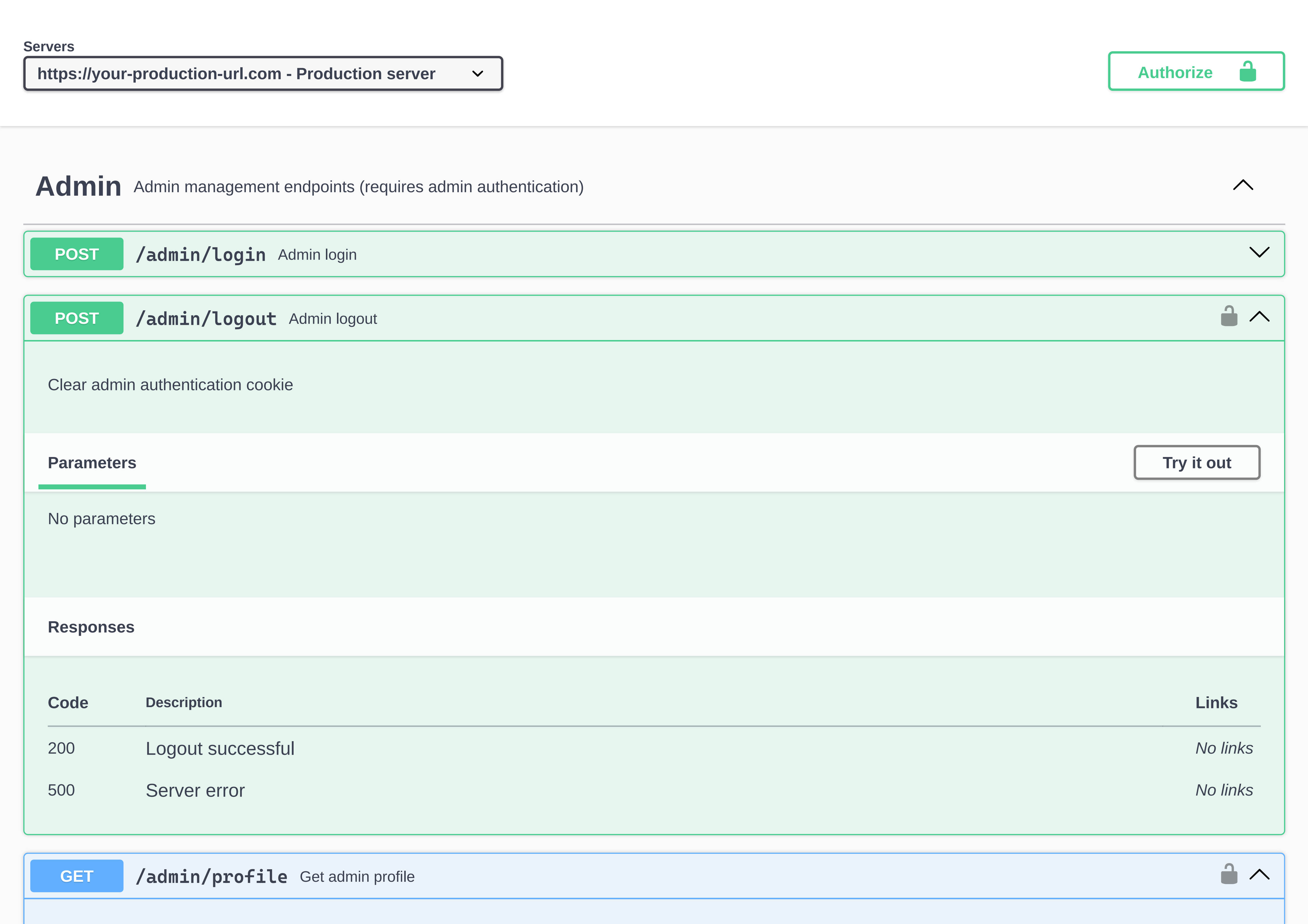
SJBU Voting System API 1.0.0 OAS 3.0

```
    # SJBU Voting System API Documentation

    A comprehensive voting system with admin management, statistics, auditing, and fraud detection.

    ## Authentication

    ### User Authentication
    - Uses JWT tokens stored in HTTP-only cookies
    - Token expires in 5 minutes for security

    ### Admin Authentication
    - Uses separate JWT tokens for admin functions
    - Token expires in 8 hours
    - Role-based access control (super_admin, admin, moderator)

    ## Rate Limiting

    - **General API**: 100 requests/15min per IP
    - **Voting**: 3 votes/5min per IP
    - **Authentication**: 5 attempts/15min per IP
    - **Admin**: 200 requests/5min per IP

    ## Caching

    - Position and candidate data cached for 10 minutes
    - Statistics cached for 30 minutes
    - Vote verification cached for 5 minutes

    ## Security Features

    - Comprehensive logging of all activities
    - Vote auditing and fraud detection
    - Rate limiting and DDoS protection
    - Input validation and sanitization
```

