

# **NIST Special Publication 1500-103**

---

## **Cast Vote Records Common Data Format Specification**

**Version 1.0**

---

John Wack  
Sam Dana  
Herb Deutsch  
John Dziurlaj  
Ian Piper

This publication is available free of charge from:  
<https://doi.org/10.6028/NIST.SP.1500-103>



# NIST Special Publication 1500-103

## Cast Vote Records Common Data Format Specification

Version 1.0

John Wack  
*Software and Systems Division  
Information Technology Laboratory, NIST*

Sam Dana  
*Prometheus Computing*

Herb Deutsch  
*Election System Software*

John Dziurlaj  
*Democracy Fund*

Ian Piper  
*Dominion Voting*

This publication is available free of charge from:  
<https://doi.org/10.6028/NIST.SP.1500-103>

November 2019  
INCLUDES UPDATES AS OF 03-31-2020; SEE APPENDIX E



U. S. Department of Commerce  
*Wilbur Ross, Secretary*

National Institute of Standards and Technology  
*Walter G. Copan, Under Secretary of Commerce for Standards and Technology and Director*

National Institute of Standards and Technology (NIST) Special Publication 1500-103, Version 1.0  
94 pages (November 2019)

NIST Special Publication series 1500 is intended to capture external perspectives related to NIST standards, measurement, and testing-related efforts. These external perspectives can come from industry, academia, government, and others. These reports are intended to document external perspectives and do not necessarily represent official NIST positions.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST publications are available at <http://www.nist.gov/publication-portal.cfm>.

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](mailto:voting@nist.gov)

## **Reports on Computer Systems Technology**

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. This document reports on ITL's research, guidance, and outreach efforts in Information Technology and its collaborative activities with industry, government, and academic organizations.

### **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 JavaScript Object Notation (JSON), unified markup language (UML), and eXtensible Markup Language (XML).

### **Keywords**

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

## Acknowledgements

The authors wish to thank their colleagues of the National Institute of Standards and Technology Voluntary Voting System Guidelines (VVSG) Interoperability Public Working Group, who reviewed drafts of this document and contributed to its technical content. The authors gratefully acknowledge and appreciate the following contributors for their keen and insightful assistance with developing this specification:

Kenneth Bennett  
*Office of Registrar-Recorder  
/County Clerk, Los Angeles*

Kim Brace  
*Election Data Services*

George Gilbert  
*Ranked Choice Voting  
Resource Center*

Katy Owens Hubler  
*Democracy Research*

Jared Marcotte  
*The Turnout*

Lauren Massa-Lochridge  
*llochridge & associates*

Neal McBurnnett  
*ElectionAudits*

John McCarthy  
*Verified Voting*

Jennifer Morrell  
*Democracy Fund*

Sarah Whitt  
*Electronic Registration  
Information Center*

In addition to the above acknowledgments, the authors also gratefully acknowledge and appreciate the National Institute of Standards and Technology's Mary Brady and Benjamin Long for their exceptional contributions in helping to improve the content of the publication. And finally, the authors also gratefully acknowledge and appreciate the significant contributions from individuals and organizations 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 selections, with usually one CVR created per sheet (page) of a ballot. Election results are produced by tabulating the collection of CVRs, and audits can be done by comparisons of the paper ballots or paper records of voter selections against the CVRs.

This specification supports three 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 JavaScript Object Notation (JSON) and eXtensible Markup Language (XML) 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 created from the UML model.

There are many complex operations performed by voting devices when voters submit their paper ballots to be scanned. These operations are mostly invisible to voters but are necessary to determine whether contest selections have been marked adequately and whether voter intent is reflected by what is marked on the ballot. This specification includes the necessary detail to capture these operations so that CVRs can be better audited and adjudicated as necessary to include write-in candidates or other issues.

This specification is geared towards the following audiences:

- Election officials
- Voting equipment manufacturers
- Election analysts and auditors
- Election-affiliated organizations
- The public

## Table of Contents

<b>Executive Summary .....</b>	<b>iii</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Why this specification is needed .....	1
1.2 Intended Audience .....	1
1.3 Document Structure .....	2
1.4 Motivation and Methodology .....	2
<b>2 Background: Cast Vote Record Creation, Contents, and Handling.....</b>	<b>3</b>
2.1 Overview of Cast Vote Records and their Generation .....	3
2.2 Counting Cast Vote Records .....	4
2.3 Adjudication of Cast Vote Records .....	4
2.4 Auditing Cast Vote Records.....	5
<b>3 Cast Vote Record UML Model Overview .....</b>	<b>6</b>
3.1 CVR Report General Structure .....	6
3.2 Interoperability Considerations .....	6
3.3 Cast Vote Record Snapshots.....	9
3.4 Representing Contest Selections and Selection Positions.....	10
3.4.1 Representing the Facts About a Contest Selection .....	12
3.4.2 HasIndication and IsAllocable .....	13
3.4.3 Ranked Choice Voting Example .....	14
3.4.4 Cumulative Voting Example .....	15
3.4.5 Range Voting Example.....	16
3.5 Identifiers Within the CVR.....	18
3.5.1 A Unique ID for Each CVR .....	18
3.5.2 An ID for Each Sheet of a Multi-sheet Paper Ballot.....	18
3.5.3 An ID for Ballots Pre-Printed with a Unique Identifier .....	18
3.5.4 An ID to Link the CVR to its Corresponding Paper Ballot.....	18
3.5.5 IDs to Support Batching.....	19
3.5.6 IDs via Associations with Other Classes .....	19
<b>4 Cast Vote Record UML Model Documentation .....</b>	<b>20</b>
4.1 Class Annotation.....	20
4.2 Class BallotMeasureContest.....	21
4.3 Class BallotMeasureSelection .....	22

4.4	Class Candidate .....	23
4.5	Class CandidateContest .....	24
4.6	Class CandidateSelection .....	25
4.7	Class CastVoteRecordReport .....	26
4.8	Class Code .....	28
4.9	Class Contest .....	29
4.10	Class ContestSelection .....	30
4.11	Class CVR .....	31
4.12	Class CVRCContest .....	33
4.13	Class CVRCContestSelection .....	35
4.14	Class CVRSnapshot .....	37
4.15	Class CVRWriteIn .....	38
4.16	Class Election .....	39
4.17	Class File .....	40
4.18	Class FractionalNumber .....	41
4.19	Class GpUnit .....	42
4.20	Class Hash .....	43
4.21	Class Image .....	44
4.22	Class ImageData .....	45
4.23	Class Party .....	46
4.24	Class PartyContest .....	47
4.25	Class PartySelection .....	48
4.26	Class ReportingDevice .....	49
4.27	Class RetentionContest .....	50
4.28	Class SelectionPosition .....	51
4.29	Enumeration AllocationStatus .....	53
4.30	Enumeration CastVoteRecordVersion .....	54
4.31	Enumeration ContestSelectionStatus .....	55
4.32	Enumeration ContestStatus .....	56
4.33	Enumeration CVRStatus .....	57
4.34	Enumeration CVRTypE .....	58
4.35	Enumeration HashType .....	59
4.36	Enumeration IdentifierType .....	60

4.37 Enumeration IndicationStatus .....	61
4.38 Enumeration PositionStatus.....	62
4.39 Enumeration ReportingUnitType.....	63
4.40 Enumeration ReportType.....	64
4.41 Enumeration VoteVariation .....	65
<b>5 Usage Examples.....</b>	<b>66</b>
5.1 Anatomy of a CVR .....	66
5.2 Basic Example .....	67
5.3 SelectionPosition .....	68
5.3.1 Position and Rank in SelectionPosition .....	68
5.3.2 Voter Made Marks (Paper Only).....	69
5.3.3 Machine Generated Indications .....	70
5.4 Handling Overvotes .....	70
5.5 Write-Ins .....	71
5.5.1 Write-In Counter .....	72
5.5.2 Adjudication of Write-Ins.....	72
5.6 CVR Snapshots .....	72
5.6.1 Current CVR.....	74
5.7 Ballot Images .....	74
5.7.1 Storing the Image as a Reference .....	74
5.7.2 Storing the Image Data.....	74
5.8 Voting Method Support .....	75
5.8.1 Rank Choice Voting.....	75
5.8.2 Cumulative Voting .....	76

### List of Appendices

<b>Appendix A— Acronyms .....</b>	<b>79</b>
<b>Appendix B— Glossary .....</b>	<b>80</b>
<b>Appendix C— References .....</b>	<b>83</b>
<b>Appendix D— File Download Locations.....</b>	<b>84</b>
<b>Appendix E— Change Log .....</b>	<b>85</b>

## 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 describes a CVR UML (Unified Modeling Language) [1] model and XML (eXtensible Markup Language) [2] and JSON (JavaScript Object Notation) [3] schemas that were generated from the model.

The primary features of this specification include:

- The capability to export raw CVRs representing contest selections and other information such as changes made to the CVRs by the vote-capturing device because of election or contest rules and changes made to the CVRs as a result of adjudication.
- The capability, for a single election, to contain collections of CVRs produced by multiple devices from multiple locations, such as voting centers or precincts.
- A data model in UML that itemizes and defines the data involved in CVRs and that is used to derive the XML and JSON schemas.

### 1.1 Why this specification is needed

The purpose of this specification is to provide interoperable data interchange formats in XML and JSON for CVRs to assist election officials, auditors, and other election analysts in collecting, aggregating, tabulating, and auditing CVRs from multiple types of vote-capture devices. An additional purpose 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 use 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.
- A UML model that is easily extensible to additional use cases.

### 1.2 Intended Audience

The intended audience of this specification includes election officials, voting system designers and developers, and 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, Background: Cast Vote Record Creation, Contents, and Handling, contains background information about how CVRs get created, their contents, and how they are handled in the election process.
- Section 3, Cast Vote Record UML Model Overview, contains an overview of the UML model structure and how it can be used for CVR exports and reports.
- Section 4, Cast Vote Record UML Model Documentation, describes the classes and enumerations in greater detail.
- Section 5, Usage Examples, contains examples of CVR structure using XML.

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

### 1.4 Motivation and Methodology

This document was written primarily to assist election officials, developers, and auditors in handling CVRs as they are created and used. At the time of writing, voting systems and the data produced do not interoperate unless they are from the same manufacturer, but even within a manufacturer's line of products there is often a lack of interoperability. This adds more complexity when attempting to integrate COTS (commercial off-the-shelf) or other manufacturer equipment for tabulations, adjudications, and audits.

NIST and a community of U.S. election officials, analysts, manufacturers, and election system technologists analyzed how CVRs are used within the election process and produced this interoperable CVR format that can be used regardless of manufacturer. This specification addresses the following use cases:

1. Interoperable exports of CVRs from devices such as scanners for import into tabulators, 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, so scanners and other devices can export CVRs in JSON or XML and validate usage against the schema.

## 2 Background: Cast Vote Record Creation, Contents, and Handling

This section contains a general overview of how CVRs are created, their contents, and how they are subsequently handled in the election process.

### 2.1 Overview of Cast Vote Records and their Generation

Simply put, a cast vote record (CVR) is an electronic record of a voter's ballot selections, and its primary purpose is to provide a record of voter selections that can be counted in an efficient manner to produce election results. A CVR is created by equipment such as a voter facing scanner in a polling place into which a voter inserts a paper ballot. CVRs also get created by batch fed scanners used to scan absentee or other types of ballots that are collected before the election or that cannot be scanned by polling place scanners for various reasons. After the polls are closed, the CVRs are collected by election officials on memory devices and subsequently copied to an election management system that aggregates and tabulates the votes.

Three primary types of voting devices that create CVRs are:

- All-electronic voting devices that a voter uses to make ballot selections and that create and store a CVR for each ballot.
- Ballot marking devices (BMDs) that function like all-electronic devices but that produce a paper record of the voter's choices that must be subsequently scanned.
- Voter-facing optical scanners used in polling places and batch-fed optical scanners used in central offices to scan paper ballots.

The scanning devices above are sometimes referred to collectively as "tabulators" because they generally have a tabulation capability, but this is not always the case.

CVRs may include other information besides voter choices, including:

- Information on all contests and contest options 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 printed 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 scanner must interpret the voter's selections according to the rules of each contest to determine which selections can be counted. This is true primarily of hand-marked paper ballot scanners in which voters may write in candidates whose names were not on the ballot or they may make mistakes that invalidate their choices such as overvoting in a contest. All-electronic devices and BMDs guide the voter how to make selections according to the contest rules, thus the CVRs they create require much less interpretation, except for write-ins.

