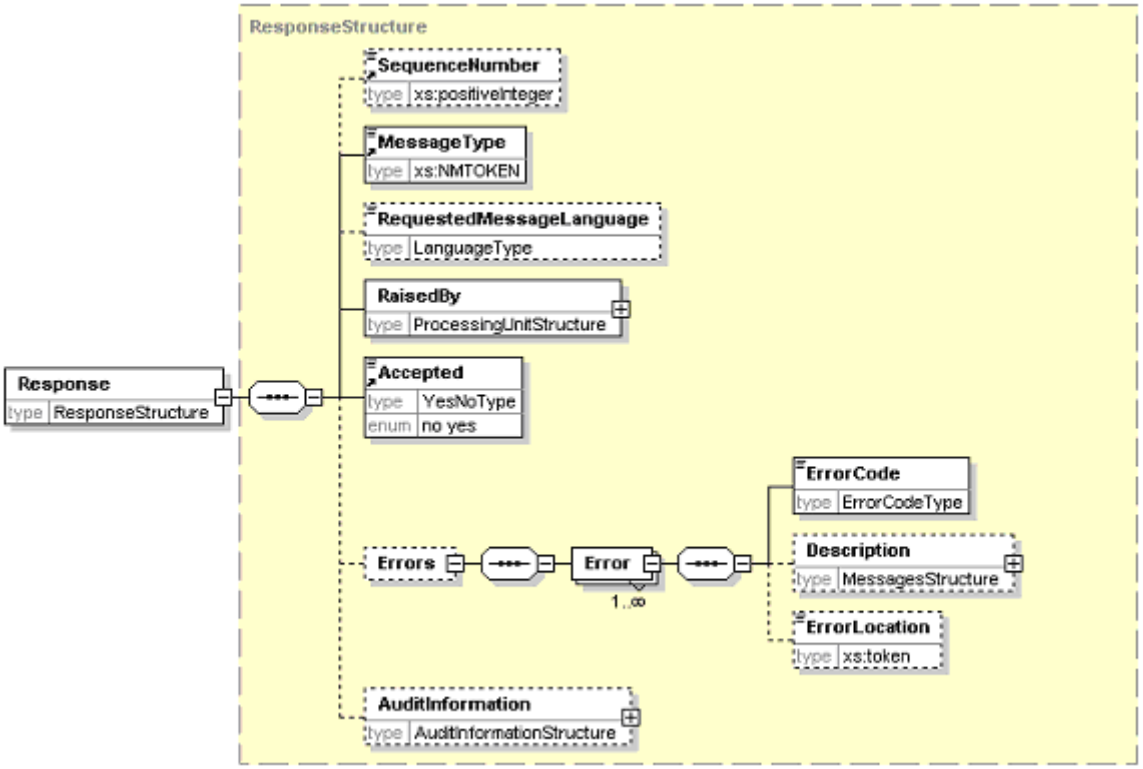
Both request and response start by identifying the source and destination as processing units.

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

995 ActionDateTime is used to specify when the action should be carried out, and then there is an
996 optional list of voters or candidates.

997 A response has a similar structure. It could be that the Action code is no longer required, so this
998 is now optional. The TransactionID must match that given in the request. The Result is either
999 a binary Success flag or a remark or both. Again, there is a date and time, but in this case it is
1000 the date and time at which the action took place.

1001 ## 6.3 Response (130)



1002

1003 ### 6.3.1 Description of Schema

1004 Some messages have a defined response message that provides useful information. However,
1005 there is a need for a more general response, either to indicate that a message has been
1006 accepted, or to indicate the reasons for rejection.

1007 The message includes information to identify the message to which the response applies (by
1008 using the same transaction id in the EML element and, if necessary, including the sequence
1009 number of the message to which the response applies in the Response element), with
1010 information on the entity raising the message, whether the message was accepted and
1011 information about the errors if it was not. The desired language for a display message can also be
1012 included to allow a downstream processor to substitute a language-specific error message if
1013 required.

1014 If the message is reporting an error, the location of the error within the message can be indicated.
1015 Usually, this will be an XPath to the location of the error. However, errors detected by an XML
1016 parser may be in a different format, such as a line number.

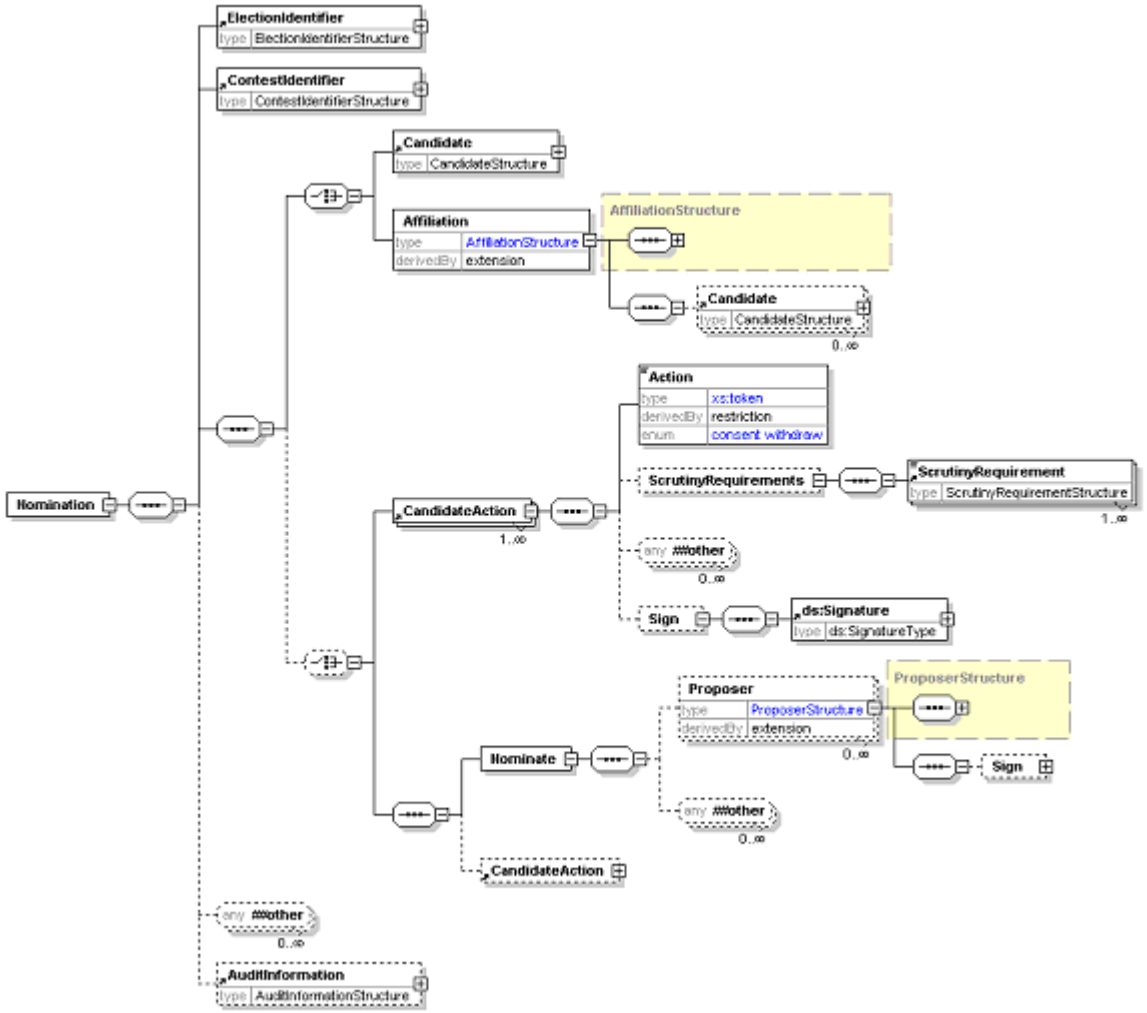
1017 Note that a single response can be raised for a series of sub-messages with the same transaction
1018 ID. This allows indication, for example, that a sub-message was missing.

1019

6.3.2 Additional EML Rules

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

6.4 Candidate Nomination (210)



6.4.1 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.

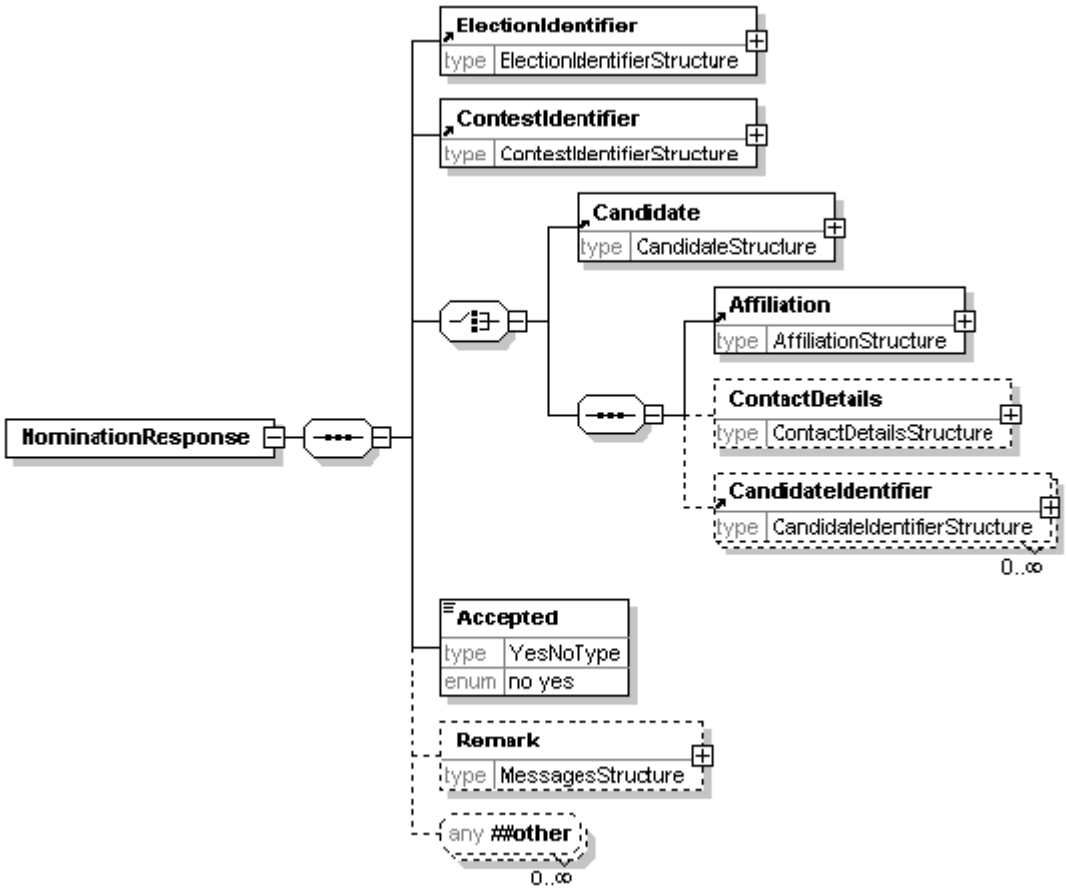
The election and contest must be specified. When a candidate is being nominated, there must be information about the candidate and one or more proposers. The candidate must supply a name. Optionally, the candidate can provide contact information, an affiliation (e.g. a political party) and textual profiles and election statements. These two items use the `MessagesStructure` to allow text in multiple languages. There is also scope to add additional information defined by the election organiser.

