DRAFT NIST Special Publication 1500-10x

Cast Vote Records Common Data Format Specification

Version 1.0

List of authors

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.1500-103

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication 1500-10x

Cast Vote Records Common Data Format Specification

Version 1.0

List of authors

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.1500-103

April 2018

U. S. Department of Commerce
Wilbur Ross, Secretary

National Institute of Standards and Technology
Walter Copan, Director

National Institute of Standards and Technology (NIST) Special Publication 1500-10x
71 pages (June 2018)

National Institute of Standards and Technology
Attn: Software and Systems Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8970) Gaithersburg, MD 20899-8930
Email: voting@nist.gov

Reports on Computer Systems Technology

Abstract

This document is a specification for a common data format for cast vote records (CVR) produced by vote-capture devices such as ballot scanners. It supports the interoperable export of CVRs from these devices and the interoperable import and export of CVRs to/from election management systems, adjudication systems, and audit systems. The specification includes examples of XML and JSON usage.

Keywords

Common data format (CDF); cast vote record (CVR); JavaScript Object Notation (JSON); unified markup language (UML); voluntary voting system guidelines (VVSG); eXtensible Markup Language (XML).

Acknowledgements

The editor wishes to thank his colleagues of the National Institute of Standards and Technology Voter Records Interchange Public Working Group, who contributed to its technical content. The editor gratefully acknowledges and appreciates the following contributors for their keen and insightful assistance with developing this specification:

In addition to the above acknowledgments, the editor also gratefully acknowledges and appreciate the significant contributions from individuals and organizations involved in the NIST Voting Interoperability Public Working Group as well as in the public and private sectors, whose thoughtful and constructive comments improved the overall quality, thoroughness, and usefulness of this publication.

Executive Summary

This document presents an interoperable, common data format specification for cast vote records (CVR), which are produced by vote-capture devices such as ballot scanners. A CVR is an electronic record of a voter's choices, sometimes with additional information such as whether the vote-capture device changed any of the voter's choices due to election rules or because some voter choices were unreadable. Election results are produced by tabulating the collection of CVRs, and audits can be done by comparisons of the paper records against the CVRs.

This specification supports 3 general use cases for CVRs:

1. Interoperable exports of CVRs from devices such as scanners for import into tabulators, election management systems (EMS), or auditing systems.
2. Interoperable exports of aggregated collections of CVRs from aggregating devices such as election management systems.
3. Update of CVRs after adjudication.

The purpose of this specification is to provide an interoperable, non-proprietary data exchange format in XML and JSON for CVRs so as to promote greater transparency to voting records produced by vote-capture devices, and to facilitate the exchange of CVRs with other devices that operate upon CVRs regardless of device manufacturer.

The specification includes a UML (Unified Markup Language) model and references XML (eXtensible Markup Language) and JSON (JavaScript Object Notation) schemas that were generated from the UML model.

This specification is geared towards the following audiences:

- Election officials;
- Voting equipment manufacturers;
- Election analysts and auditors;
- Election-affiliated organizations and the public.

Table of Contents

List of Appendices

1 Introduction

This document is a specification for a common data format (CDF) for cast vote records (CVR) produced by vote-capture devices such as ballot scanners and subsequently tabulated, adjudicated, and audited by other voting devices such as election management systems (EMS). The specification also describes XML (eXtensible Markup Language) [1] and JSON (JavaScript Object Notation) [2] schemas that were generated from a UML model.

Primary features of this specification include:

- Capability to export raw CVRs representing voter choices as well as other information such as changes made to the CVRs by the vote-capturing device because of election/contest rules and changes made to the CVRs as a result of adjudication.
- Capability, for a single election, to contain aggregated collections of CVRs produced by multiple devices from multiple locations, e.g., voting centers or precincts.
- A data model in UML (Unified Modeling Language) [5] that itemizes and defines the data involved in CVRs and that is used to derive the XML and JSON schemas.

1.1 Purpose

The purpose of this specification is to provide interoperable data interchange formats in XML and JSON for CVRs so as to assist election officials, auditors, and other election analysts in collecting and aggregating CVRs from multiple types of vote-capture devices, and tabulation or audit of the CVRs by multiple types of devices, potentially from different manufacturers. An additional purpose of this specification is to provide greater transparency to CVRs and operations performed on them. Advantages of using this specification include:

- Interoperable data interchange formats for CVRs to remove reliance on proprietary data formats.
- Capability to utilize the same interoperable format for CVR creation, analysis and update, tabulation, adjudication, and audit.
- Greater freedom to use devices from different manufacturers for operations involving CVRs.
- Consistent handling of voting variations such as ranked choice voting (RCV).
- A UML model that is easily extensible to additional use cases.

1.2 Audience

The intended audience of this specification includes election officials, voting system designers and developers, as well as others in the election community including the general public. Some background in election administration and voting equipment is useful in understanding the material in this specification.

1.3 Document Structure

This specification is laid out as follows:

- Section 2 contains background information regarding how CVRs get generated, their contents, and how they are handled in the election process.
- Section 3 contains an overview of the UML model structure and how it can be used for CVR exports and reports.
- Section 4 describes the classes and enumerations in greater detail.
- Section 5 contains examples of CVRs using JSON and XML.

Appendices contain acronyms, definitions, references, and URLs for downloading the associated JSON and XML schemas.

1.4 Motivation and Methodology

This document was motivated primarily to assist election officials and developers and auditors in handling CVRs as they are generated and utilized. The current varying systems involved and the data produced generally do not interoperate unless they are from the same manufacturer, which adds more complexity when using other equipment for tabulations, adjudications, and audits. NIST and a community of U.S. election officials, analysts, and election system technologists analyzed how CVRs are produced and used in the election process, so as to produce an interoperable CVR format that can be used regardless of manufacturer. This specification implements the following use cases:

1. Interoperable exports of CVRs from devices such as scanners for import into tabulators, election management systems (EMS), or auditing systems.
2. Interoperable exports of aggregated collections of CVRs from aggregating devices such as election management systems.
3. Update of CVRs because of adjudication.

JSON and XML schemas were generated from the UML model, such that scanners and other devices can export CVRs in JSON or XML and validate usage against the schema.

2 Background on Cast Vote Records Generation and Use Cases

The section contains a general overview of how CVRs get generated, their contents, and how they are subsequently handled in the election process. With that as background, an overview of the UML model structure follows, showing how the model can be used for reports of cast vote records.

2.1 Overview of Cast Vote Records and their Generation

A cast vote record (CVR) is, simply put, an electronic record of a voter's ballot choices. A CVR is generated by equipment that a voter uses, generally at the polling place, to vote in an election. CVRs are also generated by other equipment used to scan absentee or other types of ballots that are collected before the election or that cannot be scanned by polling place equipment for various reasons. After the polls are closed, the CVRs are collected by election officials, often on memory devices, and subsequently counted at a central location. The primary purpose, then, of a CVR is to provide an artifact of voter choices that can be counted in an efficient manner so as to produce election results.

The voting equipment can be thought of, then, as digitizers that provide an electronic record of the voter's choices so that they can be counted accurately and efficiently, given that counting paper ballots by hand is slower and inefficient. There are three primary types of voting devices that generate CVRs:

- All-electronic, touch screen voting devices that a voter uses to make ballot choices and that generate and store a CVR for each ballot.
- Ballot marking devices (BMDs) that function like all-electronic touch-screen devices but that produce a paper record of the voter's choices that must be subsequently scanned.
- Precinct-count optical scanners (PCOS) used in polling places and central-count optical scanners (CCOS) used in central offices to scan paper ballots.

The equipment above are often referred to collectively as "tabulators" because they generally have some tabulation capability. For example, a scanner may scan thousands of ballots over the course of a day and may generate a final tabulated report that shows how many votes each candidate/ballot question selection received.

CVRs may include other information besides voter choices, including the following:

- Information on all contests and ballot selections on the ballot in addition to those marked.
- The ballot style associated with the CVR.
- The precinct or location associated with the CVR.
- The equipment that produced the CVR.
- The political party associated with the ballot for partisan primaries.
- Images of the entire ballot and images of write-in areas on the ballot.
- An identifier that is also impressed on the ballot as it is scanned.
- Indications of how the scanner has interpreted various marks.

This specification includes support for the above items.

2.2 Counting Cast Vote Records

To produce a CVR that is countable, the equipment must also interpret the voter's choices according to the rules of each contest so as to determine which choices can be counted. This is true primarily of hand-marked paper ballot scanners, whereas all-electronic devices and BMDs essentially guide the voter how to make choices according to the contest rules, thus the CVRs they generate are generally countable, excepting write-ins.

Scanners must first interpret the ballot and detect where voters have made marks and whether those marks meet manufacturer-specific criteria for validity, i.e., whether a mark is placed in the right location and is sufficiently solid so as to constitute an intentional ballot choice made by the voter. Each device in the jurisdiction must also be "programmed" with the election specific information for the polling place that it will be used in so that the scanners can apply election rules to the detected marks. Thus, scanners may perform interpretation based on a number of different factors, including:

- A scanner may flag marks as being marginal, i.e., as not meeting the criteria for validity, and therefore not count those marks.
- Voters may vote for more than the allowable number of choices (overvote) and the scanner must know not to count any of the choices made by the voter for that particular contest.
- Likewise, voters may undervote a contest, and the scanner must record that the contest was undervoted.
- Depending on the voting method in place for a particular contest, the scanner must "know" how to interpret the voter marks in the context of the voting method.
- A scanner may generate voter marks in the case of straight party voting where a voter can decide to vote for all candidates of a particular party by making a single straight-party selection at the top of the ballot.
- A scanner may invalidate voter marks in the case of straight party voting where a voter selects the straight party choice but votes for the other party in various contests. Depending on local election rules, the votes in those contests or perhaps the entire ballot could be invalidated.

Typically, CVR-generating equipment will export a collection of CVRs that will include some tabulation of the contests on the ballots. This CVR collection may be copied to a memory device or otherwise transferred to a central location, where it can be combined with other CVR exports so as to produce election results.

2.3 Adjudication of Cast Vote Records

After a CVR collection has been exported, a number of the CVRs may require additional inspection and adjustment as part of a process known as adjudication, which may be done on an election management system (EMS) by election officials. Write-ins are generally the most common reason: on ballots produced by BMDs the write-in names could be spelled differently or incorrectly, and for scanned paper ballots, either the ballots themselves must be inspected or the images of the write-in areas of the ballot that were made by the scanner. There are a number of other reasons why ballots may require adjudication, such as

- The ballot was unreadable by the scanner.
- The voter may have marked the ballot in ways that are difficult to interpret, e.g., the voter may have circled the ovals instead of filling in the ovals.
- The marks made by the voter were borderline marginal.

The CVR itself does not necessarily need to be updated or rewritten as a result of adjudication, however the capability to update the CVR offers more convenience in that the complete collection could be retabulated as necessary, e.g., for recounts or audits. This specification provides the capability to update the CVR with multiple annotations made by adjudicators, recording the following items:

- The adjudicator name(s).
- Time stamp of when the adjudication(s) was made.
- The adjudication, i.e., the action taken by the adjudicator(s).
- The status of the item being adjudication prior to the adjudication(s).

This is discussed in more detail in Section 4.1.

2.4 Auditing Cast Vote Records

CVRs need to be audited against the paper ballots so that the election results can be verified to be accurate. There are a variety of methods for auditing, and the CVR should be able to accommodate these methods, that is, contain information that is needed for these methods.

Sections 2.1 and 2.2 list various items that should be part of a CVR, such as its corresponding ballot style, precinct, indications of interpretations, and so forth, which are needed for various aspects of audits. In general, this specification supports auditing by providing the following:

- Support for ballot-level auditing, i.e., there is an identifier that can be linked to an ID on the corresponding paper ballot provided the scanner creating the CVR can impress an identifier on the ballot as it is scanned.
- Support to include adjustments to vote selections made by adjudicators and associated history.
- If the scanner does post-processing of the ballot selections, indications of what actions the scanner took, e.g., "voter selection overridden by contest rules".
- Indications of marginal marks, mark quality/density (if scanner is capable).
- A CVR can include signed/hashed references to an associated image of the ballot or images of write-ins made by the voter.
- Improved handling of multi-sheet ballots so that one can more accurately count the number of multi-sheet ballots scanned even if page 1 or any other page is missing.

- Capability to include batch IDs or other batch-related information.

.

3 Cast Vote Record UML Model Overview

This section presents an overview of the CVR UML model, showing how it is structured and how it can be used for various voting methods. Section 4 contains more detailed information on specific classes and enumerations, and Section 5 contains several CVR examples using JSON and XML.

In general, the JSON and XML formats closely follow the structure of the UML model. Thus, the examples of CVR structure in this section apply also to JSON and XML. The UML model is shown in Figures 1 and 2.