When creating a cast vote record, ballot scanners must first interpret the ballot and detect where voters have made marks and whether those marks meet manufacturer-specific criteria for validity, that is, whether a mark is placed in the right location and is sufficiently formed so it constitutes an intentional ballot selection made by the voter. Each device in the jurisdiction must also be programmed with election specific information for the polling place that it will be used in, so the scanners can apply that polling place's election rules to the detected marks. Scanners may perform interpretation based on a number of different factors, including:

- A scanner may flag marks as being marginal, that is not meeting the criteria for validity, and therefore not counting those marks.
- Voters may vote for more than the allowable number of options (overvote) and the scanner must know not to count *any* of the selections made by the voter for that particular contest.
- Likewise, voters may undervote a contest, and the scanner must record that the contest was undervoted. The marked selections are still counted.
- The scanner must be programmed to tabulate the voter marks according to the voting method in place for a particular contest.
- A scanner may create indirect selections 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 the entire ballot could be invalidated.

Typically, CVR-generating equipment will export a collection of CVRs that may also include a tabulated report. 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 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 EMS by election officials. Write-ins are the most common reason:

1. On ballots produced by BMDs the write-in names could still be spelled differently or incorrectly, and
2. For scanned paper ballots, either the ballots themselves or the images of the write-in areas of the ballot that were made by the scanner must be inspected.

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, for example, the voter may have circled the ovals instead of filling them in.
- The scanner detected one or more overvotes.
- The scanner detected that the entire ballot was blank.

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).

This is discussed in more detail in Section 4.1, Class Annotation.

## 2.4 Auditing Cast Vote Records

CVRs need to be audited against their paper counterparts so that election results can be verified to be accurate. This specification supports auditing by providing the following as options:

- Support for ballot-level comparison auditing, that is, there is an identifier in the CVR that can be linked to an ID printed on the corresponding paper ballot.
- Support to include adjustments to contest selections made by adjudicators.
- Different snapshots of the CVR can be created, one for the original scan, one for after election rules have been applied, and others as needed for adjudications.
- Indications of marginal marks, mark quality/density can be associated with contest selections.
- A CVR can include signed/hashed references to an associated image of the ballot or images of write-ins made by the voter.
- Capability to include batch information such as batch IDs and sequence within the batch.

### 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, Cast Vote Record UML Model Documentation, contains information on specific classes and enumerations, and Section 5, Usage Examples, discusses several CVR examples using 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 class and enumeration names are shown in a different font to distinguish them from surrounding text. For example, the name of the root class in the UML model is `CastVoteRecordReport`.

#### 3.1 CVR Report General Structure

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

- Creates CVRs, such as a scanner or BMD, and/or
- Processes and generates a report of CVRs such as an EMS.

In the case of an EMS, the report could consist of aggregations of CVRs from multiple creating devices and, if desired, multiple creating locations, i.e., polling places. In an election conducted in a large county, for example, there could be many CVR reports exported from creating devices (i.e., scanners), and the county could import these reports into a central EMS and issue potentially one single aggregated report.

The CVRs in the report can each be associated with the following:

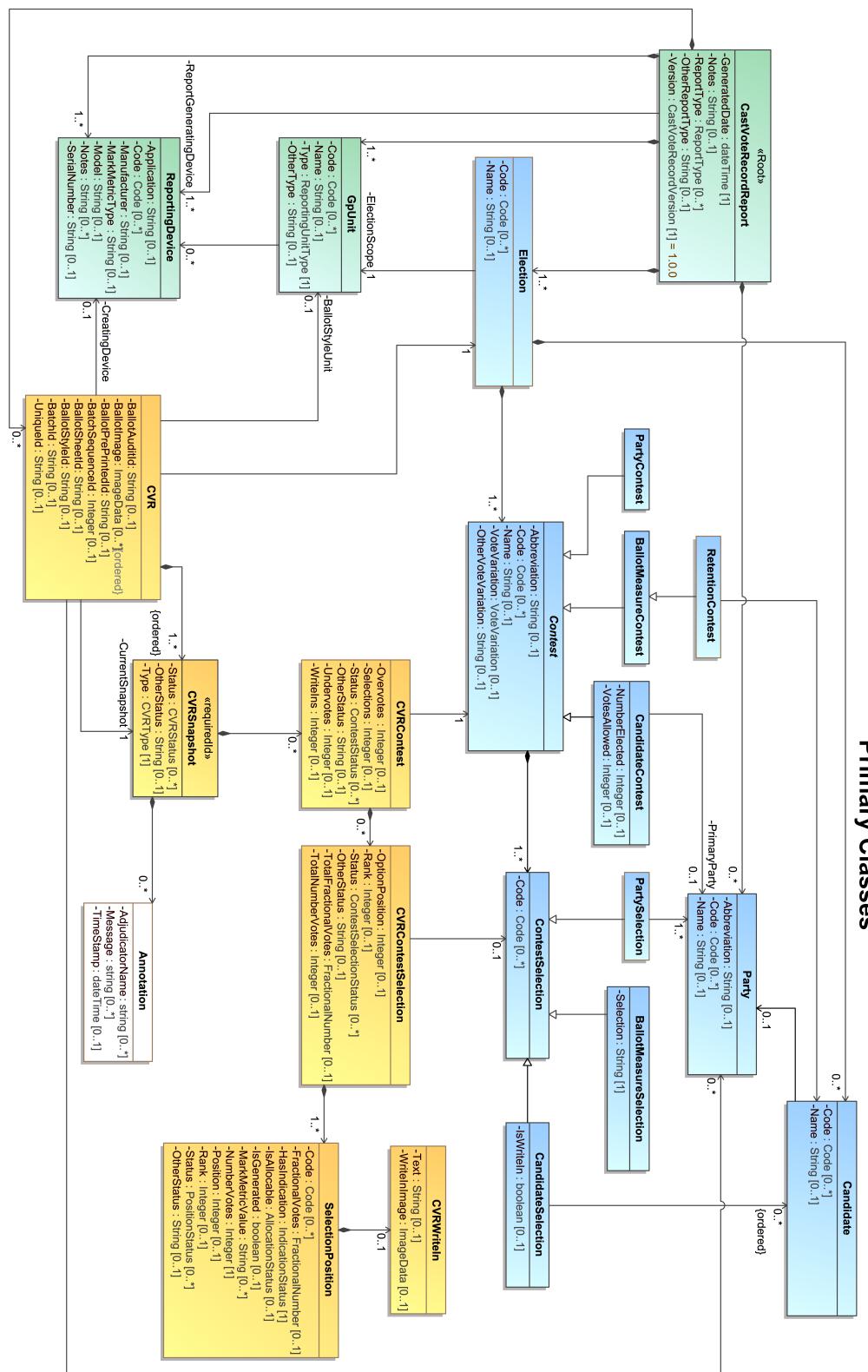
- The election,
- Precinct or split-precinct geography that corresponds to the ballot style, or
- Creating device.

As well, the report itself can identify the election(s), where the CVRs were created and the creating device(s).

Figures 1 and 2 show the classes in the UML model and the enumerations used in a number of class attributes.

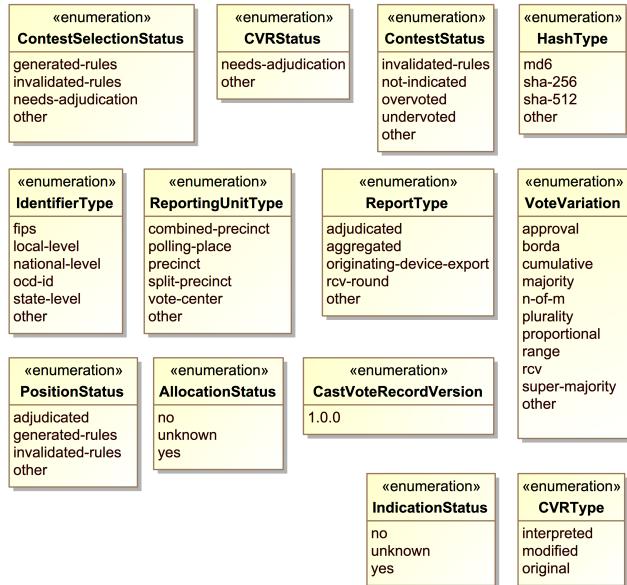
#### 3.2 Interoperability Considerations

A major impediment to interoperability of CVRs across different manufacturers is that typically, the contest selections in a manufacturer's CVR consist of codes and the CVR thus appears as a structured series of codes and contest votes. To understand the CVR, one must know what the codes mean and their structure, and this information is usually opaque and unavailable to others



**Figure 1 - CVR UML Model - Classes**

## Enumerations



## Secondary Classes

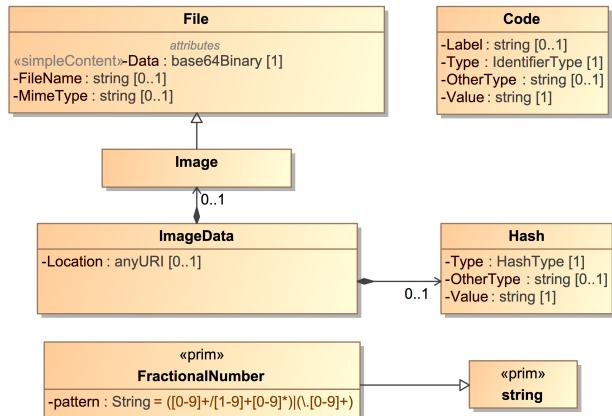


Figure 2 - CVR UML Model - Enumerations and Other Classes

(including auditors)<sup>1</sup>. An important feature of the UML model is that it allows for expanded information in the CVR to be included, in a space-saving manner, to identify contests and contest selections and the associated indications of marks. While a manufacturer's codes can still be included, one can also include interoperable codes and descriptions so that others can understand the CVR and CVRs can be exchanged across devices and manufacturers. The UML model also allows for all contests and contest options, whether selected or not, to be included in the CVR if desired.

<sup>1</sup> There were space-savings benefits to using structured codes with older devices that had less memory than today's devices.

However, it would be inefficient and wasteful of space to repeat static and identical contest or candidate information in each CVR, and it would be better to define this static information only once and then point to it as needed. To effect this, the model treats the static information about contests, contest options, devices, and political geography as static *objects* that, once defined, can be linked and connected with the dynamic contest selection information. A static object in concrete terms means a definition in XML/JSON for the item consisting solely of static attributes related to the item, i.e., a contest code ID used in the election or a contest or candidate name, neither of which will change within the context of the CVR report.

- Date/Time of report generation
- Definitions for
  - Device generating the report and its location (e.g., precinct, county office, etc.)
  - Election associated with the collection of CVRs
  - CVR creating device(s) in the report
  - Location of creating device(s) in the report (e.g., vote center)
  - Identification of the ballot style area(s) associated with CVRs in the report (e.g., the precincts/splits served by the vote center)
  - Contest option object definitions referenced as needed by the CVRs
    - **CVR 1:**
      - Corresponding ballot style and other audit-related details
      - CVRSnapshot
        - Type – original or interpreted or modified
        - Is this the snapshot to be tabulated? Yes/No
        - Links to contest option and contest selection objects defined previously
        - Indications associated with the contest selections
      - CVRSnapshot (if there are more)
    - **CVR 2 (and subsequent CVRs)**

**Figure 3 - CVR report structure**

As shown in Figure 3, the CVR report structure consists of objects created for the election, the devices, political geography, contests, and contest options, followed by instances of CVR classes. As stated, each CVR can optionally include a link to the creating device, the corresponding ballot-style area, or the election.

### 3.3 Cast Vote Record Snapshots

Figure 3 shows a CVR optionally including one or more instances of CVRSnapshot. A CVRSnapshot is a record of the voter's selections at a particular point in time. There are three types of snapshots, which are listed in the CVRTyp enumeration:

- **original** – the version created when a marked paper ballot is scanned and *before* interpretation of the voter’s selections according to contest/election rules.
- **interpreted** – the version created after interpretation of the voter’s selections recorded in the **original** version, or the version created by a DRE (Direct Record Electronic) device or from a BMD’s encoded voter selections.
- **modified** – the version created after modifications have been made to the **interpreted** version as a result of adjudication, e.g., because of write-ins on the ballot.