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

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

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

6.5 Response to Nomination (220)



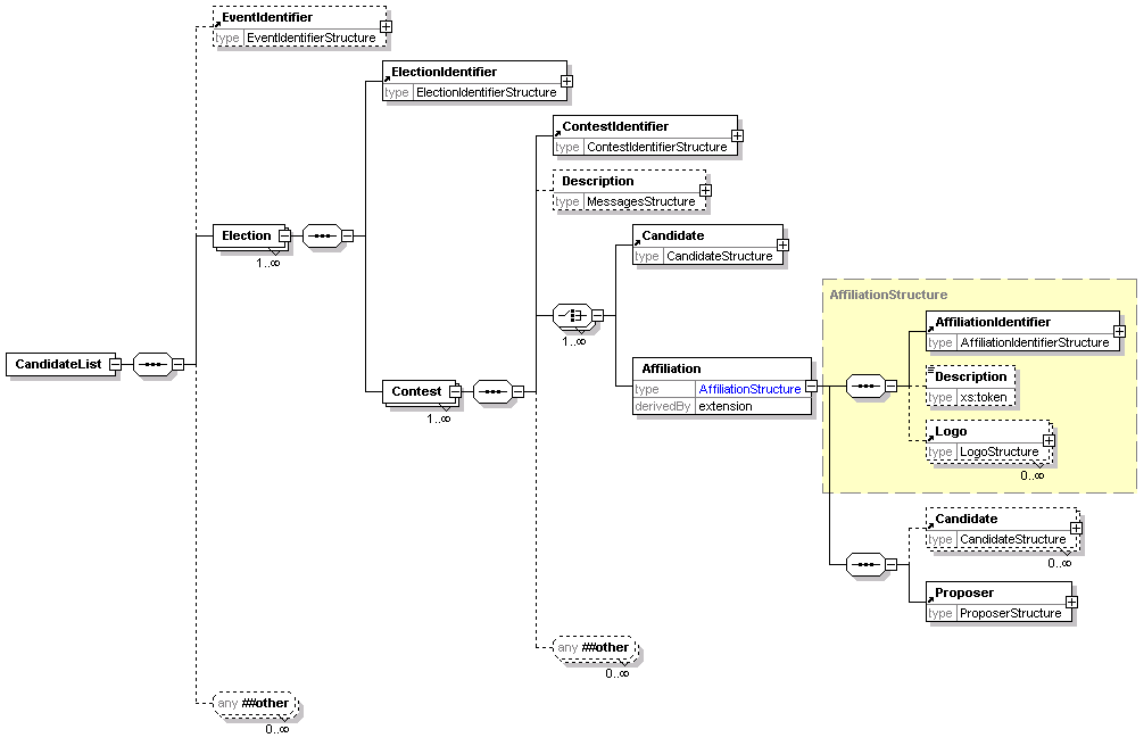
6.5.1 Description of Schema

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

6.5.2 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

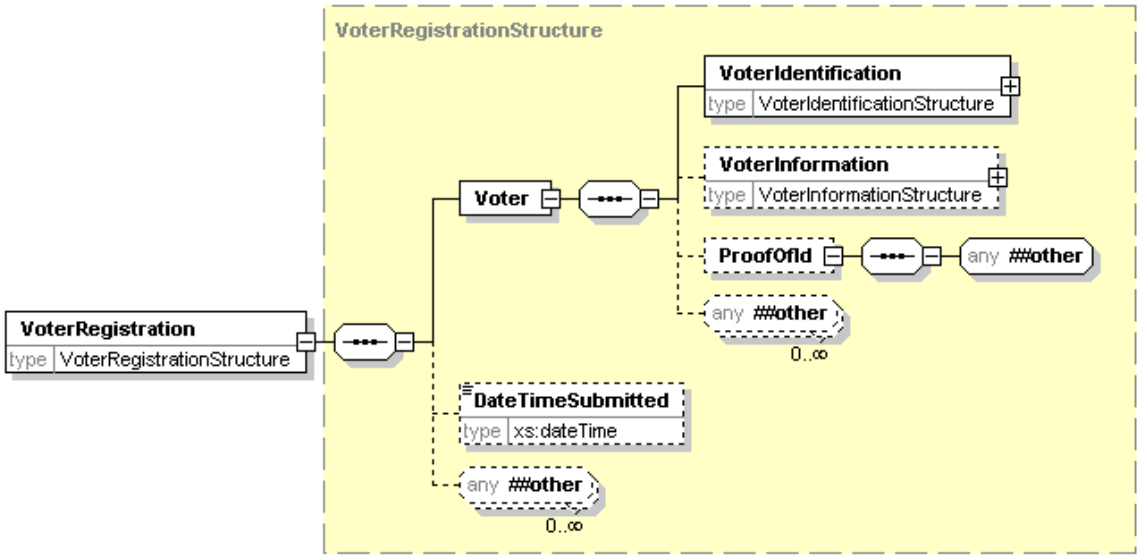
6.6 Candidate List (230)



6.6.1 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.

6.7 Voter Registration (310)



6.7.1 Description of Schema

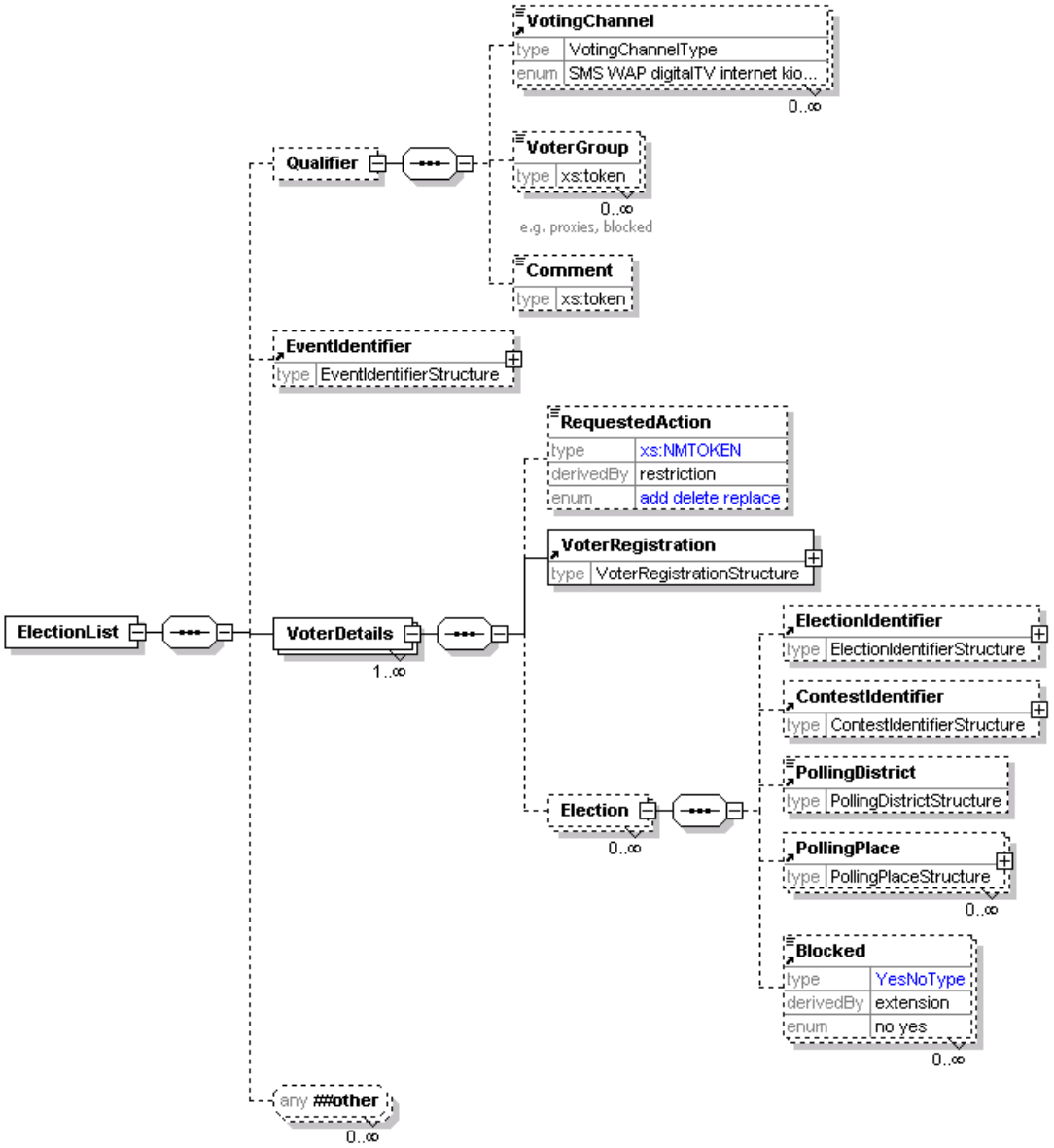
This schema is used for messages registering voters. It uses the `VoterIdentificationStructure`. The `VoterInformationStructure` is used unchanged. Proof of ID can be provided.

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

6.7.2 EML Additional Rules

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

6.8 Election List (330)



Element	Attribute	Type	Use	Comment
Blocked	Reason	xs:token	optional	
	Channel	VotingChannelType	optional	

6.8.1 Description of Schema

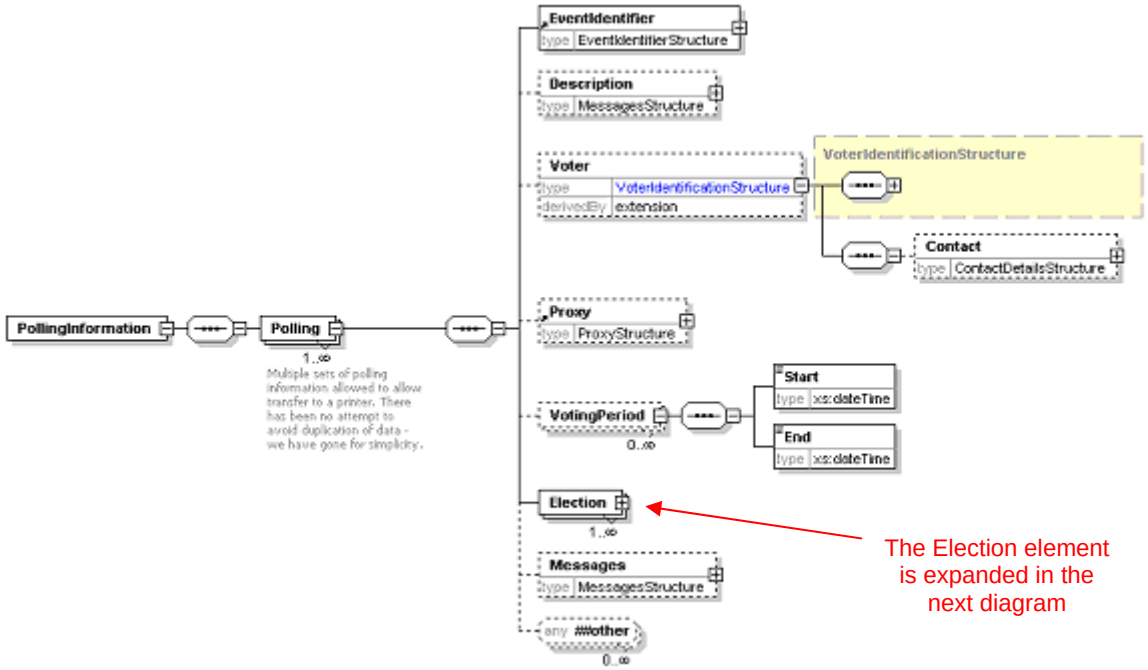
This schema is primarily used for messages communicating the list of eligible voters for an election or set of elections. It can also be used for any other purpose that involves the transfer of voter information where the 120-interDB message is not appropriate. Partial lists are allowed through the use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

