



Clear Ballot

ClearVote 2.1

**ClearDesign Accessible
Definition File Guide**

ClearDesign Accessible Definition File Guide

Clear Ballot Part Number: 100133-10017

Copyright © 2012–2020 Clear Ballot Group. All rights reserved.

This document contains proprietary and confidential information consisting of trade secrets of a technical and commercial nature. The recipient may not share, copy, or reproduce its contents without express written permission from Clear Ballot Group.

ClearAccess, ClearAudit, Clear Ballot, ClearCast, ClearCount, ClearData, ClearDesign, ClearVote, DesignServer, DesignStation, Image-to-Ballot Traceability, MatchPoint, ScanServer, ScanStation, Speed Accuracy Transparency, Visualization of Voter Intent, Visual Verification, and Vote Visualization are trademarks or registered trademarks of Clear Ballot Group.

ScandAll PRO is a trademark of Fujitsu Limited. All rights reserved. Other product and company names mentioned herein are the property of their respective owners.

Clear Ballot Group
2 Oliver Street, Suite 200
Boston, MA 02109
857-250-4961
clearballot.com

Document history

Date	Description	Version	Author
06/21/2019	Initial version	1.0	Joe Srednicki
11/04/2019	Updated cover page	1.0.1	Joe Srednicki
02/12/2020	Minor edits	1.0.2	Joe Srednicki



Table of contents

Preface	5
Chapter 1. Introduction	6
1.1 What is an ADF file?	6
1.2 What is an ADFx file?	6
Chapter 2. Contents of the ADF file	7
2.1 Contents of config.txt	7
2.2 Contents of election.json	8
Chapter 3. Contents of the ADFx file	20

Preface

This section defines the purpose of this document. It contains the following subsections.

- About this document
- Scope of this document
- Intended audience

About this document

This document describes the format of the accessible definition file (ADF) exported by ClearDesign. ClearAccess imports the ADF to set up an election.



A ClearVote® system can comprise the ClearAccess®, ClearAudit®, ClearCast®, ClearCount®, and ClearDesign® products. Jurisdictions are not required to purchase all products. You can ignore references to any ClearVote products that are not part of your voting system. Also ignore implementation options that are not relevant to your policies and procedures.

Scope of this document

This document contains the following chapters:

- [Chapter 1. Introduction](#)
- [Chapter 2. Contents of the ADF file](#)
- [Chapter 3. Contents of the ADFX file](#)

Intended audience

The document is for state and federal election officials and their voting system test laboratories as part of the Technical Data Package (TDP) required to certify the ClearVote system for use. This document is also used by Clear Ballot personnel who support election officials and staff.

Chapter 1. Introduction

This chapter describes an unencrypted ADF file and an encrypted ADFx file.

1.1 What is an ADF file?

An accessible definition file (ADF) is a ZIP archive file created by ClearDesign that describes an election, including the data to support the accessible voting system. The ADF file uses HMACs to verify that the data created by ClearDesign has not been altered.

ClearAccess uses the ADF to load an election on the ClearAccess product.

The name of the ADF file ends in “adf.zip”.

1.2 What is an ADFx file?

ClearDesign version 2.0 and later versions produce an encrypted ADF file called ADFx. The ADFx file contains the functionality of an ADF file, but with encryption added for security.

The encrypted ADF file ends in “adfx.zip”.

Chapter 2. Contents of the ADF file

This chapter lists the contents of the unencrypted ADF file. The ADF is used to transfer data from ClearDesign to ClearAccess. The ADF contains all data necessary to produce the electronic ballot in ClearAccess.

The ADF Zip archive contains the following files (Table 2-1).

Table 2-1. Files contained in the ADF Zip archive

File	Contains
config.txt	Configuration and validation information
election.json	The election and ballot definitions in a JSON format
template.html	The HTML template for displaying the election ballot

2.1 Contents of config.txt

Table 2-2 lists the fields in config.txt.

Table 2-2. Fields in config.txt

Field	Description
format	A string that defines this file format. This string must be "CLEARBALLOT_ADF"
version	The version of the ADF file format.
ClearDesign version	The version of the software that created the ADF file.
electionName	The name of the election.
electionDate	The date of the election in the format YYYY-MM-DD.
creationDate	The time and date the file was created in the ISO format.
mediaDate	The date the media was created format YYYY-MM-DD.
mediaVersion	The version of the election data. When the data changes, this number increments, and the mediaCopy is reset to 0.
mediaCopy	The copy number of the media. Each time media is created for the same version, the mediaCopy is incremented.
mediaHash	The SHA256 hash of the media, for simple identifying purposes.