Each instance of the snapshot includes only the contest options and contest selections that were voted and that are valid for that snapshot. One of the snapshots must be identified as the snapshot to include in tabulations. Figure 4 below shows the structure in more detail.

- **CVRSnapshot**
    - Status of the snapshot – needs adjudication, other
    - ID of an associated batch and sequence number within a batch
    - Snapshot type – original, interpreted, modified (required)
    - Annotation information (if needed)
    - **CVRContest** – link to a voted contest in this snapshot
      - **CVRContestSelection** – identifies a selection in the contest
        - **SelectionPosition** – position of the selection and other facts
        - **SelectionPosition** (if there are more in the contest)
    - **CVRContest** (if there are more in this snapshot)
  - **CVRSnapshot** (if there are more)

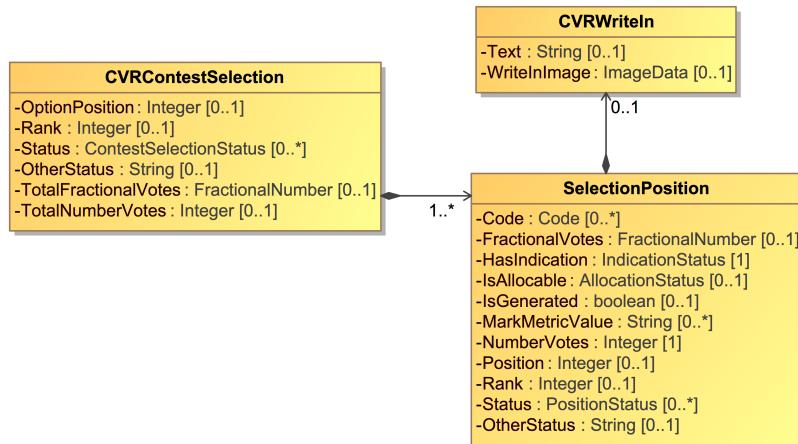
**Figure 4 - CVRSnapshot Structure**

Depending on how this specification may be implemented, all three types of snapshots could be used in a CVR, or two, or one. For CVRs created by devices such as a DRE or a BMD that encodes voter selections, there is no **original** snapshot, thus using **interpreted** makes more sense. For hand-marked paper ballots, use of the **original** and **interpreted** snapshots allows for traceability of all marks detected on the ballot and how the scanner interpreted them according to mark density values and contest rules. Generally, the **modified** snapshot would be created by adjudication applications so as to provide traceability for adjudications of voter-made marks or adjudication of write-ins.

### 3.4 Representing Contest Selections and Selection Positions

As shown in Figure 3 and Figure 4, **CVRSnapshot** includes **CVRContest** for linking contest objects to those contests that were selected by the voter. **CVRContest** then includes **CVRContestSelection** for each contest selection made by the voter. **CVRContestSelection** then links contest object definitions with the voted contest selections, e.g., for linking a candidate code within a candidate object to the voted selection for that candidate, thereby associating the code with the voted selection.

CVRContestSelection also includes an important class, SelectionPosition, shown in Figure 5, for recording information about each selected position within a contest selection such as, does the position represent a vote? The name of this class is important, because contest options on the ballot each have a position, thus each contest selection included in the CVR must be associated with that position. In practical terms, each position can be thought of as a bubble on the ballot that can be filled in by the voter. For example, each contest selection for a candidate in the Presidential contest will be associated with one position, that being the bubble next to the candidate. Accordingly, SelectionPosition will indicate position 1 for the candidate selected in the contest.



**Figure 5 - SelectionPosition Class**

Other types of contest selections may have multiple positions, such as for Ranked Choice Voting (RCV) in which each candidate could be considered as a row on a ballot, with associated rankings for each candidate, each represented as a bubble on the ballot. In this case, it is important to know the position of the selected bubble so as to know its associated ranking. Figure 6 shows an example RCV contest in which there are 3 candidates and each candidate can be ranked from 1 to 3, thus there are 3 bubbles for each candidate and three possible positions. In this example, SelectionPosition will indicate position 1 for Shapiro, 3 for Walsh, and 2 for Kurt:

### Member of County Council at Large

Contest Option	1st	2nd	3rd
Ileen Shapiro	[x]	[ ]	[ ]
Debbie Walsh	[ ]	[ ]	[x]
Sandra Kurt	[ ]	[x]	[ ]

**Figure 6 - RCV Contest with 3 Positions per Selection**

### 3.4.1 Representing the Facts About a Contest Selection

`SelectionPosition` is also used to state facts concerning the position(s) associated with contest selections, such as whether, at a position, there is an indication of a voter selection in a contest and whether it is potentially countable. For scanned paper ballots, scanner-resident software that has been programmed with the appropriate contest rules typically interprets the voter selections and makes those decisions as to

- Whether a voter mark is of sufficient quality to be considered as a contest selection, and
- Whether the contest selection obeys the rules of the contest and can be consequently counted as a vote.

However, CVRs also could be simply recorded by a scanner but then exported to another device where the interpretation takes place.

Thus, in the interests of transparency, the facts that are recorded in `SelectionPosition` must be just that: only facts and not reflective of decisions as to validity of marks and countability of votes. Where software has made a decision as to the countability of a contest selection, it is indicated clearly. Using the facts, however, one can more accurately adjudicate or override the decisions made by software.

Therefore, this specification uses terminology very carefully in its class, attribute names, and descriptions. Before further consideration of `SelectionPosition`, the following glossary terms must be understood.

#### 3.4.1.1 Use of *Mark*

The word *Mark* is used in this specification *only* to mean a scanner-detectable mark on a paper ballot that requires some associated measurement of quality to determine whether the mark represents a tabulatable contest selection. While the mark would presumably have been made by the voter, a flaw on the ballot could be detected by the scanner and considered as a mark<sup>2</sup>, albeit an ambiguous one.

A *mark* is important to differentiate from a machine-made representation of a contest selection that was made via the voter using a BMD or DRE. The machine-made representation does not require an associated measurement of quality; it is either present or not.

The thresholds for mark quality may be set in state or local election law and thus can vary by jurisdiction. Scanners can include the measurements in the CVR by first indicating the name or type of the quality measurement in `ReportingDevice.MarkMetricType`, and then using, for each mark, `SelectionPosition.MarkMetricValue` to indicate the mark measurement.

---

<sup>2</sup> For example, a crease in a paper ballot or a smudge that runs through a contest option's oval or checkbox could be detected as a human-made mark.

### 3.4.1.2 Use of Selection Indication

The term *Selection Indication* or *Indication* is used as a broader term to mean either (a) a mark presumably made by a human or (b) a machine-made representation of a mark. Selection indications can come from the following sources:

- A flaw on the paper ballot detected as a mark made by the voter.
- A mark made by the voter on the paper ballot; the mark could be ambiguous or could meet scanner thresholds for quality.
- A mark made by a ballot marking device onto a full-face paper ballot.
- An indication made by the voter using a DRE or by a ballot marking device using a bar code to represent voter selections.
- An indication made by the scanner in certain cases as a result of applying contest rules to the voter's indication<sup>3</sup>.
- An indication made by an adjudicator.

### 3.4.1.3 Use of Allocable

*Allocable* is used, then, to indicate whether the selection indication is potentially countable, that is, whether it can be given over or *allocated* to a later tabulation process that may count it as a vote if it meets all tabulation criteria. In most cases, allocable means countable, but here, *countable* is avoided, as it implies a decision has been made to count the selection indication.

## 3.4.2 HasIndication and IsAllocable

SelectionPosition uses two attributes that act as booleans to show decisions made by software as to whether selection indications are allocable. The first, HasIndication, indicates whether software has decided that there is, indeed, a selection indication that can be potentially counted. The following table shows the mapping of selection indications to HasIndication values:

Selection Indication	HasIndication
machine-readable mark from a paper ballot	yes
machine-generated mark from a DRE or BMD	yes
marginally machine-readable mark	unknown
no mark or no mark undetected	no

Table 1 - Selection Indication to HasIndication Mapping

---

<sup>3</sup> For example, a voter may choose a straight party option, and the scanner may, in the CVR, create selection indications for each contest according to the straight party contest rules.

When there is no indication of a selection indication, most likely the contest selection was not voted but is being included in the CVR.

The second boolean-like attribute, `IsAllocable`, is used for indicating a decision made by software as to whether a selection indication is allocable to the tabulator process for counting as a vote. In effect, this indicates if there is a vote in a contest selection that could be counted. When `HasIndication` has a value of unknown, `IsAllocable` need not be included.

### 3.4.3 Ranked Choice Voting Example

Figure 7 shows an example RCV (ranked choice voting) contest for President, in which up to 4 candidates can be selected, each with an associated rank. In this example, Smith was ranked as 1, Front was ranked as 3, Hillberry was ranked as 2, but the voter did not rank James.

**Presidential Contest**

Contest Option	1st	2nd	3rd	4th
Jack Smith/Steven Ritz	[x]	[ ]	[ ]	[ ]
Layla Front/Robert Brak	[ ]	[ ]	[x]	[ ]
James Hillberry/Mark Jafrate	[ ]	[x]	[ ]	[ ]
Anna James/Mary Kalman	[ ]	[ ]	[ ]	[ ]

**Figure 7 - RCV contest with multiple contest selections; one voter mark per selection**

`CVRContest` will include a `CVRContestSelection` instance for each of the three contest selections that were selected. Each `CVRContestSelection` will include a `SelectionPosition` instance for the bubble filled in by the voter (in this case, only one bubble for each contest selection was filled in; depending on the RCV contest rules, more than one bubble filled in for a given contest selection may be an overvote).

A partial example of the structure representing the contest in Figure 7 is as follows:

- `CVRContest`:
  - Link to the Presidential contest object
  - `CVRContestSelection`:
    - Link to candidate object for Jack Smith/Steven Ritz
    - Total number of votes represented by the contest selection = 1
    - `SelectionPosition`:
      - Position = 1 (or Rank = 1)
      - `HasIndication` = yes
      - `IsAllocable` = yes

- Number of votes = 1
- CVRContestSelection:
  - Link to candidate object for Layla Front/Robert Brak
  - Total number of votes represented by the contest selection = 1
  - SelectionPosition:
    - Position = 3 (or Rank = 3)
    - HasIndication = yes
    - IsAllocable = yes
    - Number of votes = 1
- CVRContestSelection:
  - Link to candidate object for James Hillberry/Mark Jafrate
  - Total number of votes represented by the contest selection = 1
  - SelectionPosition:
    - Position = 2 (or Rank = 2)
    - HasIndication = yes
    - IsAllocable = yes
    - Number of votes = 1

#### 3.4.4 Cumulative Voting Example

Figure 8 shows an example for cumulative voting in which multiple voter marks are allowable for each contest selection, with a total number of votes (or score) included for each candidate.

Three votes can be allocated across the five candidates, however only two candidates received votes, Ford and Hill. Thus, CVRContest will include two instances of CVRContestSelection, one for Ford and one for Hill. Each CVRContestSelection includes SelectionPosition for each bubble filled into the left of each candidate's name, for a maximum of three. In this case, there will be two SelectionPosition instances for Henry Ford and one for Mary Hill.

Cumulative Voting Contest

1	2	3	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Joe Smith
<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	Henry Ford
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Jane Doe
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Fred Rubble
<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	Mary Hill

You may offer up to 3 votes.

Figure 8 - Cumulative Voting contest in which multiple voter marks per selection are allowed

A partial example of the structure representing the contest in Figure 8 is as follows:

- CVRContest:
  - Link to contest object for this contest
  - CVRContestSelection1:
    - Link to candidate object for Henry Ford
    - Total number of votes represented by the contest selection = 2
    - SelectionPosition:
      - Position = 1
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 1
    - SelectionPosition:
      - Position = 2
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 1
  - CVRContestSelection2:
    - Link to candidate object for Mary Hill
    - Total number of votes represented by the contest selection = 2
    - SelectionPosition:
      - Position = 3
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 1

### 3.4.5 Range Voting Example

Lastly, Figure 9 shows an example of a contest using range voting, in which each candidate receives a score, which is analogous to number of votes. In this example, one bubble can be filled in for each contest selection, and each candidate will have an associated score or number of votes, with the maximum score of 9. The position of the bubble is highly meaningful, as the bubble in position 1 represents no votes and the bubble in position 10 represents 9 votes.