Contact SJBU Development Team



Servers: https://your-production-url.com - Production server

Admin

Admin management endpoints (requires admin authentication)

POST `/admin/login` Admin login

POST `/admin/logout` Admin logout

Clear admin authentication cookie

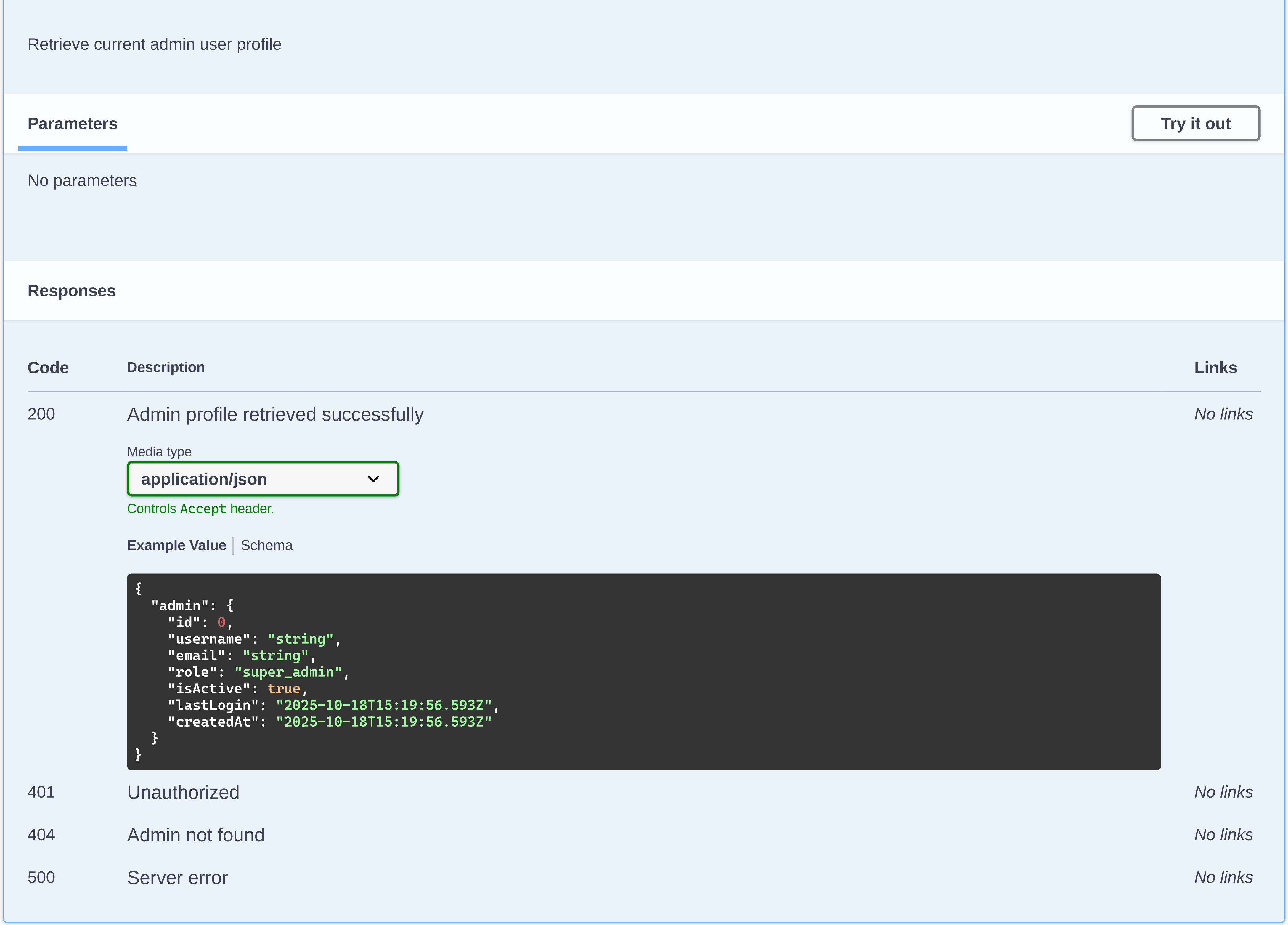
Parameters

No parameters

Responses

Code	Description	Links
200	Logout successful	*No links*
500	Server error	*No links*