Table 2-2. Fields in config.txt (continued)

Field	Description
htmlHmac	The SHA256 HMAC of the HTML data. Used to validate the data.
electionHmac	The SHA256 HMAC of the election data. Used to validate the data.
electionCode	The hashed code used by the election administrator to validate the data.
pollworkerCode	The hashed code used by the poll worker to validate the data.
votingCode	The hashed code used by the voting session to validate the data.

2.2 Contents of election.json

This file contains the election and ballot definitions.

The election.json file for version 7.

```
audios: (dictionary) The dictionary of audio recording
{
  <key>: (string) The name of the audio entity model
  <value>: (dictionary) The dictionary of entity id's and language audio
  {
    <key>: (integer) the id of the entity
    <value>: (dictionary) The dictionary of the language id to the audio
    {
      <key>: (integer) the id of the language
      <value>: (string) The base64 encoding of the audio
    }
  }
},
ballotGroupStyles: (list of dictionaries) the list of BallotGroupStyles in the election
[{
  id: (integer) the id of the entity
  sortSeq: (integer) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: the abbreviation of the entity
  exportID: (string - optional) the export id of the entity
  importID: (string - optional) the import id of the entity
}]
```

```

ballotGroups: (list of dictionaries) the list of BallotGroups in the election
[{
  id: (integer) the id of the entity
  sortSeq: (integer) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: the abbreviation of the entity
  exportID: (string - optional) the export id of the entity
  importID: (string - optional) the import id of the entity
  ballotGroupStyleId: (integer) the id of the ballotGroupStyle
}]
ballotLayouts: The list of ballotLayouts in the election
[{
  id: (integer) The id of the ballotLayout
  name: (integer) The name of the ballotLayout
  type: (string) The type of the BallotLayout ('Card', 'CardStyle', 'Ballot', 'BallotStyle')
  voterGroupId: (integer): The voterGroup id for the BallotLayout
  cardSequence: (integer) The card sequence within the ballot
  cardTemplateId: (integer) The cardTemplate used by the ballot
  layoutStyleId: (integer) The layoutStyle id used by the ballotLayout
  ballotContests: (list of dictionaries) The list of ballotContests
  [
    {
      contestId: (integer) The contestId of the contest for the BallotContest
      ballotChoices: (list of dictionaries) The list of ballotChoices for the ballotContest
      [
        {
          candidateId: (integer) The candidate Id
          voterGroupIds: (list of integers) The list of voterGroupId's for the candidate choice[
          ]
        }
      ]
    }
  ]
  ballotSetId: (integer) The ballotSet id
  ballotSplits: (list of dictionaries) The list of ballotSplits
  [
    {
      ballotGroupId: (integer) The ballotGroupId associated with the BallotSplit
      ballotId: (integer) The ballotId associated with the BallotSplit
      ballotSequence: (integer) The ballotSequence for the BallotSplit
      precinctId: (integer) The precinct id for the BallotSplit
      splitId: (integer) The split id for the BallotSplit
    }
  ]
  cards: (list of dictionaries) The list of cards associated with the ballot
  [
    {
      id: (integer) the id of the entity
      sortSeq: (integer) the sort sequence of the entity
      name: (integer) the name of the entity
      shortName: (string) the short name of the entity
      abbreviation: (string) the abbreviation of the entity
      voterGroupId: (integer) The voterGroup id for the Card
    }
  ]
}

```



```
cardSequence: (integer) The card sequence within the ballot
cardTemplateId: (integer) The cardTemplate used by the Card
layoutStyleId: (integer) The layoutStyle id used by the Card
contests: (list of dictionaries) The list of contests on the card
[{
    contestId: (integer) The contest id
    side: (integer) The side of the card
    rect: (dictionary) The outer rectangle for the contest
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    },
    textRect: (dictionary) The rectangle for the contest text
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    }
},
candidateRect: (dictionary) The rectangle for all the candidates
{
    top: (float) The top position of the rectangle in timing mark coordinates
    left: (float) The left position of the rectangle in timing mark coordinates
    height: (float) The height of the rectangle in timing mark coordinates
    width: (float) The width of the rectangle in timing mark coordinates
},
candidates: (list of dictionaries) The list of candidates
[{
    candidateId: (integer) The candidate id
    side: (integer) The side of the card
    voterGroupId: (integer) The voterGroupId or -1 if more than one
    voterGrouprect: (dictionary) The out rectangle for the candidate
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    },
    textRect: (dictionary) The rectangle for the candidate text
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    },
}
```

```
voteMark: (dictionary) The rectangle for the vote mark
{
    top: (float) The top position of the rectangle in timing mark coordinates
    left: (float) The left position of the rectangle in timing mark coordinates
    height: (float) The height of the rectangle in timing mark coordinates
    width: (float) The width of the rectangle in timing mark coordinates
},
}]
}]
Headers: (list of dictionaries) The list of headers on the card
{
    headerId: (integer) The header id
    side: (integer) The side of the card
    rect: (dictionary) The outer rectangle of the header
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    }
    textRect: (dictionary) The rectangle for the text
    {
        top: (float) The top position of the rectangle in timing mark coordinates
        left: (float) The left position of the rectangle in timing mark coordinates
        height: (float) The height of the rectangle in timing mark coordinates
        width: (float) The width of the rectangle in timing mark coordinates
    }
},
}]
}]
```

```
ballotSets: (list of dictionaries) The list of ballotSet in the election
[{
    id: (integer) the id of the entity
    sortSeq: (integer) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    splitIdentifier: (string) The ballotSplit identifier for the cards ('name' or 'ballotSequence')
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
}]
```