Range Voting Contest

Score each candidate by filling a number (0 is worst; 9 is best)										
Candidate A	0	1	2	3	4	5	6	7	8	9
Candidate B	0	1	2	3	4	5	6	7	8	9
Candidate C	0	1	2	3	4	5	6	7	8	9

**Figure 9 - Range Voting contest with multiple contest selections; one voter mark per selection**

A partial example of the structure representing the contest in Figure 9 is as follows:

- CVRContest:
  - Link to contest object for this contest
  - CVRContestSelection:
    - Link to candidate object for Candidate A
    - Total number of votes represented by the contest selection = 0
    - SelectionPosition:
      - Position = 1
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 0
  - CVRContestSelection:
    - Link to candidate object for Candidate B
    - Total number of votes represented by the contest selection = 9
    - SelectionPosition:
      - Position = 10
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 9
  - CVRContestSelection:
    - Link to candidate object for Candidate C
    - Total number of votes represented by the contest selection = 7
    - SelectionPosition:
      - Position = 8
      - HasIndication = yes
      - IsAllocable = yes
      - Number of votes = 7

### 3.5 Identifiers Within the CVR

The CVR can contain potentially up to ten different identifiers whose purpose in general is to make CVRs easier to use in tabulation and more useful in auditing. In particular, the CVR can contain an ID that is common to its corresponding paper ballot so as to support ballot-level comparison audits in which paper ballots are compared against their corresponding CVRs.

Seven of the IDs are attributes of the CVR class because the information they represent will be different for each CVR. There are three additional ID-related classes that are linked from the CVR as the result of associations with `BallotStyleUnit`, `Creating Device`, and `Election`; these classes contain static information that will be the same for each CVR. The different types of IDs can grow confusing; thus this section describes how they are used.

#### 3.5.1 A Unique ID for Each CVR

First, every CVR, as it is created by a device such as a scanner or BMD, is assigned a unique ID so that it can be individually referenced. This ID, `CVR::UniqueId`, applies to the entire CVR.

#### 3.5.2 An ID for Each Sheet of a Multi-sheet Paper Ballot

Generally, if a ballot contains multiple sheets, a CVR will be created for each sheet; for a 3-page ballot, three CVRs will be created. If each sheet of the ballot contains an indication of the sheet number, e.g., sheet 2 contains an indication that it is sheet 2, the creating device can include this indication within the CVR, using `CVR::BallotSheetId`.

If the paper ballot is two-sided, a scanner may still consider the ballot as consisting of one sheet and create just one CVR instead of a CVR for each side. In this case, `CVR::BallotSheetId` would apply to both sides of the ballot.

#### 3.5.3 An ID for Ballots Pre-Printed with a Unique Identifier

In some cases, paper ballots may be pre-printed with a unique identifier such as a sequence number. In this case, the scanner can include the pre-printed identifier in the CVR using `CVR::BallotPrePrintedId`.

#### 3.5.4 An ID to Link the CVR to its Corresponding Paper Ballot

For ballot-level comparison audits, there must be a means for pairing a CVR to its corresponding paper ballot. Scanners may print an ID on a ballot as it is scanned and then include that ID in `CVR::BallotAuditId`.

### **3.5.4.1 An ID to Link the CVR to its Corresponding Ballot Style ID**

The CVR can be linked to the ballot style ID that was used in creating the paper ballot. This ID is contained in `CVR::BallotStyleId`.

### **3.5.5 IDs to Support Batching**

There are two identifiers both related to batching, that is, grouping paper ballots as they are scanned into separate batches for the purposes of auditing by batch. The first identifier, `CVRSnapshot::BatchId`, is used to identify the batch. The second identifier, `CVRSnapshot::BatchSequenceId`, is used to identify the position or sequence of the ballot within the batch.

### **3.5.6 IDs via Associations with Other Classes**

Lastly, the CVR class can be associated with 3 other classes, each class containing static information related to tabulation and auditing. These classes are referenced (linked) from each CVR because the information within each class is entirely static and thus need not be repeated in each CVR. These classes are:

1. `BallotStyleUnit` – contains static information about the political geography corresponding to the ballot's ballot style. Most likely the political geography will correspond to a precinct or a precinct split.
2. `CreatingDevice` – contains static information about the device that created the CVR, including the manufacture, model, serial number, and the type of mark metric measure used by the device.
3. `Election` – contains identifying information about the election, including the name and any associated codes.

## 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 associations with other classes, curly braces are used around those attribute names, e.g., if ClassA has an association with ClassB that is named "Automobile", then the table of attributes for ClassA would include "{Automobile}" as one of the attributes.

Class attributes and enumeration values are alphabetically ordered with some exceptions, e.g., "OtherStatus" comes after "Status" in some classes, and "other" is last in enumeration values.

### 4.1 Class Annotation

Annotation is used to record annotations made by one or more adjudicators.

[CVRSnapshot](#) includes Annotation.

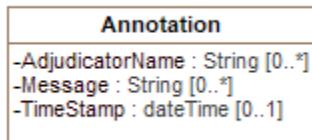


Figure 10 - Class Annotation

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).
TimeStamp	0..1	dateTime	The date and time of the annotation.

## 4.2 Class BallotMeasureContest

BallotMeasureContest is a subclass of [Contest](#) and is used to identify the type of contest as involving one or more ballot measures. It inherits attributes from [Contest](#).

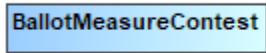


Figure 11 – Class BallotMeasureContest

### 4.3 Class BallotMeasureSelection

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

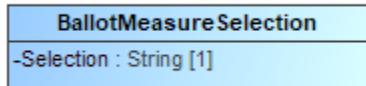


Figure 12 - BallotMeasureSelection

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.

#### 4.4 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.

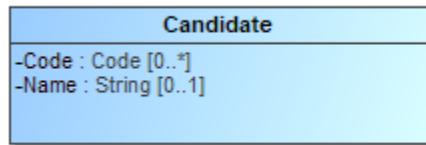


Figure 13 - Class Candidate

Attribute	Multiplicity	Type	Attribute Description
Code	0..*	<a href="#">Code</a>	A code or identifier associated with the candidate.
Name	0..1	<a href="#">String</a>	Candidate's name as listed on the ballot.
{Party}	0..1	<a href="#">Party</a>	The party associated with the candidate.

## 4.5 Class 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](#).



Figure 14 - Class CandidateContest

Attribute	Multiplicity	Type	Attribute Description
NumberElected	0..1	Integer	The number of candidates to be elected in the contest.
{PrimaryParty}	0..1	<a href="#">Party</a>	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 Class CandidateSelection

CandidateSelection is a subclass of [ContestSelection](#) and is used for candidates, including for write-in candidates.



Figure 15 - Class CandidateSelection

Attribute	Multiplicity	Type	Attribute Description
{Candidate}	0..*	<a href="#">Candidate</a>	The candidate associated with the contest 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.

## 4.7 Class CastVoteRecordReport

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

`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 contest options that were voted and the indication(s)/mark(s) made.

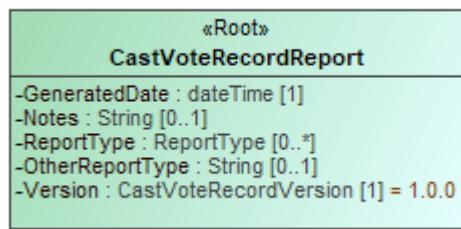


Figure 16 - Class `CastVoteRecordReport`

Attribute	Multiplicity	Type	Attribute Description
{CVR}	0..*	<a href="#">CVR</a>	Used to include instances of <a href="#">CVR</a> classes, one per cast vote record in the report.
{Election}	1..*	<a href="#">Election</a>	Used to include the election(s) associated with the CVRs.
GeneratedDate	1	<code>dateTime</code>	Identifies the time that the election report was created.
{GpUnit}	1..*	<a href="#">GpUnit</a>	Used to include the political geography, i.e., location, for where the cast vote record report was created and for linking cast vote records to their corresponding precinct or split (or otherwise smallest unit).
Notes	0..1	<code>String</code>	Notes that can be added as appropriate, presumably by an adjudicator.
{Party}	0..*	<a href="#">Party</a>	The party associated with the ballot sheet for a partisan primary.
{ReportGeneratingDevice}	1..*	<a href="#">ReportingDevice</a>	Identifies the device used to create the CVR report.

Attribute	Multiplicity	Type	Attribute Description
{ReportingDevice}	1..*	<a href="#">ReportingDevice</a>	The device creating the report. The reporting device need not necessarily be the creating device, i.e., for an aggregated report, the reporting device could be an EMS used to aggregate and tabulate cast vote records.
ReportType	0..*	<a href="#">ReportType</a>	The type of report, using the <a href="#">ReportType</a> enumeration.
OtherReportType	0..1	String	If <a href="#">ReportType</a> is 'other', this contains the report type.
Version	1	<a href="#">CastVoteRecordVersion</a>	The version of the CVR specification being used (1.0).

## 4.8 Class Code

Code is used in [Election](#), [GpUnit](#), [Contest](#), [Candidate](#), and [Party](#) to identify an associated code and the type of code.

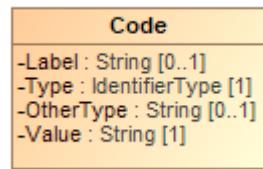


Figure 17 - Class Code

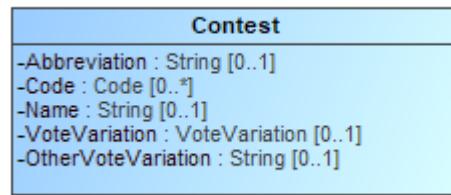
Attribute	Multiplicity	Type	Attribute Description
Label	0..1	String	A label associated with the code, used as needed.
Type	1	<a href="#">IdentifierType</a>	Used to indicate the type of code, from the <a href="#">IdentifierType</a> enumeration.
OtherType	0..1	String	If Type is 'other', the type of code.
Value	1	String	The value of the code, i.e., the identifier.

## 4.9 Class Contest

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 cast vote record, such as a contest, its voted contest selection(s), and the mark(s) associated with the selection(s).

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

1. [PartyContest](#) - used for straight party contests,
2. [BallotMeasureContest](#) - used for contests, and
3. [CandidateContest](#) - used for candidate contests.



**Figure 18 - Class Contest**

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

## 4.10 Class ContestSelection

ContestSelection represents a contest selection in a contest. [Contest](#) can include an instance of ContestSelection for each contest selection in the contest or, as desired, all contest selections.

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

1. [BallotMeasureSelection](#) - used for ballot measures,
2. [CandidateSelection](#) - used for candidate selections, and
3. [PartySelection](#) - used for straight party selections.

Instances of [CVRCContestSelection](#) subsequently link to the contest selections as needed so as to tie together the contest, the contest selection, and the mark(s) made for the contest selection.

ContestSelection contains one attribute, [Code](#), that can be used to identify the contest selection and thereby eliminate the need to identify it using the subclasses.



Figure 19 - Class ContestSelection

Attribute	Multiplicity	Type	Attribute Description
Code	0..*	<a href="#">Code</a>	Code used to identify the contest selection.

## 4.11 Class CVR

CVR constitutes a cast vote record, generated by a ballot scanning device, containing indications of contests and contest options chosen by the voter, as well as other information for auditing and annotation purposes. 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 created.

[CastVoteRecordReport](#) includes multiple instances of CVR as applicable.

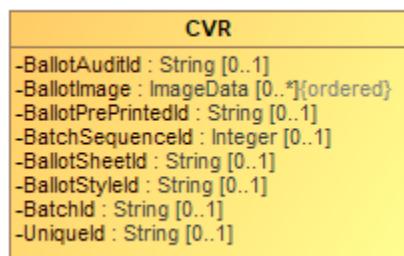


Figure 20 - Class CVR

Attribute	Multiplicity	Type	Attribute Description
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..*	<a href="#">ImageData</a>	An image of the ballot sheet created by the scanning device.
BallotPrePrintedId	0..1	String	A unique identifier for the ballot (or sheet of a multi-sheet ballot) that this CVR represents, used if ballots are pre-marked with unique identifiers. If provided, this number would be the same on all CVRs that represent individual sheets from the same multi-sheet ballot. This identifier 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 <a href="#">BallotAuditId</a> attribute.
BallotSheetId	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

Attribute	Multiplicity	Type	Attribute Description
			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 <a href="#">BallotAuditId</a> attribute.
BallotStyleId	0..1	String	An identifier of the ballot style associated with the corresponding ballot.
{BallotStyleUnit}	0..1	<a href="#">GpUnit</a>	Identifies the smallest unit of geography associated with the corresponding ballot, typically a precinct or split-precinct.
BatchId	0..1	String	The identifier for the batch that includes this CVR.
BatchSequenceId	0..1	Integer	The sequence number of the corresponding paper ballot within a batch.
{CreatingDevice}	0..1	<a href="#">ReportingDevice</a>	Identifies the device that created the CVR.
{CurrentSnapshot}	1	<a href="#">CVRSnapshot</a>	Identifies the snapshot that is currently tabulatable.
{CVRSnapshot}	1..*	<a href="#">CVRSnapshot</a>	Identifies the repeatable portion of the CVR that links to contest selections and related information.
{Election}	1	<a href="#">Election</a>	Used to identify an election with which the CVR is associated.
{Party}	0..*	<a href="#">Party</a>	Identifies the party associated with a CVR, typically for partisan primaries.
UniqueId	0..1	String	The sequence number for this CVR. This represents the ordinal number that this CVR was processed by the tabulating device.