3.1 CVR Report General Structure

The UML model implements a report of cast vote records exported by a device that

- Generates CVRs, such as a scanner or BMD, and/or
- Processes/Tabulates collections of CVR exports, such as an EMS.

The CVRs in a report can be organized in various ways, including according to an election, to the originating device, as well as to ballot style. The CVR reports could be imported into a central tabulator such as an EMS, and the tabulator could export, in the same file, an aggregated report consisting of collections of CVRs from different generating devices. Thus, in an election conducted in, say, a county, there could be many CVR reports, each specific to a scanner or BMD, and the county could import these reports into a central EMS and issue potentially one single aggregated report consisting of all the specific reports.

A central feature of the UML model concerns how it deals with the items that compose the CVR. The model treats contests, candidates and ballot measures, ballot selections, and voter marks all as separate objects that can be linked together in various ways so as to accommodate different types of voting methods. To form a CVR, the different objects are linked together to identify, for a given CVR, which contests and ballot selections were voted and the marks associated with the votes (as well as other marks such as marginal or made by the scanner according to contest rules or via adjudication).

To save on file size, the CVR report structure is arranged similarly to the NIST 1500-100 Election Results Reporting specification (https://www.nist.gov/itl/voting/nist-election-results-common-data-format-specification) in the following respect: the objects for all contests and candidates in the report are defined first, and then the individual CVRs in the report link to these definitions as needed. Thus, the exporting device must build the report by first identifying all contests that were voted and the associated candidates and ballot measures, and then defining objects for each of these items. As part of the process, duplicates are thrown out, so that each unique object is defined only once.

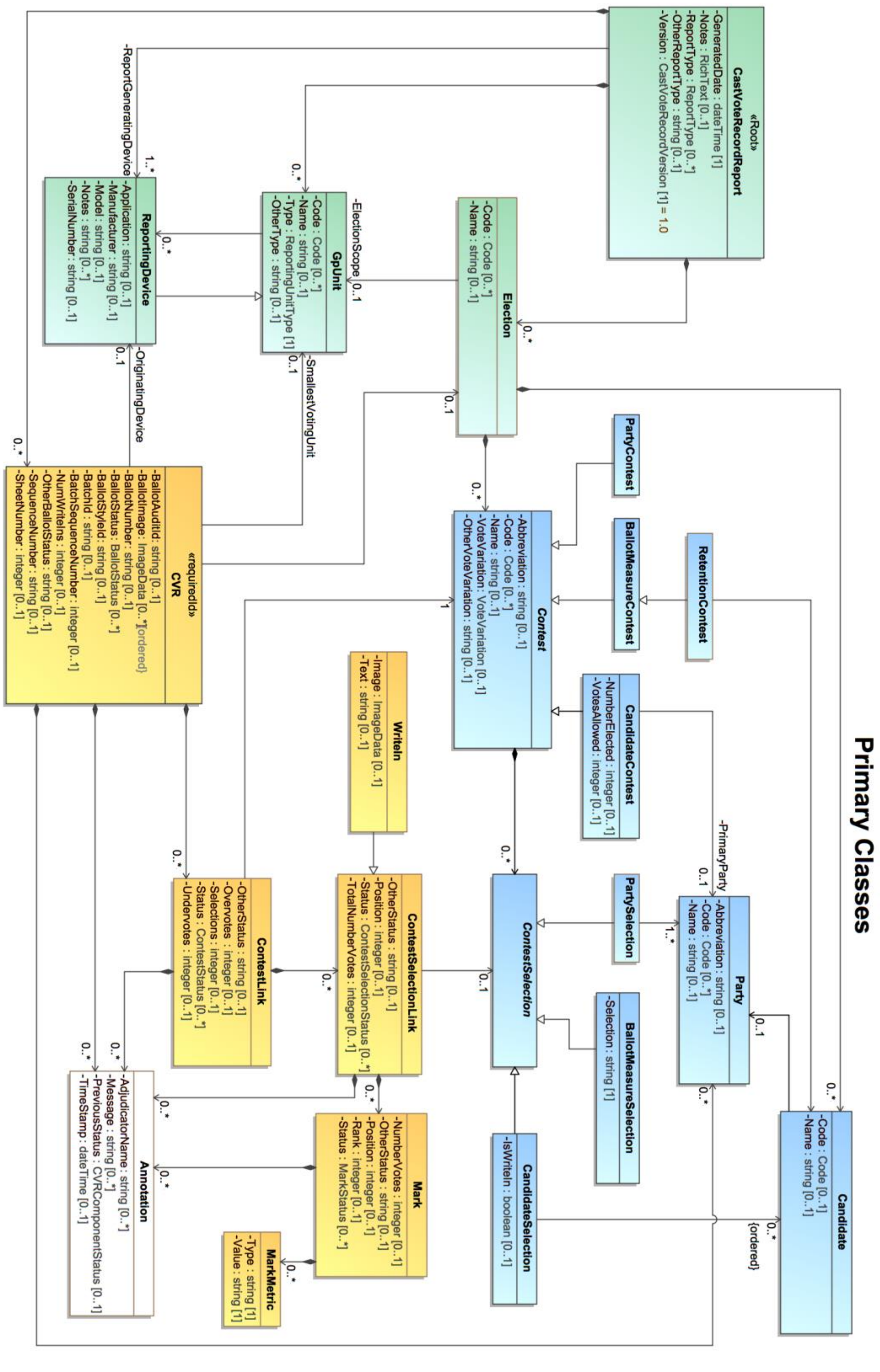

Primary Classes
«Root»
CastVoteRecordReport
-GeneratedDate : dateTime [1]
-Notes : RichText [0..1]
-ReportType : ReportType [0..*]
-OtherReportType : string [0..1]
-Version : CastVoteRecordVersion [1] = 1.0
Election
-Code : Code [0..*]
-Name : string [0..1]
GpUnit
-Code : Code [0..*]
-Name : string [0..1]
-Type : ReportingUnitType [1]
-OtherType : string [0..1]
ReportingDevice
-Application : string [0..1]
-Manufacturer : string [0..1]
-Model : string [0..1]
-Notes : string [0..*]
-SerialNumber : string [0..1]
«requiredId»
CVR
-BallotAuditId : string [0..1]
-BallotImage : ImageData [0..*]{ordered}
-BallotNumber : string [0..1]
-BallotStatus : BallotStatus [0..*]
-BallotStyleId : string [0..1]
-BatchId : string [0..1]
-BatchSequenceNumber : integer [0..1]
-NumWriteIns : integer [0..1]
-OtherBallotStatus : string [0..1]
-SequenceNumber : string [0..1]
-SheetNumber : integer [0..1]
Contest
-Abbreviation : string [0..1]
-Code : Code [0..*]
-Name : string [0..1]
-VoteVariation : VoteVariation [0..1]
-OtherVoteVariation : string [0..1]
PartyContest
BallotMeasureContest
RetentionContest
CandidateContest
-NumberElected : integer [0..1]
-VotesAllowed : integer [0..1]
WriteIn
-Image : ImageData [0..1]
-Text : string [0..1]
ContestSelection
PartySelection
BallotMeasureSelection
-Selection : string [1]
CandidateSelection
-IsWriteIn : boolean [0..1]
Party
-Abbreviation : string [0..1]
-Code : Code [0..*]
-Name : string [0..1]
Candidate
-Code : Code [0..1]
-Name : string [0..1]
ContestLink
-OtherStatus : string [0..1]
-Overvotes : integer [0..1]
-Selections : integer [0..1]
-Status : ContestStatus [0..*]
-Undervotes : integer [0..1]
ContestSelectionLink
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Status : ContestSelectionStatus [0..*]
-TotalNumberVotes : integer [0..1]
Mark
-NumberVotes : integer [0..1]
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Rank : integer [0..1]
-Status : MarkStatus [0..*]
MarkMetric
-Type : string [1]
-Value : string [1]
Annotation
-AdjudicatorName : string [0..*]
-Message : string [0..*]
-PreviousStatus : CVRComponentStatus [0..1]
-TimeStamp : dateTime [0..1]
-ReportGeneratingDevice
-ElectionScope
-OriginatingDevice
-SmallestVotingUnit
-PrimaryParty
{ordered}

Figure 1 - CVR UML Model - Classes

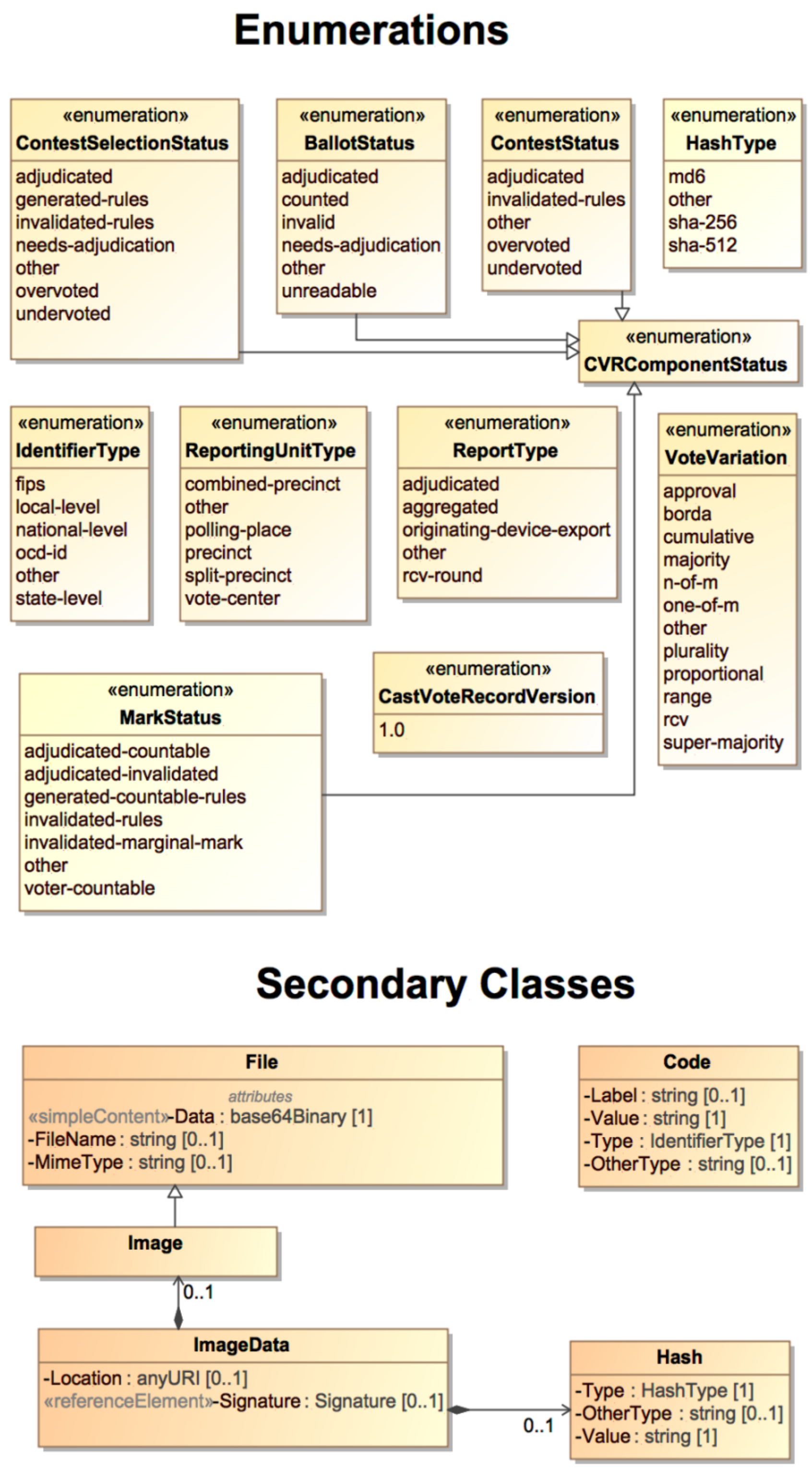
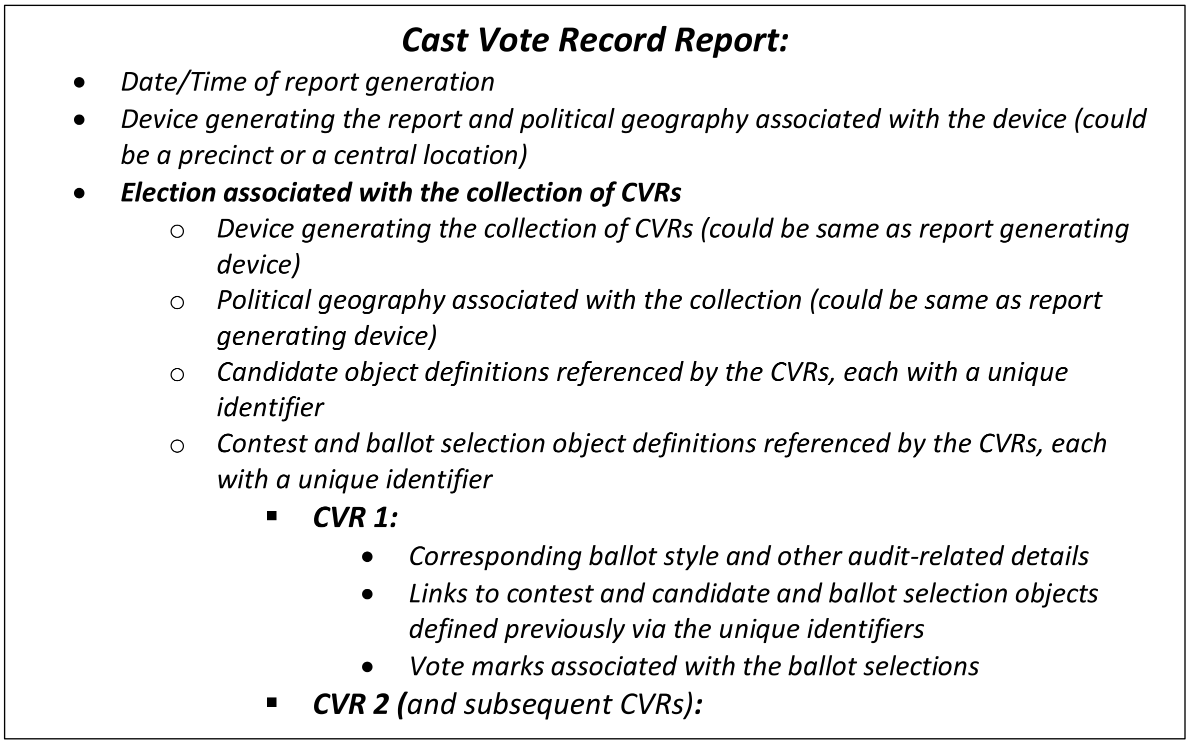