ballotStyles: (list of dictionaries) The list of ballotStyles in the election

{

 id: (integer) the id of the entity
 sortSeq: (integer) the sort sequence of the entity
 name: (string) the name of the entity
 shortName: (string) the short name of the entity
 abbreviation: (string) the abbreviation of the entity
 ballotSetId: (integer) The BallotSet id for the entity
 voterGroupId: (integer) The voterGroup id for the entity

}]

ballots: (list of dictionaries) The list of ballots in the election

{

 id: (integer) the id of the entity
 sortSeq: (integer) the sort sequence of the entity
 name: (string) the name of the entity
 shortName: (string) the short name of the entity
 abbreviation: (string) the abbreviation of the entity
 ballotStyleId: (integer) The ballotStyle id for the entity

}]

cardTemplates: (list of dictionaries) The list of cardTemplates in the election

{

 id: (integer) the id of the entity
 sortSeq: (integer) the sort sequence of the entity
 name: (string) the name of the entity
 shortName: (string) the short name of the entity
 abbreviation: (string) the abbreviation of the entity
 height: (integer) The height of the card in 100th of an inch
 width: (integer) The width of the card in 100th of an inch
 backColumns: (integer) The number of logical columns on the back
 backOrientation: (string) The orientation of the back 'L' or 'P'
 frontColumns: (integer) The number of logical columns on the front
 frontOrientation: (string) The orientation of the front 'L' or 'P'
 ovalPosition: (string) The oval positions 'L' or 'R'
 colsPerInch: (integer) The number of horizontal timing marks per inch
 rowsPerInch: (integer) The number of vertical timing marks per inch

}]

contests: (list of dictionaries) The list of contests in the election

{

 id: (integer) the id of the entity
 sortSeq: (integer) the sort sequence of the entity
 name: (string) the name of the entity
 shortName: (string) the short name of the entity
 abbreviation: (string) the abbreviation of the entity
 type: (string) The type of contest ('C', 'P', 'S', 'Q', 'R')
 voteFor: (integer) The number to vote for

```

numColumns: (integer) the number of logical columns the contest is to span
partyPreferenceId: (integer or None) the id of the party preference contest if there is one
straightPartyId: (integer or None) the id of the straight party contest if there is one
candidateColumns: (integer) The number of columns to put the candidates in
candidateRows: (integer) The number of rows to allocate per candidate
exportID: (string - optional) The export id of the entity
importID: (string - optional) The import id of the entity
ballotChoices: ( list of dictionaries) ballotChoices for the contest
{
    candidateId: (integer) The candidate id associated with the choice
    voterGroupIds: (list of integers) The list of voterGroups id associated with the choice
    [
    ]
ballotText: (list of dictionaries) The ballot text for the contest
{
    id: (integer) The id of the ballot text record
    languageId: (integer) The language associated with the ballot text record
    ballotText: (integer) The ballot text
}
candidates: (list of dictionaries) The list of candidates for the contest
{
    id: (integer) the id of the entity
    sortSeq: (integer) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
    ballotText: (list of dictionaries) The ballot text for the contest
    {
        id: (integer) The id of the ballot text record
        languageId: (integer) The language associated with the ballot text record
        ballotText: (string) The ballot text
    }
    rotationGroup: (integer) The rotation group for the candidate
    type: (string) The type of candidate ('candidate', 'write-in', 'label-only')
    voterGroupIds: (list of integers) The list of voterGroup ids for the candidate
    [
    ]
}
entityStyle: (dictionary) The entityStyle overrides for the contest
{
    id: (integer) the id of the entity
    sortSeq: (integer) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    backgroundColor: (string or None) the background color in CSS format
    borderColor: (string or None) The border color in CSS format