## 4.12 Class CVRContest

CVRContest class is included by [CVRSnapshot](#) for each contest on the ballot that was voted, that is, whose contest options contain indications that may constitute a vote. CVRContest includes [CVRContestSelection](#) for each contest option in the contest containing an indication or write-in.

[CVRSnapshot](#) can also include CVRContest for every contest on the ballot regardless of whether any of the contest options contain an indication, for cases where the CVR must include all contests that appeared on the ballot.

CVRContest 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 "choose 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.

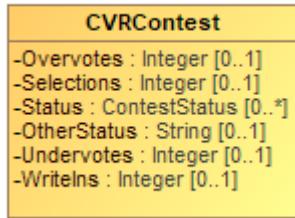


Figure 21 - Class CVRContest

Attribute	Multiplicity	Type	Attribute Description
{Contest}	1	<a href="#">Contest</a>	Used to link to an instance of <a href="#">Contest</a> specific to the contest at hand, for the purpose of specifying information about the contest such as its contest identifier.
{CVRContestSelection}	0..*	<a href="#">CVRContestSelection</a>	Used to include information about a contest selection in the contest, including the associated indication(s).
Overvotes	0..1	Integer	The number of votes lost due to overvoting.
Selections	0..1	Integer	Used to indicate the number of possible contest selections in the contest.

Attribute	Multiplicity	Type	Attribute Description
Status	0..*	<a href="#">ContestStatus</a>	The status of the contest, e.g., overvoted, undervoted, from the <a href="#">ContestStatus</a> enumeration.
OtherStatus	0..1	String	Used when Status is 'other' to include a user-defined status.
Undervotes	0..1	Integer	The number of votes lost due to undervoting.
WriteIns	0..1	Integer	The total number of write-ins in the contest.

#### 4.13 Class CVRCContestSelection

CVRCContestSelection is used to link a contest option containing an indication with information about the indication, such as whether a mark constitutes a countable vote, or whether a mark is determined to be marginal, etc. [CVRCContest](#) includes an instance of CVRCContestSelection when an indication for the selection is present, and [CVRCContestSelection](#) then includes [SelectionPosition](#) for each indication present. To tie the indication to the specific contest selection, CVRCContestSelection links to an instance of [ContestSelection](#) that has previously been included by [Contest](#).

Since multiple indications per contest option are possible for some voting methods, CVRCContestSelection can include multiple instances of [SelectionPosition](#), one per indication. CVRCContestSelection can also be used for the purpose of including, in the CVR, all contest options in the contest regardless of whether indications are present. In this case, CVRCContestSelection would not include [SelectionPosition](#) if no indication is present but would link to the appropriate instance of [ContestSelection](#).

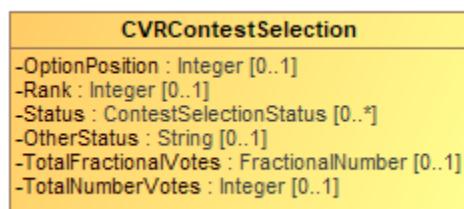


Figure 22 - Class CVRCContestSelection

Attribute	Multiplicity	Type	Attribute Description
{ContestSelection}	0..1	<a href="#">ContestSelection</a>	Used to link to an instance of a contest selection that was previously included by <a href="#">Contest</a> .
OptionPosition	0..1	Integer	Used to include the ordinal position of the contest option as it 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.

Attribute	Multiplicity	Type	Attribute Description
{SelectionPosition}	1..*	<a href="#">SelectionPosition</a>	Used to include further information about the indication/mark associated with the contest selection. Depending on the voting method, multiple indications/marks per selection may be possible.
Status	0..*	<a href="#">ContestSelectionStatus</a>	Contains the status of the contest selection, e.g., 'needs-adjudication' for a contest requiring adjudication, using values from the <a href="#">ContestSelectionStatus</a> enumeration.
OtherStatus	0..1	String	Used when Status is 'other' to include a user-defined status.
TotalFractionalVotes	0..1	<a href="#">FractionalNumber</a>	For cumulative or range and other similar voting variations, contains the total proper fractional number of votes across all indications/marks.
TotalNumberVotes	0..1	Integer	For cumulative or range and other similar voting variations, contains the total number of votes across all indications/marks.

#### 4.14 Class CVRSnapshot

CVRSnapshot contains a version of the contest selections for a CVR; there can be multiple versions of CVRSnapshot within the same CVR. [Type](#) specifies the type of the snapshot, i.e., whether interpreted by the scanner according to contest rules, modified as a result of adjudication, or the original, that is, the version initially scanned before contest rules are applied. [CVR](#) includes CVRSnapshot.

Other attributes are repeated in each CVRSnapshot because they may differ across snapshots, e.g., the contests could be different as well as other status.

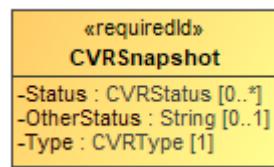


Figure 23 - Class CVRSnapshot

Attribute	Multiplicity	Type	Attribute Description
{Annotation}	0..*	<a href="#">Annotation</a>	Used to include an annotation associated with the CVR snapshot.
{CVRContest}	0..*	<a href="#">CVRContest</a>	Identifies the contests in the CVR.
Status	0..*	<a href="#">CVRStatus</a>	The status of the CVR.
OtherStatus	0..1	String	When Status is 'other', contains the ballot status.
Type	1	<a href="#">CVRTyp</a>	The type of the snapshot, e.g., original.

#### 4.15 Class CVRWriteIn

CVRWriteIn is used when the contest selection is a write-in. It has attributes for the image or text of the write-in.



Figure 24 - Class CVRWriteIn

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	<a href="#">ImageData</a>	Used for an image of the write-in, typically made by a scanner when scanning a paper ballot.

#### 4.16 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.

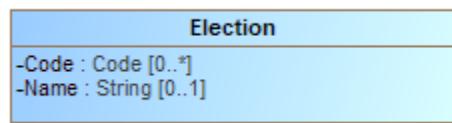


Figure 25 - Class Election

Attribute	Multiplicity	Type	Attribute Description
{Candidate}	0..*	<a href="#">Candidate</a>	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..*	<a href="#">Code</a>	Used for a code associated with the election, e.g., a precinct identifier if the election scope is a precinct.
{Contest}	1..*	<a href="#">Contest</a>	Used for establishing a collection of contest definitions that will be referenced by the CVRs.
{ElectionScope}	1	<a href="#">GpUnit</a>	Used to identify the election scope, i.e., the political geography corresponding to the election.
Name	0..1	<a href="#">String</a>	A text string identifying the election.

#### 4.17 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.



Figure 26 - Class File

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.

#### 4.18 Class FractionalNumber

A proper fractional value represented using fractional or decimal notation.

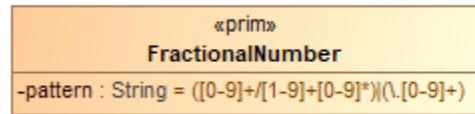


Figure 27 - Class FractionalNumber

Attribute	Multiplicity	Type	Attribute Description
pattern		String	Pattern describing the allowed values for a <a href="#">FractionalNumber</a> .

## 4.19 Class GpUnit

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 creates 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 [BallotStyleUnit](#) to link a CVR to the smallest political subdivision that uses the same ballot style as was used for the voter's ballot.

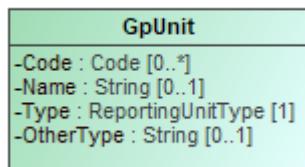


Figure 28 - Class GpUnit

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

## 4.20 Class Hash

Hash is used to specify a hash associated with a file such as an image file of a scanned ballot.

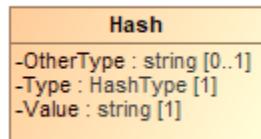
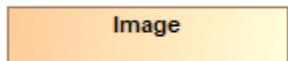


Figure 29 - Class Hash

Attribute	Multiplicity	Type	Attribute Description
Type	1	<a href="#">HashType</a>	The type of the hash, from the <a href="#">HashType</a> enumeration.
OtherType	0..1	String	If Type is 'other', the type of the hash.
Value	1	String	The hash value, encoded as a string.

#### 4.21 Class Image

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



**Figure 30 - Class Image**

## 4.22 Class ImageData

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.

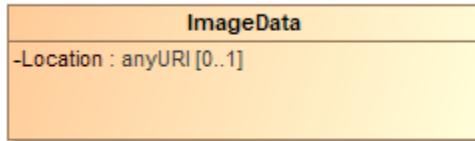


Figure 31 - Class ImageData

Attribute	Multiplicity	Type	Attribute Description
{Hash}	0..1	<a href="#">Hash</a>	A hash value for the image data, used for verification comparisons against subsequent copies of the image.
{Image}	0..1	<a href="#">Image</a>	The image of an individual ballot sheet created by the scanner, could possibly include both sides of a two-sided ballot sheet depending on the scanner's configuration.
Location	0..1	anyURI	A pointer to the location of the image file.

## 4.23 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.

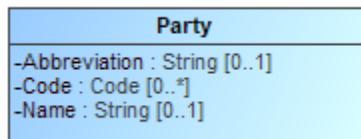


Figure 32 - Class Party

Attribute	Multiplicity	Type	Attribute Description
Abbreviation	0..1	String	Short name for the party, e.g., "DEM".
Code	0..*	<a href="#">Code</a>	A code associated with the party.
Name	0..1	String	Official full name of the party, e.g., "Republican".

#### 4.24 Class 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](#).

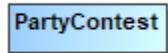


Figure 33 - Class PartyContest

## 4.25 Class PartySelection

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

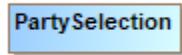


Figure 34 - Class PartySelection

Attribute	Multiplicity	Type	Attribute Description
{Party}	1..*	<a href="#">Party</a>	The party associated with the contest selection.

## 4.26 Class ReportingDevice

ReportingDevice 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 created the CVR report. [CVR](#) refers to it as [CreatingDevice](#) to specify the device that created the CVRs.

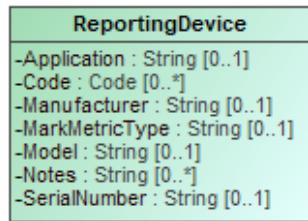


Figure 35 - Class ReportingDevice

Attribute	Multiplicity	Type	Attribute Description
Application	0..1	String	The application associated with the reporting device.
Code	0..*	<a href="#">Code</a>	A code associated with the reporting device.
Manufacturer	0..1	String	Manufacturer of the reporting device.
MarkMetricType	0..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.
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.

#### 4.27 Class 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](#).

**RetentionContest**

**Figure 36 - Class RetentionContest**

Attribute	Multiplicity	Type	Attribute Description
{Candidate}		<a href="#">Candidate</a>	Identifies the candidate in the retention contest.

## 4.28 Class SelectionPosition

[CVRContestSelection](#) includes SelectionPosition to specify a voter's indication/mark in a contest option, and thus, a potential vote. The number of potential SelectionPositions that could be included by CVRContestSelection is the same as the number of ovals next to a particular option. There will be usually 1 instance of [SelectionPosition](#) for plurality voting, but there could be multiple instances for RCV, approval, cumulative, or other vote variations in which a voter can select multiple options per candidate.