Figure 2 - CVR UML Model - Enumerations and Other Classes

The exporting device then creates CVRs by, for each CVR, linking the objects together as necessary to identify the contests and candidates and ballot selections specific to a CVR and that were voted, and then adding the vote marks where they are indicated. This will generally save file space over simply repeating each contest, etc., object definition for each CVR.

3.1.1 General Structure Examples

To clarify the CVR report structure, Figure 3 shows a basic example:



Cast Vote Record Report:

- *Date/Time of report generation*
- *Device generating the report and political geography associated with the device (could be a precinct or a central location)*
- ***Election associated with the collection of CVRs***
    - *Device generating the collection of CVRs (could be same as report generating device)*
    - *Political geography associated with the collection (could be same as report generating device)*
    - *Candidate object definitions referenced by the CVRs, each with a unique identifier*
    - *Contest and ballot selection object definitions referenced by the CVRs, each with a unique identifier*
        - ***CVR 1:***
            - *Corresponding ballot style and other audit-related details*
            - *Links to contest and candidate and ballot selection objects defined previously via the unique identifiers*
            - *Vote marks associated with the ballot selections*
        - ***CVR 2 (**and subsequent CVRs**):***

Figure 3 - CVR Report Basic Structure

As discussed in Section 2, a CVR can potentially include all contests and ballot selections regardless of whether they were voted, thus the exporting device would include all contests and ballot selections, indicating those that were voted.

To further clarify the report structure, Figures 4 and 5 show examples of a CVR collection exported from a precinct scanner located in a vote center that serves three precincts. There are several ways in which the report can be structured. The example in Figure 4 is essentially the same structure as was shown in Figure 3. In this case, the report generating device would be the scanner, the election would be synonymous to a precinct, and each precinct's CVRs would then be grouped accordingly.

Cast Vote Record Report:

- *Date/Time of report generation*
- *Device generating the report – device located at the Vote Center*
- ***Election associated with the first collection of CVRs***
  - *Political geography associated with the collection – Precinct 1*
  - *Candidate object definitions referenced by CVRs for Precinct 1*
  - *Contest and ballot selection object definitions referenced by CVRs for Precinct 1*
    - ***CVR 1-1, CVR 1-2, CVR 2-3, …***
- ***Election associated with the second collection of CVRs***
  - *Political geography associated with the collection – Precinct 2*
  - *Candidate object definitions referenced by CVRs for Precinct 2*
  - *Contest and ballot selection object definitions referenced by CVRs for Precinct 2*
    - ***CVR 2-1, CVR 2-2, CVR 2-3, …***

Figure 4 - CVR Report Structure for a Voting Center - Example 1

Example 4 defines contest and candidate objects for each precinct. But, the ballots in a vote center will differ by only those contests that are specific to a precinct. Example 5 improves on this by using a single collection of CVRs. This requires linking each CVR to its respective precinct.

Cast Vote Record Report:

- *Date/Time of report generation*
- *Device generating the report – device located at the Vote Center*
- ***Election associated with the collection of CVRs***
  - *Political geography associated with the collection – the Vote Center*
  - *Candidate object definitions referenced by CVRs*
  - *Contest and ballot selection object definitions referenced by CVRs*
    - ***CVR 1 linked to political geography associated with its precinct***
    - ***CVR 2 linked to political geography associated with its precinct***
    - ***…***

Figure 5 - CVR Report Structure for a Voting Center - Example 2

Space is preserved by defining all contests, etc., at the vote center only once. Figure 4's structure may be more useful if housing collections whose corresponding ballots differ significantly as opposed to only slightly. Or, a combination of the two structures may be more useful, depending on many factors such as number of collections, extent to which they differ, voting methods in use, etc.

3.2 Cast Vote Record Class Structure

This section contains further explanation of the CVR class and its associated classes that are used to construct the CVRs that were shown in the previous CVR report examples. The construction is straightforward in most cases for plurality voting. Other voting methods may complicate the structure to some degree.

Before describing the CVR structure, it will be helpful to better understand the structure of contests, candidates, and ballot selections object definitions, as well as their associated vote marks. From the UML models from Figures 1 and 2, the Contest and Candidate classes contain optional attributes. The most likely to be used will be Code, for containing a code associated with the object. At the expense of the report filesize, one can include other attributes such as the name of the object. As implemented in JSON or XML, each of the objects also contains a unique identifier. A CVR uses the identifiers to link to those contests that were voted.

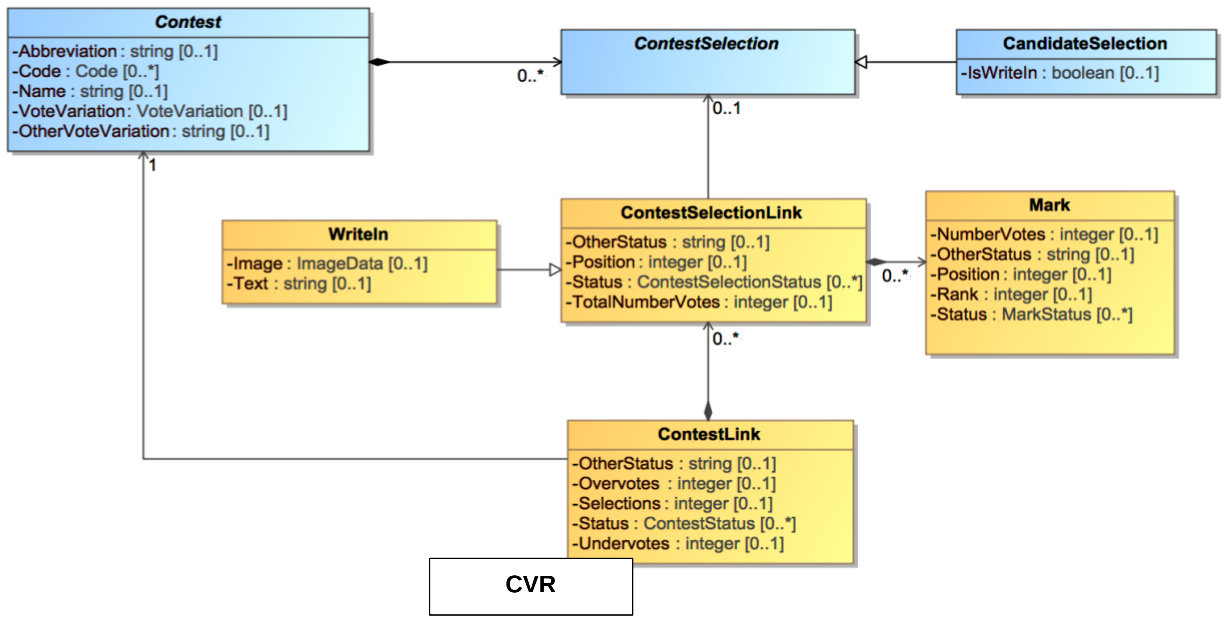

Contest
-Abbreviation : string [0..1]
-Code : Code [0..*]
-Name : string [0..1]
-VoteVariation : VoteVariation [0..1]
-OtherVoteVariation : string [0..1]
0..*
ContestSelection
CandidateSelection
-IsWriteIn : boolean [0..1]
0..1
1
WriteIn
-Image : ImageData [0..1]
-Text : string [0..1]
ContestSelectionLink
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Status : ContestSelectionStatus [0..*]
-TotalNumberVotes : integer [0..1]
0..*
Mark
-NumberVotes : integer [0..1]
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Rank : integer [0..1]
-Status : MarkStatus [0..*]
0..*
ContestLink
-OtherStatus : string [0..1]
-Overvotes : integer [0..1]
-Selections : integer [0..1]
-Status : ContestStatus [0..*]
-Undervotes : integer [0..1]
CVR

Figure 6 - SubClasses of CVR Class

Figure 6 shows that the Contest class defines its ballot selections using the ContestSelection class. For JSON and XML, the ballot selections link to the candidate objects using their unique identifiers. The ballot selections also contain a unique identifier, which the CVR uses to link to those ballot selections that were voted. In this way, a CVR can link to those contest, candidate, and ballot selection objects that were voted, and then associate the vote marks accordingly.

A ballot selection is analogous to a line or row on a ballot, such as for selecting a candidate in a contest. Vote marks are analogous to the bubbles associated with the ballot selection.

Lastly, Figure 6 shows two other important classes used for linking the objects. The CVR class uses ContestLink to link to those contest objects that were voted. ContestLink uses ContestSelectionLink to link to the voted ballot selections and their associated vote marks.

With that as background, Figure 7 shows an example of a CVR, listing all possible attributes:

CVR:

- *Header information (all optional):*
    - *ID of the device that created the CVR*
    - *ID of corresponding ballot style*
    - *ID of the associated party if a primary election*
    - *ID to link the CVR with its corresponding paper ballot*
    - *ID of an associated batch and sequence number within the batch*
    - *Page number if a multi-page ballot*
    - *CVR status – counted, adjudicated, needs adjudication, etc.*
- ***ContestLink** (attributes all optional)**:***
    - *Link to a contest object that was voted*
    - *Contest status - overvoted/undervoted, adjudicated, invalidated, etc.*
    - *Number votes lost due to overvoting/undervoting*
    - *Total number of ballot selections voted in the contest*
    - ***ContestSelectionLink** (attributes all optional)**:***
        - *Link to a ballot selection that was voted*
        - *If write-in, the image or text of the write-in*
        - *Ballot selection status - overvoted/undervoted, adjudicated, invalidated, etc.*
        - *Position of the ballot selection on the ballot*
        - *Total number of votes represented by the ballot selection*
        - ***Mark** (attributes all optional)**:***
            - *Vote mark status - marked by voter, by rules, by adjudication, marginal, etc.*
            - *Position of the vote mark bubble on the ballot*
            - *Number of votes represented by the vote mark*
            - *Rank – for RCV in which the bubble contains a ranking*
            - *Mark quality metric*
        - ***Additional Marks as needed per voting method***
    - ***Additional ContestSelectionLinks as needed per voting method***
- ***Additional ContestLinks as needed for other contests that were voted***

Figure 7 - CVR Structure

Typically, only those attributes that are absolutely needed would be included, thus all attributes are optional. MarkMetric represents a vendor-specific measurement of the mark's quality, used to determine whether the mark is marginal or valid.

CVR includes ContestLink only for contest objects that are voted, i.e., that contain ballot selections and/or that are undervoted. If the contest is entirely undervoted, ContestLink would include only the number of votes lost due to undervoting.