```

```
borderBottom: (integer or None) the bottom border width in pixels
borderLeft: (integer or None) The left border width in pixels
borderRight: (integer or None) The right border width in pixels
borderTop: (integer or None) The top border width in pixels
font: (string or None) The font name to use
justify: (string or None) The text justification 'left', 'center', 'right', 'full'
marginBottom: (integer or None) The bottom margin width in pixels
marginLeft: (integer or None) The left margin width in pixels
marginRight: (integer or None) The right margin width in pixels
marginTop: (integer or None) The top margin width in pixels
option: (string or None) The font options ('bold', 'italics', 'underline')
paddingBottom: (integer or None) The bottom padding width in pixels
paddingLeft: (integer or None) The left padding width in pixels
paddingRight: (integer or None) The right padding width in pixels
paddingTop: (integer or None) The top padding width in pixels
size: (integer or None) The font size in points
textBackgroundColor: (string or None) The text background color in CSS format
textColor: (string or None) The text color in CSS format
}
voterGroupIds: (list of integers) The list of voterGroup id's for the contest
[
]
}]
districtCategories: (list of dictionaries) The list of district categories in the election
//{
  id: (integer) the id of the entity
  sortSeq: (string) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  exportID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
}
districts: (list of dictionaries) The list of districts in the election
//{
  id: (integer) the id of the entity
  sortSeq: (string) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  exportID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
  districtCategoryId: The district categories id
}
]
```



```

election: The election record
{
  id: (integer) the id of the entity
  sortSeq: (string) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  exportID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
  electionDate: (string) The date of the election
}

headers: The list of headers in the election
[{
  id: (integer) the id of the entity
  sortSeq: (integer) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  export ID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
  ballotSetId: (integer or None) the ballotSet id for the entity
  numColumns: (integer) The number of logical columns for the header
  location: (string) The location for the header
  placement: (string) The placement for the header
  startColumn: (integer) The starting column for the header
  startSortSeq: (integer or None) The sort sequence of the first contest linked to this header
  endSortSeq: (integer or None) The sort sequence of the last contest linked to this header
  type: (string) The type of header ('card-heder', 'card-footer', 'contest-header', 'contest-footer')
  voterGroupId: (integer) the voterGroup id for the header (-1 for all voterGroups)
  ballotText: (list of dictionaries) The ballot text for the contest
  [
    {
      id: (integer) The id of the ballot text record
      languageld: (integer) The language associated with the ballot text record
      ballotText: (string) The ballot text
    }
  ]
  entityStyle: (dictionary) The entityStyle overrides for the contest
  {
    id: (integer) the id of the entity
    sortSeq: (string) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    backgroundColor: (string or None) the background color in CSS format
    borderColor: (string or None) The border color in CSS format
    borderBottom: (integer or None) the bottom border width in pixels
    borderLeft: (integer or None) The left border width in pixels
    borderRight: (integer or None) The right border width in pixels
  }
}

```

```
borderTop: (integer or None) The top border width in pixels
font: (string or None) The font name to use
justify: (string or None) The text justification 'left', 'center', 'right', 'full'
marginBottom: (integer or None) The bottom margin width in pixels
marginLeft: (integer or None) The left margin width in pixels
marginRight: (integer or None) The right margin width in pixels
marginTop: (integer or None) The top margin width in pixels
option: (string or None) The font options ('bold', 'italics', 'underline')
paddingBottom: (integer or None) The bottom padding width in pixels
paddingLeft: (integer or None) The left padding width in pixels
paddingRight: (integer or None) The right padding width in pixels
paddingTop: (integer or None) The top padding width in pixels
size: (integer or None) The font size in points
textBackgroundColor: (string or None) The text background color in CSS format
textColor: (string or None) The text color in CSS format
}
}]
}
```

languages: (list of dictionaries) The list of languages for the election

```
[{
  id: (integer) the id of the entity
  sortSeq: (string) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  exportID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
  isDefault: (boolean) Flag for the default language
},
```