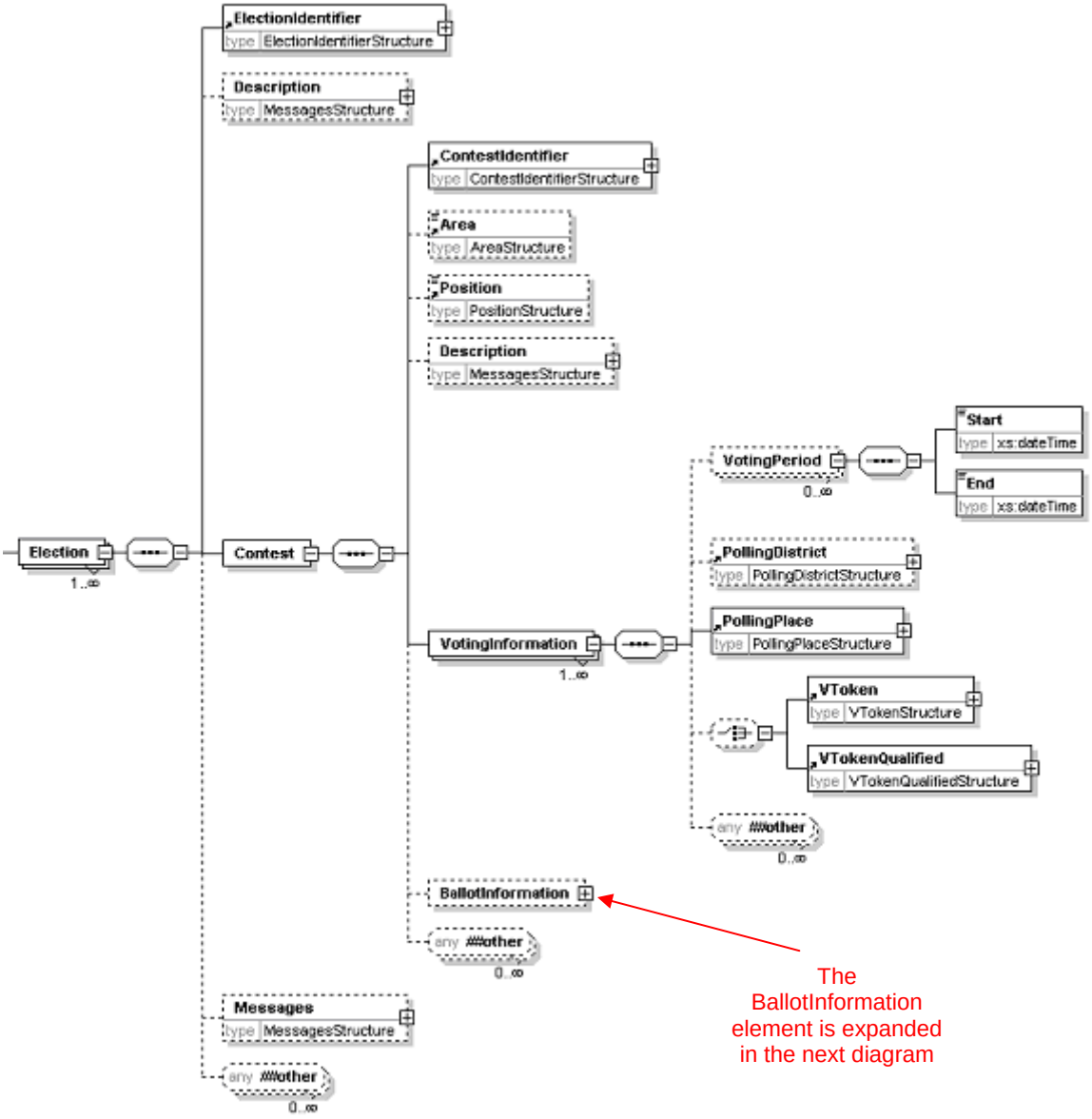
For each voter, information is provided about the voter himself or herself, and optionally about the elections and contests in which the voter can participate. The information about the voter is the same as that defined in the 310-voterregistration schema. Added to this can be a list of elections, each 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.

6.8.2 EML Additional Rules

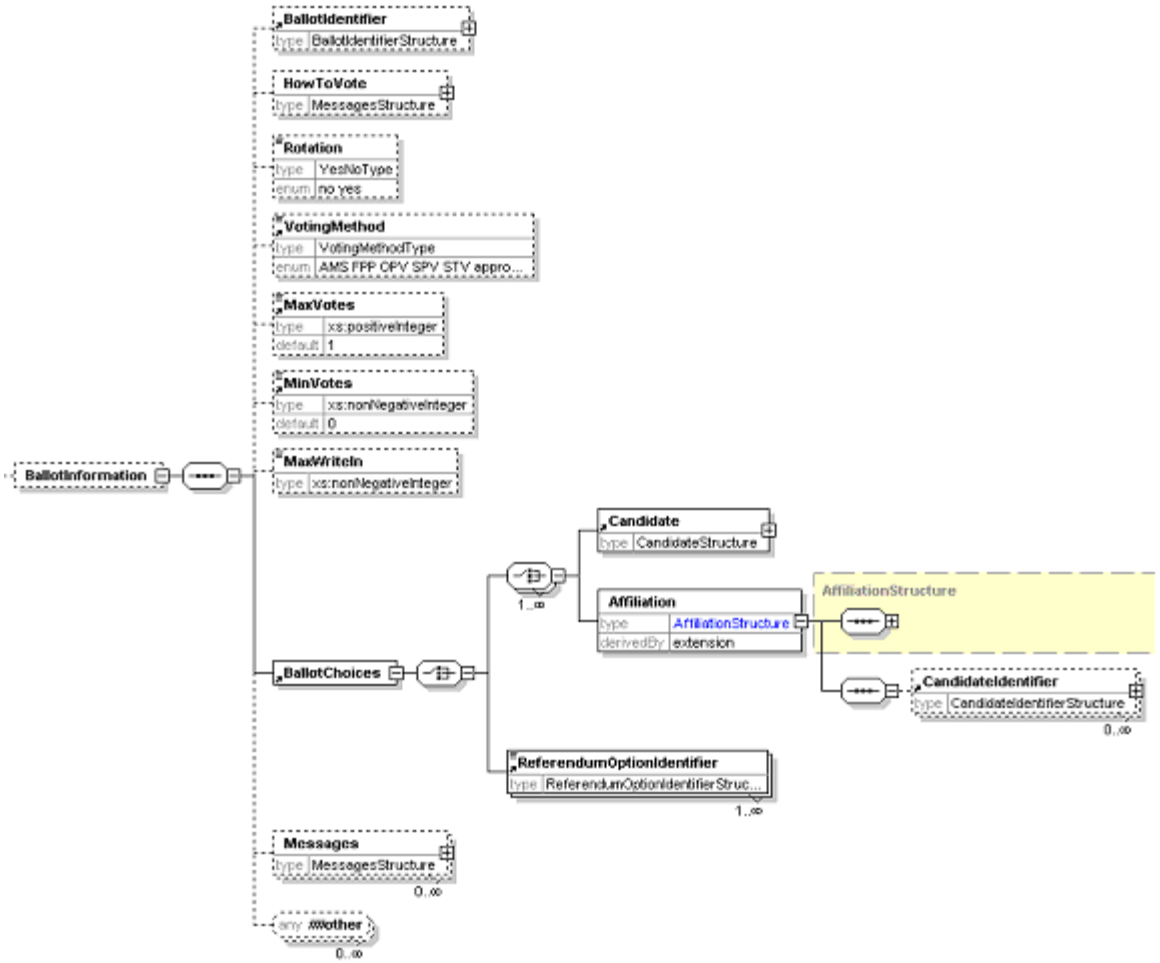
Error Code	Error Description
3330-002	The polling district can only be included for either the voter or the election.
3330-003	The polling place can only be included for either the voter or the election.

6.9 Polling Information (340)





1094



1095

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

1096 ### 6.9.1 Description of Schema

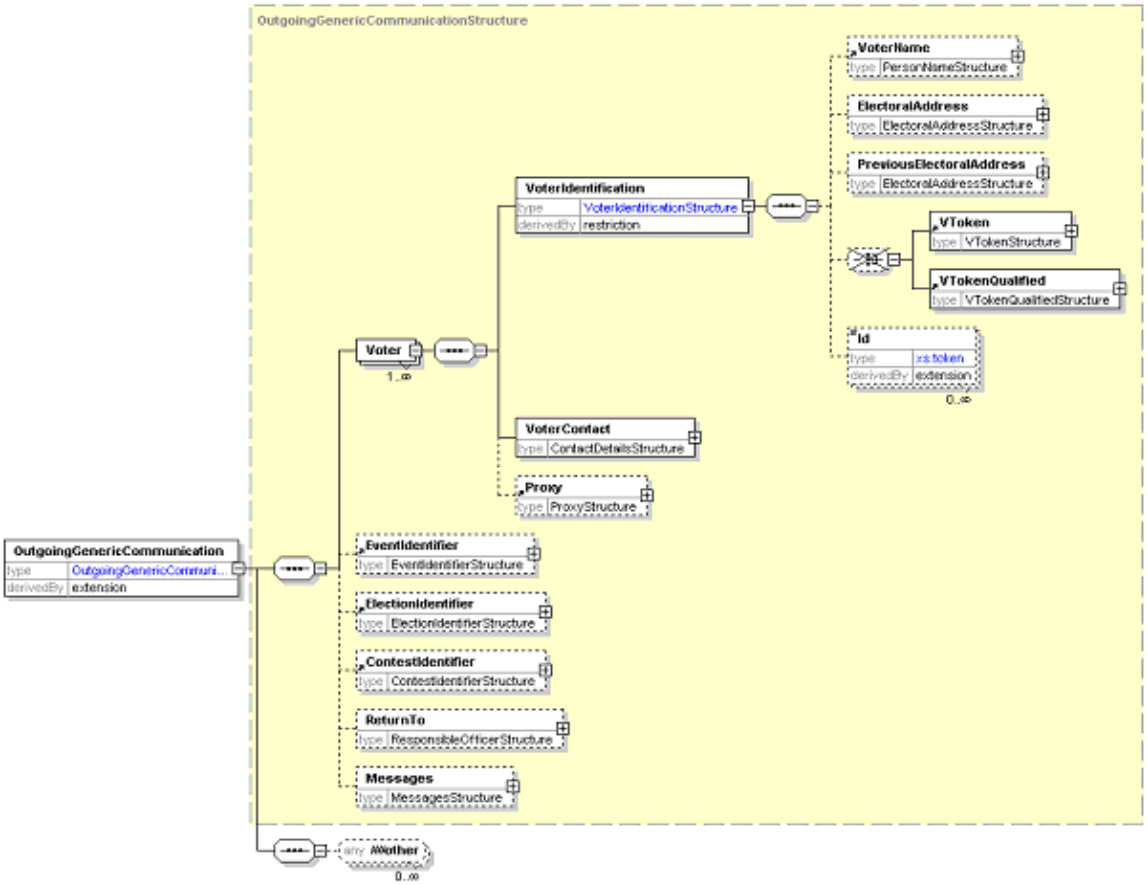
1097 The polling information message defined by this schema is sent to a voter to provide details of
1098 how to vote. It can also be sent to a distributor, so multiple sets of information are allowed. In the
1099 case of SMS voting, ballot information may also be required, so this can be included. Either one
1100 or several sets of polling information may be sent to each voter for any election event.

1101 Some information about the voter and any proxy may be included, for example to print on a
1102 polling card. This can also include a mailing address for a distributor to use.

1103 Information about the elections and contests is included for the benefit of the voter. For each
1104 voting channel, this includes where to vote (which could be a polling station, address for postal
1105 voting, URL for Internet voting, phone number for SMS voting etc) and the times that votes can
1106 be placed. Use of the `DisplayOrder` attribute on these allows the display or printing of
1107 information to be tailored from within the XML message.

Ballot information may be included if required. This is a subset of the information defined in the 410-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.

