OASIS

ELECTION AND VOTER SERVICES TECHNICAL COMMITTEE

ELECTION MARK-UP LANGUAGE (EML): XML Schemas

DOCUMENT CONTROL

Abstract

Date	Version	Status
29 April 02	1.0	Committee Specification for TC approval

Change History			
Date	**Version**	**Status**	**Editor/ Author**
22 Mar 02	0.3a	Draft e-Voting Schemas for public consultation	Paul Spencer
13 Feb 02	0.2a	Draft e-Voting Schemas for internal comment	Paul Spencer
07 Feb 02	0.1a	Draft e-Voting Schemas for internal comment	Paul Spencer

OASIS COPYRIGHT NOTICES

(A) "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."

(B) "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."

(C) "Copyright (C) The Organization for the Advancement of Structured Information Standards [OASIS] (date). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

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

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

(D) "OASIS has been notified of intellectual property rights claimed in regard to some or all of the contents of this specification. For more information consult the online list of claimed rights."

CONTENTS

1 Introduction

This document describes the Election Markup Language (EML) being developed under the auspices of the OASIS Election and Voter Services Technical Committee.

It should be treated as a draft - when complete and reviewed, it will be restructured into the document style and formatting used by the committee.

Currently, schema styles, naming and approach to namespaces match those used within the UK Government. The review process should examine this and result in any changes necessary.

1.1 Compliance

The schemas covered by this document comply with the provisions of the Voting Process models [1] and meet the requirements of the scenarios [2].

1.2 Conformance

To conform to this specification, a system must implement all parts of this specification that are relevant to that system's functionality. The required schema set will normally be part of the purchasing criteria and should indicate schema version numbers. For example, in the future, the specification for an election list system might specify that a conforming system must accept and generate XML messages conforming to the following schemas:

Schema	Accept	Generate
EML110	v1.0	
EML310	v2.0, v2.1	
EML320	v1.0, v2.0	v2.0
EML330		v1.1
EML340		v1.0
EML350		v1.0
EML360		v1.3

A conforming system will thus conform both to the relevant parts of this specification and to the accompanying schemas.

2 Schema Outline

2.1 Structure

The Election Markup Language specification defines a vocabulary (the EML core) and a message syntax (the individual message schemas). Thus most voting-related terms are defined as elements in the core with the message schemas referencing these definitions. The core also contains data type definitions so that types can be re-used with different names (for example, there is a common type to allow messages in different channel formats), or used as bases for deriving new definitions.

There is a third category of schema document within EML - the EML externals. This schema document contains definitions that are expected to be changed on a national basis. Currently this comprises the name and address elements, which are based on the OASIS Extensible Name and Address Language [3], but may be replaced by national standards such as those contained in the UK Government Address & Personal Details schemas . Such changes can be made by replacing just this single file.

As well as these, several external schemas are used. The W3C has defined standard schemas for XML [4], XLink [5] and XML signature [6]. OASIS has defined schemas for the extensible Name and Address Language (xNAL) [3]. As part of the definition of EML, the E&VSTC of OASIS has defined a schema for the Timestamp used within EML. All these schemas use their appropriate namespaces, and are accessed using `xsd:import` directives.

Each message (or message group) type is specified within a separate a schema document. All messages use the `EML` element from the election core as their document element. Elements declared in the individual schema documents are as descendents of the `EML` element. The example instance messages in the scenarios document demonstrate this structure.

2.2 IDs

XML elements which contain the names of certain data items may have an identifier, represented as an ID attribute. This identifier exists purely because of the channel (e.g. Internet voting) being used andis likely to be system generated. Some IDs are optional and some are mandatory as shown here:

Element	ID Opt/Man
BallotName	O
CandidateName	M
ContestName	M
ElectionEventName	M
ElectionName	M
LocationName	O
OptionName	M
VoterName	O

Note that this needs further consideration. For example, is it reasonable for the Election ID to be mandatory when a voter indicates a preferred voting channel?

Other items will have IDs related to the voting process independently of the channel being used. For example, a voter might be associated with an electoral roll number or a reference on a company share register. These IDs are coded as elements.

Each `schema` element has an `id` attribute that relates to the message numbering scheme in the Process document. Each message also carries this number.

2.3 Displaying Messages

Many e-voting messages are intended for some form of presentation to a user, be it through a browser, a mobile device, a telephone or another mechanism. These messages need to combine highly structured information (such as a list of the names of candidates in an election) with more loosely structured, often channel-dependent information (such as voting instructions).

Such messages start with one or more `Display` elements, such as:

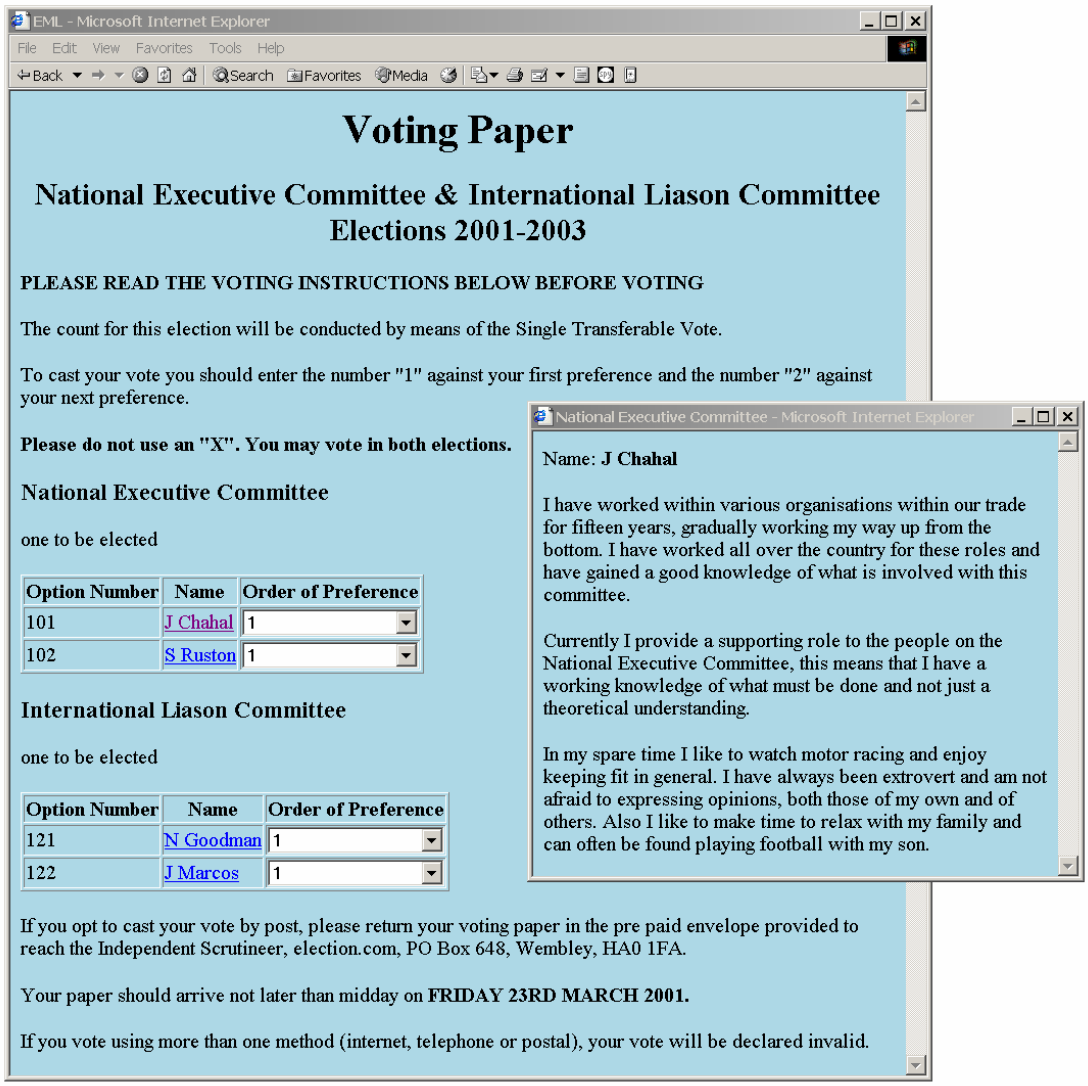
```
<?xml version="1.0" encoding="UTF-8"?>
<EML
  Id="410"
  SchemaVersion="0.1"
  xml:lang="en"
  xmlns="http://www.govtalk.gov.uk/temp/voting"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.govtalk.gov.uk/temp/voting
                      ..\schemas\ballot.xsd">
  <Display Format="html">
    <Stylesheet Type="text/xsl">../stylesheets/ballot.xsl</Stylesheet>
    <Stylesheet Type="text/css">../stylesheets/eml.css</Stylesheet>
  </Display>
  <Ballots>
    ...
```

This example shows a `Display` element providing information to the receiving application about an XSL stylesheet which transforms the message into HTML for displaying the ballot in a Web browser. The xml:lang attribute on the `EML` element indicates that the message content is in English. Other `Display` elements can be added to cover other formats. In the `Display` element in the example, the XSLT stylesheet reference is followed by a CSS stylesheet reference. In this case, the XSLT stylesheet referenced will pick up the reference to the CSS stylesheet as it transforms the message, and generate appropriate output to enable the displaying browser to apply that cascading stylesheet to the resulting HTML.

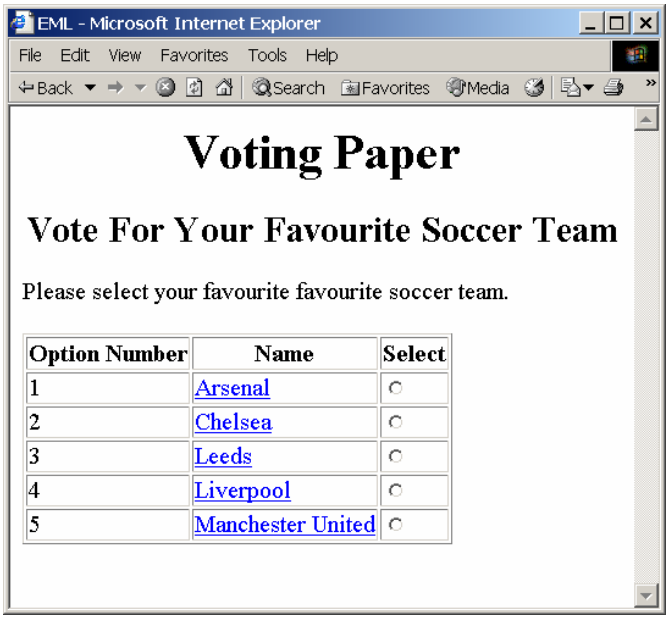
Not all information in a message will need to be displayed, and the creator of the message might have views on the order of display of the information. To allow stylesheets to remain generic, many elements in the schemas can have a `DisplayOrder` attribute. The values of these attributes determine the layout of the display (or the spoken voice if transforming to, for example, VoiceXML[5]), even when using a generic stylesheet.

When displaying messages in HTML, the expectation is that generic stylesheets will cover most cases, with the stylesheet output being embedded in a web page generated from an application-specific template. Similarly, voice applications might have specific welcome and sign-off messages, while using a generic stylesheet to provide the bulk of the variable data.

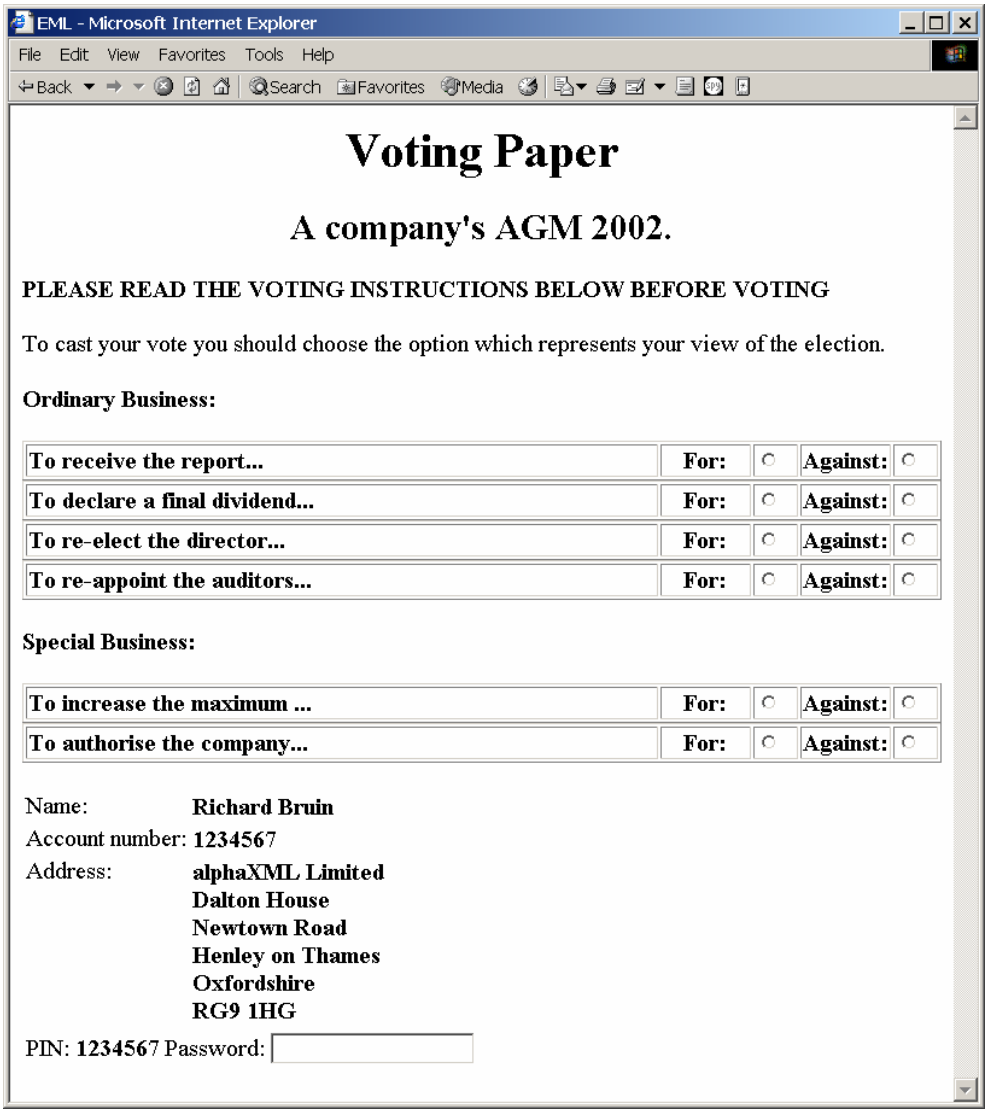
The three screen shots show the effect of using the same XSL stylesheet on the ballots for scenarios 2, 3 and 4 defined in the scenarios document [2]. In the first picture, clicking on the name of a candidate has popped up a window with additional details.