[SelectionPosition](#) contains additional information about the mark to specify whether the mark is countable, as well as information needed for certain voting methods. [MarkMetricValue](#) specifies the measurement of a mark on a paper ballot. The measurement is assigned by the scanner for measurements of mark density or quality and would be used by the scanner to indicate whether the mark is a valid voter mark representing a vote or is marginal.

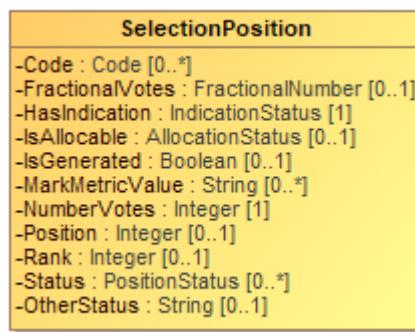


Figure 37 - Class SelectionPosition

Attribute	Multiplicity	Type	Attribute Description
Code	0..*	Code	Code used to identify the contest selection position.
{CVRWriteIn}	0..1	CVRWriteIn	Used to store information regarding a write-in vote.
FractionalVotes	0..1	FractionalNumber	The proper fractional number of votes represented by the position.
HasIndication	1	IndicationStatus	Whether there is a selection indication present.
IsAllocable	0..1	<a href="#">AllocationStatus</a>	Whether this indication should be allocated to the contest option's accumulator.
IsGenerated	0..1	Boolean	Whether this indication was generated by election rules, if not present, False is assumed.
MarkMetricValue	0..1	String	The value of the mark metric, represented as a string.

Attribute	Multiplicity	Type	Attribute Description	
NumberVotes	1	Integer	The number of votes represented by the position, usually 1 but may be more depending on the voting method.	
Position	0..1	Integer	The ordinal position of the selection position within the contest option.	
Rank	0..1	Integer	For the RCV voting variation, the rank chosen by the voter, for when a position can represent a ranking.	
Status	0..*	<a href="#">PositionStatus</a>	Status of the position, e.g., "generated-rules" for generated by the machine, from the PositionStatus enumeration. If no values apply, use 'other' and include a user-defined status in OtherStatus.	
OtherStatus	0..1	String	Used when Status is "other" to include a user-defined status.	

#### 4.29 Enumeration AllocationStatus

Used in [SelectionPosition](#)::IsAllocable to indicate whether the [SelectionPosition](#)::NumberVotes should be allocated to the underlying contest option counter.



**Figure 38 – Enumeration AllocationStatus**

Value	Value Description
no	To not allocate votes to the contest option's accumulator.
unknown	When the decision to allocate votes is unknown, such as when adjudication is needed.
yes	To allocate votes to the contest option's accumulator.

#### 4.30 Enumeration CastVoteRecordVersion

To identify the version of the CVR specification being used, i.e., version 1.0.0. This will need to be updated for different versions of the specification.



Figure 39 – Enumeration CastVoteRecordVersion

Value	Value Description
1.0.0	Fixed value for the version of this specification.

### 4.31 Enumeration ContestSelectionStatus

Used in [CVRCContestSelection](#)::Status to identify the status of a contest selection in the CVR.

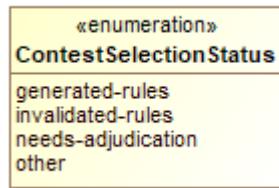


Figure 40 – Enumeration ContestSelectionStatus

Value	Value Description
generated-rules	To indicate that the contest selection was created per contest rules.
invalidated-rules	To indicate that the contest selection was invalidated by the creating device because of contest rules.
needs-adjudication	To indicate that the contest selection was flagged by the creating device for adjudication.
other	Used in conjunction with <a href="#">CVRCContestSelection</a> ::OtherStatus when no other value in this enumeration applies.

#### 4.32 Enumeration ContestStatus

Used in [CVRContest::Status](#) to identify the status of a contest in which contest selection(s) were made.

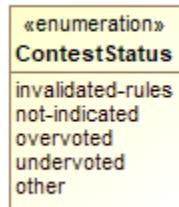


Figure 41 – Enumeration ContestStatus

Value	Value Description
invalidated-rules	To indicate that the contest has been invalidated by the creating device because of contest rules.
not-indicated	For a <a href="#">CVRContest</a> with no <a href="#">SelectionPosition</a> , i.e. to specify the position contains no marks or other indications.
overvoted	To indicate that the contest was overvoted.
undervoted	To indicate that the contest was undervoted.
other	Used in conjunction with <a href="#">CVRContest::OtherStatus</a> when no other value in this enumeration applies.

### 4.33 Enumeration CVRStatus

Used in [CVRSnapshot::Status](#) to identify the status of the CVR.

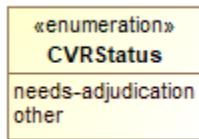


Figure 42 - Enumeration CVRStatus

Value	Value Description
needs-adjudication	To indicate that the CVR needs to be adjudicated.
other	Used in conjunction with <a href="#">CVRSnapshot::OtherStatus</a> when no other value in this enumeration applies.

#### 4.34 Enumeration CVRTypE

Used in [CVRSnapshot::Type](#) to indicate the type of snapshot.

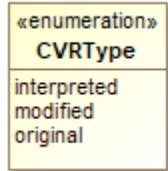


Figure 43 - Enumeration CVRTypE

Value	Value Description
interpreted	Has been adjudicated.
modified	After contest rules applied.
original	As scanned, no contest rules applied.

#### 4.35 Enumeration HashType

Used in [Hash::Type](#) to indicate the type of hash being used for an image file.

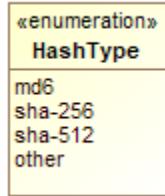


Figure 44 - Enumeration HashType

Value	Value Description
md6	To indicate that the MD6 message digest algorithm is being used.
sha-256	To indicate that the SHA 256-bit signature is being used.
sha-512	To indicate that the SHA 512-bit signature is being used.
other	Used in conjunction with <a href="#">Hash::OtherType</a> when no other value in this enumeration applies.

#### 4.36 Enumeration IdentifierType

Used in [Code](#)::Type to indicate the type of code/identifier being used.

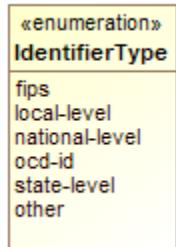


Figure 45 - Enumeration IdentifierType

Value	Value Description
fips	To indicate that the identifier is a Federal Information Processing Standard (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 Open Civic Data Identifier (OCD-ID).
ocd-id	To indicate that the identifier is from the OCD-ID scheme.
state-level	To indicate that the identifier is from a state-level scheme, i.e., unique to a particular state.
other	Used in conjunction with <a href="#">Code</a> ::OtherType when no other value in this enumeration applies.

#### 4.37 Enumeration IndicationStatus

Used in [SelectionPosition](#)::HasIndication to identify whether a selection indication is present.

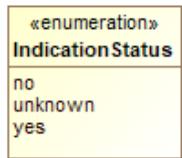


Figure 46 - Enumeration IndicationStatus

Value	Value Description
no	There is no selection indication.
unknown	It is unknown whether there is a selection indication, e.g., used for ambiguous marks.
yes	There is a selection indication present.

#### 4.38 Enumeration PositionStatus

Used in [SelectionPosition::Status](#) to identify the status of a selection indication.

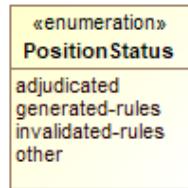


Figure 47 - Enumeration PositionStatus

Value	Value Description
adjudicated	Used if the indication was adjudicated.
generated-rules	Used if the indication was generated by the creating device per contest rules.
invalidated-rules	Used if the indication was invalidated by the creating device because of contest rules.
other	Used in conjunction with <a href="#">SelectionPosition::OtherStatus</a> when no other value in this enumeration applies.

#### 4.39 Enumeration ReportingUnitType

Used in [GpUnit](#)::Type to indicate a type of political geography.

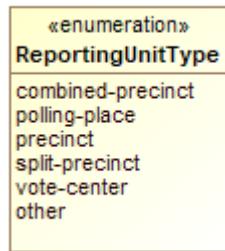


Figure 48 - Enumeration ReportingUnitType

Value	Value Description
combined-precinct	To indicate a combined precinct.
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.
other	Used in conjunction with <a href="#">GpUnit</a> ::OtherType when no other value in this enumeration applies.

#### 4.40 Enumeration ReportType

Used in [CastVoteRecordReport::ReportType](#) to indicate the type of the CVR report.

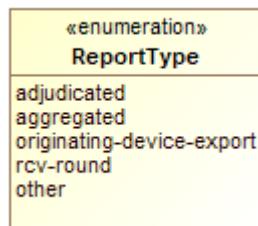


Figure 49 - Enumeration ReportType

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.
rcv-round	To indicate that the report is the result of a ranked choice voting round.
other	Used in conjunction with <a href="#">CastVoteRecordReport::OtherReportType</a> when no other value in this enumeration applies.

#### 4.41 Enumeration VoteVariation

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

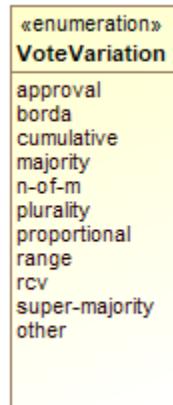


Figure 50 - Enumeration VoteVariation

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.
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.
other	Used in conjunction with <a href="#">Contest::OtherVoteVariation</a> when no other value in this enumeration applies.

## 5 Usage Examples

This section contains examples showing how to use this specification in various voting scenarios. The examples refer to several XML files whose locations can be found in Appendix D. Some of the sections use examples from Section 3 but show them in XML.

### 5.1 Anatomy of a CVR

This section refers to the file for Example 1.

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

A 1500-103 instance (in XML or JSON) may contain one or more CVRs, which in turn must contain one or more CVRSnapshots, each representing a CVR at a specific point in time. The file is divided roughly into two parts: the CVR elements at the beginning followed by other elements for defining the election and its contests, candidates, and contest selections so that the CVR elements can link to them as necessary. Lines 205 to 244 describe an election containing 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's `ObjectId` attributes. For example, the contest definition starting on line 228 contains:

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

so that CVR elements can link to this contest definition by using `_C1`:

```
<ContestId>_C1</ContestId>
```

Importantly, *the object identifiers are not the same as the codes 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 3 to 204 contain the CVR elements. Each CVR element includes at least one CVRSnapshot. Each CVRSnapshot represents a particular type, such as the original captured from a scanner, or after it has been interpreted (i.e., business rules have been applied), or otherwise modified. The CVRSnapshot element includes one or more CVRCandidate elements, which link to the voted contest whose object identifier is `_C1`, thereby identifying that contest within the report file. It then includes CVRCandidateSelection, which links to a contest option that was selected by the

voter. Each CVR element also includes an optional sequence number (SequenceNumber); this isn't required but could be helpful to auditors.

## 5.2 Basic Example

This section refers to the file for Example 2.

Consider the following contest:

<p><b><u>For Treasurer of State</u></b></p> <p><input checked="" type="checkbox"/> <i>Connie Pillich</i> <input type="checkbox"/> <i>Josh Mandel</i></p>
--

**Figure 51 - Treasurer Contest**

which can be represented with the following XML fragment:

```
<cdf:CVRContest>
  <cdf:ContestId>_5TS</cdf:ContestId>
  <cdf:CVRContestSelection>
    <cdf:ContestSelectionId>_1ECP</cdf:ContestSelectionId>
    <cdf:Position>1</cdf:Position>
    <cdf:SelectionPosition>
      <cdf:HasIndication>yes</cdf:HasIndication>
      <cdf:IsAllocable>yes</cdf:IsAllocable>
      <cdf:NumberVotes>1</cdf:NumberVotes>
    </cdf:SelectionPosition>
    <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
  </cdf:CVRContestSelection>
  <cdf:Position>1</cdf:Position>
  <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
  ...
</cdf:CVRContest>
```

The ContestSelectionId value of \_1ECP represents the reference to the selected contest option:

```
<cdf:Candidate ObjectID="_1ECP">
  ...
  <cdf:Name>Connie Pillich</cdf:Name>
  ...
</cdf:Candidate>
```

By dereferencing `_5TS`, we can see this does indeed represent a contest selection of Connie Pillich for Treasurer of State:

```
<cdf:Contest xsi:type="cdf:CandidateContest" ObjectId="_5TS">
  ...
  <cdf:Name>For Treasurer of State</cdf:Name>
  <cdf:VoteVariation>n-of-m</cdf:VoteVariation>
  <cdf:VotesAllowed>1</cdf:VotesAllowed>
</cdf:Contest>
```

## 5.3 SelectionPosition

Section 3.4 discussed SelectionPosition in detail; this section contains additional information about usage and examples.