1113 # 6.10 Outgoing Generic Communication (350a)



1114

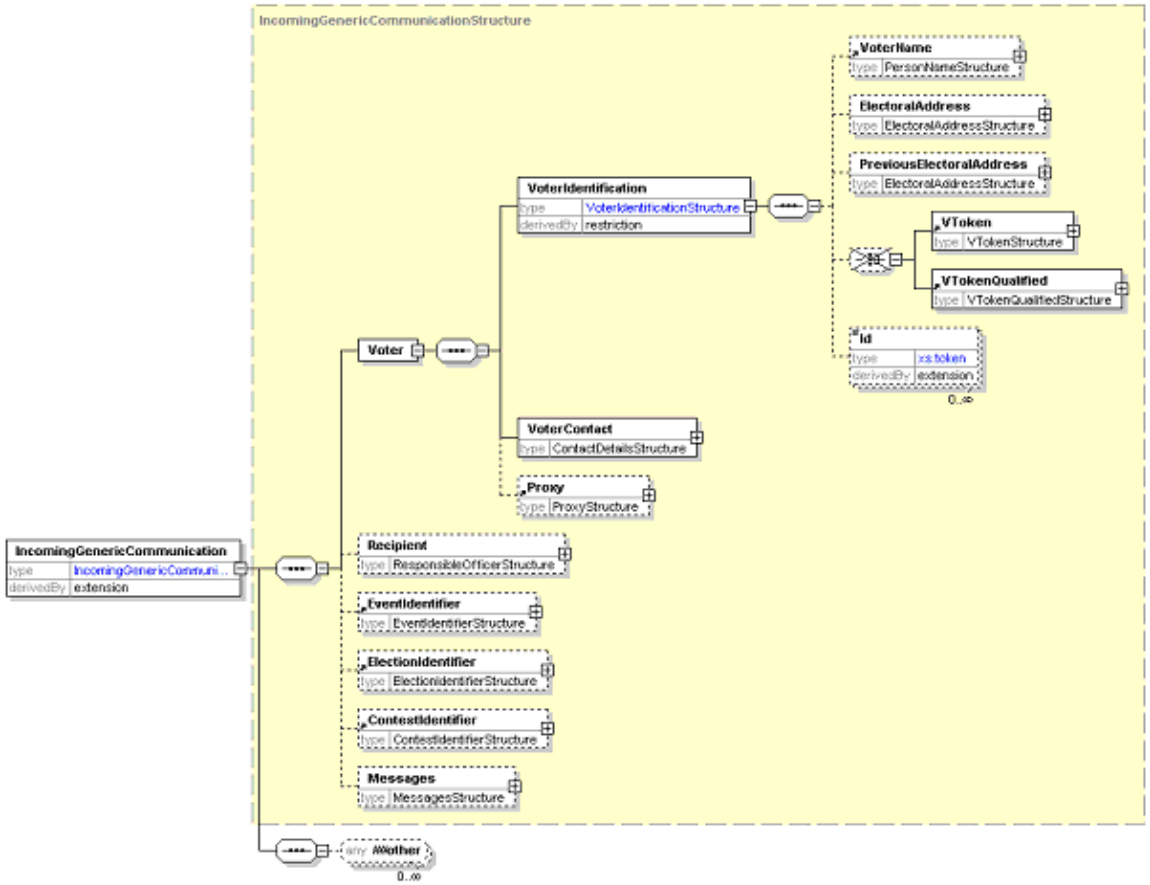
1115 ## 6.10.1 Description of Schema

1116 This schema provides a common structure for communications to the voter. Individual message
1117 types can be designed based on extensions of this schema.

1118 The voter must always provide a name and might provide one or more identifiers. These are
1119 shown as a restriction of the `VoterIdentificationStructure`, the restriction being to leave
1120 out the `VToken` and `VTokenQualified`. Contact details are also required, and it is expected that
1121 at least one of the allowed contact methods will be included. Inclusion of proxy information is
1122 optional.

1123 The identifiers for the election event, election and contest are optional. There is then an element
1124 in which a message can be placed in any of several different formats according to the channel
1125 being used.

6.11 Incoming Generic Communication (350b)



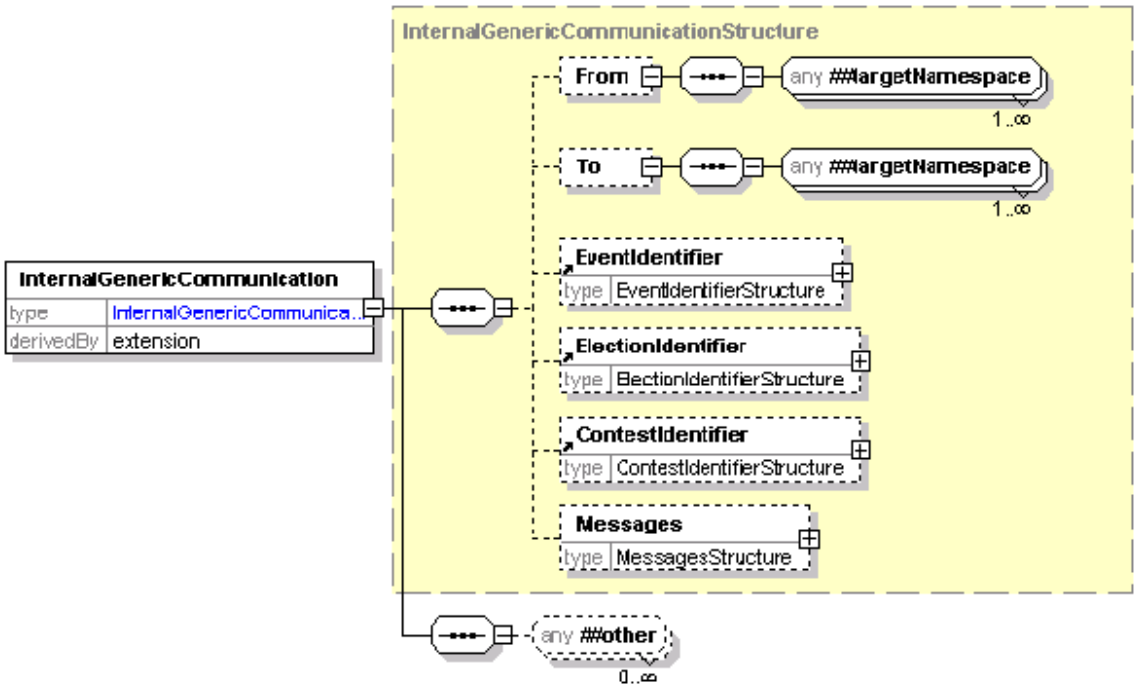
6.11.1 Description of Schema

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.

6.12 Internal Generic (350c)



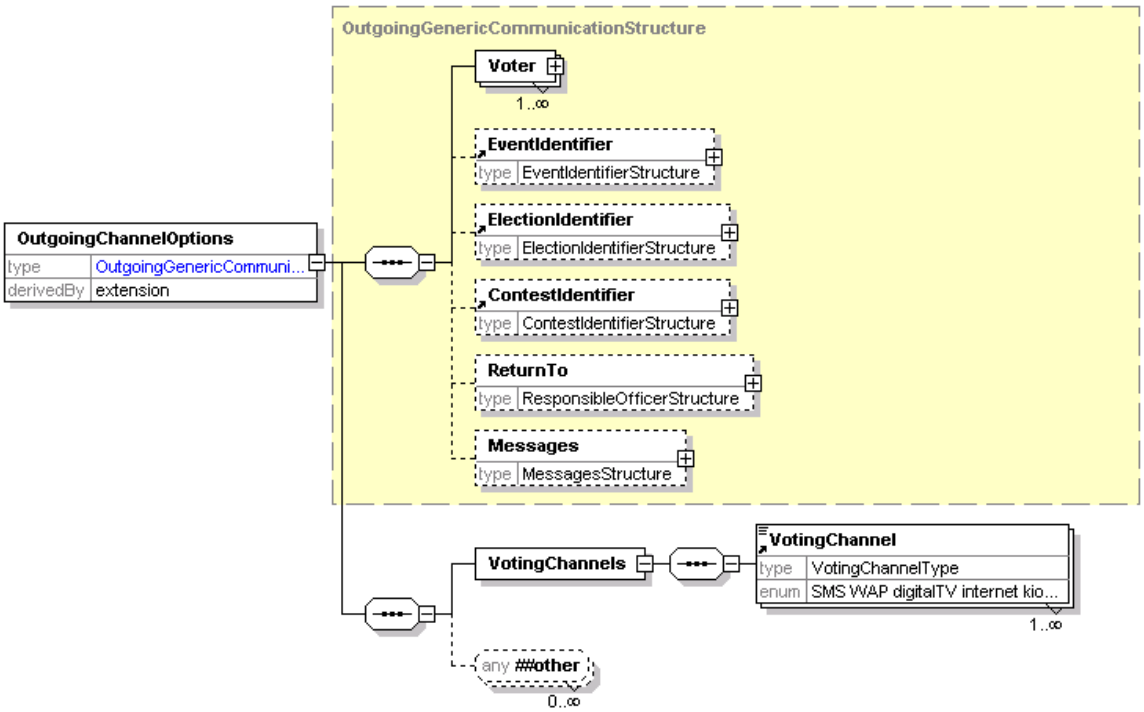
6.12.1 Description of Schema

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

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

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.

1150 # 6.13 Outgoing Channel Options (360a)

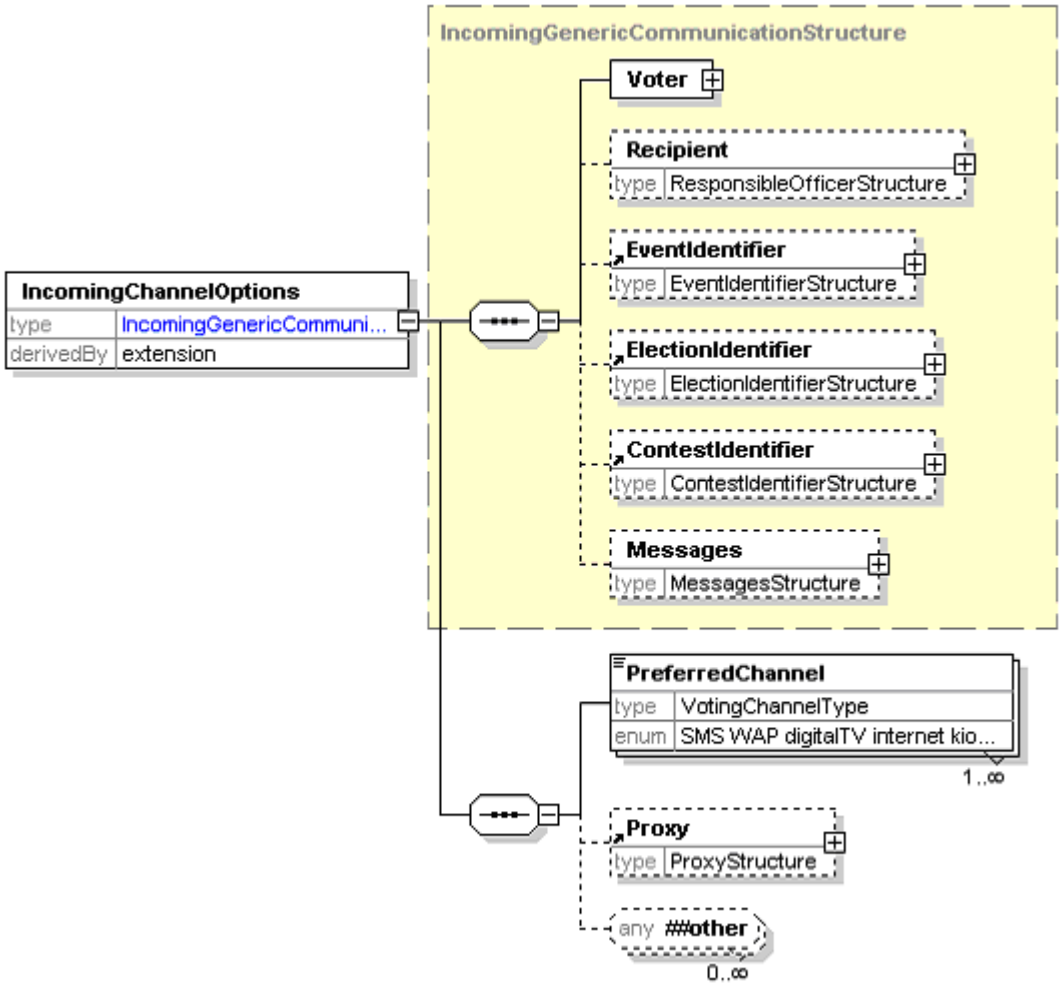


1151

1152 ## 6.13.1 Description of Schema

1153 This schema is used for messages offering a set of voting channels to the voter. It is an extension
1154 of schema 350a. A message conforming to this schema will include a list of allowed channels,
1155 either to request general preferences or for a specific election event or election within the event.