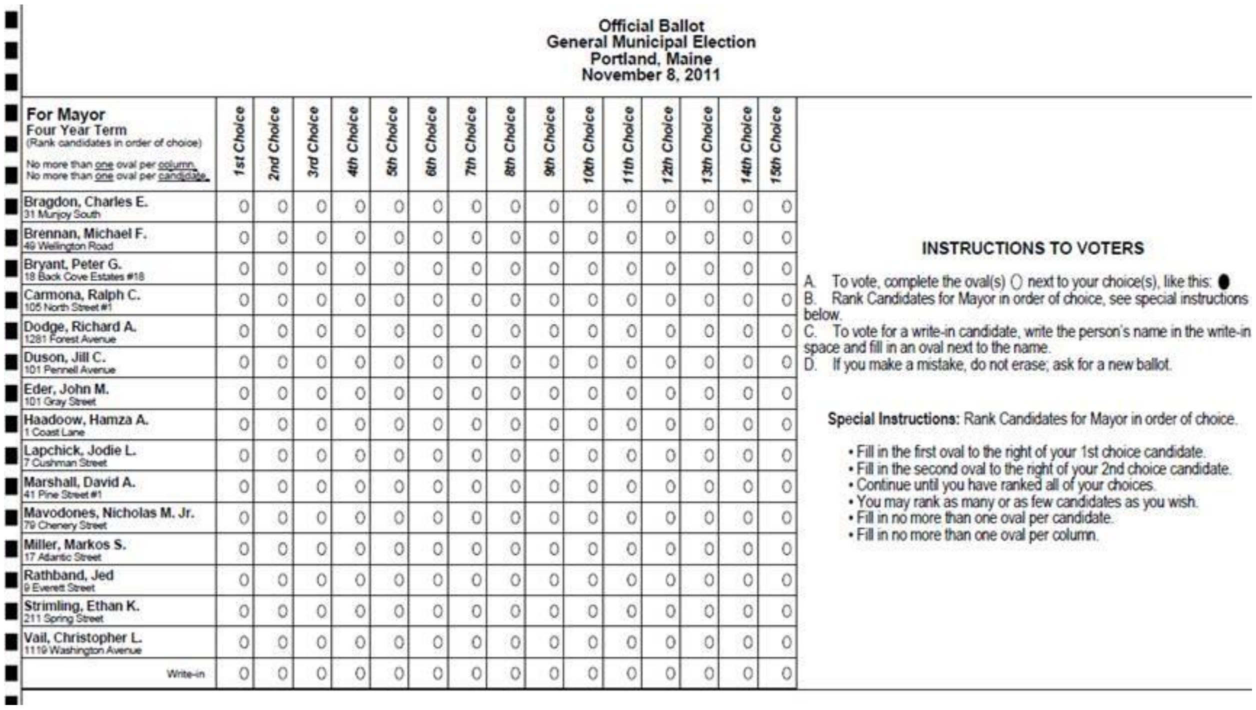
ContestLink includes ContestSelectionLink to link to ballot selection objects when:

- The voter made a mark designated by the scanner as a valid mark.
- The voter made a mark designated by the scanner as a marginal mark.
- The scanner invalidated a voter mark or generated a new mark because of contest rules.
- An adjudication invalidated a mark or generated a new mark.

The following section goes into more detail regarding the use of ContestSelectionLink and Mark for different voting methods.

3.2.1 Use of ContestSelectionLink and Mark for Voting Methods

Figure 7 showed the possibility of multiple ballot selections in a contest and multiple vote marks in a ballot selection. Depending on the voting method being used, multiple ballot selections could be allowable or could constitute overvotes; the same is true for multiple ballot selections.



Official Ballot
General Municipal Election
Portland, Maine
November 8, 2011

For Mayor **Four Year Term** (Rank candidates in order of choice) No more than one oval per column. No more than one oval per candidate.	1st Choice	2nd Choice	3rd Choice	4th Choice	5th Choice	6th Choice	7th Choice	8th Choice	9th Choice	10th Choice	11th Choice	12th Choice	13th Choice	14th Choice	15th Choice
Bragdon, Charles E. 31 Munjoy South	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Brennan, Michael F. 49 Wellington Road	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Bryant, Peter G. 18 Back Cove Estates #18	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Carmona, Ralph C. 105 North Street #1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Dodge, Richard A. 1281 Forest Avenue	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Duson, Jill C. 101 Pennell Avenue	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Eder, John M. 101 Gray Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Haadoow, Hamza A. 1 Coast Lane	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Lapchick, Jodie L. 7 Cushman Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Marshall, David A. 41 Pine Street #1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Mavodones, Nicholas M. Jr. 79 Chenery Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Miller, Markos S. 17 Atlantic Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Rathband, Jed 9 Everett Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Strimling, Ethan K. 211 Spring Street	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Vail, Christopher L. 1119 Washington Avenue	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Write-in	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

INSTRUCTIONS TO VOTERS

A. To vote, complete the oval(s) ○ next to your choice(s), like this: ●
B. Rank Candidates for Mayor in order of choice, see special instructions below.
C. To vote for a write-in candidate, write the person's name in the write-in space and fill in an oval next to the name.
D. If you make a mistake, do not erase; ask for a new ballot.

Special Instructions: Rank Candidates for Mayor in order of choice.

- Fill in the first oval to the right of your 1st choice candidate.
- Fill in the second oval to the right of your 2nd choice candidate.
- Continue until you have ranked all of your choices.
- You may rank as many or as few candidates as you wish.
- Fill in no more than one oval per candidate.
- Fill in no more than one oval per column.

Figure 8 - RCV contest with multiple ballot selections; one vote mark per selection

Figure 8 shows an RCV contest, in which 15 candidates can be chosen, each with an associated rank. ContestLink includes ContestSelectionLink for each candidate selection in which a mark was made, i.e., a bubble was filled in to the right of a candidate's name. ContestSelectionLink includes Mark for each bubble filled in. Depending on the RCV contest rules, more than one bubble filled in for a given candidate ballot selection may be an overvote. A partial example of the structure representing the contest in Figure 8 is as follows:

- ContestLink:
    - Link to contest object for this contest
    - Total number of voted ballot selections = 3
    - Number of votes lost to undervoting = 12
    - ContestSelectionLink:
        - Link to candidate object for Charles E. Bragdon
        - Mark:
            - Position = 3 and/or Rank = 3
            - Status = marked by voter
    - ContestSelectionLink:
        - Link to candidate object for Michael F. Brennan
        - Mark:
            - Position = 1 and/or Rank = 1
            - Status = marked by voter
    - ContestSelectionLink:
        - Link to candidate object for Peter G. Bryant
        - Mark:
            - Position = 2 and/or Rank = 2
            - Status = marked by voter

Figure 9 below shows an example for cumulative voting in which multiple vote marks are allowable for each ballot selection, with a total number of votes (or score) included for each candidate.

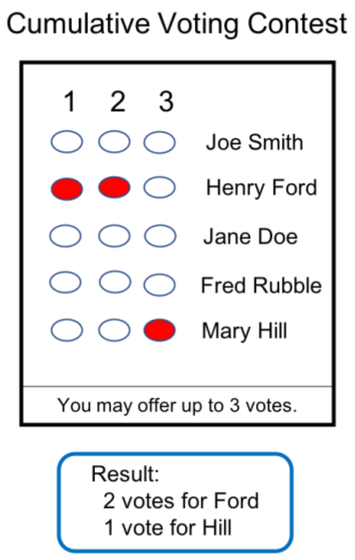


Figure 9 - Cumulative Voting contest with multiple ballot selections; multiple vote marks per selection

In Figure 9, 3 votes can be allocated across the 5 candidates, however only 2 candidates received votes, Ford and Hill. Thus, ContestLink need only include 2 instances of ContestSelectionLink, one for Ford and one for Hill. Each ContestSelectionLink includes Mark for each bubble filled in to the left of each candidate's name, for a maximum of 3. In this case, there will be one instance of Mark for Henry Ford and two instances of Mark for Mary Hill. A partial example of the structure representing the contest in Figure 9 is as follows:

- ContestLink:
    - Link to contest object for this contest
    - Total number of voted ballot selections = 2
    - ContestSelectionLink:
        - Link to candidate object for Henry Ford
        - Total number of votes represented by the ballot selection = 2
        - Mark:
            - Position = 1
            - Status = marked by voter
            - Number of votes represented by the mark = 1
        - Mark:
            - Position = 2
            - Status = marked by voter
            - Number of votes represented by the mark = 1
    - ContestSelectionLink:
        - Link to candidate object for Mary Hill
        - Total number of votes represented by the ballot selection = 2
        - Mark:
            - Position = 3
            - Status = marked by voter
            - Number of votes represented by the mark = 1

Lastly, Figure 10 shows an example of a contest using range voting, in which each candidate receives a score, which is analogous to number of votes.

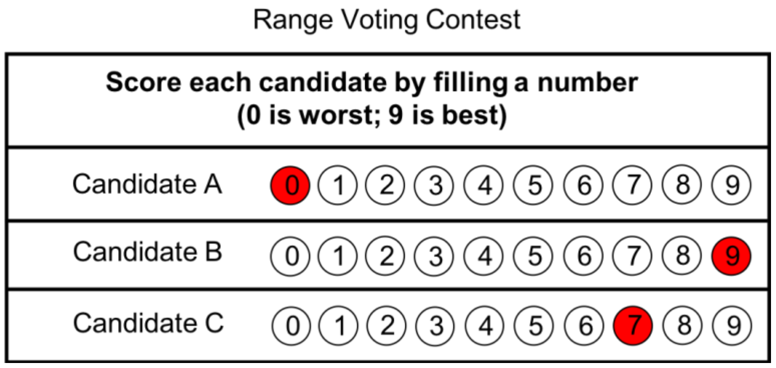


Figure 10 - Range Voting contest with multiple ballot selections; one vote mark per selection

In this example, one bubble can be filled in for each ballot selection, and each candidate will have an associated score or number of votes. A partial example of the structure representing the contest in Figure 10 is as follows:

- ContestLink:
    - Link to contest object for this contest
    - Total number of voted ballot selections = 3
    - ContestSelectionLink:
        - Link to candidate object for Candidate A
        - Total number of votes represented by the ballot selection = 0
        - Mark:
            - Position = 1
            - Status = marked by voter
    - ContestSelectionLink:
        - Link to candidate object for Candidate B
        - Total number of votes represented by the ballot selection = 9
        - Mark:
            - Position = 10
            - Status = marked by voter
    - ContestSelectionLink:
        - Link to candidate object for Candidate C
        - Total number of votes represented by the ballot selection = 7
        - Mark:
            - Position = 8
            - Status = marked by voter

The following Section 4 describes the classes and enumerations in the UML model and provides additional explanation of usage. Section 5 contains several CVR examples using XML.

4 Cast Vote Record UML Model Documentation

This section contains documentation and discussion of the features included in the CVR UML model. As noted previously, this model was used in deriving the XML and JSON schemas, and the schema usage closely follows that of the UML model.

The UML classes are described first, followed by the enumerations. Each description contains an image of the class (from the UML model) and a table containing details about each of the class's attributes. To denote that certain class attributes derive from the class's relationships with other classes, curly braces are used around those attribute names, e.g., if ClassA has a relationship with ClassB that is named "Automobile", then the table of attributes would include "{Automobile}" as one of the attributes.

4.1 Class Annotation

Annotation is used to record annotations made by one or more adjudicators. The annotation can be specific to an entire CVR, or to a particular contest, ballot selection, or ballot mark. Multiple annotations per item can be included. CVR, ContestLink, ContestSelectionLink, and Mark all include Annotation.

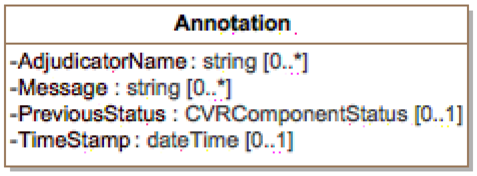

Annotation
-AdjudicatorName : string [0..*]
-Message : string [0..*]
-PreviousStatus : CVRComponentStatus [0..1]
-TimeStamp : dateTime [0..1]

Figure 11 - Annotation

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
AdjudicatorName	0..*	string	The name(s) of the adjudicator(s).
Message	0..*	string	A message created by the adjudicator(s).
PreviousStatus	0..1	CVRComponentStatus	The status of the item being adjudicated prior to the recording of this annotation. The values come from the ContestSelectionStatus, BallotStatus, ContestStatus, or Mark enumerations. If the previous status was specified in a class's other status attribute, e.g., ContestLink.OtherContestStatus, PreviousStatus would contain this other status.
TimeStamp	0..1	dateTime	The date and time of the annotation.

4.2 Class Candidate

Candidate identifies a candidate in a contest on the voter's ballot. Election includes instances of Candidate for each candidate in a contest; typically, only those candidates who received votes would be included.