Screen shot of the ballot for scenario 2



Screen shot of the ballot for scenario 3



Screen shot of the ballot for scenario 4

2.4 Namespaces

The message schemas and the core schema are associated with the namespace `urn:oasis:names:tc:evs:schema:eml`. Use is also made of the external namespaces for XLink and xNAL, identified here using the prefixes `xlink:` and `xnal:`.

Version 2 of xNAL will have a namespace when it is released, and this will be used here. Currently, an invalid namespace is being used and the `elementFormDefault` has been set to "qualified" so that references to it can be qualified.

2.5 Extensibility

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

2.6 Conventions

Within this specification, the following conventions are used throughout:

- Element and attribute names are shown in `Courier` font.
- *Editorial comments are shown like this.*
- Diagrams are shown as generated by XML Spy v4.3, which was also used to generate the schemas and samples.
- 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.

3 Schema Descriptions

This section describes the schemas that make up EML. For data types and elements with complex content, diagrams of the structure are shown. These are expanded to show the complete structure other than where an element is accessed by reference or corresponds to a data type described elsewhere. If the element is derived from a type (rather than being an exact correspondence), the derived structure is shown.

3.1 Core

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

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 markup 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 datatype defined in XML Schema part 2 [8] 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;
- an address might be an `ElectoralAddress` or a `MailingAddress`, and so will specify a new element based on the datatype; and
- 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` datatype.

Currently, the name and address data types are taken from the xNAL schemas as mentioned previously. Investigation is needed to evaluate other schemas for inclusion, embodying agreed definitions for widely used data types such as email addresses and telephone numbers.

The following schema components are defined in `emlcore.xsd`. In the descriptions that follow, element definitions are not shown where they are an example of an obviously-named data type.

Elements	Complex Data Types	Simple Data Types
Affiliation AuditInformation BallotName CandidateName ContestName ElectionEventName ElectionName ElectionRuleId ElectionStatement EML EventEnd EventStart LocationName MaxVotes MinVotes OptionName Profile Proposer Seal VoterName VotingChannel VotingMethod VToken	AuditInformationStructure BallotNameStructure CandidateNameStructure ContactDetailsStructure ContestNameStructure ElectionEventNameStructure ElectionNameStructure EmailStructure IncomingGenericCommunicationStructure LocationNameStructure MessagesStructure OptionNameStructure OutgoingGenericCommunicationStructure ProposerStructure SealStructure TelephoneStructure VoterIdentificationStructure VoterInformationStructure VoterNameStructure VTokenStructure	ElectionRuleIdType EmailType TelephoneNumberType VotingChannelType VotingMethodType YesNoType

3.1.1 Simple Data Types

3.1.1.1 ElectionRuleIdType

The election rule ID is used to identify a rule governing an election. For example, a professional society may have a rule that, within a single election event, only a certain class of membership is entitled to vote in one election. The ID can be described as either an `xsd:NMTOKEN` (intended when it references a known document or database) or a URI.

3.1.1.2 EmailType

This is a string with a maximum length of 129 characters and a pattern `[^@]+@[^@]+`. This allows any characters except the @ symbol, followed by an @ symbol and another set of characters excluding this symbol.

3.1.1.3 TelephoneNumberType

Since this must allow for various styles of international telephone number, the pattern has been kept simple. The pattern is `\+?[0-9\(\)\-\s]{1,35}`. This allows an optional plus sign, then between 1 and 35 characters with a combination of digits, brackets, the dash symbol and white space.

3.1.1.4 VotingChannelType

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

- `SMS`
- `WAP`
- `digitalTV`
- `internet`
- `kiosk`
- `polling`
- `postal`

- telephone
- other

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

3.1.1.5 VotingMethodType

The VotingMethod type holds the enumerated values for the type of election (such as *first past the post* or *single transferable vote*). The full set of enumerations is:

- FPP
- OPV
- SPV
- STV
- additonalmember
- approval
- block
- partylist
- supplementary
- other

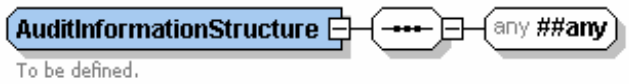
3.1.1.6 YesNoType

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

3.1.2 Complex Data Types

3.1.2.1 AuditInformationStructure

This data type is awaiting definition. For the moment, it allows any element content.



3.1.2.2 BallotNameStructure

The ballot name structure defines a string with two optional attributes: Id and DisplayOrder.

3.1.2.3 CandidateNameStructure

The candidate name structure defines a string with a mandatory Id and optional DisplayOrder attribute.

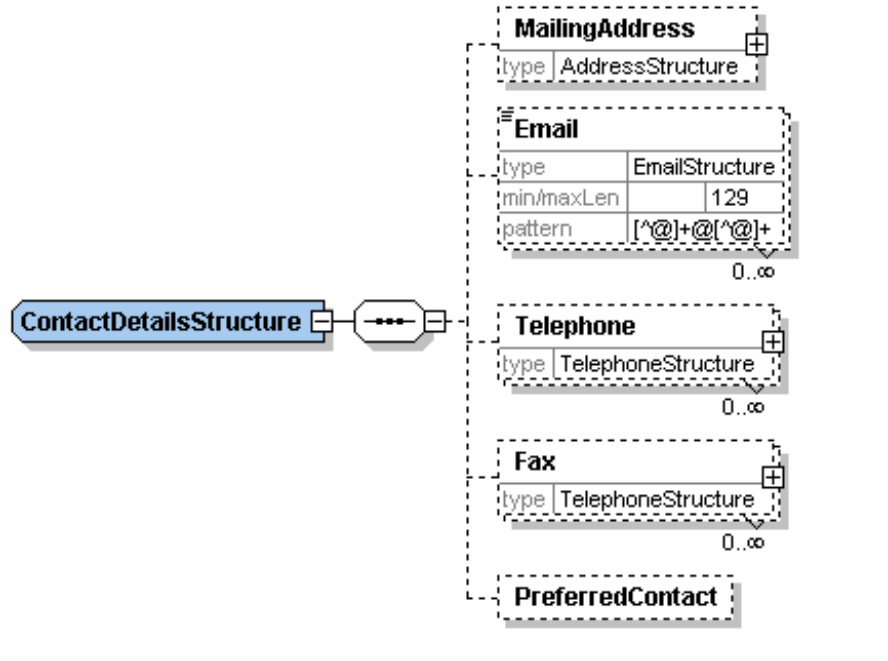
3.1.2.4 ContactDetailsStructure

This data type allows for a set of contact details. Each can be qualified through attributes as shown in the descriptions of e.g. EmailStructure below. The PreferredContact is an XLink to a definition of the preferred means of contact. The destination of this link could be part of this structure or could be elsewhere in this or another document. The use of this mechanism is illustrated in the scenario for voter registration for a UK Parliamentary Election.

As an example of the use of PreferredContact and the Preferred attributes on email addresses and phone and fax numbers, consider the case of an election officer needing to contact a person. The officer should take note of the preferred method of contact. If this is

unsuitable, for example the preferred method is by post, but the need for contact is urgent, the officer might decide that the telephone is the appropriate contact method, see several phone numbers and use the one whose `Preferred` attribute has a value of `yes`.

Feedback would welcomed on this mechanism for defining the preferred contact method.



3.1.2.5 ContestNameStructure

The contest name structure defines a string with a mandatory `Id` and optional `DisplayOrder` attribute.

3.1.2.6 ElectionEventNameStructure

The election event name structure defines a string with a mandatory `Id` and optional `DisplayOrder` attribute.

3.1.2.7 EmailStructure

This is an extension of the `EmailType` and adds a `Preferred` attribute of type `YesNoType`. This indicates which of several email addresses is preferred.

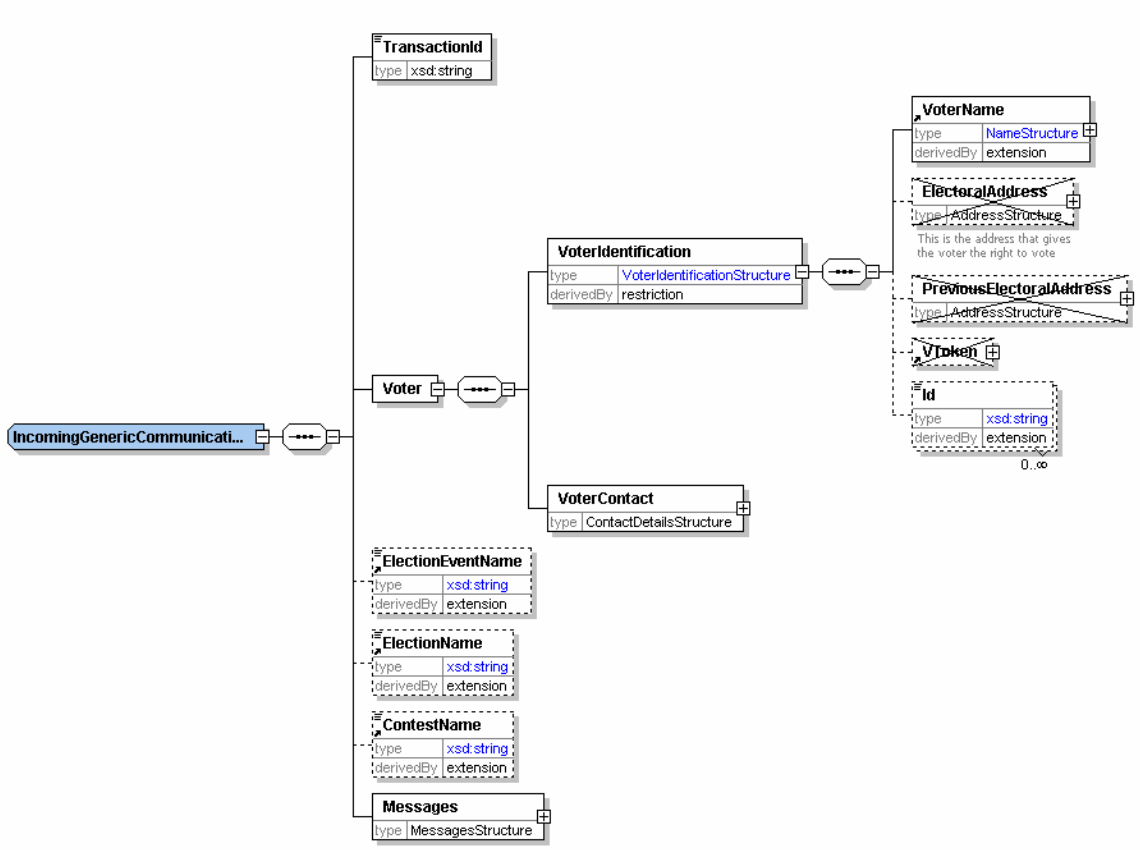
3.1.2.8 IncomingGenericCommunicationStructure

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 schema.

The `TransactionId` is used to reference an outgoing message to which this is a response or to provide a reference for a response.

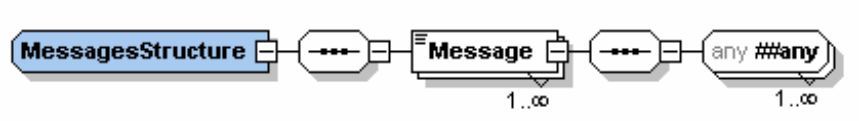
The voter must always provide a name and might provide one or more identifiers. These are shown as a restriction of the `VoterIdentificationStructure`. Contact details are also required, and it is expected that at least one of the allowed methods will be included.

The names of 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.



3.1.2.9 MessagesStructure

This data type is used for general display information. The `Messages` element contains a `DisplayOrder` attribute. The `Message` element contains a `Format` attribute indicating the type of output intended (HTML, WAP, VoiceXML etc.).



3.1.2.10 OptionNameStructure

The option name structure defines the name of a candidate (when a person) or choice (when a resolution) and is a string with a mandatory `Id` and optional `DisplayOrder` attribute.