6.14 Incoming Channel Options (360b)

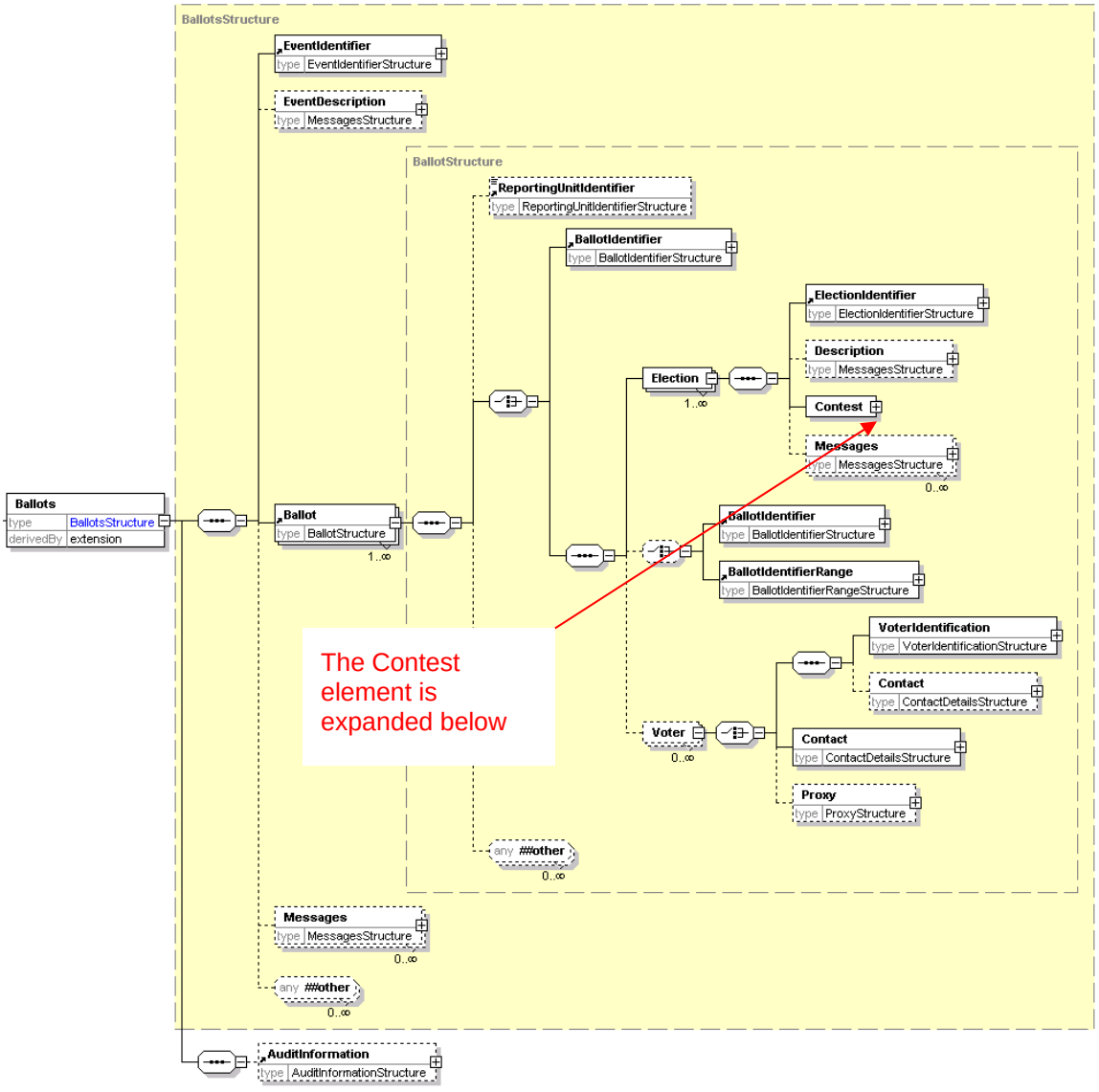


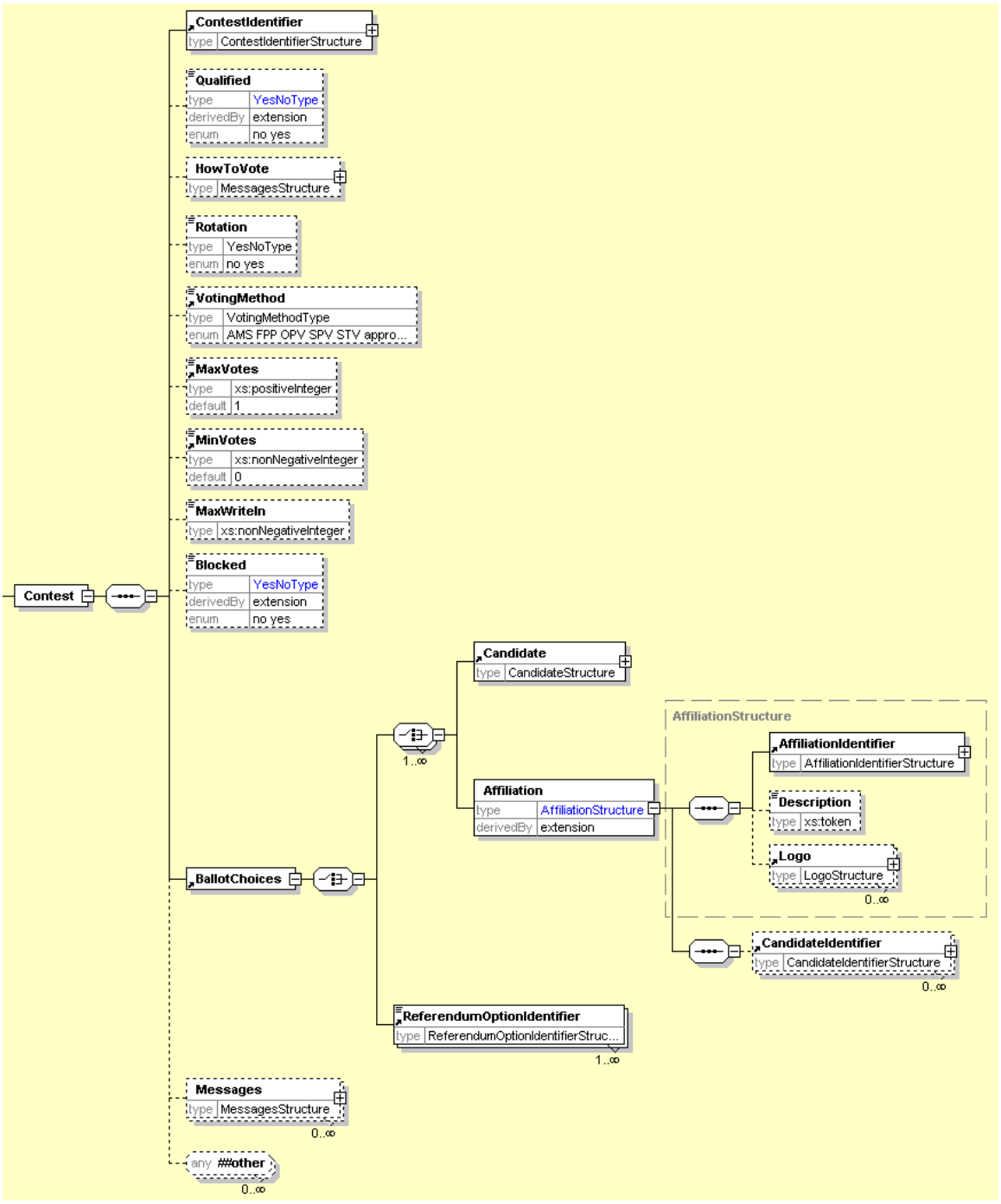
6.14.1 Description of Schema

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

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

6.15 Ballots (410)





1167
1168

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	

6.15.1 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.

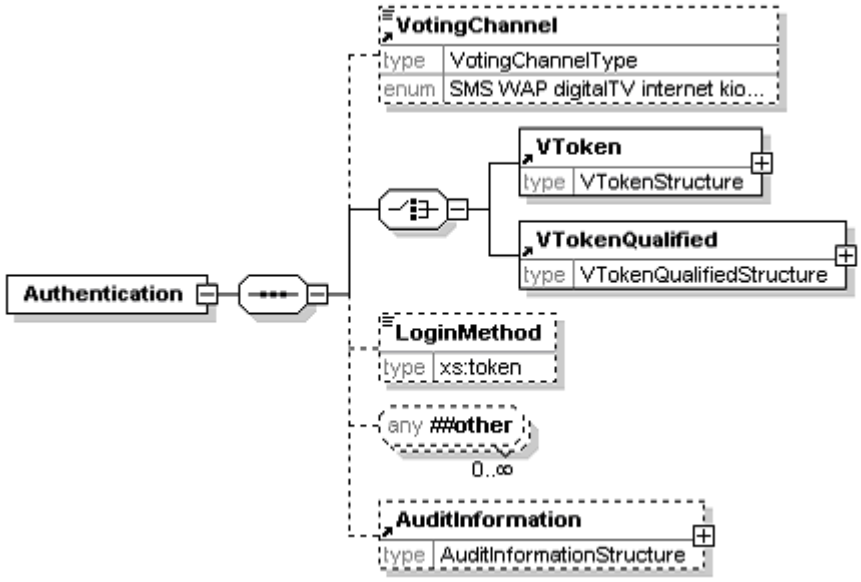
In the simplest case, a distributor can be sent information about the election event and a ballot ID to indicate the ballot to print.

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`).

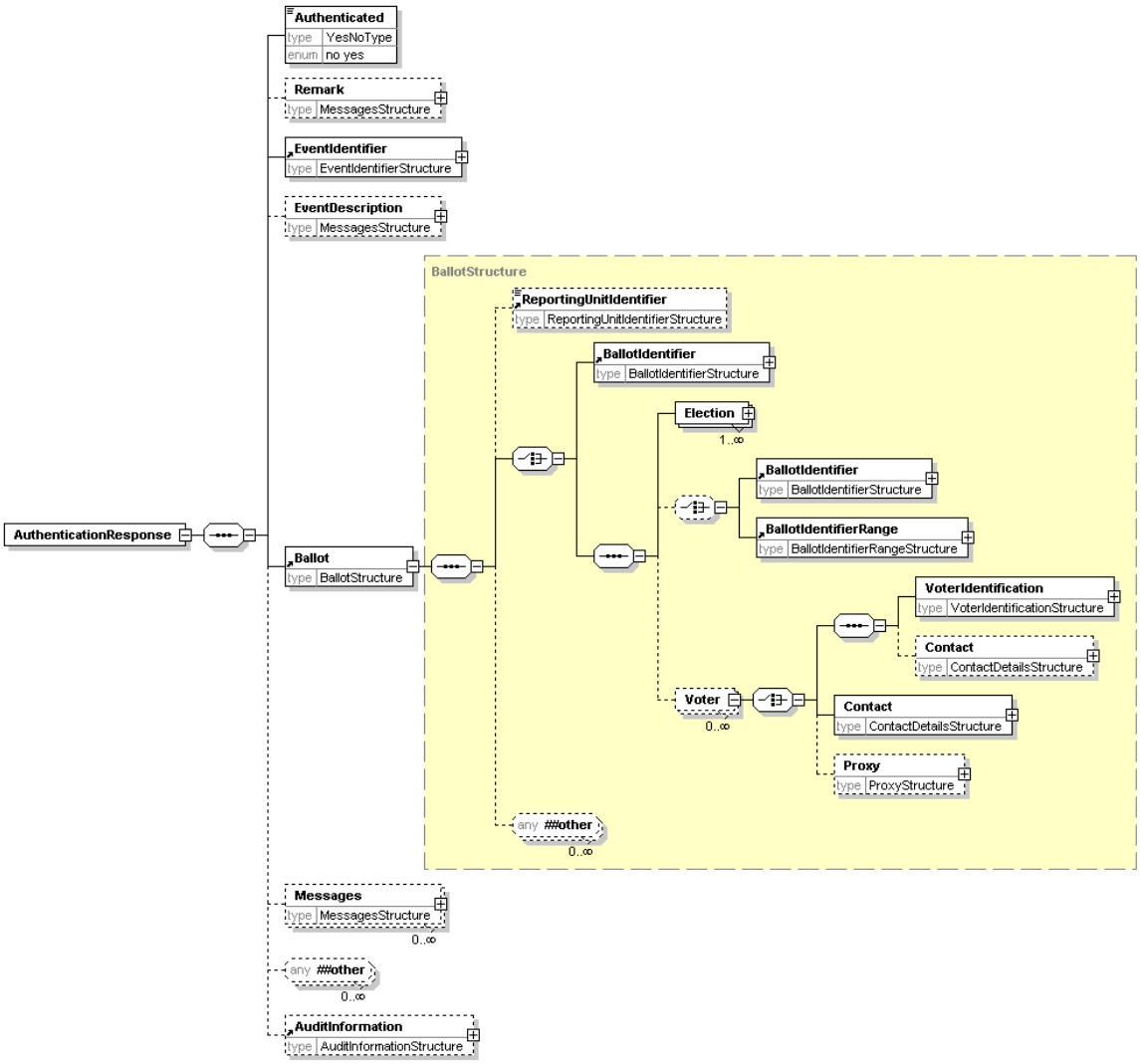
6.16 Authentication (420)



6.16.1 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.