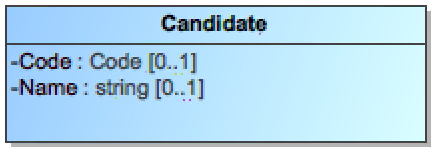


Figure 12 - Candidate

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Code	0..1	Code	A code or identifier associated with the candidate.
Name	0..1	String	Candidate's name as listed on the ballot.
Party	0..1	Party	The party associated with the candidate.

4.3 Class CVR

CVR constitutes a CVR, generated by a scanner or other vote capture device, containing indications of the contests and ballot selections chosen by the voter, as well as other information for auditing and annotation purposes. CastVoteRecordReport includes multiple instances of CVR as applicable. Each sheet of a multi-page paper ballot is represented by an individual CVR, e.g., if all sheets of a 5-sheet ballot are scanned, 5 CVRs will be generated.

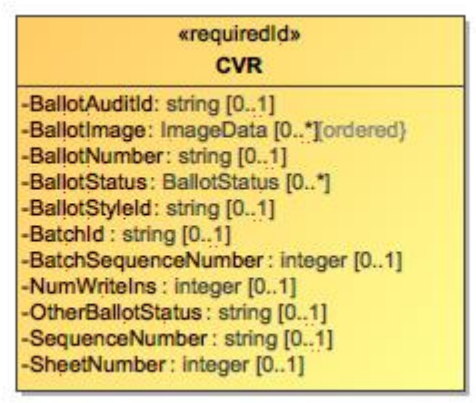

«requiredId»
CVR
-BallotAuditId: string [0..1]
-BallotImage: ImageData [0..*]{ordered}
-BallotNumber: string [0..1]
-BallotStatus: BallotStatus [0..*]
-BallotStyleId: string [0..1]
-BatchId : string [0..1]
-BatchSequenceNumber : integer [0..1]
-NumWriteIns : integer [0..1]
-OtherBallotStatus : string [0..1]
-SequenceNumber : string [0..1]
-SheetNumber : integer [0..1]

Figure 13 - CVR

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{ContestLink }	0..*	ContestLink	Identifies the contests in the CVR.
{Party}	0..*	Party	Identifies the party associated with a CVR, typically for partisan primaries.
{Annotation}	0..*	Annotation	Used to include an annotation associated with the CVR as a whole.
{Election}	0..1	Election	
BallotAuditId	0..1	string	A unique identifier for this CVR, used to link the CVR with the corresponding audit record, e.g., a paper ballot. This identifier may be impressed on the corresponding audit record as it is scanned, or otherwise associated with the corresponding ballot.
BallotImage	0..*	ImageData	An ordered list of pointers to image files associated with the corresponding ballot, made by the scanner as the ballot is scanned. The images could represent the entire ballot or parts of the ballot, e.g., for write-in marks made by a voter on a paper ballot.
BallotNumbe r	0..1	string	A unique number for the ballot (or sheet of a multi-sheet ballot) that this CVR represents, used if ballots are pre-marked with unique numbers. If provided, this number would be the same on all CVRs that

			represent individual sheets from the same multi-sheet ballot. This number is not the same as one that may be impressed on the corresponding ballot as it is scanned or otherwise associated with the corresponding ballot; see the BallotAuditId attribute.
BallotStatus	0..*	BallotStatus	The status of the CVR, e.g., adjudicated, counted, etc.
BallotStyleId	0..1	string	An identifier of the ballot style associated with the corresponding ballot.
BatchId	0..1	string	The identifier for the batch that includes this CVR.
BatchSequenceNumber	0..1	integer	The sequence number of the corresponding paper ballot within a batch.
NumWriteIns	0..1	integer	The total number of write-ins on the ballot, that is, the sum of write-ins in contests on the ballot.
OriginatingDevice	0..1	ReportingDevice	Identifies the device used to generate the CVR.
OtherBallotStatus	0..1	string	When BallotStatus is 'other', contains the ballot status.
SequenceNumber	0..1	string	The sequence number for this CVR. This represents the ordinal number that this CVR was processed by the tabulating device.
SheetNumber	0..1	integer	Indicates the sheet number of the multi-sheet ballot being represented.
SmallestVotingUnit	0..1	GpUnit	Identifies the smallest unit of geography associated with the corresponding ballot, typically a precinct or split-precinct.

4.4 Class CastVoteRecordReport

The root class/element; attributes pertain to the status and format of the report and when generated.

CastVoteRecordReport includes multiple instances of CVR, one per CVR or sheet of a multi-page cast vote record. CastVoteRecordReport also includes multiple instances of Contest, typically only for those contests that were voted so as to reduce file size. The Contest instances are later referenced by other classes to link them to ballot selections that were voted and the mark(s) made.

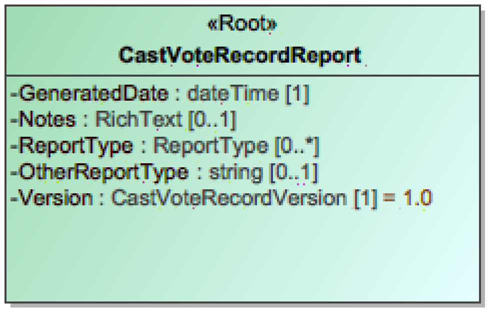

«Root»
CastVoteRecordReport
-GeneratedDate : dateTime [1]
-Notes : RichText [0..1]
-ReportType : ReportType [0..*]
-OtherReportType : string [0..1]
-Version : CastVoteRecordVersion [1] = 1.0

Figure 14 - CastVoteRecordReport

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{CVR}	0..*	CVR	Used to include instances of CVR classes, one per CVR in the report.
{Election}	0..*	Election	Used to include the election associated with the CVRs.
{GpUnit}	0..*	GpUnit	Used to include the political geography, i.e., location, for where the CVR report was generated and for linking CVRs to their corresponding precinct or split (or otherwise smallest unit).
GeneratedDate	1	dateTime	Identifies the time that the election report was generated.
Notes	0..1	RichText	Notes that can be added as appropriate, presumably by an adjudicator.
OtherReportType	0..1	string	If ReportType is 'other', this contains the report type.
ReportGeneratingDevice	1..*	ReportingDevice	Identifies the device used to generate the CVR report.
ReportType	0..*	ReportType	The type of report, using the ReportType enumeration.
Version	1	CastVoteRecordVersion	The version of the CVR specification being used (1.0).

4.5 Class Code

Code is used in Election, GpUnit, Contest, Candidate, and Party to identify an associated code as well as the type of code.

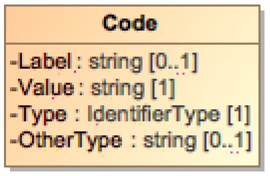


Figure 15 - Code

Attribute Detail:

Attribute	**Multiplicity**	**Type**	**Attribute Description**
Label	0..1	string	A label associated with the code, used as needed.
OtherType	0..1	string	If Type is 'other', the type of code.
Type	1	IdentifierType	Used to indicate the type of code, from the IdentifierType enumeration.
Value	1	string	The value of the code, i.e., the identifier.

4.6 Class Contest and SubClasses BallotMeasureContest, CandidateContest, PartyContest, RetentionContest

Contest represents a contest on the ballot. CastVoteRecordReport initially includes an instance of Contest for each contest on the ballot. Other classes can subsequently reference the instances as necessary to link together items on the CVR, such as a contest, its voted ballot selection(s), and the mark(s) associated with the selection(s).

ContestSelection has three subclasses, each used for a specific type of ballot selection: These subclasses inherit Contest's attributes.

- BallotMeasureContest - used for contests,
- CandidateContest - used for candidate contests,
- PartyContest - used for straight party contests, and
- RetentionContest – used for (judicial) retention contests.

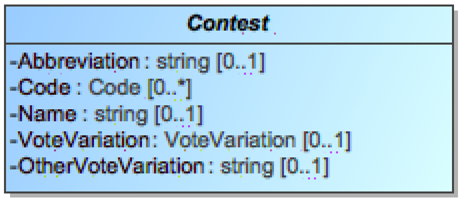

Contest
-Abbreviation : string [0..1]
-Code : Code [0..*]
-Name : string [0..1]
-VoteVariation : VoteVariation [0..1]
-OtherVoteVariation : string [0..1]

Figure 16 - Contest

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{ContestSelection}	0..*	ContestSelection	Identifies the ballot selections in the contest.
Abbreviation	0..1	string	An abbreviation associated with the contest.
Code	0..*	Code	A code or identifier used for this contest.
Name	0..1	string	Title or name of the contest, e.g., "Governor" or "Question on Legalization of Gambling".
OtherVoteVariation	0..1	string	If VoteVariation is 'other', the vote variation for this contest.
VoteVariation	0..1	VoteVariation	The vote variation for this contest, from the VoteVariation enumeration.

4.6.1 SubClass BallotMeasureContest

BallotMeasureContest is a subclass of Contest and is used for ballot measure contests. It inherits attributes from Contest.

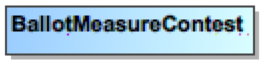


Figure 17 - BallotMeasureContest

4.6.2 SubClass CandidateContest

CandidateContest is a subclass of Contest and is used to identify the type of contest as involving one or more candidates. It inherits attributes from Contest.

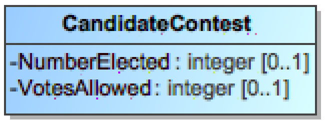

CandidateContest
-NumberElected : integer [0..1]
-VotesAllowed : integer [0..1]

Figure 18 - CandidateContest

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
NumberElected	0..1	integer	The number of candidates to be elected in the contest.
PrimaryParty	0..1	Party	The party associated with the contest, if a partisan primary.
VotesAllowed	0..1	integer	The number of votes allowed in the contest, e.g., 3 for a 'choose 3 of 5 candidates' contest.

4.6.3 SubClass PartyContest

PartyContest is a subclass of Contest and is used to Identify the type of contest as involving a straight party selection. It inherits attributes from Contest.

Figure 19 - PartyContest

4.6.4 SubClass RetentionContest

RetentionContest is a subclass of BallotMeasureContest and is used to Identify the type of contest as involving a retention, such as for a judicial retention. While it is similar to BallotMeasureContest, it contains a link to Candidate that BallotMeasureContest does not. RetentionContest inherits attributes from Contest.

Figure 20 - RetentionContest

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Candidate		Candidate	Identifies the candidate in the retention contest.

4.7 Class ContestLink

ContestLink class is included by CVR for each contest on the ballot that was voted, that is, whose ballot selections contain marks that may constitute a vote. ContestLink includes ContestSelectionLink for each ballot selection in the contest containing a mark.

CVR can also include ContestLink for every contest on the ballot regardless of whether any of the ballot selections contain a mark, for cases where the CVR must include all contests that appeared on the ballot.

ContestLink attributes are for including summary information about the contest. Overvotes plus Undervotes plus TotalVotes must equal the number of votes allowable in the contest, e.g., in a "chose 3 of 5" contest in which the voter chooses only 2, then Overvotes = 0, Undervotes = 1, and TotalVotes = 2, which adds up to the number of votes allowable = 3.

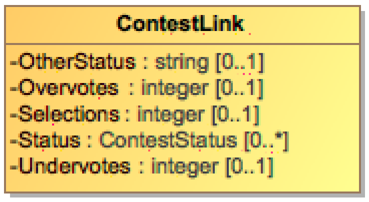

ContestLink
-OtherStatus : string [0..1]
-Overvotes : integer [0..1]
-Selections : integer [0..1]
-Status : ContestStatus [0..*]
-Undervotes : integer [0..1]

Figure 21 - ContestLink

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{Contest}	1	Contest	Used to link to an instance of Contest specific to the contest at hand, for the purpose of specifying information about the contest such as its contest identifier.
{ContestSelectionLink}	0..*	ContestSelectionLink	Used to include information about a ballot selection in the contest, including the associated vote marks(s).
{Annotation}	0..*	Annotation	Used to include an annotation(s) for adjudicating aspects of the contest.
OtherStatus	0..1	string	Used when Status is 'other' to include a user-defined status.
Overvotes	0..1	integer	The number of votes lost due to overvoting. Identifies a contest in which a ballot selection was made as well as the ballot selection(s) made in the contest. Overvotes plus Undervotes plus TotalVotes must equal the number of votes allowable in the contest, e.g., in a "chose 3 of 5" contest in which the voter chooses only 2, then Overvotes = 0, Undervotes = 1, and TotalVotes = 2, which adds up to the number of votes allowable = 3.

Selections	0..1	integer	Used to indicate the number of possible ballot selections in the contest.
Status	0..*	ContestStatus	The status of the contest, e.g., overvoted, undervoted, from the ContestStatus enumeration. If no values apply, use 'other' and include a user-defined status in OtherStatus.
Undervotes	0..1	integer	The number of votes lost due to undervoting.