3.1.2.11 OutgoingGenericCommunicationStructure

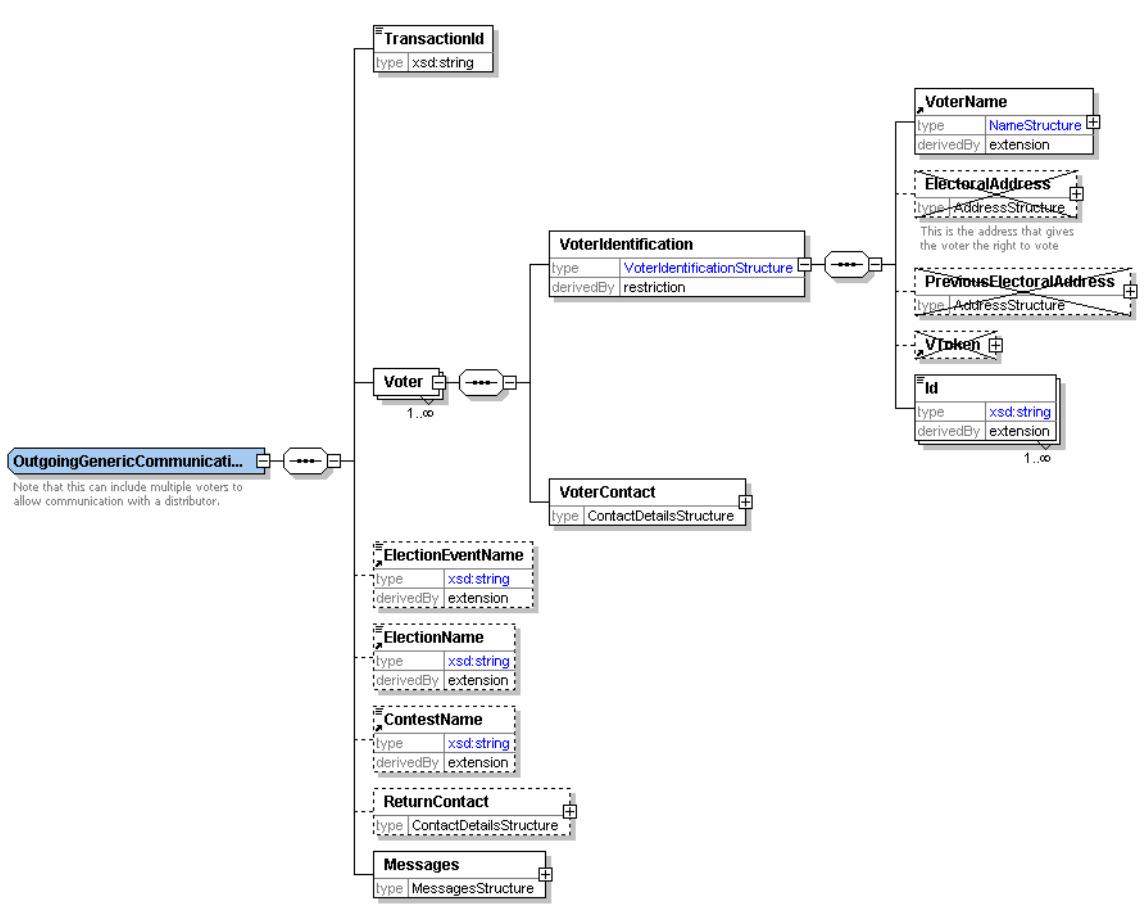
This data type provides a common structure for outgoing communications. Individual message types, such as that used for requesting the selection of a preferred voting channel (schema 360a) are based on extensions of this data type.

Unlike the schema for incoming communications, messages to multiple voters are allowed to enable this schema to be used to describe messages being sent to a distributor (such as a printer or email bureau).

The `TransactionId` is used to provide a reference to be used in a response or to reference an incoming message to which this is a response

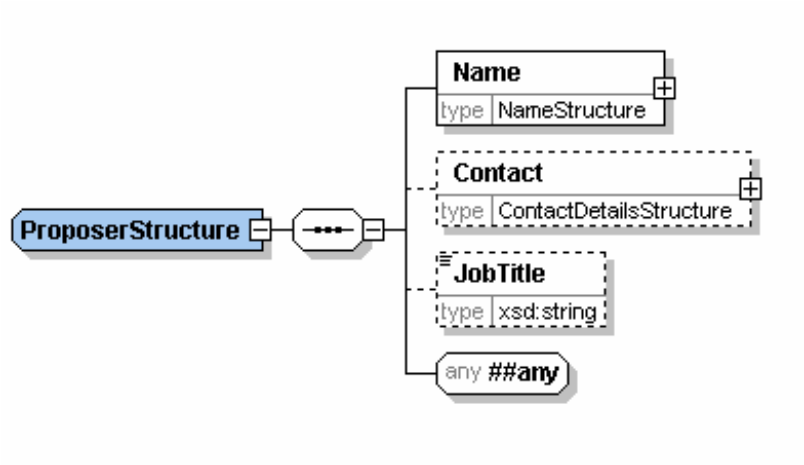
Each voter must have a name and one or more identifiers. These are shown as a restriction of the `VoterIdentificationStructure`. Contact details are also required, and it is expected that at least one of the allowed methods will be included.

The names of the election event, election and contest are optional. There may also be contact information provided to allow a reply. There is then an element in which a message can be placed in any of several different formats according to the channel being used.



3.1.2.12 ProposerStructure

A proposer proposes, seconds or endorses an option. A name is always required, and additional information might be needed.



3.1.2.13 SealStructure

The seal is used to protect information such as a vote, VToken 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.

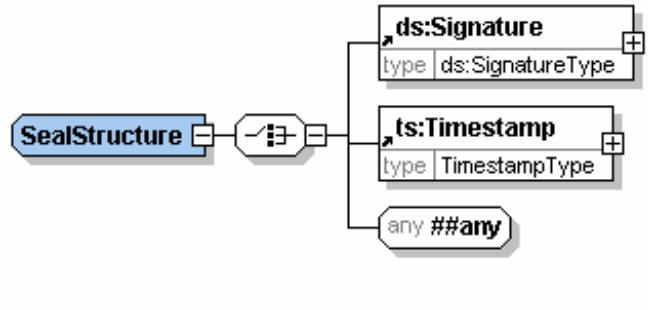
If a message is to be divided, each part must be separately sealed to protect the integrity of the data. For example, if votes in several elections are entered on a single ballot, and these votes are being counted in separate locations, each vote must be separately sealed.

A seal may be any structure which provides the required integrity characteristics, including:

- an XML signature (as defined in http://www.w3.org/2000/09/xmldsig)
- a time-stamp (see Appendix A)
- other mechanisms

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.

The other mechanism may be used, for example a combinations of an authentication mechanism and timestamps that will provid integrity of the vote, authentication of the identity of the system that collected the vote, and proof of the time that the vote was cast.



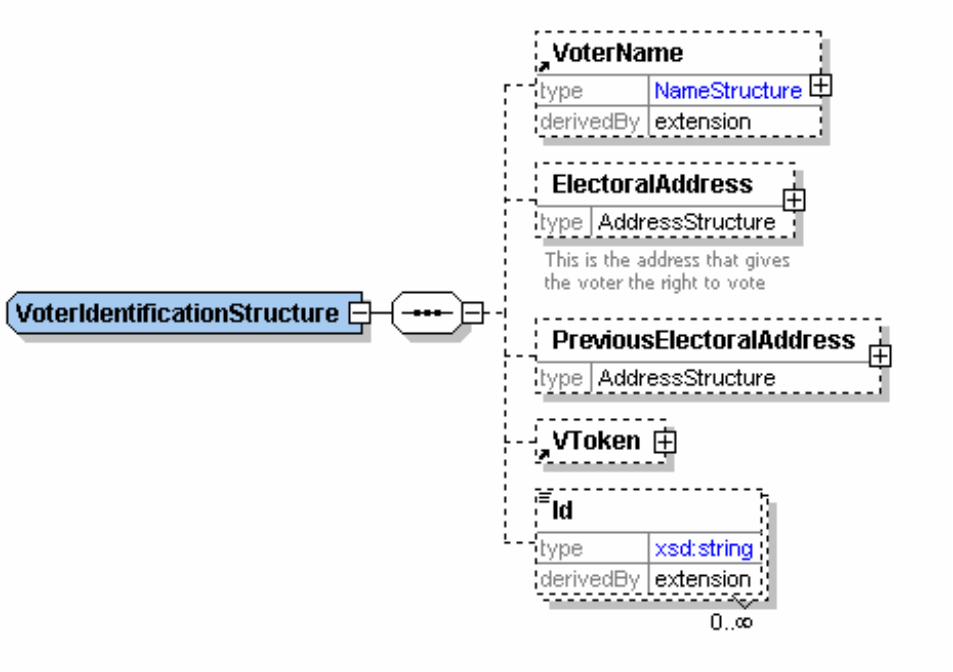
3.1.2.14 TelephoneStructure

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

3.1.2.15 VoterIdentificationStructure

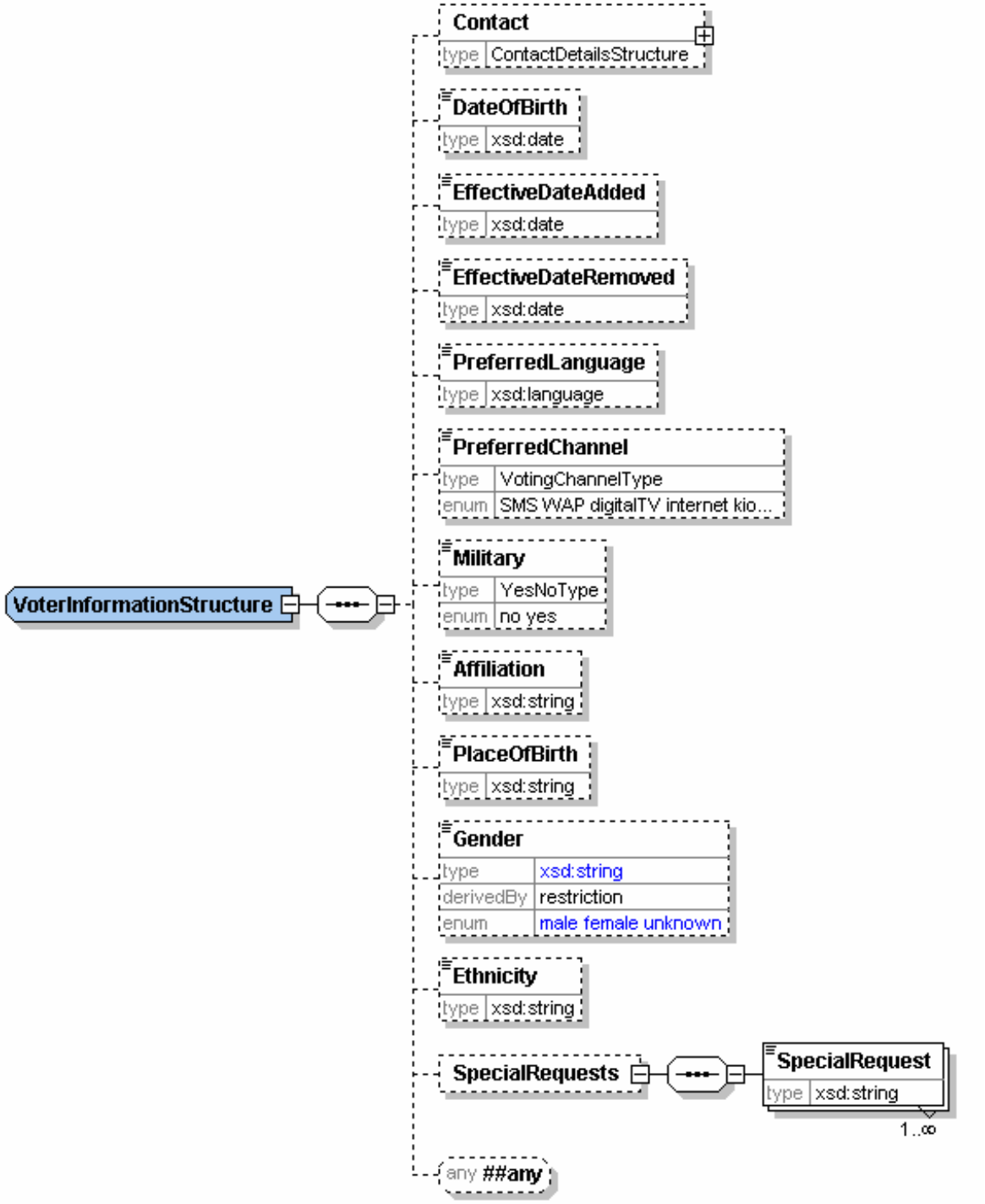
This is used wherever identification of a voter is required. It contains the voter's name and electoral address (using definitions from xNAL), the voting token and a number of identifiers (such as an electoral roll number). It may also include a previous electoral address if this is required (for example, because a voter has not been at his or her current address for more than a predefined period).

This has been produced as a complex data type rather than an element since it is expected that it will usually be restricted (for example, many uses will make the `VoterName` mandatory).



3.1.2.16 VoterInformationStructure

This contains more information about the voter. It contains all the information that would typically be included on an electoral roll other than that used for identification of the voter. It contains an `xsd:any` element for extensibility. This has been produced as a complex data type rather than an element since it is expected that it will usually be restricted.



3.1.2.17 VoterName

The voter name structure defines a string with two optional attributes: `Id` and `DisplayOrder`.