1200 # 6.17 Authentication Response (430)



1201

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	

1202 ## 6.17.1 Description of Schema

1203 The authentication response is a response to message 420. It indicates whether authentication
1204 succeeded using the `Authenticated` element, and might also present the ballot to the user.
1205 This is a restriction of the Ballots element to allow only a single ballot per reply.

6.18 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	

6.18.1 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.

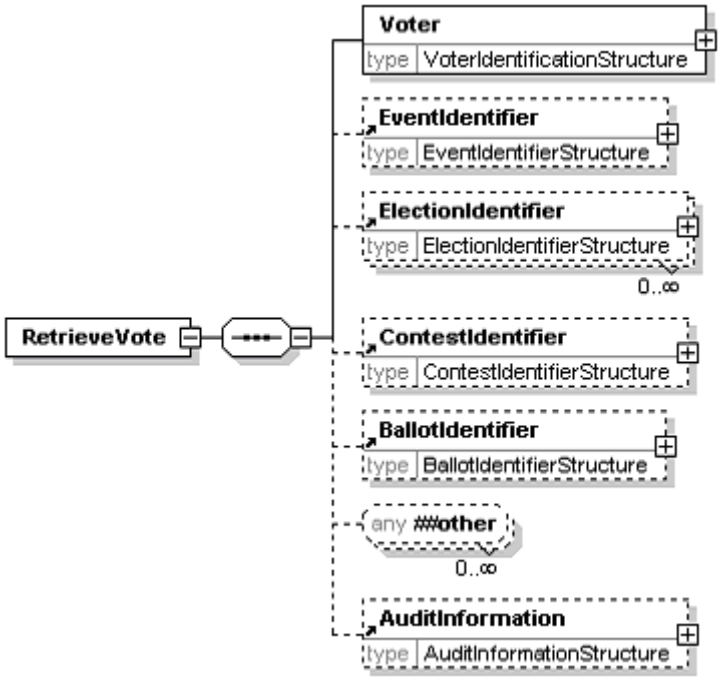
For each election, the contest is identified, with a set of, possibly sealed, votes. The votes are sealed at this level if there is a chance that the message will be divided, for example so that votes in different elections can be counted in different locations.

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`.

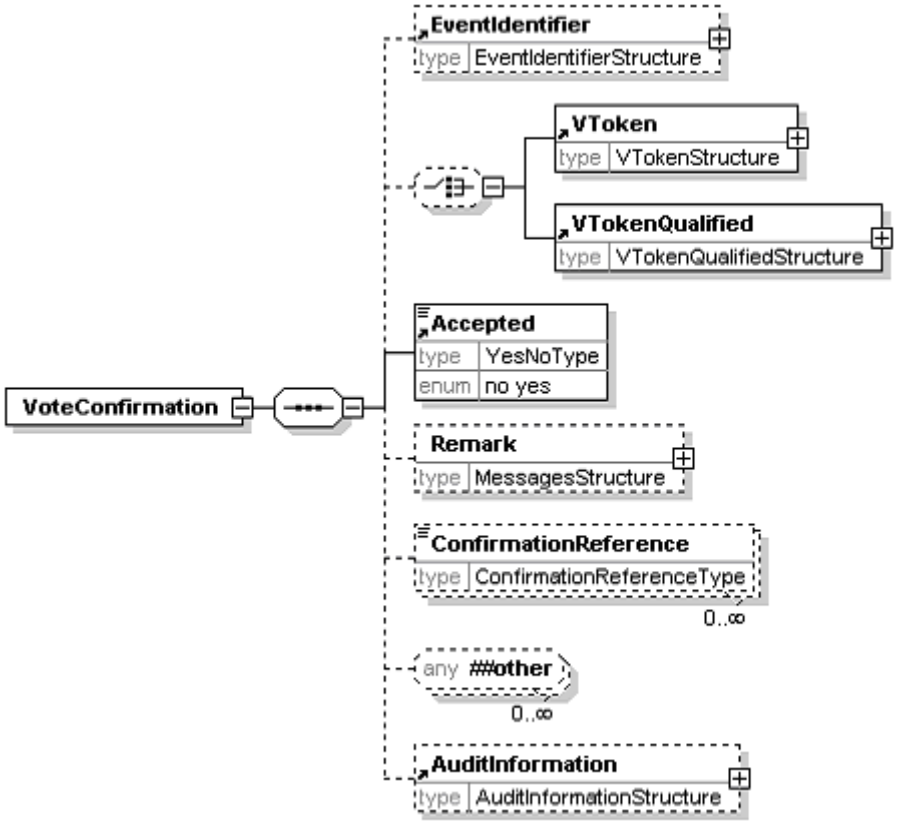
6.19 Retrieve Vote (445)



6.19.1 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.

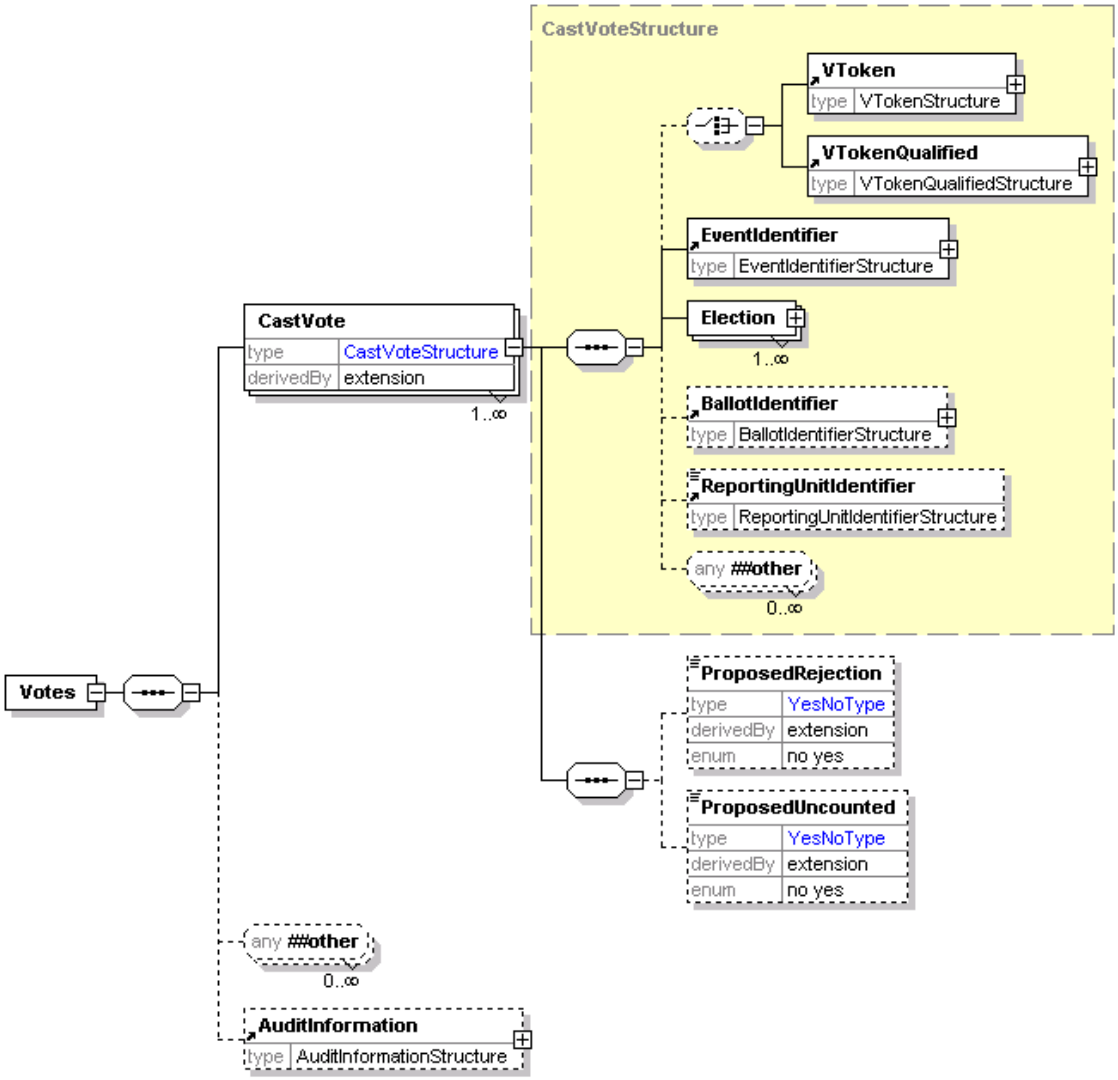
6.20 Vote Confirmation (450)



6.20.1 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.

1241 # 6.21 Votes (460)



1242

1243 See 440-CastVote for the detail of the `CastVoteStructure`.

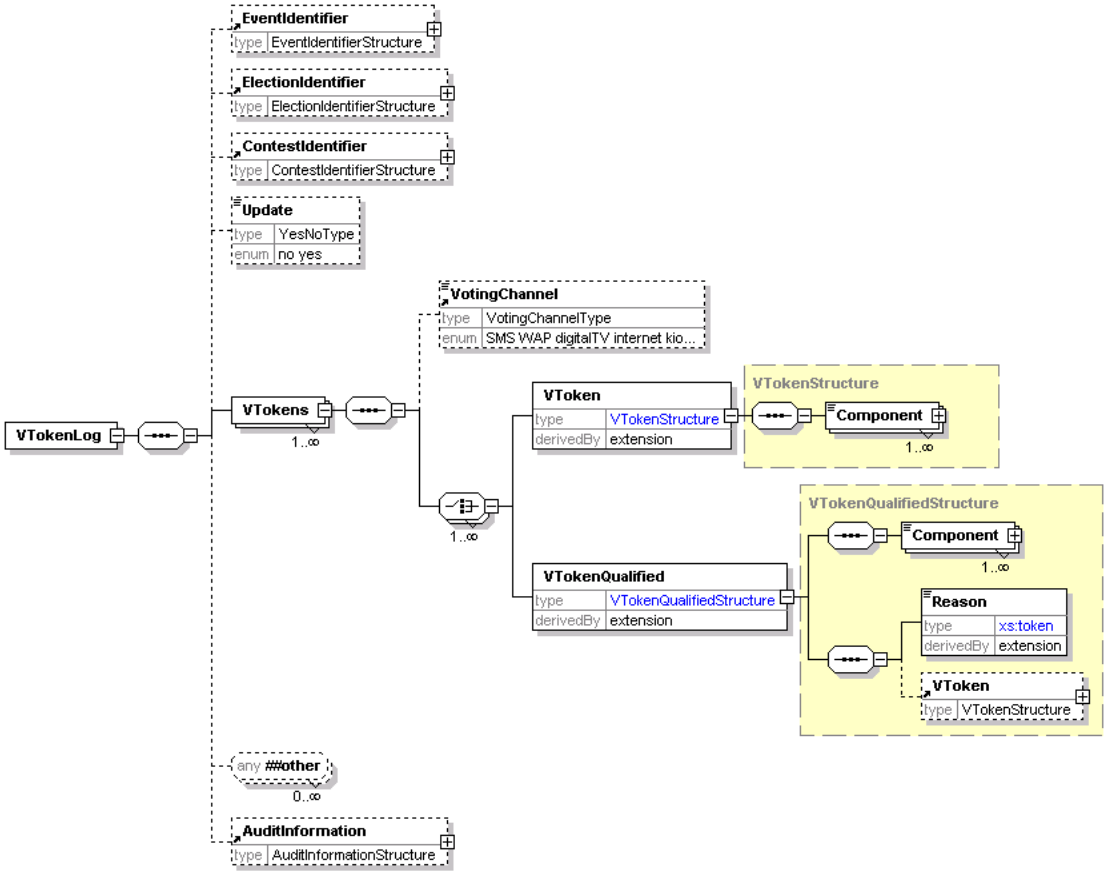
Element	Attribute	Type	Use	Comment
CastVote	Spoilt	xs:token	optional	
Contest	Spoilt	xs:token	optional	
Selection	Value	VotingValueType	optional	
	ShortCode	ShortCodeType	optional	
Candidate	Value	VotingValueType	optional	
ProposedRejection	Reason	xs:token	optional	
	ReasonCode	xs:token	required	
	Objection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

	Objection	YesNoType	optional	

6.21.1 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.