4.8 Class ContestSelection and SubClasses BallotMeasureSelection, CandidateSelection, PartySelection

ContestSelection represents a ballot selection in a contest. Contest can include an instance of ContestSelection for each ballot selection in the contest. Typically, there will be ballot selections included only for those contests their ballot selections that were voted, i.e., that contain a mark.

ContestSelection has three subclasses, each used for a specific type of ballot selection:

- BallotMeasureSelection - used for ballot measures,
- CandidateSelection - used for candidate selections, and
- PartySelection - used for straight party selections.

Instances of ContestSelectionLink subsequently link to the ballot selections as needed so as to tie together the contest, the ballot selection, and the mark(s) made for the ballot selection.

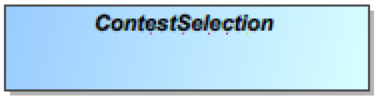


Figure 22 - ContestSelection

4.8.1 SubClass BallotMeasureSelection

BallotMeasureSelection is a subclass of ContestSelection and is used for ballot measures. The voter's selected response to the ballot selection (e.g., "yes" or "no") may be in English or other languages as utilized on the voter's ballot.

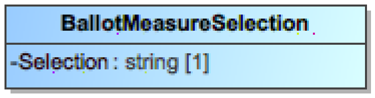


Figure 23 - BallotMeasureSelection

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Selection	1	String	The voter's selection, i.e., 'yes' or 'no', in English or in other languages as utilized on the voter's ballot.

4.8.2 SubClass CandidateSelection

CandidateSelection is a subclass of ContestSelection and is used for candidates, including for write-in candidates.

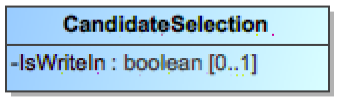


Figure 24 - CandidateSelection

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Candidate	0..*	Candidate	The candidate associated with the ballot selection. For contests involving a ticket of multiple candidates, an ordered list of candidates as they appeared on the ballot would be created.
IsWriteIn	0..1	boolean	A flag to indicate if the candidate selection is associated with a write-in.

4.8.3 SubClass PartySelection

PartySelection is a subclass of ContestSelection and is used typically for a ballot selection in a straight-party contest.

Figure 25 - PartySelection

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Party	1..*	Party	The party associated with the ballot selection.

4.9 Class ContestSelectionLink and SubClass WriteIn

ContestSelectionLink is used to link those ballot selections that contain a mark to information about the mark, such as whether the mark constitutes a countable vote, or whether the mark is determined to be marginal, etc. ContestLink includes an instance of ContestSelectionLink when a mark for the selection is present, and ContestSelectionLink then includes Mark for each mark present. To tie the mark to the specific ballot selection, ContestSelectionLink links to instances of ContestSelection that have previously been included by Contest.

Since multiple marks per ballot selection are possible for some voting methods, ContestSelectionLink can include multiple instances of Mark, one per mark. ContestSelectionLink can also be used for the purpose of including, in the CVR, all ballot selections in the contest regardless of whether marks are present. In this case, ContestSelectionLink would not include Mark if no mark is present but would link to the appropriate instance of ContestSelection.

ContestSelectionLink has one subtype, WriteIn, whose attributes are used to include information about the write-in including the text of the write-in or an image of the write-in, whichever may be present.

Need to regen the image.

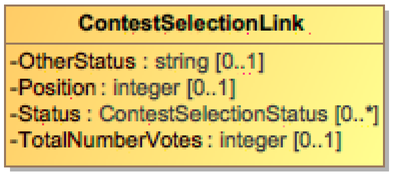

ContestSelectionLink
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Status : ContestSelectionStatus [0..*]
-TotalNumberVotes : integer [0..1]

Figure 26 - ContestSelectionLink

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{ContestSelection}	0..1	ContestSelection	Used to link to an instance of a ballot selection that was previously included by Contest.
{Annotation}	0..*	Annotation	Used to include an annotation(s) for adjudicating aspects of the ballot selection.
{Mark}	0..*	Mark	Used to include further information about the mark associated with the ballot selection. Depending on the voting method, multiple marks per selection may be possible.
OtherStatus	0..1	string	Used when Status is 'other' to include a user-defined status.
Position	0..1	integer	Used to include the ordinal position of the ballot selection as is appeared on the ballot.

Rank	0..1	integer	For the RCV voting variation, the rank chosen by the voter, for when a contest selection can represent a ranking.
Status	0..*	ContestSelectionStatus	Contains the status of the ballot selection, e.g., 'marked-voter' for marked by the voter, using values from the ContestSelectionStatus enumeration. If no values apply, use 'other' and include a user-defined status in OtherStatus.
TotalNumberVotes	0..1	integer	For cumulative or range voting variations, contains the total number of votes indicated across all marks.

4.9.1 SubClass WriteIn

WriteIn is a subclass of ContestSelectionLink and is used when the ballot selection is a write-in. It inherits attributes from ContestSelectionLink and has additional attributes for the image or text of the write-in.

Need to regen the image.

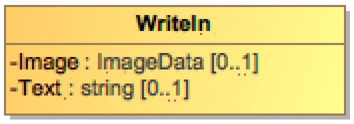


Figure 27 - WriteIn

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Text	0..1	String	Used for the text of the write-in, typically present when the CVR has been created by electronic ballot marking equipment.
WriteInImage	0..1	ImageData	Used for an image of the write-in, typically made by a scanner when scanning a paper ballot.

4.10 Class Election

Election defines instances of the Contest and Candidate classes so that they can be later referenced in CVR classes. Election includes an instance of Contest for each contest in the election and includes an instance of Candidate for each candidate. This is done to utilize file sizes more efficiently; otherwise each CVR would need to define these instances separately and much duplication would occur.

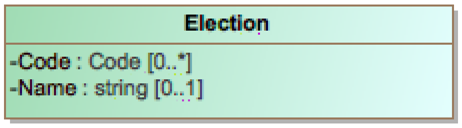


Figure 28 - Election

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{Contest}	0..*	Contest	Used for establishing a collection of contest definitions that will be referenced by the CVRs.
{Candidate}	0..*	Candidate	Used to establish a collection of candidate definitions that will be referenced by the CVRs. The contests in each CVR will reference the candidate definitions.
Code	0..*	Code	Used for a code associated with the election, e.g., a precinct identifier if the election scope is a precinct.
ElectionScope	0..1	GpUnit	Used to identify the election scope, i.e., the political geography corresponding to the election.
Name	0..1	string	A text string identifying the election.

4.11 Class GpUnit and SubClass ReportingDevice

Used for identifying a geographical unit for various purposes, including:

- the reporting unit of the report generation device, e.g., a precinct location of a scanner that generates the collection of CVRs,
- the geographical scope of the election, or the unit of geography associated with an individual CVR.

CastVoteRecordReport includes instances of GpUnit as needed. Election references GpUnit as ElectionScope, for the geographical scope of the election. CVR references GpUnit as SmallestVotingUnit, to link a CVR to the smallest political subdivision that the CVR "belongs" to.

GpUnit has one subclass, ReportingDevice, used to identify a specific device by its manufacturer, model, a serial number. CVR references ReportingDevice to link a CVR to the device that generated it. Election references ReportingDevice to link to the device that was used to generate the CVR report.

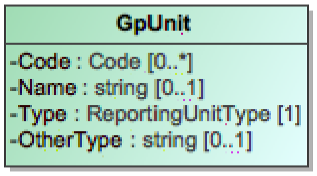


Figure 29 - GpUnit

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{ReportingDevice}	0..*	ReportingDevice	The collection of CVRs associated with the reporting unit and the reporting device.
Code	0..*	Code	A code associated with the geographical unit.
Name	0..1	string	Name of the geographical unit.
OtherType	0..1	string	Used when Type is 'other' to include a user-defined type.
Type	1	ReportingUnitType	Contains the type of geographical unit, e.g., precinct, split-precinct, vote center, using values from the ReportingUnitType enumeration. If no values apply, use 'other' and include a user-defined type in OtherType.

4.11.1 SubClass ReportingDevice

ReportingDevice is a subclass of GpUnit and is used to specify a voting device as the “political geography” at hand. CastVoteRecordReport refers to it as "ReportGeneratingDevice" and uses it to specify the device that generated the CVR report. CVR refers to it as "OriginatingDevice" to specify the device that generated the CVRs.

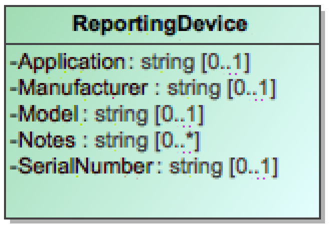

ReportingDevice
-Application : string [0..1]
-Manufacturer : string [0..1]
-Model : string [0..1]
-Notes : string [0..*]
-SerialNumber : string [0..1]

Figure 30 - ReportingDevice

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Application	0..1	string	The application associated with the reporting device.
Manufacturer	0..1	string	Manufacturer of the reporting device.
Model	0..1	string	Manufacturer's model of the reporting device.
Notes	0..*	string	Additional explanatory notes as applicable.
SerialNumber	0..1	string	Serial number or other identification that can uniquely identify the reporting device.

4.12 Class ImageData and Associated Classes File, Image, Hash

ImageData is used to specify an image file such as for a write-in or the entire ballot. It works with several other classes, as follows:

- File with SubClass Image – to contain either a filename for an external file or the file contents, and
- Hash – to contain cryptographic hash function data for the file.

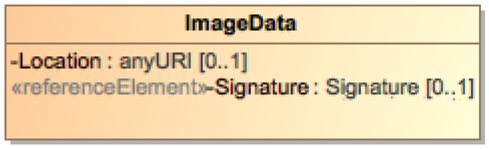

ImageData
-Location : anyURI [0..1]
«referenceElement»-Signature : Signature [0..1]

Figure 31 - ImageData

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{Image}	0..1	Image	
{Hash}	0..1	Hash	
Location	0..1	anyURI	A pointer to the location of the image file.
Signature	0..1	Signature	Either a signature for the embedded image, or a detached signature for the image available at the specified location.

4.12.1 Class File

Used to hold the contents of a file or identify a file created by the scanning device. The file generally would contain an image of the scanned ballot or an image of a write-in entered by a voter onto the scanned ballot. SubClass Image is used if the file contains an image.

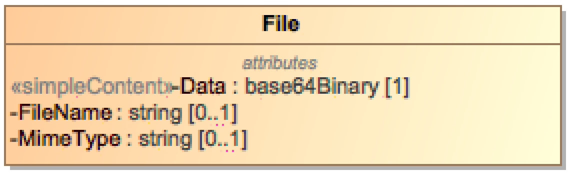

File
attributes
«simpleContent»-Data : base64Binary [1]
-FileName : string [0..1]
-MimeType : string [0..1]

Figure 32 - File

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Data	1	base64Binary	Contains the base64 binary contents of the file.
FileName	0..1	string	Contains the name of the file or an identifier of the file.
MimeType	0..1	string	The mime type of the file, e.g., image/jpeg.

4.12.2 SubClass Image

Used by File for a file containing an image, e.g., an image of a write-in on a paper ballot.

Figure 33 - Image

4.12.3 Class Hash

Contains a hash associated with a file.

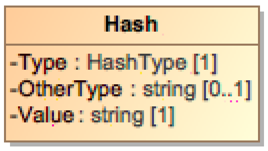


Figure 34 - Hash

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
OtherType	0..1	string	If Type is 'other', the type of the hash.
Type	1	HashType	The type of the hash, from the HashType enumeration.
Value	1	string	The hash value, encoded as a string.

4.13 Class Mark

ContestSelectionLink includes Mark to specify a voter's mark in a ballot selection. and thus, a potential vote. The number of potential Marks that should be included by ContestSelectionLink is, for paper ballots, the same as the number of ovals next to a particular choice. There will be usually 1 instance of Mark for plurality voting, but there could be multiple instances for RCV, approval, cumulative, or other vote variations in which a voter can select multiple candidates.

Mark contains additional information about the mark to specify whether the mark is countable, as well as information needed for certain voting methods. It includes MarkMetric for assigning a quality metric to the mark, and also includes Annotation.