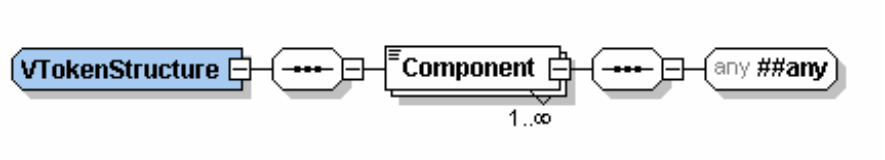
3.1.2.18 VTokenStructure

The `VToken` contains the information required to authenticate the voter's right to vote in a specific election or contest. A `VToken` 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 a various stages during the voting process (such as PIN, password and other

coded data elements). The totality of the VToken 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 VToken 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. Other election rules require a link to be maintained between a vote and a voter, in which case a link is maintained between the VToken data and the voter's identity.

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



3.1.3 Elements

Elements are defined here if:

- their type is a generic EML type such as MessagesStructure rather than a specific type such as AuditInformationStructure;
- they are derived from an EML data type by extension or restriction; or
- they are of a data type defined in XML Schema part 2 [8].

3.1.3.1 Affiliation

This is a text string used to identify the affiliation of a candidate in an election.

3.1.3.2 ElectionName

The election name is a string with a mandatory Id and optional DisplayOrder attribute.

3.1.3.3 ElectionStatement

This is the candidate's message to voters and is an extension of the MessagesStructure to allow multiple languages.

3.1.3.4 EML

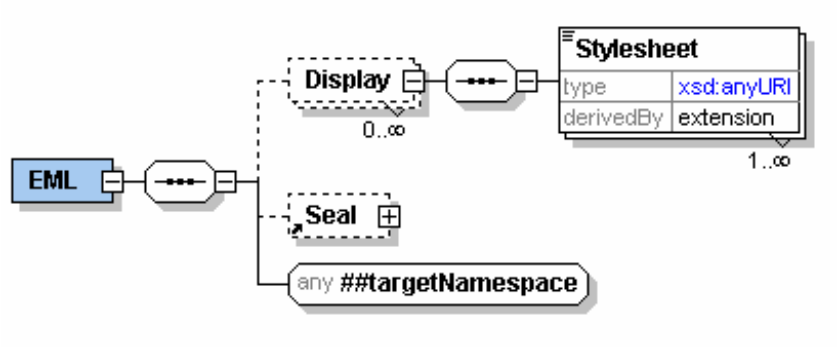
This element is used as the document element for all Election Markup Language messages. It has three attributes: an Id that relates to the Id of the associated message in the Process document, a SchemaVersion that indicates the full version number of the schema with which the message was designed to comply, and an xml:lang that indicates the language of the message content.

The EML element can contain multiple Display elements. These contain Stylesheet elements that indicate a MIME type (using the Type attribute) and a URI as the element value. The Display element has a Format attribute that indicates the target channel for the display (such as HTML). The reason for having multiple Display elements is to allow the same message to be presented appropriately through different channels.

The EML element can also contain a Seal element. This is used to seal the complete message so that any tampering can be detected.

In general, there will only be a single `Stylesheet` element per `Display` element. More are allowed so that the output of an XSLT transformation to HTML can contain a reference to a CSS stylesheet to be used to display the tansformed message.

Finally, the `EML` element can contain any other element from the EML namespace. These will be elements such as `Ballots` and `VoterRegistration` defined in other schema documents.



3.1.3.5 EventEnd

This is the end date/time of the election event in `xsd:dateTime` format.

3.1.3.6 EventStart

This is the start date/time of the election event in `xsd:dateTime` format.

3.1.3.7 LocationName

The location name is a string with two optional attributes: `Id` and `DisplayOrder`.

3.1.3.8 MaxVotes

The maximum number of votes allowed (also known as the vote limit). This is an `xsd:positiveInteger` and defaults to a value of `1`.

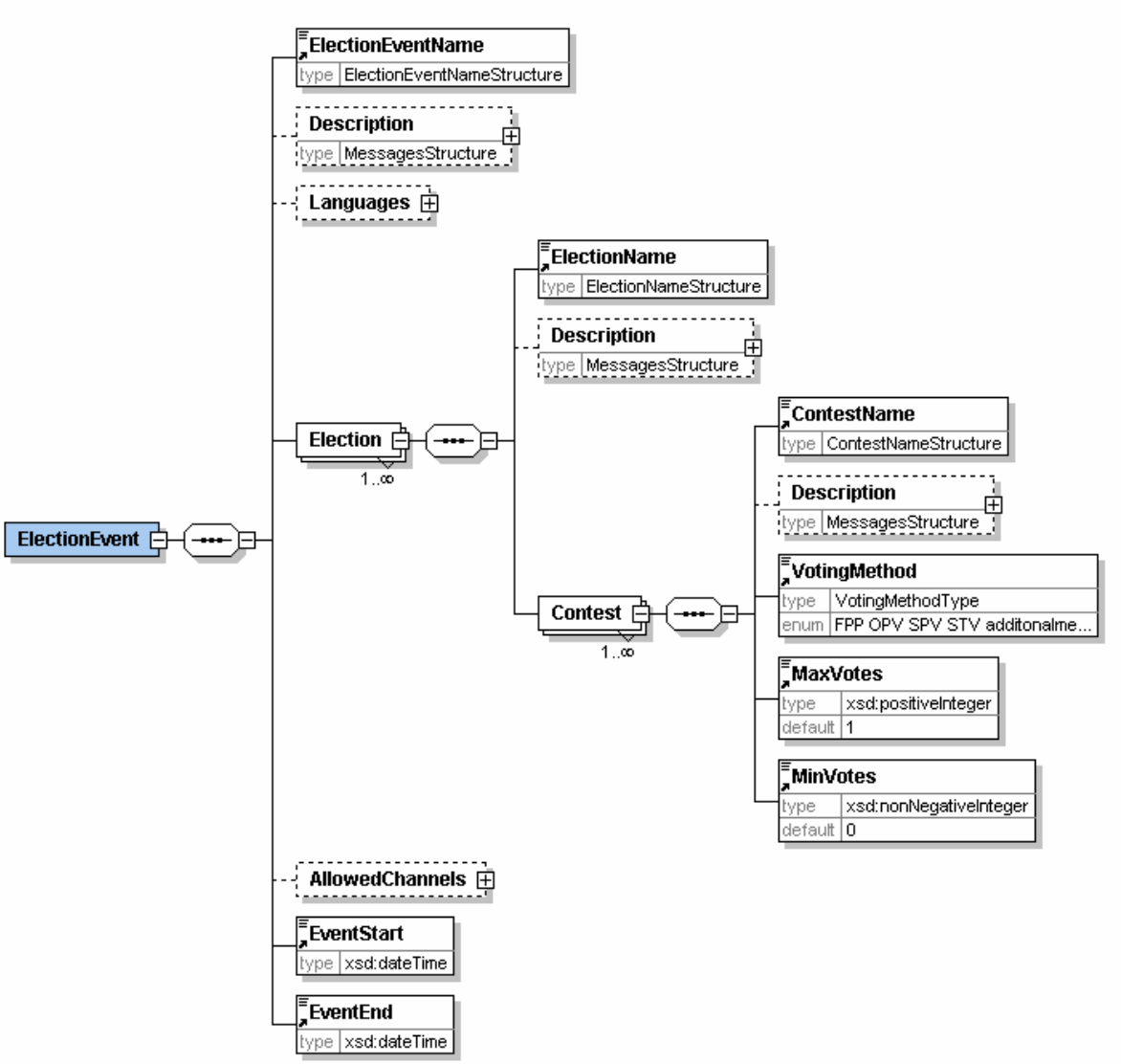
3.1.3.9 MinVotes

The minimum number of votes allowed. This is an `xsd:nonNegativeInteger` and defaults to a value of `0`.

3.1.3.10 Profile

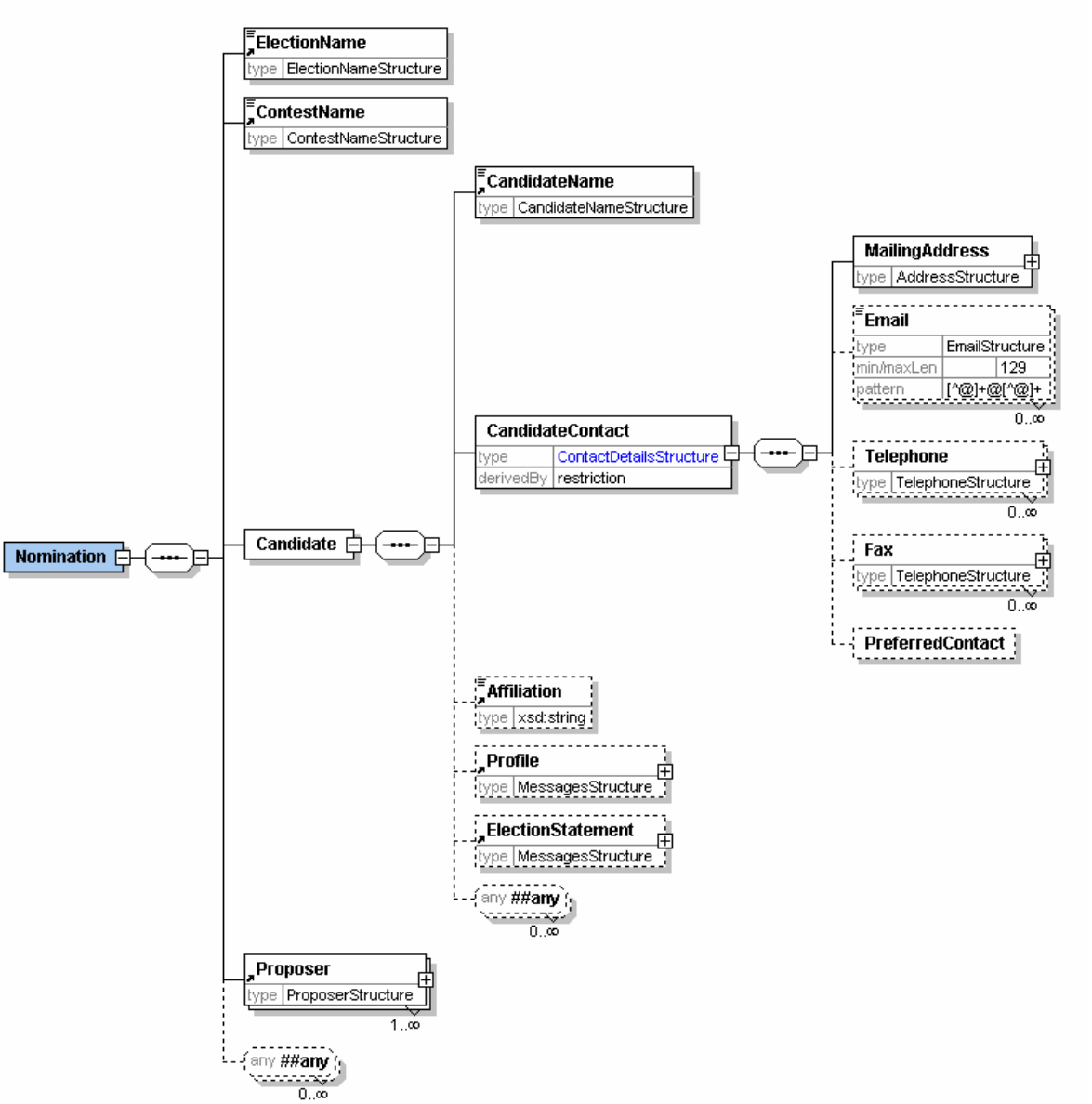
This is the candidate's profile statement and is an extension of the `MessagesStructure` to allow multiple languages.

3.2 110 - Election Event



This schema is used for messages providing information about an election or set of elections. An event has a start and end date and time, a list of allowed voting channels, a list of the languages in which information is to be available and a set of one or more elections. Each election may have multiple contests, each of which can have a different voting method (e.g. *first past the post* or *single transferable vote*). Some voting methods will specify the maximum and minimum numbers of votes, but if these are omitted, they default to sensible values.