### 5.3.1 Position and Rank in SelectionPosition

It is important to know, when tabulating a CVR, the position on the ballot corresponding to the selection indication. There may be effectively only one position for those contest options in which one selection at most is possible, such as for a single candidate. However, consider the following contest in which three candidates are to be ranked for a county council position:

**Member of County Council at Large**

Contest Option	1st	2nd	3rd
Ileen Shapiro	[x]	[ ]	[ ]
Debbie Walsh	[ ]	[ ]	[x]
Sandra Kurt	[ ]	[x]	[ ]

**Figure 52 - Importance of Position and Rank**

Each contest option has three possible choices and each possible position indicates a ranking. The selection of Sandra Kurt's contest option corresponds to the following XML fragment:

```
<cdf:CVRContestSelection>
  <cdf:ContestSelectionId>_1HSK</cdf:ContestSelectionId>
  <cdf:OptionPosition>3</cdf:Position>
  <cdf:SelectionPosition>
    <cdf:HasIndication>yes</cdf:HasIndication>
    <cdf:IsAllocable>yes</cdf:IsAllocable>
    <cdf:NumberVotes>1</cdf:NumberVotes>
    <cdf:Position>2</cdf:Position>
  </cdf:SelectionPosition>
  <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
```

From the above, Kurt's (\_1HSK) position on the ballot is third and she was ranked second. This is represented by setting `CVRContestSelection.SelectionPosition` to 3 and `SelectionPosition.OptionPosition` to 2. Depending on how the contest options are structured, `SelectionPosition.Rank` could be used instead, e.g., for RCV contests.

### 5.3.2 Voter Made Marks (Paper Only)

A mark (made by a voter) may be associated with one or more `MarkMetricValues`, which is an implementation dependent measure of a mark.

When a metric is used, its type (`MarkMetricType`) must be first specified by the `ReportingDevice` playing the role of the CVR's `CreatingDevice`.

```
<cdf:ReportingDevice ObjectID="rd">
    <cdf:MarkMetricType>AJAX</cdf:MarkMetricType>
</cdf:ReportingDevice>
```

Thus, the `MarkMetricType` used is expected to be the same for all marks originating from the same `CreatingDevice`. From the above example, we can see that the mark has a quality measurement of type AJAX (a fictional quality measurement).

The `IsGenerated` boolean can be used to indicate whether the indication is the result of a mark on the ballot or whether it was generated by the application of contest rules or through adjudication. However, setting it to `false` does not confirm that an indication meets scanner-determined thresholds for mark quality, thus `HasIndication` is used. It can tell us if the mark met the threshold or logic of a `MarkMetricType` to be considered a selection indication for the contest option (machine interpretation), or if adjudication resulted in the capture of a selection (human interpretation).

The following table, repeated from Section 3, shows the mapping of mark types to `HasIndication` values:

Mark type	HasIndication
machine-readable mark	yes
marginally machine-readable mark	unknown
machine unreadable mark	no

Figure 53 - Mapping of Mark Types to HasIndication Values

### 5.3.3 Machine Generated Indications

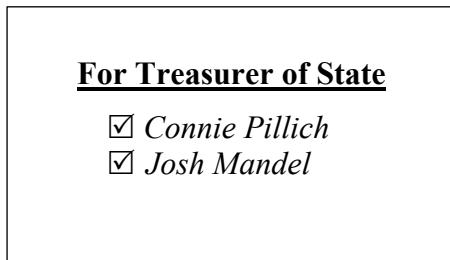
If an indication was generated by machine, such as in the indirect selections of straight party voting, IsGenerated can be set to true. HasIndication will have, of course, a value of yes.

```
<cdf:SelectionPosition>
    <cdf:HasIndication>yes</cdf:HasIndication>
    <cdf:IsAllocable>yes</cdf:IsAllocable>
    <cdf:IsGenerated>true</cdf:IsGenerated>
    <cdf:NumberVotes>1</cdf:NumberVotes>
    <cdf>Status>generated-rules</cdf>Status>
</cdf:SelectionPosition>
```

### 5.4 Handling Overvotes

This section only applies to paper ballots.

Consider again the treasurer contest, this time overvoted:



**Figure 54 - Overvoted Contest**

This can be represented with the XML below:

```
<cdf:CVRContest>
    <cdf:ContestId>_5TS</cdf:ContestId>
    <cdf:CVRContestSelection>
        <cdf:ContestSelectionId>_1ECP</cdf:ContestSelectionId>
        <cdf:OptionPosition>1</cdf:OptionPosition>
        <cdf:SelectionPosition>
            <cdf:HasIndication>yes</cdf:HasIndication>
            <cdf:IsAllocable>no</cdf:IsAllocable>
            <cdf:NumberVotes>1</cdf:NumberVotes>
        </cdf:SelectionPosition>
        <cdf:TotalNumberVotes>0</cdf:TotalNumberVotes>
    </cdf:CVRContestSelection>
    <cdf:CVRContestSelection>
        <cdf:ContestSelectionId>_1EJM</cdf:ContestSelectionId>
        <cdf:OptionPosition>2</cdf:OptionPosition>
        <cdf:SelectionPosition>
            <cdf:HasIndication>yes</cdf:HasIndication>
```

```
<cdf:IsAllocable>no</cdf:IsAllocable>
  <cdf:NumberVotes>1</cdf:NumberVotes>
</cdf:SelectionPosition>
  <cdf:TotalNumberVotes>0</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
  <cdf:Overvotes>1</cdf:Overvotes>
  <cdf:Undervotes>0</cdf:Undervotes>
</cdf:CVRContest>
```

Note that the indications are still accounted for, even though the votes will not be allocated to the contest option accumulators for Connie Pillich nor John Mandel, but instead to the overvote accumulator.

*Where adjudication is warrented, IsAllocable should be set to unknown.*

## 5.5 Write-Ins

Consider the following plurality contest in which one vote can be allocated across the 4 possible choices:

<b><u>For Governor</u></b>
<input type="checkbox"/> Edward FitzGerald
<input type="checkbox"/> John Kasich
<input type="checkbox"/> Anita Rios
<input checked="" type="checkbox"/> Write-In (John Smith)

**Figure 55 - Write-In Selection**

The write-in box was selected, and “John Smith” is the candidate name. This can be represented with the following XML fragment:

```
<cdf:CVRContest>
  <cdf:ContestId>_1GO</cdf:ContestId>
  <cdf:CVRContestSelection>
    <cdf:OptionPosition>4</cdf:OptionPosition>
    <cdf:SelectionPosition>
      <cdf:CVRWriteIn>
        <cdf:Text>John Smith</cdf:Text>
      </cdf:CVRWriteIn>
      <cdf:HasIndication>yes</cdf:HasIndication>
      <cdf:IsAllocable>unknown</cdf:IsAllocable>
      <cdf:NumberVotes>1</cdf:NumberVotes>
    </cdf:SelectionPosition>
```

```
<cdf>Status>needs-adjudication</cdf>Status>
  </cdf:CVRContestSelection>
</cdf:CVRContest>
```

Note that this fragment is the original CVR from the `CreatingDevice` and thus we do not yet know the validity of the write-in (it has not yet been adjudicated). Still we can say some things about it:

- The text of the write-in is John Smith, represented using the `Text` element.
- `SelectionPosition` represents both the selection of the write-in contest option and the write-in itself. Therefore, it is not possible for one to be valid but the other not.

If John Smith is determined to be a valid write-in, then the following may occur:

- `IsAllocable` is set to yes
- `CVRContestSelection` is linked to the `ContestSelection` associated with the candidate.

Some systems may not be capable of tabulating votes for the candidate underlying a write-in.

### 5.5.1 Write-In Counter

If desired, the `CVRContest` may contain the number of write-ins, i.e., the number of write-in contest options selected. This includes options that were selected, but no candidate was specified (e.g., a filled oval with an empty line).

### 5.5.2 Adjudication of Write-Ins

Adjudication can accomplish two things:

1. Determine if the name represents a valid write-in option, i.e., does the write-in text represent a valid write-in option?
2. Determine if the contest selection should be allocated.

(2) is different from (1), in that if it is determined that the write-in text represents a valid write-in option, it could still be overwritten by interpretation of voter intent.

## 5.6 CVR Snapshots

A CVR can be used throughout various points in the election lifecycle:

- Capture of contest selections
- Interpretation of contest selections

- Adjudication of contest selections
- Other operations

If a downstream system needs to modify the CVR, e.g., to add a CVRCContestSelection as the result of adjudication, a new CVRSnapshot should be created.

Consider the following XML fragment:

```
<cdf:CVRSnapshot ObjectId="css-02">
    <cdf:CVRCContest>
        <cdf:ContestId>_6RC</cdf:ContestId>
        <cdf:CVRCContestSelection>
            <cdf:ContestSelectionId>_1FMZ</cdf:ContestSelectionId>
            <cdf:OptionPosition>1</cdf:OptionPosition>
            <cdf:SelectionPosition>
                <cdf:HasIndication>yes</cdf:HasIndication>
                <cdf:IsAllocable>unknown</cdf:IsAllocable>
                <cdf:MarkMetricValue>76</cdf:MarkMetricValue>
                <cdf:NumberVotes>1</cdf:NumberVotes>
            </cdf:SelectionPosition>
        </cdf:CVRCContestSelection>
    </cdf:CVRCContest>
    <cdf>Status>needs-adjudication</cdf>Status>
    <cdf>Type>original</cdf>Type>
</cdf:CVRSnapshot>
```

This represents a CVR having a single marked contest, in which the indication is unknown (e.g., the mark is marginal). The Status of the CVRSnapshot is needs-adjudication so as to flag a downstream system or process.

Say that an adjudicator determines the mark is a result of the voter resting their pen on the contest option position, and not voter intent to make a selection for Mark Zetzer. Thus, HasIndication is set to no, and a new CVRSnapshot is created recording this action:

```
...
<cdf:CVRSnapshot ObjectId="css-03">
    <cdf:Annotation>
        <cdf:AdjudicatorName>Mark Kennamond</cdf:AdjudicatorName>
        <cdf:Message>Resting Mark, Mark Zetzer</cdf:Message>
        <cdf:TimeStamp>2018-05-16T12:10:09</cdf:TimeStamp>
    </cdf:Annotation>
    <cdf:CVRCContest>
        <cdf:ContestId>_6RC</cdf:ContestId>
        <cdf:CVRCContestSelection>
            <cdf:ContestSelectionId>_1FMZ</cdf:ContestSelectionId>
            <cdf:Position>1</cdf:Position>
            <cdf:SelectionPosition>
                <cdf:HasIndication>no</cdf:HasIndication>
                <cdf:Mark>
```

```
<cdf:MarkMetricValue>76</cdf:MarkMetricValue>
</cdf:Mark>
<cdf:NumberVotes>1</cdf:NumberVotes>
</cdf:SelectionPosition>
<cdf:TotalNumberVotes>0</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
</cdf:CVRContest>
<cdf:Type>interpreted</cdf:Type>
</cdf:CVRSnapshot>
```

Information about the adjudication is conveyed via the `Annotation` element. We can see the name of the adjudicator and the description of the changes to the CVR. There can be as many `Annotation` elements as required to describe the changes made to the CVR.

*Each `CVRSnapshot` should represent a set of changes to a CVR during a phase of processing. It is not necessary to create a separate `CVRSnapshot` for every adjudication change, however; this is left to the developer.*

### 5.6.1 Current CVR

If a system is looking to tabulate a set of CVRs, it must know for each CVR, which `CVRSnapshot` is the currently tabulatable record. This is achieved by using a `CurrentSnapshot` reference from CVR to the relevant `CVRSnapshot`.

## 5.7 Ballot Images

If a scanner is capable of capturing raster ballot images, then that data can be stored alongside the structured CVR. Ballot images can either be referenced from the CVR as a URI, or stored within it, as base64 encoded binary.

### 5.7.1 Storing the Image as a Reference

```
<BallotImage>
  <Location>http://192.168.1.1/imageserver/ballot1056.jpeg</Location>
</BallotImage>
```

### 5.7.2 Storing the Image Data

```
<BallotImage>
  <Image FileName="CVR1_Ballot.jpg"
    MimeType="image/jpeg">Q1ZSIEltYWdl</Image>
</BallotImage>
```

## 5.8 Voting Method Support

This specification supports all major voting methods currently in use in the United States (plurality, cumulative, N of M, range, RCV). The following two examples deal with RCV and cumulative.