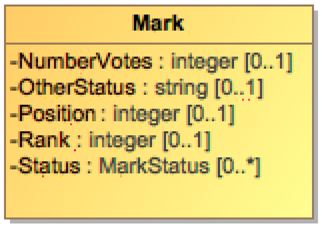

Mark
-NumberVotes : integer [0..1]
-OtherStatus : string [0..1]
-Position : integer [0..1]
-Rank : integer [0..1]
-Status : MarkStatus [0..*]

Figure 35 - Mark

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
{Annotation}	0..*	Annotation	Used to include an annotation(s) for adjudicating aspects of the mark.
{MarkMetric }	0..*	MarkMetric	Included to assign a quality metric to the mark.
NumberVotes	0..1	integer	The number of votes represented by the mark, usually 1 but may be more depending on the voting method.
OtherStatus	0..1	string	Used when Status is 'other' to include a user-defined status.
Position	0..1	integer	The ordinal position of the mark within the ballot selection.
Rank	0..1	integer	For the RCV voting variation, the rank chosen by the voter, for when a mark (bubble) can represent a ranking.
Status	0..*	MarkStatus	Status of the mark, e.g., "marked-voter" for marked by the voter, from the MarkStatus enumeration. If no values apply, use 'other' and include a user-defined status in OtherStatus.

4.14 Class MarkMetric

Mark includes MarkMetric to specify some generic measurement of a voter's mark on a paper ballot. The measurement will be typically assigned by a scanner, such as for mark density or quality, and would be used by the scanner to indicate whether the mark is intentional or marginal.

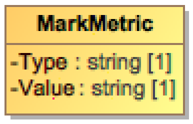


Figure 36 - MarkMetric

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Type	1	string	The type of metric being used to determine quality. The type must be specific enough that the attached value can be accurately verified later, e.g., 'Acme Mark Density' may be a sufficiently specific type.
Value	1	string	The value of the mark metric, represented as a string.

4.15 Class Party

Party is used for describing information about a political party associated with the voter's ballot. CVR includes instances of Party as needed, e.g., for a CVR corresponding to a ballot in a partisan primary, and CandidateContest references Party as needed to link a candidate to their political party.

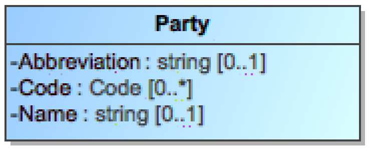


Figure 37 - Party

Attribute Detail:

Attribute	Multiplicity	Type	Attribute Description
Abbreviation	0..1	string	Short name for the party, e.g., "DEM".
Code	0..*	Code	A code associated with the party.
Name	0..1	String	Official full name of the party, e.g., "Republican".

4.16 Enumeration ContestSelectionStatus

Used in ContestSelectionLink.Status to identify the status of a ballot selection in the CVR.

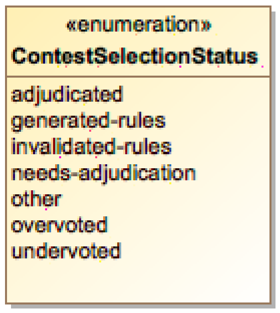


Figure 38 - ContestSelectionStatus

Enumeration Values:

Value	Value Description
adjudicated	To indicate that the ballot selection was adjudicated.
generated-rules	To indicate that the ballot selection was generated per contest rules.
invalidated-rules	To indicate that the ballot selection was invalidated by the generating device because of contest rules.
needs-adjudication	To indicate that the ballot selection was flagged by the generating device for adjudication.
other	Used in conjunction with ContestSelectionLink.OtherStatus when no other value in this enumeration applies.
overvoted	To indicate that the ballot selection was overvoted.
undervoted	To indicate that the ballot selection was undervoted.

4.17 Enumeration BallotStatus

Used in CVR.Status to identify the status of the CVR.

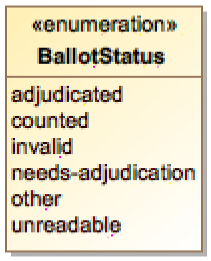


Figure 39 - BallotStatus

Enumeration Values:

Value	Value Description
adjudicated	To indicate that the CVR has been adjudicated.
counted	To indicate that the CVR has been counted.
invalid	To indicate that the associated ballot is invalid (not a ballot).
needs-adjudication	To indicate that the CVR needs to be adjudicated.
other	Used in conjunction with CVR.OtherStatus when no other value in this enumeration applies.
unreadable	To indicate that the ballot was not entirely readable by the scanner.

4.18 Enumeration CastVoteRecordVersion

To identify the version of the CVR specification being used, i.e., version 1.0. This will need to be updated for different version of the specification.

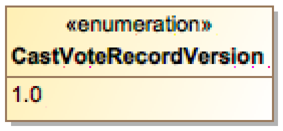


Figure 40 - CastVoteRecordVersion

Enumeration Values:

Value	Value Description
1.0	

4.19 Enumeration ContestStatus

Used in ContestLink.Status to identify the status of a contest in which ballot selection(s) were made.

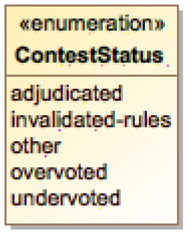


Figure 41 - ContestStatus

Enumeration Values:

Value	Value Description
adjudicated	To indicate that the contest has been adjudicated.
invalidated-rules	To indicate that the contest has been invalidated by the generating device because of contest rules.
other	Used in conjunction with ContestLink.OtherStatus when no other value in this enumeration applies.
overvoted	To indicate that the contest was overvoted.
undervoted	To indicate that the contest was undervoted.

4.20 Enumeration CVRComponentStatus

Used in Annotation.PreviousStatus to identify the previous status of the item being annotated. Values for CVRComponentStatus come from the ContestSelectionStatus, BallotStatus, ContestStatus, and MarkStatus enumerations.

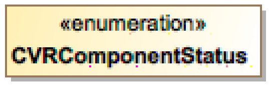


Figure 42 - CVRComponentStatus

4.21 Enumeration HashType

Used in Hash.Type to indicate the type of hash being used for an image file.

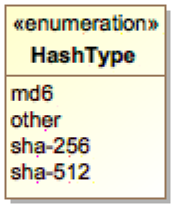


Figure 43 - HashType

Enumeration Values:

Value	**Value Description**
md6	To indicate that the MD6 message digest algorithm is being used.
other	Used in conjunction with Hash.OtherType when no other value in this enumeration applies.
sha-256	To indicate that the SHA 256-bit signature is being used.
sha-512	To indicate that the SHA 512-bit (32-byte) signature is being used.

4.22 Enumeration IdentifierType

Used in Code.Type to indicate the type of code/identifier being used.

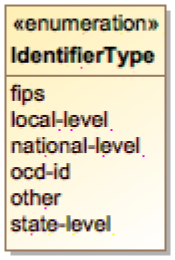


Figure 44 - IdentifierType

Enumeration Values:

Value	Value Description
fips	To indicate that the identifier is a FIPS code.
local-level	To indicate that the identifier is from a local-level scheme, i.e., unique to a county or city.
national-level	To indicate that the identifier is from a national-level scheme other than FIPS or OCD-ID.
ocd-id	To indicate that the identifier is from the OCD-ID scheme.
other	Used in conjunction with Code.OtherType when no other value in this enumeration applies.
state-level	To indicate that the identifier is from a state-level scheme, i.e., unique to a particular state.

4.23 Enumeration MarkStatus

Used in Mark.Status to identify the status of a vote mark.

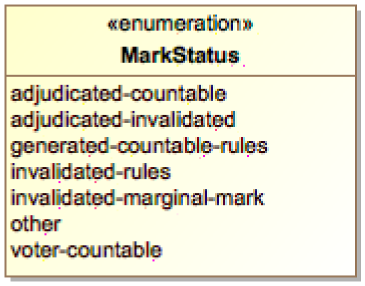


Figure 45 - MarkStatus

Enumeration Values:

Value	Value Description
adjudicated-countable	To indicate that the vote mark was adjudicated and is countable, i.e., can be subsequently counted.
adjudicated-invalidated	To indicate that the vote mark was invalidated because of adjudication and is not countable.
generated-countable-rules	To indicate that the vote mark was generated by the generating device per contest rules and is countable.
invalidated-rules	Used to indicate that the vote mark was invalidated by the generating device because of contest rules and is not countable.
invalidated-marginal-mark	To indicate that the vote mark contains a marginal mark by the voter and is not countable.
voter-countable	To indicate that the vote mark was made by the voter and is recognized as valid and is countable.
other	Used in conjunction with Mark.OtherStatus when no other value in this enumeration applies.

4.24 Enumeration ReportingUnitType

Used in GpUnit.Type to indicate the type of political geography, e.g., that is being referenced.

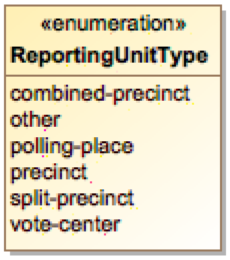


Figure 46 - ReportingUnitType

Enumeration Values:

Value	Value Description
combined-precinct	To indicate a combined precinct.
other	Used in conjunction with GpUnit.OtherType when no other value in this enumeration applies.
polling-place	To indicate a polling place.
precinct	To indicate a precinct.
split-precinct	To indicate a split-precinct.
vote-center	To indicate a vote-center.

4.25 Enumeration ReportType

Used in CastVoteRecordReport.ReportType to indicate the type of the CVR report.

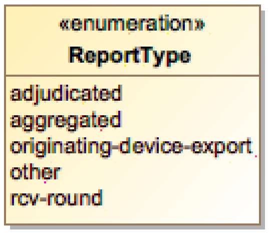


Figure 47 - ReportType

Enumeration Values:

Value	Value Description
adjudicated	To indicate that the report contains adjudications.
aggregated	To indicate that the report is an aggregation of device reports.
originating-device-export	To indicate that the report is an export from a device such as a scanner.
other	Used in conjunction with CVR.OtherReportType when no other value in this enumeration applies.
rcv-round	To indicate that the report is the result of a ranked choice voting round.

4.26 Enumeration VoteVariation

Used in Contest.VoteVariation to indicate the vote variation (vote method) used to tabulate the contest.

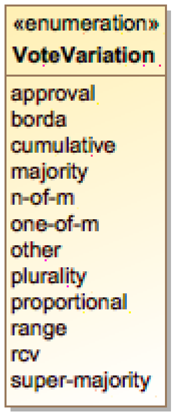


Figure 48 - VoteVariation

Enumeration Values:

Value	Value Description
approval	To indicate approval voting.
borda	To indicate the borda count method.
cumulative	To indicate cumulative voting.
majority	To indicate majority voting.
n-of-m	To indicate the N of M voting method.
one-of-m	To indicate the 1 of M voting method.
other	Used in conjunction with Contest.OtherVoteVariation when no other value in this enumeration applies.
plurality	To indicate plurality voting.
proportional	To indicate proportional voting.
range	To indicate range voting.
rcv	To indicate Ranked Choice Voting (RCV).
super-majority	To indicate the super majority voting method.

5 XML Usage Examples

This section contains several examples of CVR reports using XML. Equivalent JSON examples can be found in the locations specified in Appendix D.

5.1 Minimal CVR Example

This specification permits a wide range of data to be stored in a CVR, ranging from minimal information about the voted contests and contest selections to expanded information about all contests on the ballot as well as other items. This section shows a minimal CVR containing only the voted contests and candidates that were selected by the voter. It contains two CVRs, each indicating a selection for a candidate in the contest. Each CVR also references an image of the corresponding scanned ballot.

The file is divided roughly into two parts: the CVR elements at the beginning followed by other elements for defining the contests, candidates, and contest selections so that the CVR elements can link to them as necessary. Lines 26-43 contain the contest, candidate, and contest selection definitions. The `CVR` elements link to these items by using identifiers defined in the contest, candidate, and contest selection "`ObjectId`" attributes. For example, the contest definition starting on line 28 contains

```
<Contest ObjectId="_C1" xsi:type="CandidateContest">
```

so that `CVR` elements can link to this contest definition by using the ID "`_C1`".

Note that the object identifiers are not the same as the external identifiers that a jurisdiction may use to identify contests or candidates. The object identifiers are entirely unique to a CVR report; the exporting application must add them as it builds the report file. These identifiers are used only as a means for linking contest, contest selections, etc., together within the report file.

Lines 2-25 contain the CVR elements. Each `CVR` element includes `ContestLink`, which links to the voted contest whose object identifier is "`_C1`", thereby uniquely identifying that contest within the report file. It then includes `ContestSelectionLink`, which links to a contest option that was selected by the voter. Each `CVR` element also includes an optional sequence number; this isn't required but could be helpful to auditors.

Lines 45-47 contain a definition for the device that generated the CVRs. This indicates that a device located at a precinct whose serial number is "1038495" generated the collection of CVRs.