3.3 210 - Nomination



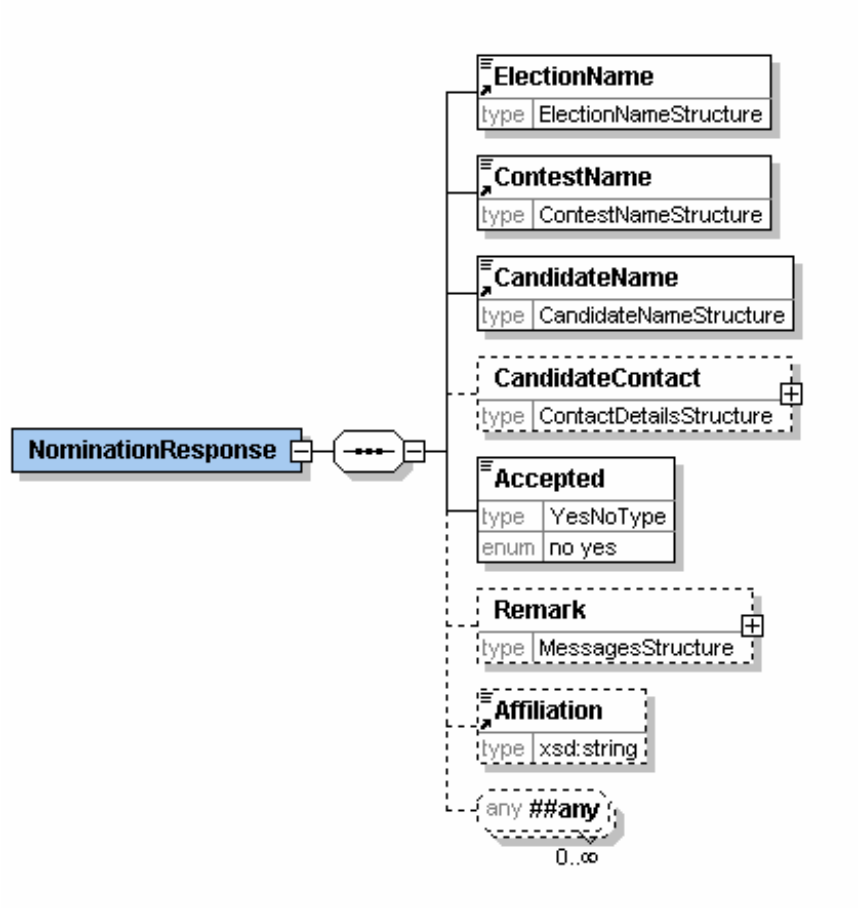
This schema is used for messages nominating candidates in an election. Note that it does not cover other forms of option nomination - only human candidates.

The election and contest must be specified as well as information about the candidate and one or more proposers. The candidate must supply name and contact information. The contact data is derived from the standard data type by making the address mandatory. Optionally, the candidate can provide an affiliation (e.g. a political party) and textual profiles and election statements. These two items extend 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.

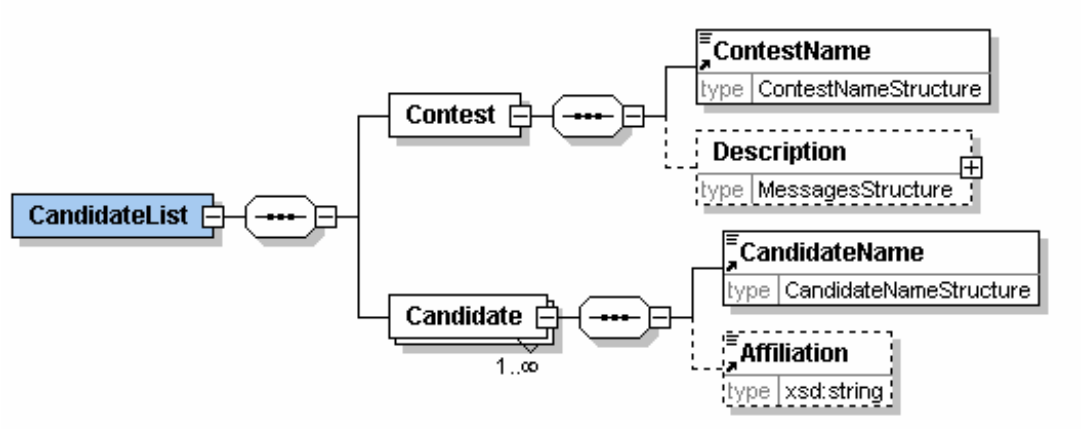
Finally, there is scope to extend the schema by adding additional information to the nomination.

3.4 220 - Nomination Response



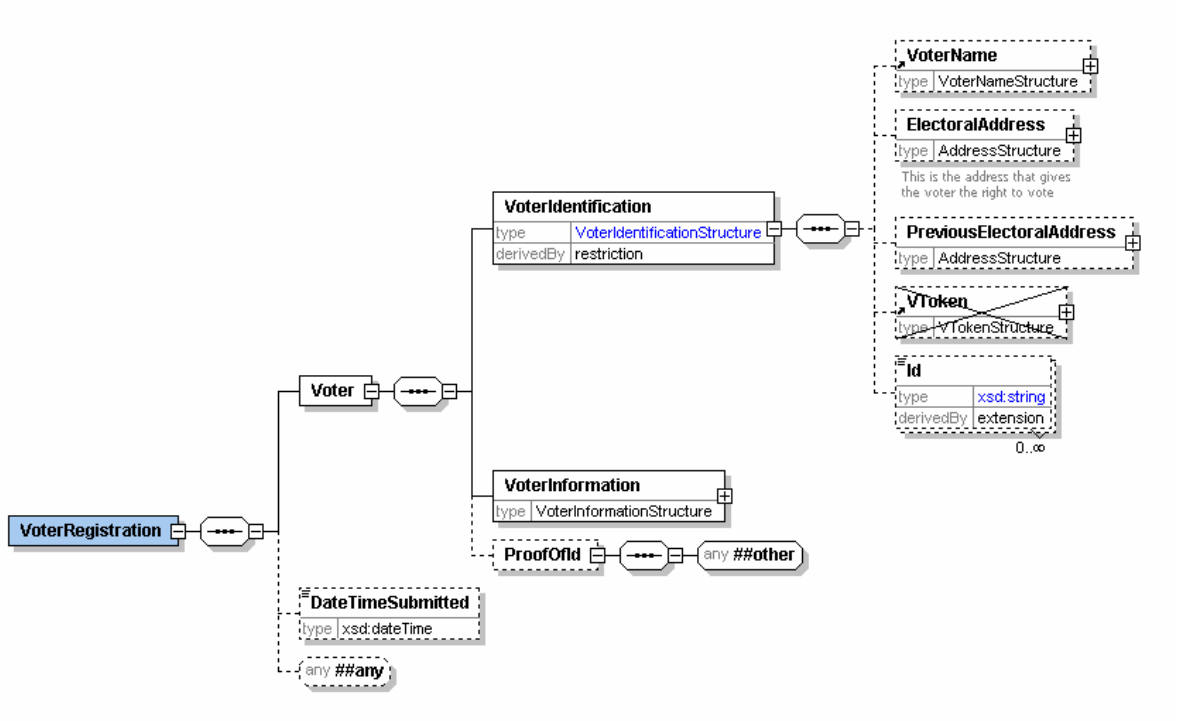
This message is sent from the election organiser to the candidate to say whether the nomination has been accepted. Along with the acceptance information and the basic information of election, contest and candidate names, the candidates contact details and affiliation can be included and a remark explaining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

3.5 230 - Candidate List



This schema is used for messages transferring candidate lists for a specified contests. It has the contest name (with its ID), optionally a contest description and then a list of candidates, each with a name and optional affiliation.

3.6 310 - Voter Registration

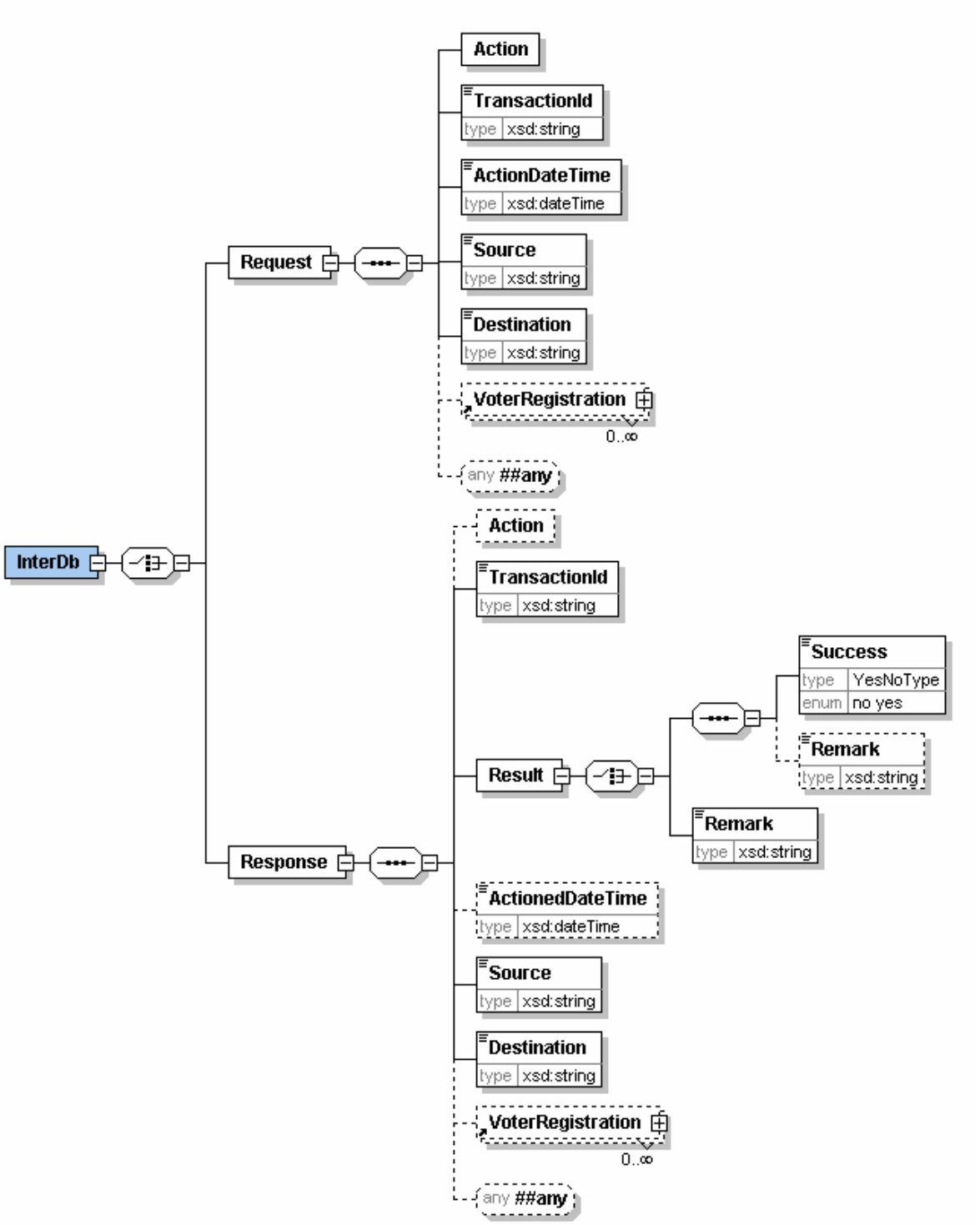


This schema is used for messages registering voters. It uses the `VoterIdentificationStructure` described in section 3.1.2.15, with the exception that no `VToken` is allowed. The `VoterInformationStructure` is used unchanged.

There is the facility to add a proof of ID and for the transmission channel (for example a trusted web site) to add a timestamp.

This schema allows any additional data to be added to the message for appropriate local extensions.

3.7 320 - Inter Database Communications

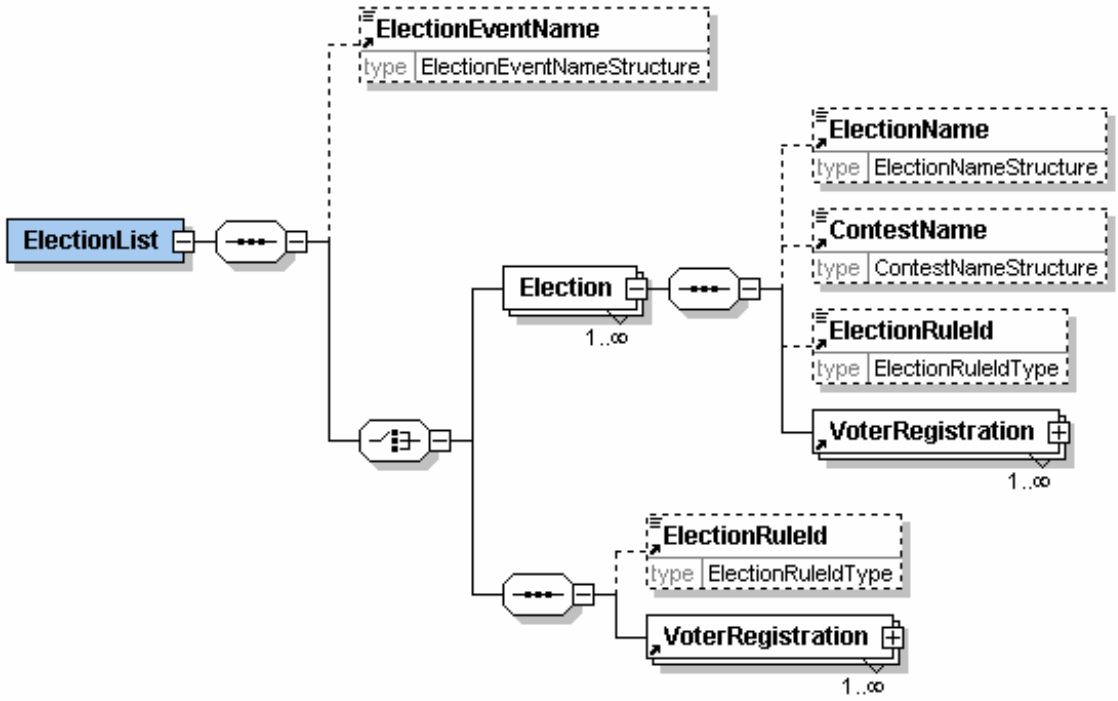


This schema is used for messages requesting services from other electoral list databases. This can, for example, be used to de-dupe databases. The schema is in two parts, so a message will be either a request or a response.

A request starts with an `Action` code and a `TransactionId` that can be used to correlate the response with the original request. The `ActionDateTime` is used to specify when the action should be carried out. The `Source` and `Destination` are used as identifiers *(should these be URIs?)*, and then there is an optional list of voters. The message can also be extended through the `xsd:any` element.