layoutStyles: (list of dictionaries) The list of layout Styles for the election

```
[{
  id: (integer) the id of the entity
  sortSeq: (string) the sort sequence of the entity
  name: (string) the name of the entity
  shortName: (string) the short name of the entity
  abbreviation: (string) the abbreviation of the entity
  exportID: (string - optional) The export id of the entity
  importID: (string - optional) The import id of the entity
}]
```

```

messages: (dictionary) The dictionary of messages for the election
{
    <key> (integer) The id of the message
    <value> (dictionary) The dictionary of language ids and text
    {
        <key> (string) The id of the language
        <value> (string) The text for the message
    }
}

options: The dictionary of options
{
    allowRecord: (boolean) Flag to allow keystroke recording. Used for certification testing only
    audioOn: (boolean) Flag controlling the default playing of audio
    cancelTimeout: (integer) The number of seconds to display the timeout warning before canceling the session
    crossEndorseOnBallotOnce: (boolean) Flag controlling whether cross-endorsed candidates are on the ballot once
    dpiSetting: (string or None) The dpi for the screen
    hasServer: (boolean) Flag indicating there is a server to get data from
    inactivityTimeout: (integer) the number of second if inactivity before showing the inactivity warning messages
    inputDevice: (string) The default input device ('screen', 'sip-puff', 'ezkey')
    mustViewAll: (boolean) Flag indicating that all contests must be viewed before showing the summary screen
    offsetX: (integer or None) The X offset for the printing of ballots. To handle variations in printers
    offsetY: (integer or None) The Y offset for the printing of ballots. To handle variations in printers
    oneTouch: (Boolean or None) Flag controlling whether a selection that would cause an over vote clears the other votes instead
    printOvalsOnly: (boolean) Flag controlling whether only ovals (not ballot text and timing marks) are printed
    screenOn: (boolean) Flag controlling whether the ballot is displayed on the screen
    straightPartyOption: (string) Straight party voting option ('exclusive', 'override', 'additive', 'combine')
    warnBlankVote: (boolean) Warn about a blank voted contest
    warnStraightParty: Warn about a changes to straight party contest
    warnUnderVote: (boolean) Warn about a under voted contest
}

precincts: (list of dictionaries) The list of precincts in the election
[{
    id: (integer) the id of the entity
    sortSeq: (string) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
}
]

```

```
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
}

splits: (list of dictionaries) The list of splits in the election
[{
    id: (integer) the id of the entity
    sortSeq: (string) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
    districtIds: (list of integers) The list of district ids the split is part of
    [
    ]
}]
voteCenterCategories: (list of dictionaries) The list of VoteCenterCategories in the election
[{
    id: (integer) the id of the entity
    sortSeq: (string) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
}]
voteCenters: (list of dictionaries) The list of VoteCenters in the election
[{
    id: (integer) the id of the entity
    sortSeq: (string) the sort sequence of the entity
    name: (string) the name of the entity
    shortName: (string) the short name of the entity
    abbreviation: (string) the abbreviation of the entity
    exportID: (string - optional) The export id of the entity
    importID: (string - optional) The import id of the entity
    voteCenterPrecincts: (list of dictionaries) The list of precincts in the voteCenter
    [
        ballotId: (integer) The ballotId for the entity
        ballotSequence: (integer) The ballotSequence for the entity
        ballotSetId: (integer) The ballotSet id for the entity
        precinctId: (integer) The precinct id for the entity
        splitId: (integer) The split id for the entity
    ]
}]
}
```

voterGroups: (list of dictionaries) The list of voterGroups in the election

[

 id: (integer) the id of the entity

 sortSeq: (string) the sort sequence of the entity

 name: (string) the name of the entity

 shortName: (string) the short name of the entity

 abbreviation: (string) the abbreviation of the entity

 exportID: (string - optional) The export id of the entity

 importID: (string - optional) The import id of the entity

 isDefault: Flag indicating this is the default (Non-partisan) voteGroup

 ballotText: (list of dictionaries) The ballot text for the contest

[

 id: (integer) The id of the ballot text record

 languageId: (integer) The language associated with the ballot text record

 ballotText: (string) The ballot text

]

]

Chapter 3. Contents of the ADFX file

An encrypted ADFX file is a ZIP file that contains the following:

- An unencrypted config.json text file
- An encrypted file containing the election data that ends in "adf.zip.cbx"

Because of encryption, you cannot open and view the "adf.zip.cbx" file.

Table 3-1 lists the fields in config.json.

Table 3-1. Fields contained in config.json

Field	Description
format	The format of the file. "format": "CLEARBALLOT_ADFX",
version	The version of the format "version": "1",
applicationName	The application that created the file. "applicationName": 'ClearDesign',
applicationVersion	The version of the application that created the file.
electionName	The election name
electionDate	The election date
jurisdictionName	The name of the jurisdiction
mediaVersion	The media version. This field changes when the data in the ADF changes.
mediaCopy	This value changes every time a ADF is created and resets to 0 when the version changes.
mediaDate	The creation date of the media in ISO format/