Lastly, lines 3-5 and lines 14-17 contain optional information about an image per ballot saved by the scanner. For example, the image of the ballot corresponding to the first CVR is filename "CVR1_Ballot.jpg" and the image type is JPEG.

```
<CastVoteRecordReport>
  <CVR>
    <BallotImage>
      <Image FileName="CVR1_Ballot.jpg" MimeType="image/jpeg"></Image>
    </BallotImage>
    <ContestLink>
      <ContestId>_C1</ContestId>
      <ContestSelectionLink>
        <ContestSelectionId>_C1CS1</ContestSelectionId>
      </ContestSelectionLink>
    </ContestLink>
    <SequenceNumber>1</SequenceNumber>
  </CVR>
  <CVR>
    <BallotImage>
      <Image FileName="CVR2_Ballot.jpg" MimeType="image/jpeg"></Image>
    </BallotImage>
    <ContestLink>
      <ContestId>_C1</ContestId>
      <ContestSelectionLink>
        <ContestSelectionId>_C1CS2</ContestSelectionId>
      </ContestSelectionLink>
    </ContestLink>
    <SequenceNumber>2</SequenceNumber>
  </CVR>
  <Election ObjectId="_EL7">
    <Candidate ObjectId="_C1CN1">
      <Code><Type>local-level</Type><Value>_C1CN1</Value></Code>
    </Candidate>
    <Candidate ObjectId="_C1CN3">
      <Code><Type>local-level</Type><Value>_C1CN3</Value></Code>
    </Candidate>
    <Code><Type>local-level</Type><Value>EL7</Value></Code>
    <Contest ObjectId="_C1" xsi:type="CandidateContest">
      <Code><Type>local-level</Type><Value>C1</Value></Code>
      <ContestSelection ObjectId="_C1CS1" xsi:type="CandidateSelection">
        <CandidateIds>_C1CN1</CandidateIds>
      </ContestSelection>
      <ContestSelection ObjectId="_C1CS2" xsi:type="CandidateSelection">
        <CandidateIds>_C1CN3</CandidateIds>
      </ContestSelection>
    </Contest>
  </Election>
  <GeneratedDate>2018-05-15T17:32:52</GeneratedDate>
  <GpUnit ObjectId="_GP1" xsi:type="ReportingDevice">
    <Type>precinct</Type><SerialNumber>1038495</SerialNumber>
  </GpUnit>
  <ReportGeneratingDeviceIds>_GP1</ReportGeneratingDeviceIds>
  <Version>1.0</Version>
</CastVoteRecordReport
```

5.2 Expanded Example Containing Vote Marks

The previous example did not contain any detailed information about the actual marks made by the voters. If the CVRs were generated as a result of using an electronic voter interface such as provided by a ballot marking device that encodes the voter's selections in a barcode, this would be appropriate; there are no actual marks and all that is needed is an indication of a voted contest/candidate. If the CVRs were generated by a ballot scanner and hand-marked paper ballots, then including some information about the voter marks may be advisable. The `Mark` element effectively represents a "bubble" on a paper ballot and for each contest there are potentially as many bubbles are there are candidate options in the contest. For example, in a contest with one ballot selection containing three bubbles representing 3 candidates where only one can be chosen, a voter could potentially overvote and mark all three bubbles.

This section shows a CVR containing the contest selection in which two of the bubbles have been marked, one being a valid and countable mark, and the other being a marginal mark. This CVR is followed by another CVR containing a contest selection in which two of the bubbles have been marked, both being valid and countable marks, however because the contest rules permit only one mark to be selected, both marks represent overvotes.

In the first CVR, lines 6-17 show two `Mark` elements, indicating that the voter made marks in two bubbles (here, bubbles in positions 1 and 3). The first mark has a quality measurement of type AJAX (a fictional quality measurement) and quality score of 95 (0 is the worst, 100 is the best).

The second mark has a quality score of 13. The scanner ruled the first mark as a valid and countable mark, and the second mark as a marginal mark. In line 18 the scanner ruled that the contest selection represents one vote, according to the following logic:

1. The first mark's quality score of 95 meets or exceeds a manufacturer-defined threshold for a valid and countable mark.
2. The second mark's quality score of 13 is below the manufacturer-defined threshold and has been ruled as a marginal mark.

In the second CVR, lines 28-39 show two `Mark` elements representing marked bubbles in positions 2 and 3. Both have quality scores of 96 and 98 respectively, thus the marks meet or exceed the the scanner's threshold for a valid and countable mark. However, the contest allows one option to be selected, not two, thus the voter apparently overvoted the contest. Accordingly, the scanner ruled both marks as invalidated according to contest rules. In line 40, the scanner ruled the contest selection as overvoted and as representing zero votes. In line 43, the scanner ruled the entire contest as overvoted.

```
1   <CVR>
2     <ContestLink>
3       <ContestId>_C1</ContestId>
```

```
      <ContestSelectionLink>
        <ContestSelectionId>_C1CS1</ContestSelectionId>
        <Mark>
          <MarkMetric><Type>AJAX</Type>Value>95</Value></MarkMetric>
          <NumberVotes>1</NumberVotes>
          <Position>1</Position>
          <Status>voter-countable</Status>
        </Mark>
        <Mark>
          <MarkMetric><Type>AJAX</Type><Value>13</Value></MarkMetric>
          <NumberVotes>0</NumberVotes>
          <Position>3</Position>
          <Status>invalidated-marginal-mark</Status>
        </Mark>
        <TotalNumberVotes>1</TotalNumberVotes>
      </ContestSelectionLink>
    </ContestLink>
    <SequenceNumber>1</SequenceNumber>
  </CVR>
  <CVR>
    <ContestLink>
      <ContestId>_C1</ContestId>
      <ContestSelectionLink>
        <ContestSelectionId>_C1CS2</ContestSelectionId>
        <Mark>
          <MarkMetric><Type>AJAX</Type><Value>96</Value></MarkMetric>
          <NumberVotes>1</NumberVotes>
          <Position>2</Position>
          <Status>invalidated-rules</Status>
        </Mark>
        <Mark>
          <MarkMetric>Type>AJAX</Type><Value>98</Value></MarkMetric>
          <NumberVotes>0</NumberVotes>
          <Position>3</Position>
          <Status>invalidated-rules</Status>
        </Mark>
        <Status>overvoted</Status>
        <TotalNumberVotes>0</TotalNumberVotes>
      </ContestSelectionLink>
      <Status>overvoted</Status>
    </ContestLink>
    <SequenceNumber>2</SequenceNumber>
  </CVR>
```

5.3 Example Showing an Adjudicated Write-in

This section shows a CVR containing a write-in from a paper ballot that has subsequently been adjudicated as being valid and countable. The scanner created an image of the write-in section of the ballot and saved it as an image file. In the CVR, the scanner included the name of the file, “CVR61WRTIN2.JPG” and the file type of “image/jpeg”. To add integrity to the CVR’s link to the file, the scanner created a hashed value of the file and added it to the CVR.

Because the scanner does not include OCR capability for write-in images, it added `needs-adjudication` to the contest selection status. An adjudicator will thus need to examine the image and make a determination as to its validity.

The example includes the Annotation element in lines 5-10. This presumes that the following actions have taken place:

1. An adjudicator named “Mark Kenamond”, line 6, used an adjudication program to examine the write-in image and determine whether the hash value stored in the CVR is valid, which it is.
2. The adjudicator determined that the voter’s write-in is a valid write-in, line 7. The adjudicator, in their message, also included the name from the write-in image, Donny Nolte.
3. The adjudicator added the corrected name of Donald Nolte in line 14.
4. The adjudicator added that the contest selection represents one vote, line 13.
5. The adjudicator added the previous status of `needs-adjudication` to the `PreviousStatus` element, line 8.
6. The adjudicator changed the status of the contest selection to `adjudicated`, line 12, and added a time stamp, line 9.

A write-in from an electronic voter interface such as provided by a ballot marking device may not require an image if the CVR is generated directly by the interface device or indirectly via an encoding of the voter’s selections such with a barcode. In this case, the device can add the text of the write-in directly to the CVR. Thus, there would be no need to use the WriteInImage element, lines 15-24. Depending on local election procedures, the CVR may still require inspection and adjudication.

The above steps may differ across jurisdictions depending on how the adjudication process works in each jurisdiction. Thus, many of the elements are optional.

```
1   <CVR>
2       <ContestLink>
3           <ContestId>_C1</ContestId>
4           <ContestSelectionLink xsi:type="WriteIn">
5               <Annotation>
6                   <AdjudicatorName>Mark Kennamond</AdjudicatorName>
7                   <Message>Valid, Donny Nolte</Message>
8                   <PreviousStatus>needs-adjudication</PreviousStatus>
9                   <TimeStamp>2018-05-16T12:10:09</TimeStamp>
```

```
                </Annotation>
                <ContestSelectionId>_C1CS1</ContestSelectionId>
                <Status>adjudicated</Status>
                <TotalNumberVotes>1</TotalNumberVotes>
                <Text>Donald Nolte</Text>
                <WriteInImage>
                    <Hash>
                        <Type>sha-512</Type>
                        <Value>156CEAE1FE594978F870E7D80DA8BAD838D9C
                            060EBBFC6521AAA93E5776250BB2C21350192B36
                            10677B988A8AF808827B50A33624FEBBB4C2068E
                            5470C830174</Value>
                    </Hash>
                    <Image FileName="CVR1WRTIN2.JPG" MimeType="image/jpeg"/>
                </WriteInImage>
            </ContestSelectionLink>
        </ContestLink>
        <SequenceNumber>61</SequenceNumber>
    </CVR>
```

5.4 Example Using the Ranked Choice Voting Method

This section describes the structure of a CVR when using Ranked Choice Voting (RCV). It does not directly address other voting methods such as range or cumulative, however the CVR for these voting methods is somewhat similar to that of RCV.

The example in this section is taken from section 3.2.1, showing a CVR that includes a ranked choice voting (RCV) contest in which 15 candidates can be ranked according to a voter's preference. As shown in Figure 8 of section 3.2.1, each of the 15 possible contest selections includes a series of bubbles, each bubble representing a distinct rank for that contest selection. Selecting more than one bubble is considered an overvote and invalidates that contest selection.

For brevity, this example shows one CVR with 3 ballot selections, as follows:

```
<CVR>
    <ContestLink>
        <ContestId>_C1</ContestId>
        <ContestSelectionLink>
            <ContestSelectionId>_C1CS1</ContestSelectionId>
            <Mark>
                <Position>3</Position>
                <Rank>3</Rank>
            </Mark>
        </ContestSelectionLink>
        <ContestSelectionLink>
            <ContestSelectionId>_C1CS2</ContestSelectionId>
            <Mark>
                <Position>2</Position>
                <Rank>2</Rank>
            </Mark>
        </ContestSelectionLink>
        <ContestSelectionLink>
            <ContestSelectionId>_C1CS3</ContestSelectionId>
            <Mark>
                <Position>4</Position>
                <Rank>4</Rank>
            </Mark>
            <Mark>
                <Position>1</Position>
                <Rank>1</Rank>
            </Mark>
            <Status>overvoted</Status>
        </ContestSelectionLink>
    </ContestLink>
    <SequenceNumber>57</SequenceNumber>
</CVR>
```

Some more description….

5.5 Example Using Auditing-Related Features

Goes over use of elements to enable better auditing.

Appendix A—Acronyms

Selected acronyms and abbreviations used in this document are defined below.

CDF	Common Data Format
EAC	Election Assistance Commission
FIPS	Federal Information Processing Standard
JSON	JavaScript Object Notation
NIST	National Institute of Standards and Technology
OCD-ID	Open Civic Data Identifiers
SP	Special Publication
UML	Unified Modeling Language
UOCAVA	Uniform and Overseas Citizens Assistance in Voting Act
VVSG	Voluntary Voting Systems Guidelines
XML	eXtensible Markup Language

Appendix B—Glossary

Selected terms used throughout this document are defined below. In some of the definitions, there is ancillary information that is not part of the definition but helpful in understanding the definition; this ancillary information is preceded with "*Note:*". Synonyms are preceded with "*Syn:*".

Appendix C—References

[1] W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation, November 26, 2008, http://www.w3.org/TR/xml/ [accessed 2/1/2016].

[2] JavaScript Object Notation, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf [accessed 2/1/2016].

[3] Election Assistance Commission, *Election Administration and Voting Survey*, http://www.eac.gov/research/election_administration_and_voting_survey.aspx [accessed 2/1/2016].

[4] Object Management Group (OMG), *UML Specification version 1.1* (OMG document ad/97-08-11) September 22, 2011, http://omg.org/ [accessed 2/1/2016].

[5] Open Civic Data, *OCD Identifiers*, http://opencivicdata.readthedocs.org/en/latest/ocdids.html [accessed 2/1/2016].

Appendix D—File Download Locations

The files associated with this specification are available for download from a NIST repository, whose address is:

http://vote.nist.gov

These files are:

- This specification,
- UML model,
- XML and JSON schemas,
- Example files, and
- Validation tools.