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

3.8 330 - Election List

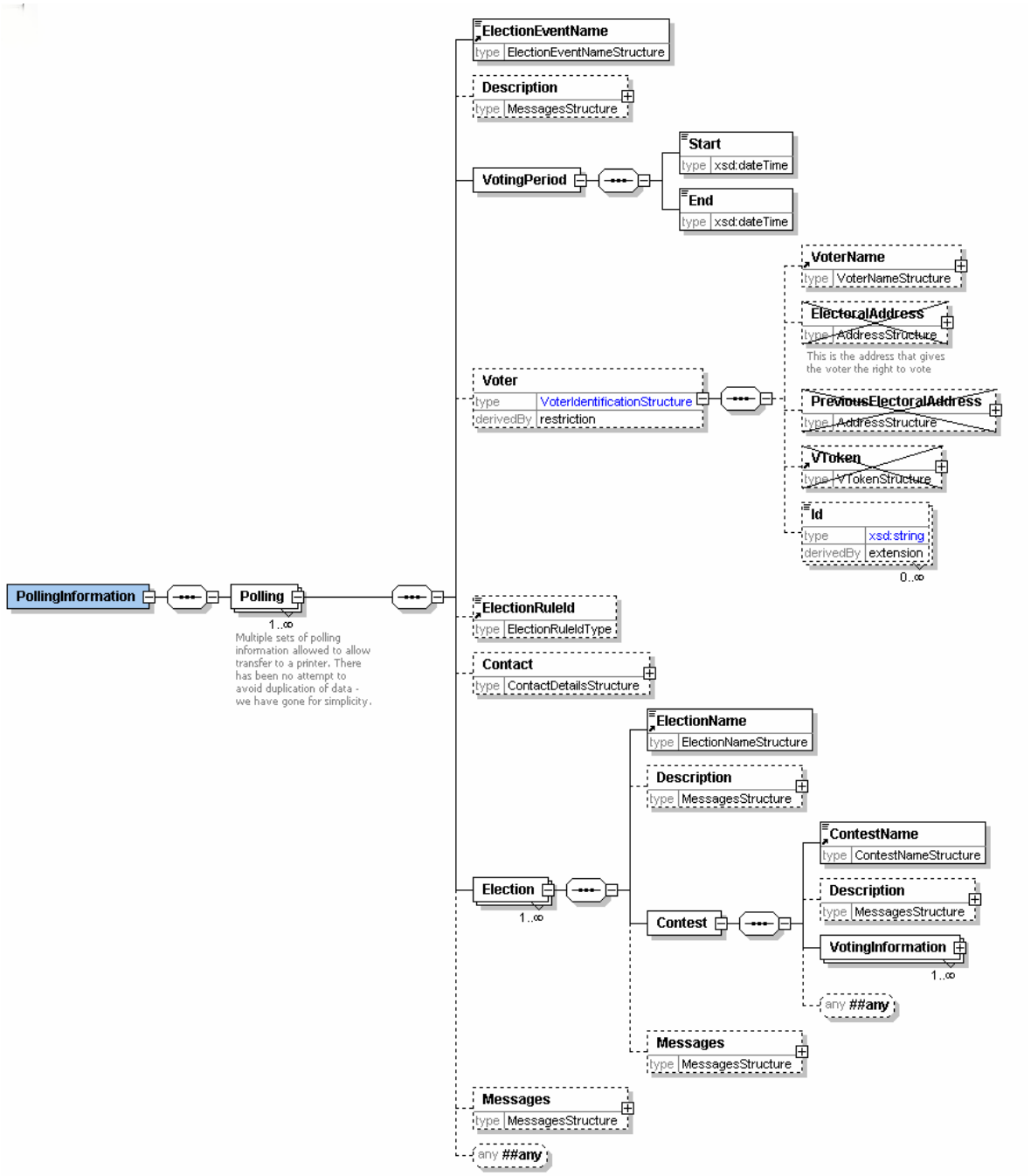


This schema is used for messages communicating the list of eligible voters for an election event or election within the event. This choice is allowed as frequently the same population will be able to vote at all elections within an event, but on other occasions the elections will have different lists.

One choice is therefore to send the election event name and ID, followed by an election rule ID and a list of voter registrations. The election rule indicates which voters in the register will be able to vote in this election event.

The other choice is to indicate the election, and optionally an individual contest, to which the voter list applies.

3.9 340 - Polling Information



The polling information messages defined by this schema are sent to voters to provide them with details of how to vote. It can also be sent to a distributor, so multiple sets of information are allowed.

One set of polling information may be sent to each voter for any election event, so the election event name is included, with the polling start and end time. Some information about the voter may be included, for example to print on a polling card.

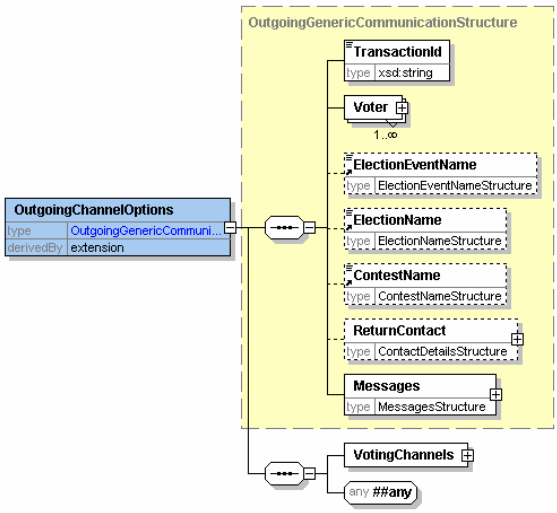
The `ElectionRuleId` can be included, and contact information for the benefit of a distributor.

Information about the elections and contests is included for the benefit of the voter, and further messages might be added. Use of the `DisplayOrder` attribute on these allows the display or printing of information to be tailored from within the XML message.

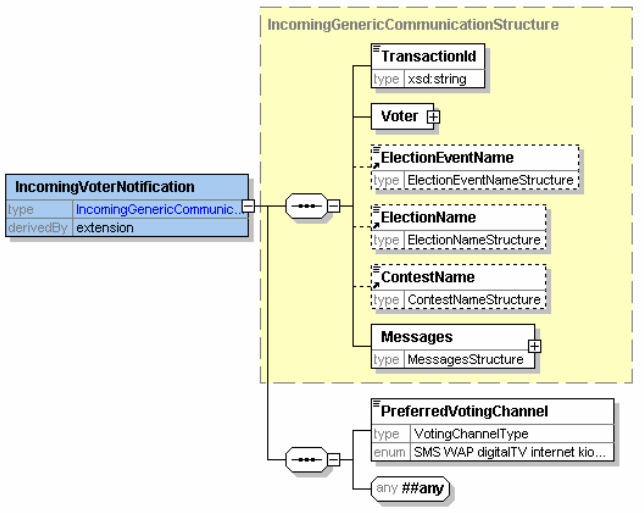
3.10 350 - Generic Communication

These two schemas (350a and 350b) extend the two corresponding data types by allowing any additional element to be appended.

3.11 360 - Channel Options



360a - Outgoing



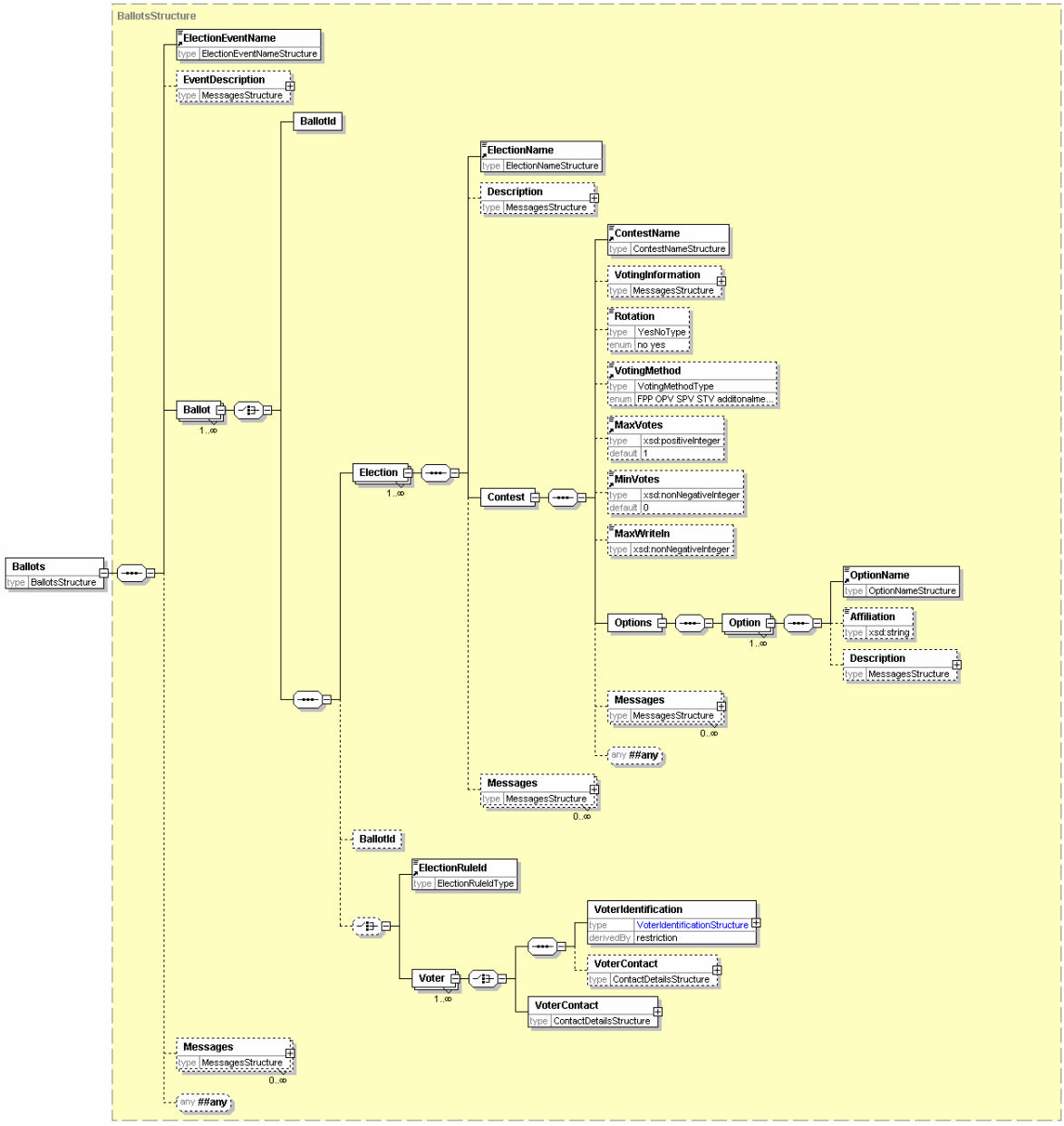
360b - Incoming

These two schemas are used for messages offering a set of voting channels to the voter and to indicate a preferred channel. 360b may be sent as an unsolicited message if this is supported within the relevant jurisdiction.

Both are extensions of the corresponding generic communications data type. The outgoing message includes a list of allowed channels, and the incoming provides a single channel.

Either message can be extended in the normal way.

3.12 410 - Ballots



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.

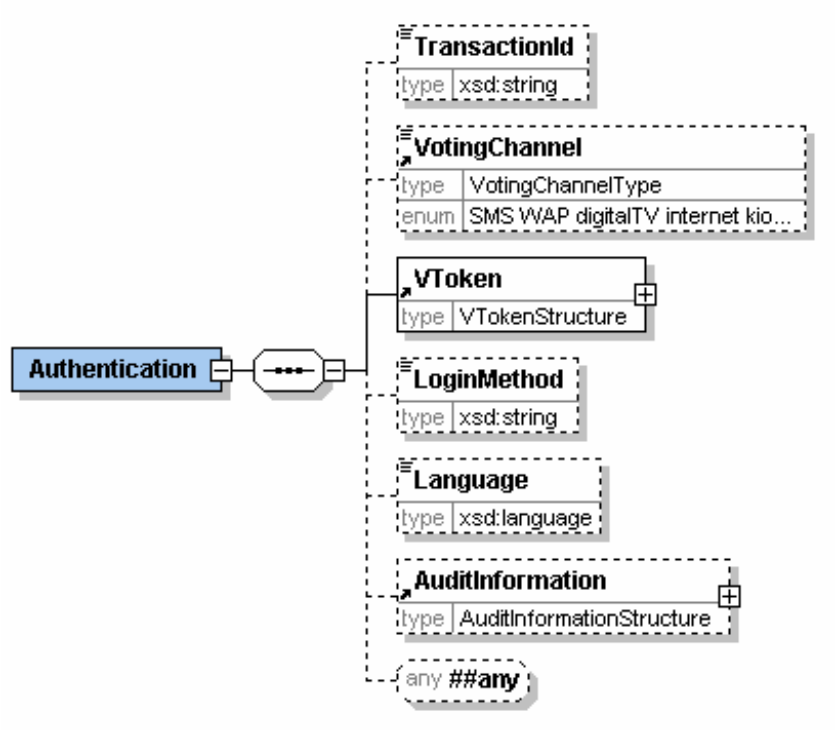
In other cases, the full information about the elections will be sent with either an election rule ID to identify the voters to whom that election applies or a set of voter names and contact information. If the ballot is being sent directly to the voter, this information is not required.