6.22 VToken Log (470)



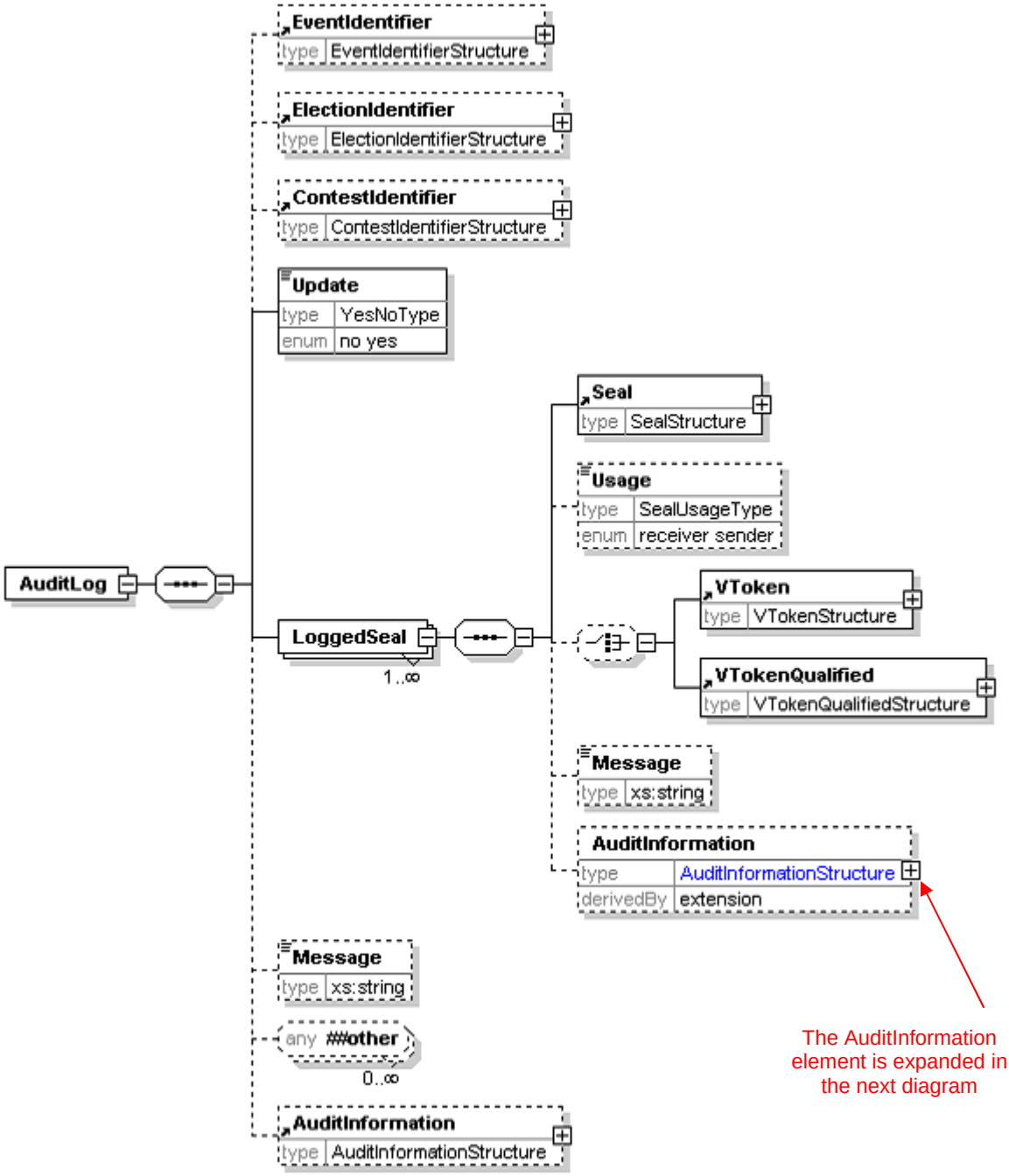
Element	Attribute	Type	Use	Comment
VToken	Status	xs:token (restricted)	required	
VTokenQualified	Status	xs:token (restricted)	required	

6.22.1 Description of Schema

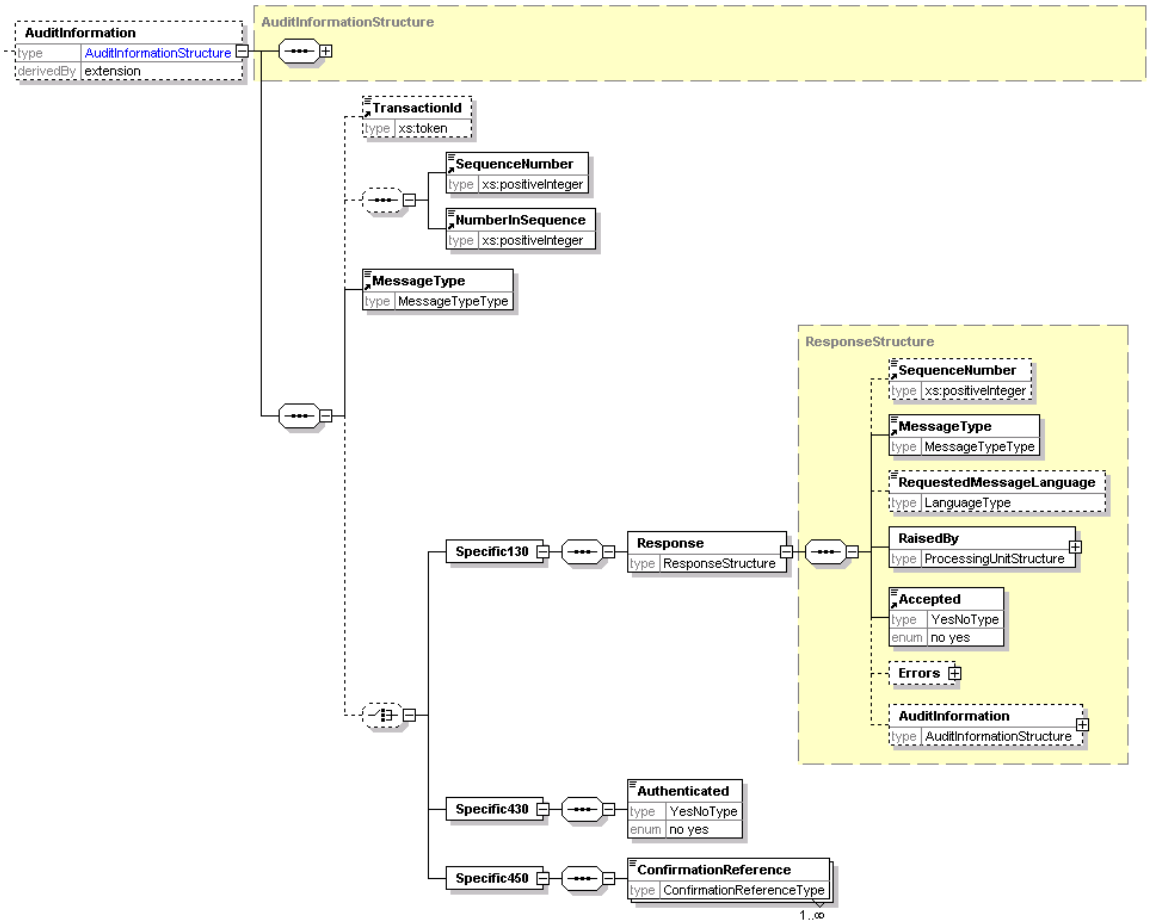
The message defined by this schema is used to add voting tokens (which may be qualified) to an audit log. The `VToken` or `VTokenQualified` is extended by the addition of a `Status` attribute with a value of `voted` or `unvoted` for the `VToken` and `voted`, `unvoted` and `withdrawn` for the `VTokenQualified`. 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.

1267

6.23 Audit Log (480)



1268



6.23.1 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:

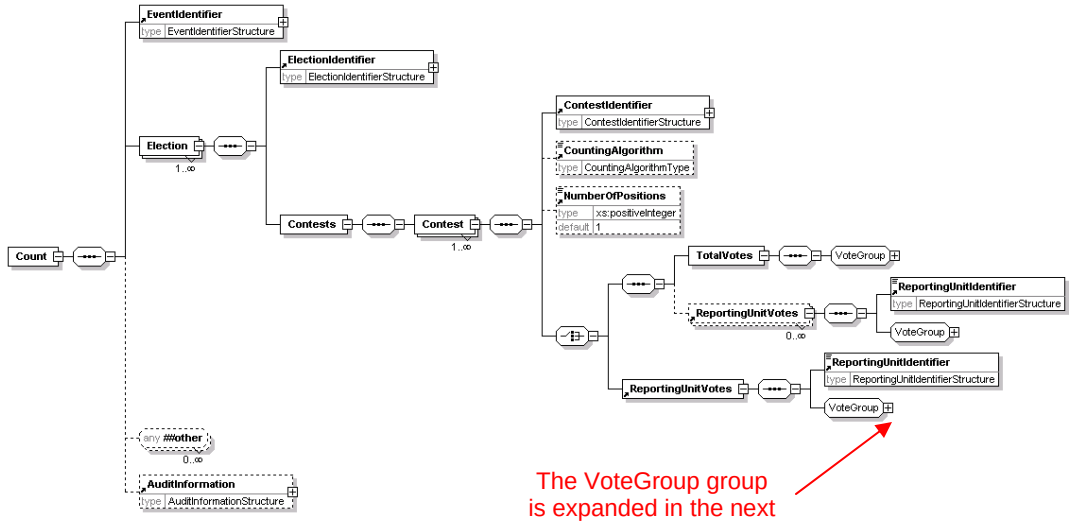
1. In some cases (e.g. a kiosk) there may be no gateway involved. In this case, the values in the Gateway column apply to the Originating Device.
2. Creators and receivers of 480 (audit log) messages may not be required to log the seals. In particlar, if an adit log message is sent per seal created or 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.