### 5.8.1 Rank Choice Voting

Consider the following RCV contest in which candidates are to be ranked from 1 to 3:

**Member of County Council at Large**

Contest Option	1st	2nd	3rd
Ileen Shapiro	[ ]	[x]	[ ]
Debbie Walsh	[x]	[ ]	[ ]
Sandra Kurt	[ ]	[ ]	[x]

**Figure 56 - RCV Example**

This can be represented with the following XML fragment:

```
<cdf:CVRContest>
    <cdf:ContestId>_9CC</cdf:ContestId>
    <cdf:CVRContestSelection>
        <cdf:ContestSelectionId>_1HIS</cdf:ContestSelectionId>
        <cdf:OptionPosition>1</cdf:OptionPosition>
        <cdf:SelectionPosition>
            <cdf:HasIndication>yes</cdf:HasIndication>
            <cdf:IsAllocable>yes</cdf:IsAllocable>
            <cdf:NumberVotes>1</cdf:NumberVotes>
            <cdf:Rank>2</cdf:Rank>
        </cdf:SelectionPosition>
        <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
    </cdf:CVRContestSelection>
    <cdf:CVRContestSelection>
        <cdf:ContestSelectionId>_1HDW</cdf:ContestSelectionId>
        <cdf:OptionPosition>2</cdf:OptionPosition>
        <cdf:SelectionPosition>
            <cdf:HasIndication>yes</cdf:HasIndication>
            <cdf:IsAllocable>yes</cdf:IsAllocable>
            <cdf:NumberVotes>1</cdf:NumberVotes>
            <cdf:Rank>1</cdf:Rank>
        </cdf:SelectionPosition>
        <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
    </cdf:CVRContestSelection>
    <cdf:CVRContestSelection>
```

```
<cdf:ContestSelectionId>_1HSK</cdf:ContestSelectionId>
<cdf:OptionPosition>3</cdf:OptionPosition>
<cdf:SelectionPosition>
    <cdf:HasIndication>yes</cdf:HasIndication>
    <cdf:IsAllocable>yes</cdf:IsAllocable>
    <cdf:NumberVotes>1</cdf:NumberVotes>
    <cdf:Rank>3</cdf:Rank>
</cdf:SelectionPosition>
<cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
<cdf:Undervotes>0</cdf:Undervotes>
</cdf:CVRContest>
```

Each candidate may be ranked using the Rank attribute. The rank may or may not be the same as the Position.

### 5.8.2 Cumulative Voting

Consider the following cumulative contest in which up to 3 votes can be allocated among the candidates:

**Member of County Council at Large**

Contest Option	1st	2nd	3rd
Ileen Shapiro	[ ]	[x]	[ ]
Debbie Walsh	[x]	[ ]	[x]
Sandra Kurt	[ ]	[ ]	[ ]

**Figure 57 - Cumulative Voting Example**

In this example, Ileen Shapiro has 1 vote and Debbie Walsh has 2. Assuming the ballot was hand marked, the following CVR fragment could be constructed:

```
<cdf:CVRContest>
<cdf:ContestId>_9CC</cdf:ContestId>
<cdf:CVRContestSelection>
    <cdf:ContestSelectionId>_1HIS</cdf:ContestSelectionId>
    <cdf:OptionPosition>1</cdf:OptionPosition>
    <cdf:SelectionPosition>
        <cdf:HasIndication>yes</cdf:HasIndication>
        <cdf:IsAllocable>yes</cdf:IsAllocable>
        <cdf:NumberVotes>1</cdf:NumberVotes>
        <cdf:Position>2</cdf:Position>
    </cdf:SelectionPosition>
    <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
<cdf:CVRContestSelection>
```

```
<cdf:ContestSelectionId>_1HDW</cdf:ContestSelectionId>
<cdf:OptionPosition>2</cdf:OptionPosition>
<cdf:SelectionPosition>
    <cdf:HasIndication>yes</cdf:HasIndication>
    <cdf:IsAllocable>yes</cdf:IsAllocable>
    <cdf:NumberVotes>1</cdf:NumberVotes>
    <cdf:Position>1</cdf:Position>
</cdf:SelectionPosition>
<cdf:SelectionPosition>
    <cdf:HasIndication>yes</cdf:HasIndication>
    <cdf:IsAllocable>yes</cdf:IsAllocable>
    <cdf:NumberVotes>1</cdf:NumberVotes>
    <cdf:Position>3</cdf:Position>
</cdf:SelectionPosition>
<cdf:TotalNumberVotes>2</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
<cdf:Undervotes>1</cdf:Undervotes>
</cdf:CVRContest>
```

Because the ballot was handmarked and Debbie Walsh received two votes, she has two SelectionIndications. If the same vote was cast on a ballot marking device, the CVR could be simplified somewhat:

```
<cdf:CVRContest>
<cdf:ContestId>_9CC</cdf:ContestId>
<cdf:CVRContestSelection>
    <cdf:ContestSelectionId>_1HIS</cdf:ContestSelectionId>
    <cdf:OptionPosition>1</cdf:OptionPosition>
    <cdf:SelectionPosition>
        <cdf:HasIndication>yes</cdf:HasIndication>
        <cdf:IsAllocable>yes</cdf:IsAllocable>
        <cdf:NumberVotes>1</cdf:NumberVotes>
    </cdf:SelectionPosition>
    <cdf:TotalNumberVotes>1</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
<cdf:CVRContestSelection>
    <cdf:ContestSelectionId>_1HDW</cdf:ContestSelectionId>
    <cdf:OptionPosition>2</cdf:OptionPosition>
    <cdf:SelectionPosition>
        <cdf:HasIndication>yes</cdf:HasIndication>
        <cdf:IsAllocable>yes</cdf:IsAllocable>
        <cdf:NumberVotes>2</cdf:NumberVotes>
    </cdf:SelectionPosition>
    <cdf:TotalNumberVotes>2</cdf:TotalNumberVotes>
</cdf:CVRContestSelection>
<cdf:Undervotes>0</cdf:Undervotes>
</cdf:CVRContest>
```

The representation of the indication for Ilene Shapiro is unchanged, but Debbie Walsh's votes have been consolidated into a single SelectionIndication, with a NumberVotes of 2.



## Appendix A—Acronyms

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

BMD	Ballot Marking Device
CDF	Common Data Format
CVR	Cast Vote Record
DRE	Direct Record Electronic
EMS	Election Management System
ID	Identifier
JSON	JavaScript Object Notation
RCV	Ranked Choice Voting
UML	Unified Modeling Language
URI	Uniform Resource Identifier
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:*”.

**Adjudication:** Process of resolving flagged cast ballots to reflect voter intent.

Common reasons for flagging include:

- write-ins,
- overvotes,
- marginal marks,
- having no contest selections marked on the entire ballot, or
- the ballot being unreadable by a scanner.

**Batch:** As used in auditing elections, a collection of paper ballots gathered as a group for tabulation or for auditing.

**Batch fed scanner:** A ballot processing device that:

- accepts stacks of hand-marked or BMD-produced paper ballots and automatically processes them until the stack is empty,
- is usually used at an election jurisdiction’s central location,
- is mostly commonly used to process absentee or mail ballots,
- usually has input and output hoppers for ballots,
- scans a ballot and rejects it if either unreadable or un-processable,
- detects, interprets, and validates contest selections,
- detects and sorts (either digitally or physically) ballots that are unreadable or un-processable, or that contain undeterminable selections, marking exceptions, or write-ins, and
- tabulates and reports contest results as required.

This unit was previously referred to as central count optical scanner or CCOS.

**Ballot marking device (BMD):**

A device that:

- permits contest options to be reviewed on an electronic interface,
- produces a human-readable paper ballot, and
- does not make any other lasting record of the voter’s selections.

<b>Cumulative voting:</b>	A voting method used in multi-seat contests where a voter is permitted to distribute allowed selections to 1 or more candidates in whole vote increments. See also proportional voting.
<b>DRE:</b>	A vote-capture device that allows: <ul style="list-style-type: none"><li>• electronic presentation of a ballot,</li><li>• electronic selection of valid contest options, and</li><li>• electronic storage of contest selections as individual records.</li></ul> It also provides a summary of these contest selections.
<b>Election management system (EMS):</b>	Set of processing functions and databases within a voting system typically used to: <ul style="list-style-type: none"><li>• define, develop, and maintain election databases,</li><li>• perform election definition and ballot layout functions,</li><li>• create ballot presentation templates for ballot printers or devices used by voters for ballot markup,</li><li>• count votes,</li><li>• consolidate and report results and maintain audit trails.</li></ul>
<b>Hash function:</b>	A data processing function, usually using a specified NIST approved cryptographic rule, to produce a fixed-length bit string from a set of data that is variable in both content and length and unique to that specific data. The resultant “hash value” cannot be used to reproduce the original data that it was derived from. It is used as a digital signature to confirm that the data being evaluated is identical to another set of data with the same hash value.
<b>N-of-M voting:</b>	Voting variation in which the voter is entitled to allocate a fixed number of votes (N) over a list of M contest options or write-in options, with the constraint that at most 1 vote may be allocated to a given contest option. This usually occurs when multiple seats are concurrently being filled in a governing body such as a city council or school board where candidates run at-large. The voter is not obliged to allocate all N votes. 1-of-M is N-of-M where N = 1.
<b>Overvote:</b>	Occurs when the number of selections made by the voter in a contest is more than the maximum number allowed in a contest.
<b>Range voting:</b>	A voting method for single-seat elections, in which voters give each candidate a score, the scores are added (or averaged), and the candidate with the highest total is elected.
<b>Ranked Choice Voting (RCV):</b>	A set of election methods which allow each voter to rank contest options in order of the voter's preference, in which votes are counted in rounds using a series of runoff tabulations to defeat contest options with the fewest votes, and which elects a winner with a

majority of final round votes in a single-winner contest and provides proportional representation in multi-winner contests.

**Schema:** A file containing definitions of data elements and attributes with rules for usage, e.g., for JSON or XML.

**Straight party override:** Explicit voter selection that overrides or supplements the vote selections made by a straight party voting option. Straight party overrides may be subject to state election rules for how they work or whether they are allowed.

**Tabulator:** A device that counts votes.

**Undervote:** Occurs when the number of voter selections in a contest is less than the maximum number allowed for that contest or when no selection is made. The number of undervotes is equal to the number of votes lost, e.g., if no selection is made in a single option contest, the number of votes lost is 1.

**Voter facing scanner:** A ballot processing device that:

- accepts hand-marked or BMD-produced paper ballots one sheet at a time;
- is usually used for in-person voting;
- permits election workers to open and close the polls; scans a ballot and rejects it if either unreadable or un-processable; detects, interprets and validates contest selections; notifies the voter of voting exceptions (such as undervotes or overvotes) or unreadable marks; stores accepted ballots in a secure container; sorts or otherwise marks ballots or ballot images that need subsequent human review; and tabulates and reports contest results after polls are closed.

This unit was previously referred to as precinct count optical scanner or PCOS.

**Voting method:** Voting style such as in-person voting, absentee voting, straight party voting, N-of-M voting, cumulative voting, or ranked choice voting.

**Write-in option:** A type of contest option that allows a voter to specify a candidate, usually not already listed as a contest option. Depending on jurisdiction rules, in some cases only previously approved names will be considered as valid write-in contest selections.

## Appendix C—References

- [1] Object Management Group (OMG), *UML Specification version 1.1* (OMG document ad/97-08-11) September 22, 2011, <http://omg.org/> [accessed 02/01/2019].
- [2] W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation, November 26, 2008, <http://www.w3.org/TR/xml/> [accessed 02/01/2019].
- [3] JavaScript Object Notation, <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> [accessed 02/01/2019].

## Appendix D—File Download Locations

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

<https://github.com/usnistgov/CastVoteRecords>

These files are also available from:

<http://vote.nist.gov>

The files include:

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

## **Appendix E—Change Log**

### **Version 1 Release 2 – March 31, 2020**

- In the UML documentation, corrected the order of UML class attributes and enumeration values in the tables so that they match the order in the UML model and in the JSON and XML schemas.
- In the UML documentation, mentions of class and attribute names are properly linked to each other. Also done for mentions of enumeration and value names.