The election information starts with the election name 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 name and ID and a list of options for which the voter can vote. There is also a set of optional information that will be required in some circumstances.

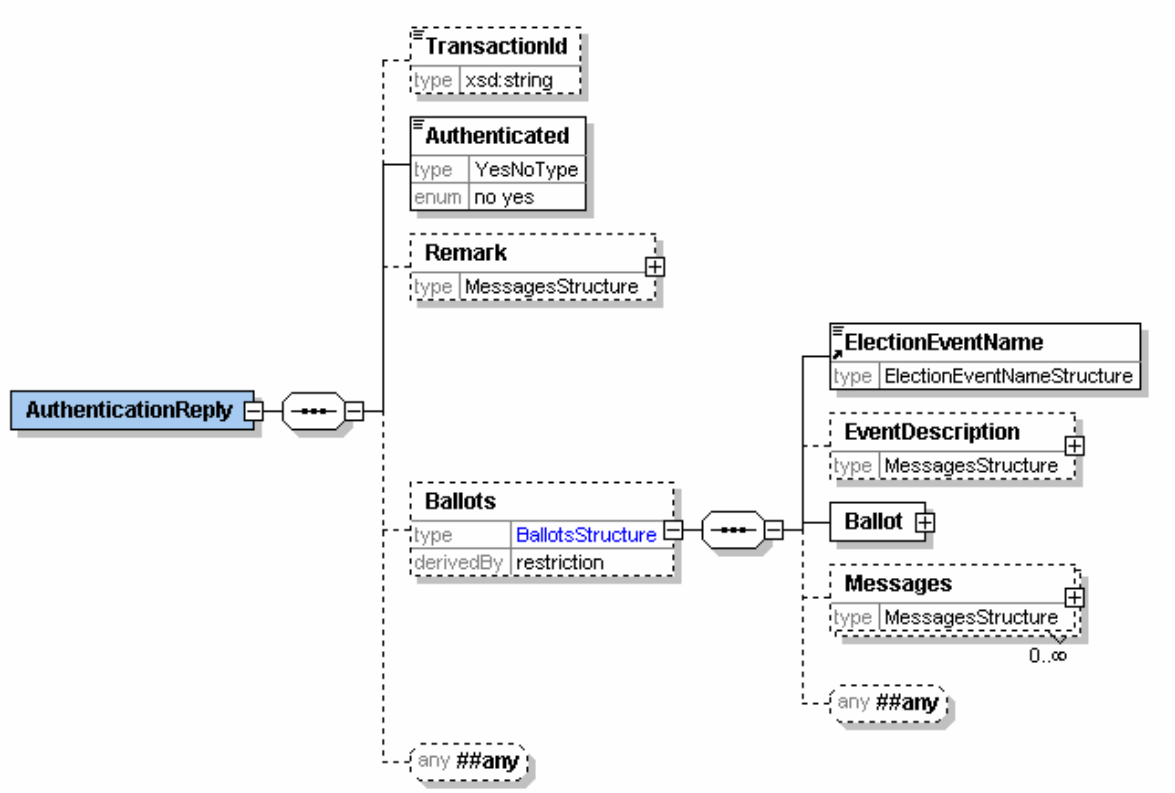
Some of this is for display to the voter (`VotingInformation` and `Messages`) and some controls the ballot and voting process (`Rotation`, `VotingMethod`, `MaxVotes`, `MinVotes`, `MaxWriteIn`).

3.13 420 - Authentication



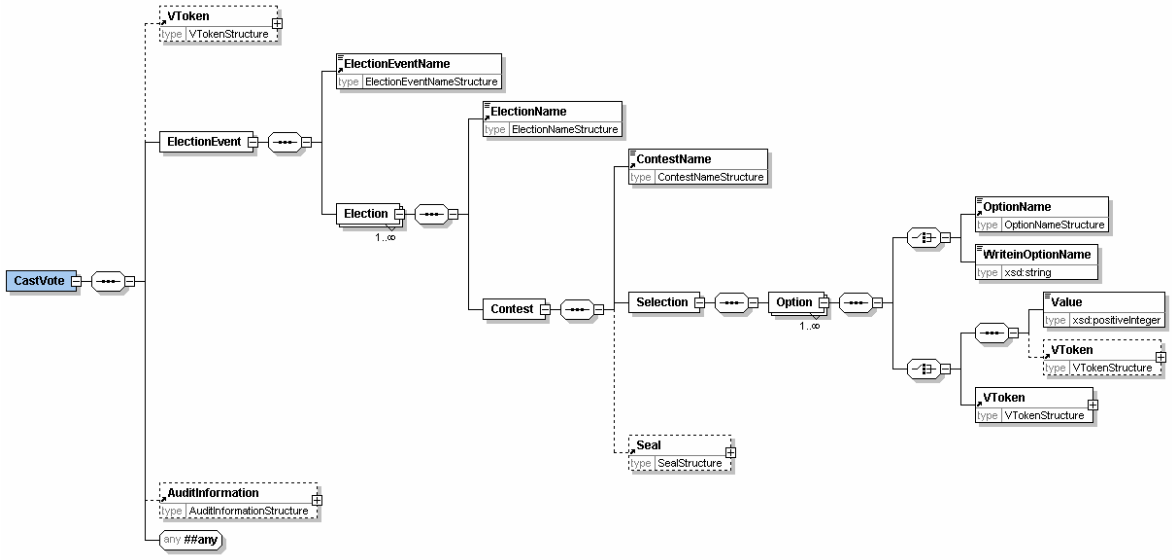
The authentication message defined by this schema may be used to authenticate a user during the voting process.

3.14 430 - Authentication Reply



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

3.15 440 - Cast Vote



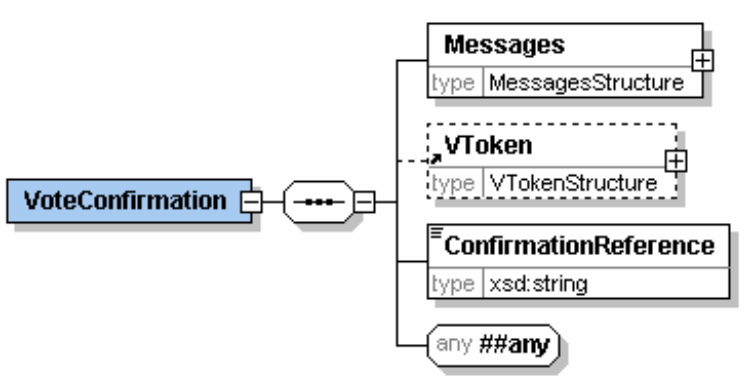
This message represents a cast vote, which comprises an optional VToken to ensure authorisation, information about the votes themselves and a set of optional audit information.

The election event is identified, together with a set of elections (if multiple elections were included on the same ballot). 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.

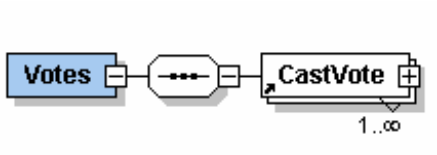
For each contest, one or more options is listed. For each of these, either the option name and ID is provided or a write-in option name for elections where this is allowed. This is accompanied by the value of the vote for that option, with an optional VToken. In some elections where it is only possible to vote for a single candidate, different VTokens may be provided for each option. In this case, only the VToken is required.

3.16 450 - Vote Confirmation



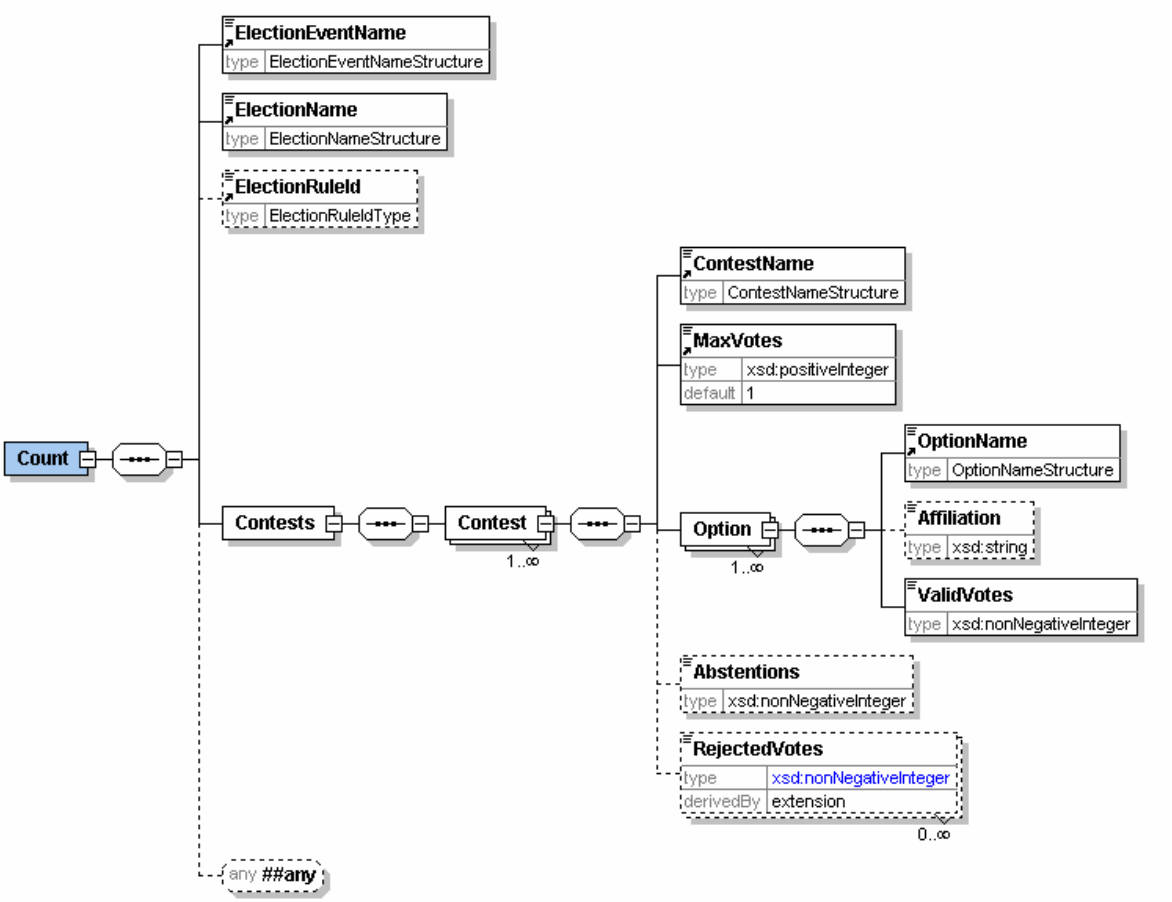
The vote confirmation message can be used to show that a vote has been accepted and provide a reference number in case of future queries. Display information can also be provided as well as additional structured information using `xsd:any`.

3.17 460 - Votes



This schema is used to define a message comprising a set of votes being transferred for counting. It is simply a set of `CastVote` elements from schema 440.

3.18 510 - Count



The count message defined by this schema is used to communicate the results of the set of contests that makes up an election.

The message therefore includes the election event name and ID, the election ID, a reference to the election rule being used and information concerning the set of contests.

Each contest indicates its name and ID, the maximum number of votes that each voter could cast, information about the votes cast for each option and the numbers of abstentions and rejected votes. The `RejectedVotes` element has `Reason` (optional) and `ReasonCode` (mandatory) attributes to indicate why the votes were rejected. The former is a textual description, and the latter a code.

For each option, the name, ID and number of valid votes is mandatory. These are optionally supplemented by an affiliation when the option is a (human) candidate.