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

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

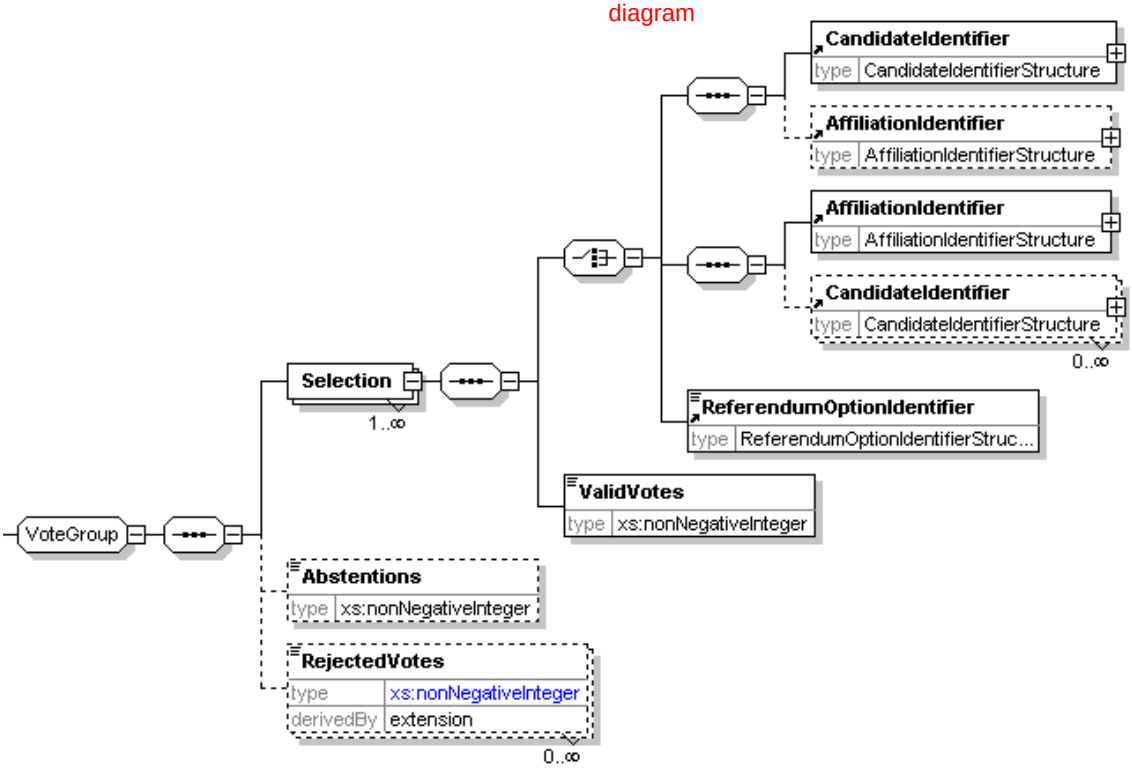
1291

6.24 Count (510)



1292

The VoteGroup group is expanded in the next diagram



1293

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	

6.24.1 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.

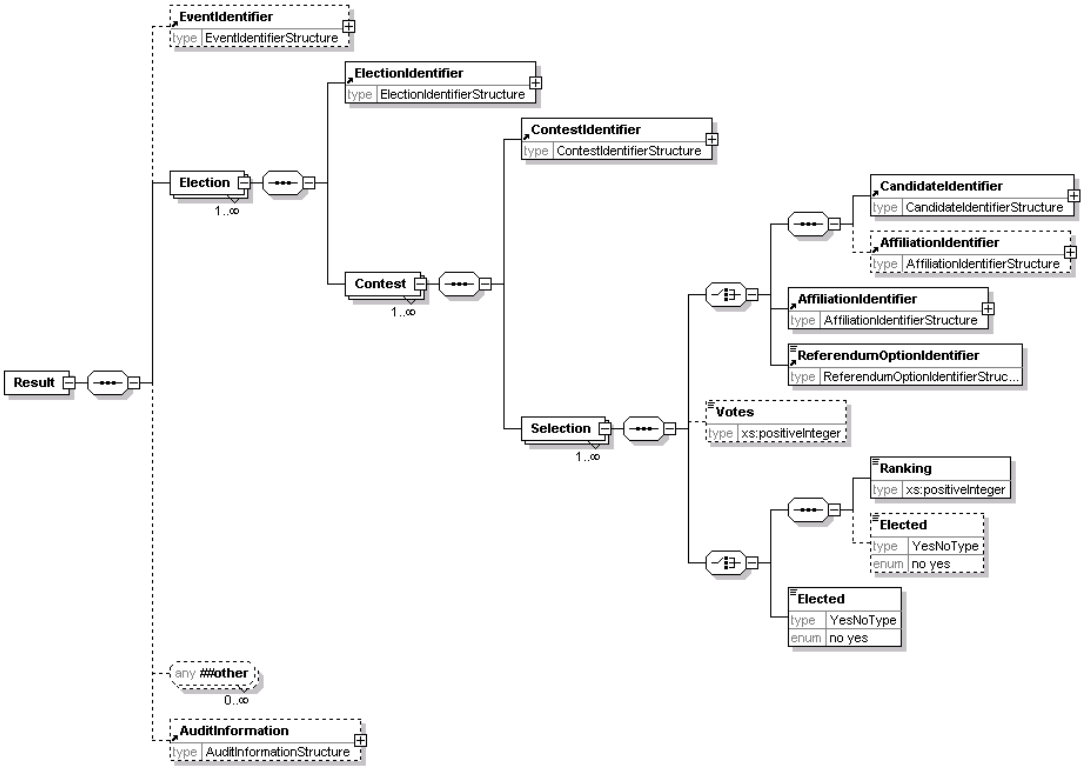
The message includes the election event identifier, and for each election, the election identifier, an optional reference to the election rule being used and information concerning the set of contests.

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 are 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.

6.25 Result (520)

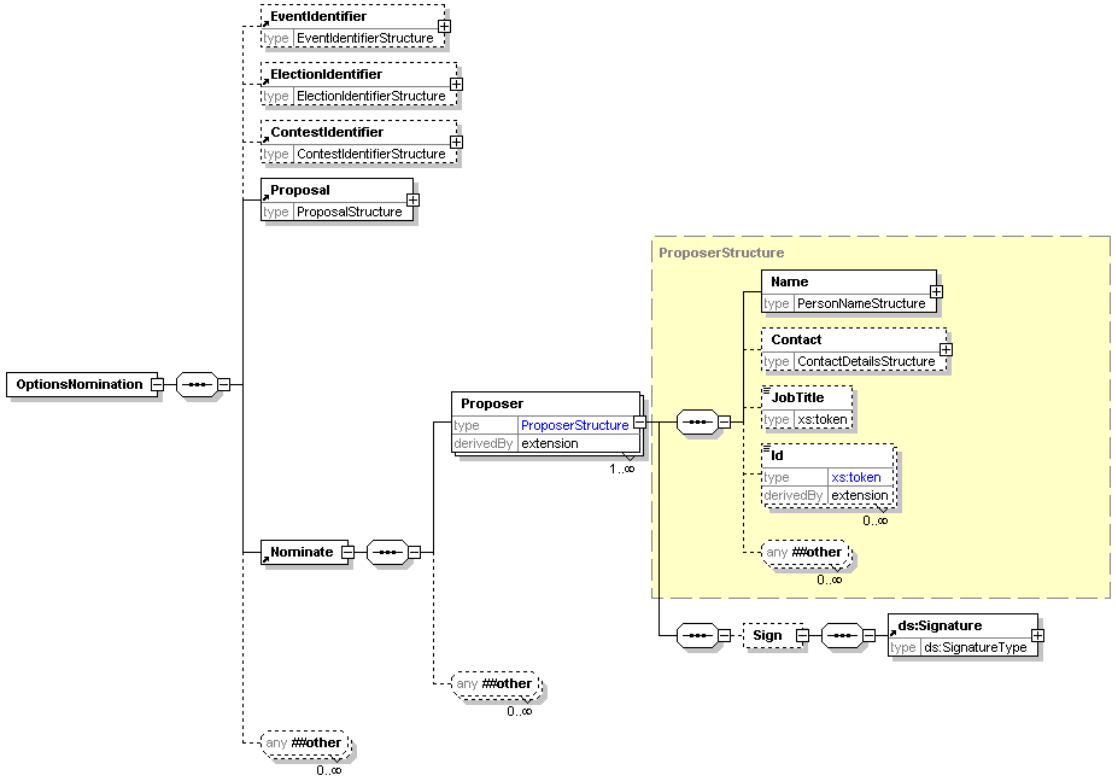


6.25.1 Description of Schema

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.

1334 # 6.26 Options Nomination (610)

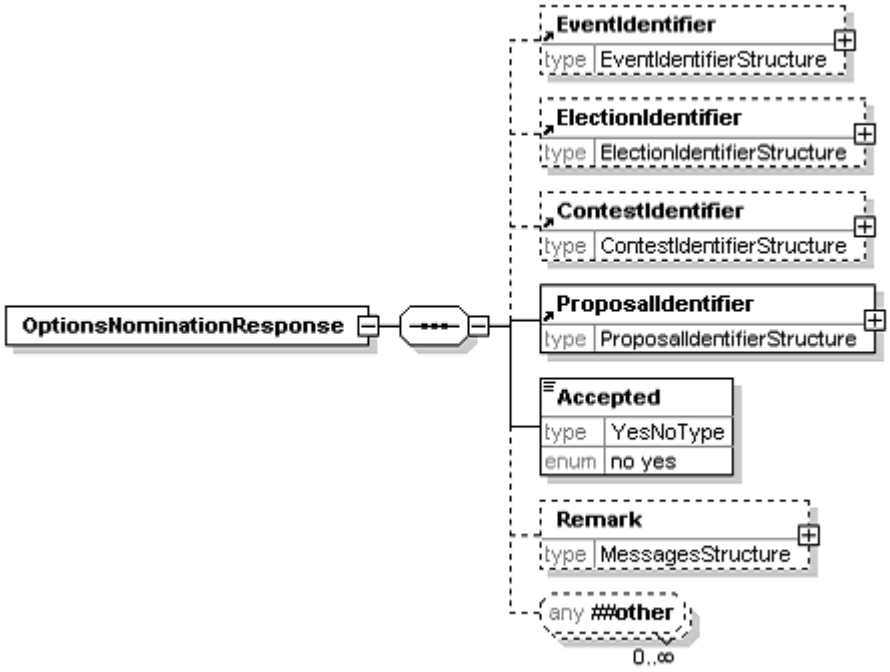


1335

1336 ## 6.26.1 Description of Schema

1337 This schema is used to submit proposals, for example for a referendum or company AGM. It uses
1338 the generic Proposal element to define the proposal itself. One of more proposers can be named
1339 and may sign the nomination.

6.27 Options Nomination Response (620)



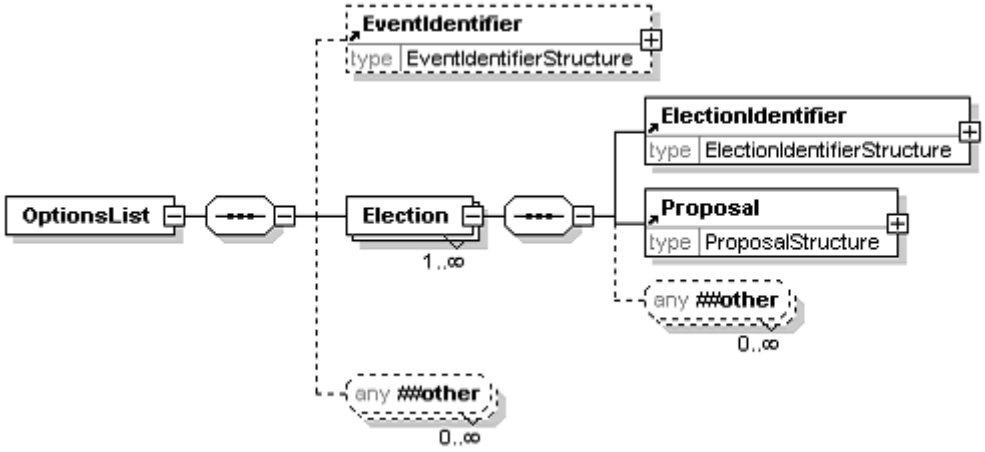
6.27.1 Description of Schema

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.

6.27.2 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

6.28 Options List (630)



6.28.1 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.

7 References

1. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types *IETF*
   http://www.ietf.org/rfc/rfc2046.txt
2. MIME Media Types *IANA*
   http://www.iana.org/assignments/media-types/
3. XML-Signature Syntax and Processing *W3C*
   http://www.w3.org/TR/xmldsig-core/
4. XML Path Language (XPath) Version 1.0 *W3C*
   http://www.w3.org/TR/xpath

Notices

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's procedures with respect to rights in OASIS specifications can be found at 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 implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.