```
1  static_diagram DIGITALVOTERLIST
2  component
3    cluster STATIONANDMANAGER
4    component
5      deferred class STATION
6        feature
7          Address : IPADDRESS
8
9          Manager : IPADDRESS
10
11         SetManager : void
12           -> address : IPADDRESS
13           require address /= void
14           ensure Manager = address
15         end
16
17         EnoughStations : BOOLEAN
18
19         ElectionInProgress : BOOLEAN
20
21         Peers : SORTED_LIST[IPADDRESS]
22
23         Database : IDATABASE
24
25         Communicator : ICOMMUNICATOR
26
27         Crypto : ICRYPTO
28
29         SetCrypto : void
30           -> newcrypto : ICRYPTO
31           require newcrypto /= void
32           ensure Crypto = newcrypto
33         end
34
35         Logger : ILOGGER
36
37         SetLogger : void
38           -> newlogger : ILOGGER
39           require newlogger /= void
40           ensure Logger = newlogger
41         end
42
43         UI : IDVLUI
44
45         IsManager : BOOLEAN
46
47         Listening : BOOLEAN
48
49         MasterPassword : VALUE
50
51         SetMasterPassword : void
52           -> password : VALUE
53           require password /= void and MasterPassword = void
54           ensure MasterPassword = password
55         end
56
57         StationActive : BOOLEAN
58           -> address : IPADDRESS
59           require address /= void
60         end
61
62         DiscoverPeers : SEQUENCE[IPADDRESS]
```

```
63              ensure result /= void
64            end
65
66            ValidMasterPassword : BOOLEAN
67              -> password : STRING
68              require password /= void
69            end
70
71            ShutDownElection : void
72
73            ExchangePublicKeys : void
74              -> address : IPADDRESS
75              require address /= void and StationActive(address)
76            end
77
78            StartListening : void
79              require not Listening
80              ensure Listening
81            end
82
83            StopListening : void
84              require Listening
85              ensure not Listening
86            end
87
88            StartElection : void
89              require not ElectionInProgress
90              ensure ElectionInProgress
91            end
92
93            EndElection : void
94              require ElectionInProgress
95              ensure not ElectionInProgress
96            end
97
98            AddPeer : void
99              -> address : IPADDRESS
100             -> key : ASYMMETRICKEY
101             require address /= void and not Peers.Contains(address)
102             ensure Peers.Contains(address)
103           end
104
105           RemovePeer : void
106             -> address : IPADDRESS
107             require address /= void and Peers.Contains(address)
108             ensure not Peers.Contains(address)
109           end
110
111           StartNewManagerElection : void
112
113           ElectNewManager : void
114             require not StationActive(Manager)
115             ensure Manager /= old Manager
116           end
117
118           RequestBallot : void
119             -> voterNumber : VOTERNUMBER
120             -> cpr : CPR
121             require Database.get(voterNumber, cpr) = NOTRECEIVED
122           end
123
124           RequestBallotCPROnly : void
```

```
125             -> cpr : CPR
126             -> password : STRING
127             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = NOTRECEIVED
128           end
129
130           BallotReceived : void
131             -> voterNumber : VOTERNUMBER
132             -> cpr : CPR
133             require Database.get(voterNumber, cpr) = NOTRECEIVED
134             ensure Database.get(voterNumber, cpr) = RECEIVED
135           end
136
137           BallotReceivedCPROnly : void
138             -> cpr : CPR
139             -> password : STRING
140             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = NOTRECEIVED
141             ensure Database.get(cpr, password) = RECEIVED
142           end
143
144           RevokeBallot : void
145             -> voterNumber : VOTERNUMBER
146             -> cpr : CPR
147             require Database.get(voterNumber, cpr) = RECEIVED
148             ensure Database.get(voterNumber, cpr) = NOTRECEIVED
149           end
150
151           RevokeBallotCPROnly : void
152             -> cpr : CPR
153             -> password : STRING
154             require password /= void and ValidMasterPassword(password)
    and Database.get(cpr, password) = RECEIVED
155             ensure Database.get(cpr, password) = NOTRECEIVED
156           end
157
158           AnnounceAddPeer : void
159             -> newPeerAddress : IPADDRESS
160             -> newPeerKey : ASYMMETRICKEY
161             require IsManager and newPeerAddres /= void
162           end
163
164           AnnounceRemovePeer : void
165             -> removePeerAddress : IPADDRESS
166             require IsManager and removePeerAddress /= void
167           end
168
169           PromoteNewManager : void
170             -> newManagerAddress : IPAddress
171             require IsManager and newManagerAddress /= void
172           end
173
174           AnnounceStartElection : void
175             require IsManager and not ElectionInProgress
176             ensure ElectionInProgress
177           end
178
179           AnnounceEndElection : void
180             require IsManager and ElectionInProgress
181             ensure not ElectionInProgress
182           end
183
```

```
184           invariant
185             Address /= void and Peers /= void
186       end
187     end
188   end
```