4 References

1 Voting Process and EML (v1.0b) *Election and Voter Services Technical Committee* OASIS *1 March 2002*

2 EML Scenarios (v0.1) *Election and Voter Services Committee* OASIS *7 March 2002*

3 eXtensible Name and Address (XNAL) Specifications and Description Document (v1.0) *Customer Information Quality Technical Committee* OASIS *8 May 2001* http://www.oasis-open.org/committees/ciq/xnal/xnal_spec.zip

4 Extensible Markup Language (XML) 1.0 (Second Edition) *Tim Bray et al* Worldwide Web Consortium *6 October 2000* http://www.w3.org/TR/REC-xml

5 XML Linking Language (XLink) (v1.0) *Steve DeRose et al* Worldwide Web Consortium *27 June 2001* http://www.w3.org/TR/xlink/

6 XML-Signature Syntax and Processing *Donald Eastlake et al* Worldwide Web Consortium *12 February 2002* http://www.w3.org/TR/xmldsig-core/

7 Voice Extensible Markup Language (VoiceXML) Version 2.0 *Scott McGlashan et al* Worldwide Web Consortium *23 October 2001* http://www.w3.org/TR/voicexml20

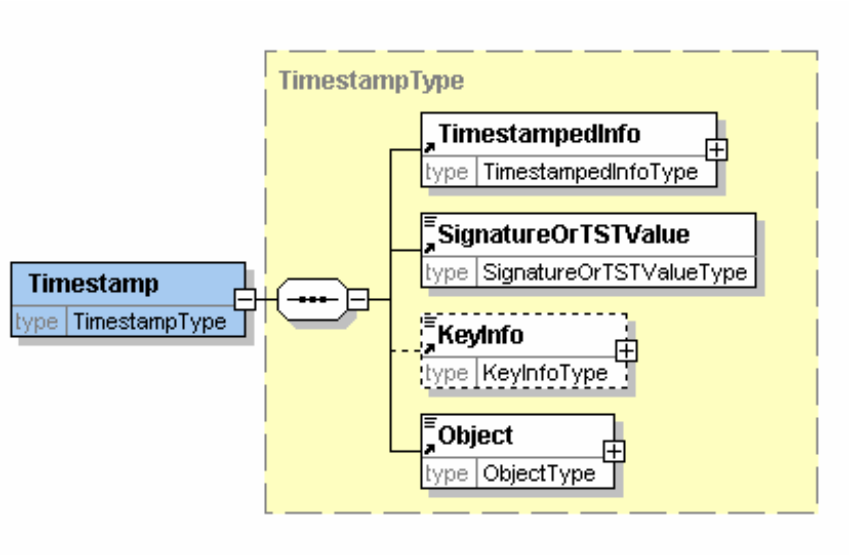
8 XML Schema Part 2: Datatypes *Paul V Biron et al* Worldwide Web Consortium 2 May 2001 http://www.w3.org/TR/xmlschema-2/

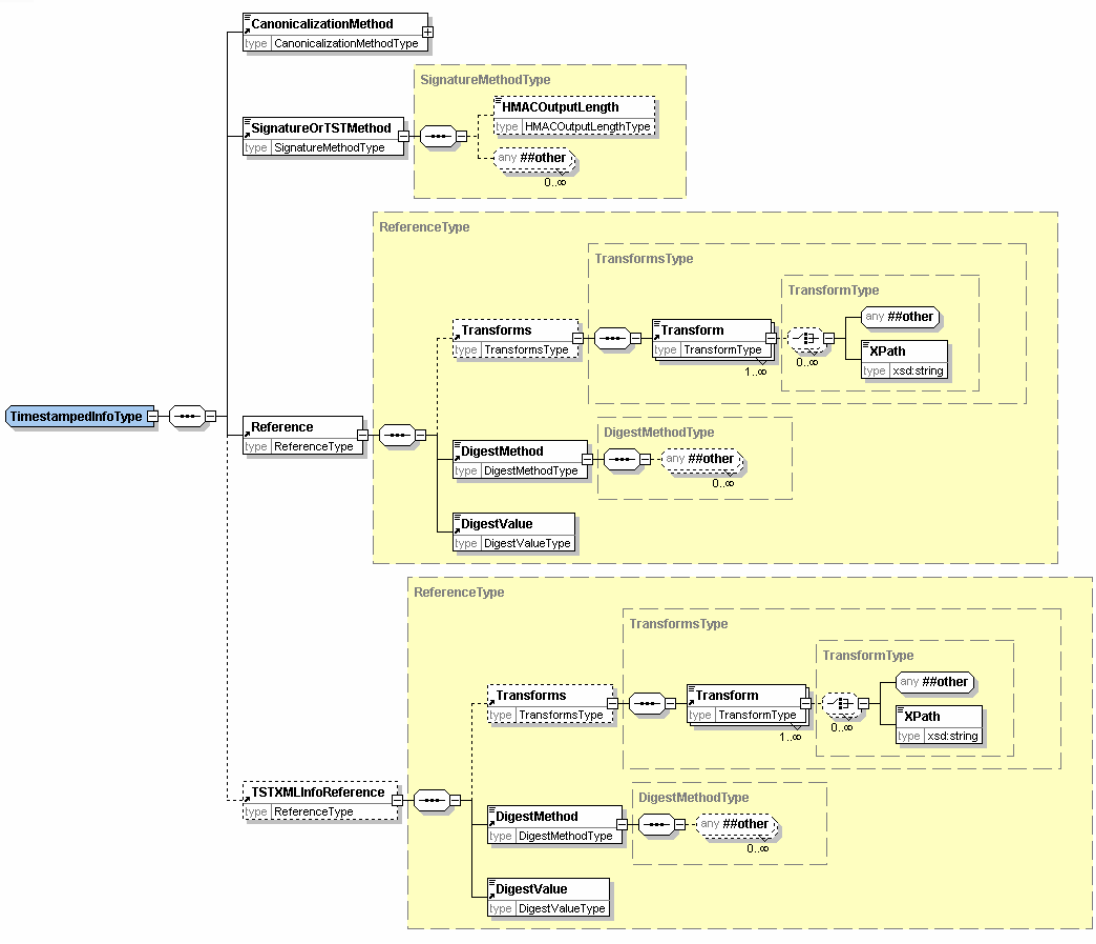
Appendix A. The Timestamp Schema

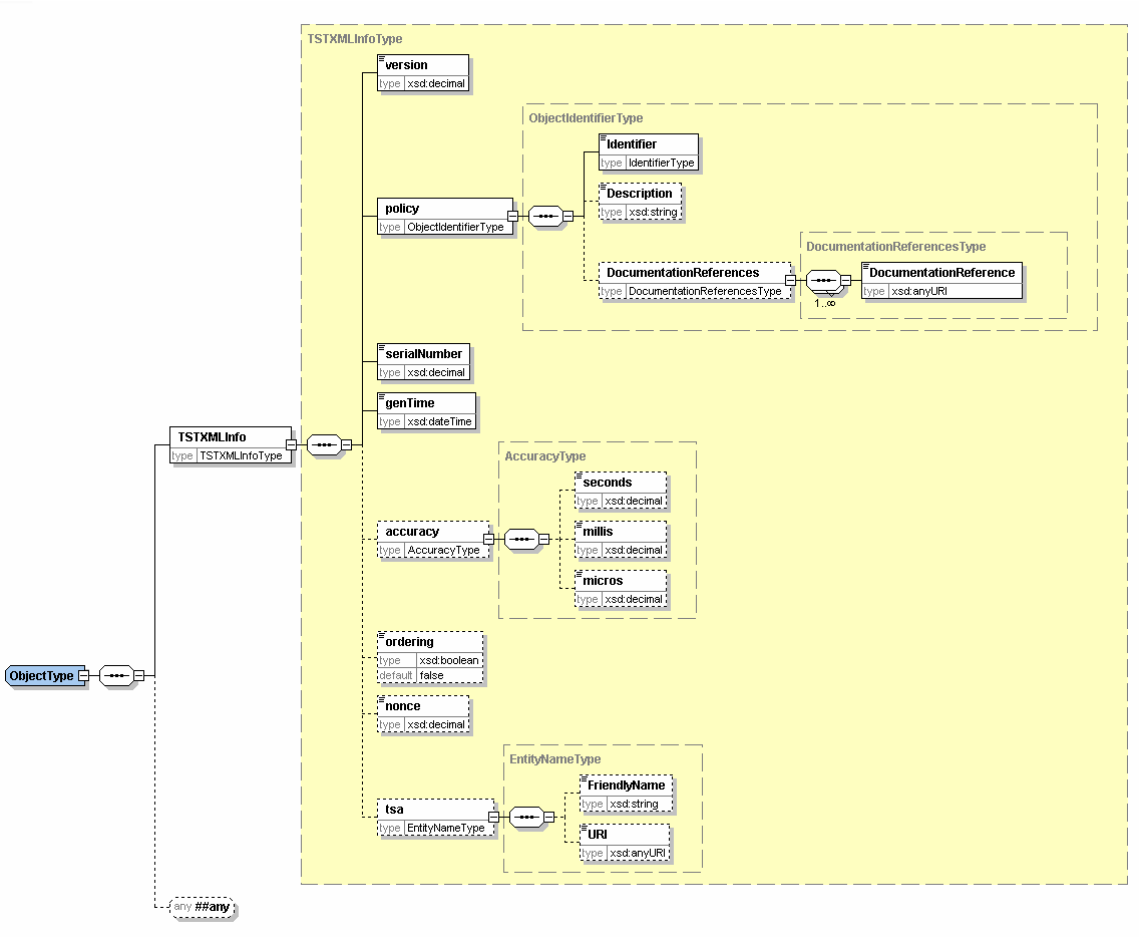
Although used as part of EML, this schema has been put in a separate namespace as it is not an integral part of the language.

A time-stamp binds a date and time to the sealed data. The time-stamp seal also protects the integrity of the data.

The structure of the time-stamp is similar to the structure of an XML Signature. The structure of the `Timestamp` element is shown here, followed by the detail of two of the four data types that are used to define its child elements.




TimestampedInfoType
CanonicalizationMethod
type CanonicalizationMethodType
SignatureOrTSTMethod
type SignatureMethodType
SignatureMethodType
HMACOutputLength
type HMACOutputLengthType
any ##other
0..∞
Reference
type ReferenceType
ReferenceType
Transforms
type TransformsType
TransformsType
Transform
type TransformType
1..∞
TransformType
any ##other
0..∞
XPath
type xsd:string
DigestMethod
type DigestMethodType
DigestMethodType
any ##other
0..∞
DigestValue
type DigestValueType
TSTXMLInfoReference
type ReferenceType
ReferenceType
Transforms
type TransformsType
TransformsType
Transform
type TransformType
1..∞
TransformType
any ##other
0..∞
XPath
type xsd:string
DigestMethod
type DigestMethodType
DigestMethodType
any ##other
0..∞
DigestValue
type DigestValueType


ObjectType
TSTXMLInfo
type TSTXMLInfoType
TSTXMLInfoType
version
type xsd:decimal
policy
type ObjectIdentifierType
ObjectIdentifierType
Identifier
type IdentifierType
Description
type xsd:string
DocumentationReferences
type DocumentationReferencesType
DocumentationReferencesType
DocumentationReference
type xsd:anyURI
1..∞
serialNumber
type xsd:decimal
genTime
type xsd:dateTime
accuracy
type AccuracyType
AccuracyType
seconds
type xsd:decimal
millis
type xsd:decimal
micros
type xsd:decimal
ordering
type xsd:boolean
default false
nonce
type xsd:decimal
tsa
type EntityNameType
EntityNameType
FriendlyName
type xsd:string
URI
type xsd:anyURI
any ##any

The timestamp structure may be used in one of two ways either:

a) Using Internet RFC 3161 binary encoded time-stamp token with the time-stamp information repeated in XML,
b) Using a pure XML encoded time-stamp.

In the case of the RFC 3161 based time-stamp, the Timestamp structure is used as follows:

- within `TimestampedInfo`:
  - `TSTOrSignatureMethod` identifies RFC 3161.
  - `Reference` contains the URI reference of the voting data being time-stamped. The `DigestValue` sub element contains the digest of the voting data being time-stamped.
  - `TSTXMLInfoReference` is not present in this case.
- `SignatureOrTSTValue` holds the RFC 3161 time-stamp token applied to the digest of `TimestampedInfo`. The `TimestampedInfo` is transformed to a canonical form using the method identified in `CanonicalizationMethod` before the digest algorithm is applied.
- `KeyInfo` contains any relevant certificate or key information.
- `Object` contains the `TSTXMLInfo` element which is a copy of the information in `SignatureOrTSTValue` converted from RFC 3161 to XML encoding. The `TSTXMLInfo` element contains:
  - version of time-stamp token format. This would be set to version 1
  - the time-stamping policy applied by the authority issuing the time-stamp,
  - the time-stamp token serial number,
  - the time that the token was issued, the contents of this element indicate the time of the timestamp.
  - optionally an indication as to whether the time-stamps are always issued in the order that requests are received
  - optionally a nonce[1] given in the request for the time-stamp token,
  - optionally the identity of the time-stamping authority

In the case of a pure XML encoded time-stamp, the Timestamp structure is used as follows:

- within `TimestampedInfo`,
  - `TSTOrSignatureMethod` identifies the algorithm used to create the signature value.
  - `Reference` contains the URI reference of the voting data being time-stamped. The `DigestValue` sub element contains the digest of the voting data being time-stamped.
  - `TSTXMLInfoReference` must be present, and contains the URI reference of `TSTXMLInfo` as contained within the `Object` element. The `DigestValue` sub element contains the digest of the `TSTXMLInfo`.
- `SignatureOrTSTValue` contains the signature value calculated over the `TimestampedInfo` using the signature algorithm identified in `TSTOrSignatureMethod` having been transformed to a canonical form using the method identified in `CanonicalizationMethod`. This signature is created by the time-stamping authority.
- `KeyInfo` contains any relevant certificate or key information.

[1] A nonce is a parameter that varies over time and is used as a defence against a replay attack.

- `Object` contains the XML encoded time-stamp information in an TSTXMLInfo element. The contents of TSTXMLInfo is the simular as for the case described above. However, in this case the information is directly signed by the time-stamping authority. The `TSTXMLInfo` element contains:
    - version of time-stamp token format: This would be set to version 2
    - the time-stamping policy applied by the authority issuing the time-stamp,
    - the time-stamp token serial number,
    - the time that the token was issued, this is the time of the timestamp.
    - optionally an indication as to whether the time-stamps are always issued in the order that requests were received
    - optionally a nonce given in the request for the time-stamp token,
    - optionally the identity of the time-stamping authority

Appendix B. W3C XML Digital Signature

Some information on the digital signature is included here, but for full information refer to the Recommendation at [6].

An XML Signature consists of:

- `SignedInfo` which includes a sequence of references to the data being signed with the digest (eg. SHA-1 hash) of the data being signed
- `SignatureValue` which contains the signature value calculated over the SignedInfo using the signature algorithm identified in `SignatureMethod` having been transformed to a canonical form using the method identified in `CanonicalizationMethod`
- `KeyInfo` contains any relevant certificate or key information.
- `Object` can contain any other information relevant to the signature