GET `/admin/profile` Get admin profile



Retrieve current admin user profile

Parameters

No parameters

Responses

Code	Description	Links
200	Admin profile retrieved successfully	*No links*

Media type

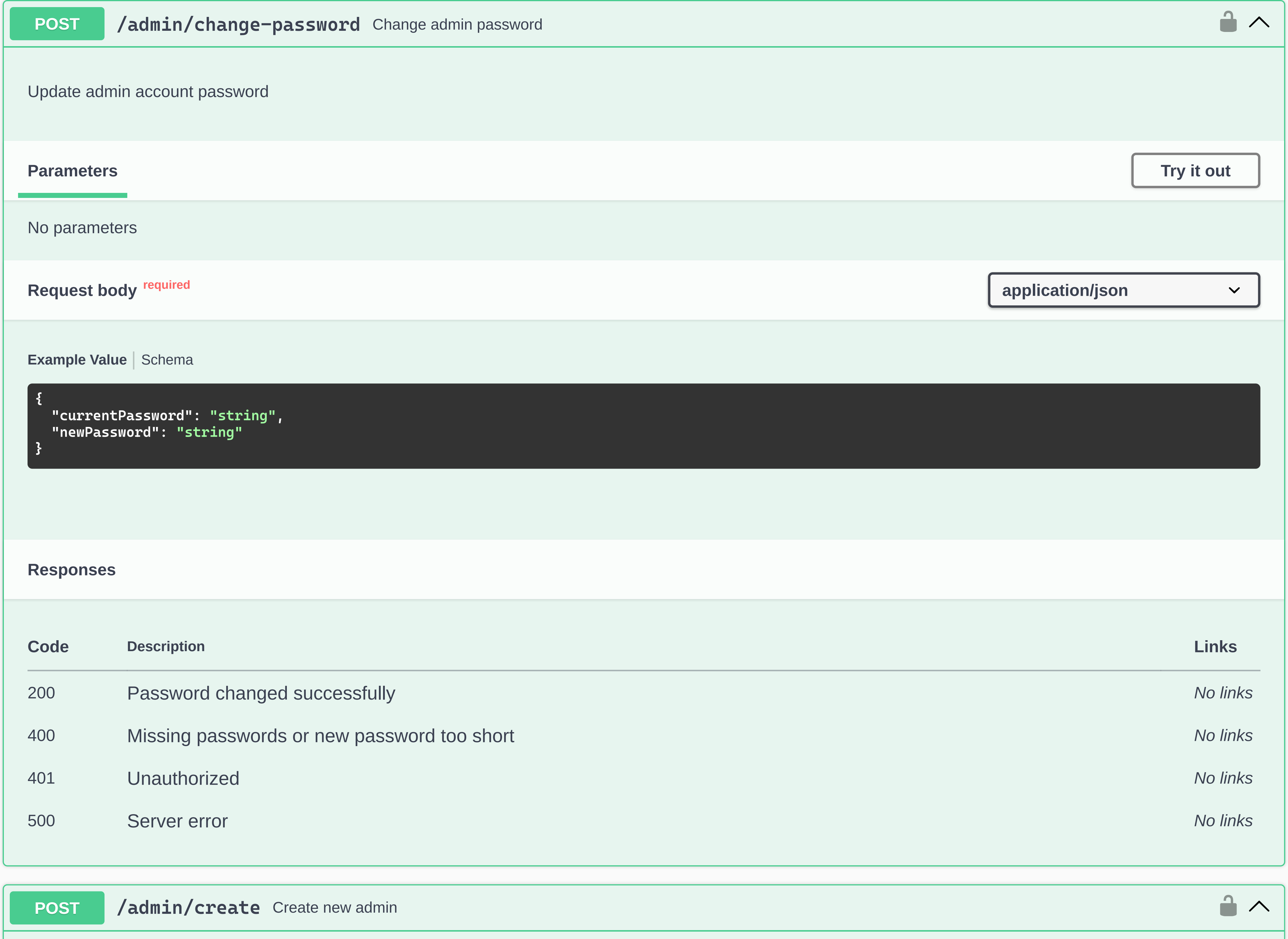
application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "admin": {
    "id": 0,
    "username": "string",
    "email": "string",
    "role": "super_admin",
    "isActive": true,
    "lastLogin": "2025-10-18T15:19:56.593Z",
    "createdAt": "2025-10-18T15:19:56.593Z"
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
404	Admin not found	*No links*
500	Server error	*No links*



POST **/admin/change-password** Change admin password

Update admin account password

Parameters

Try it out

No parameters

Request body required

application/json

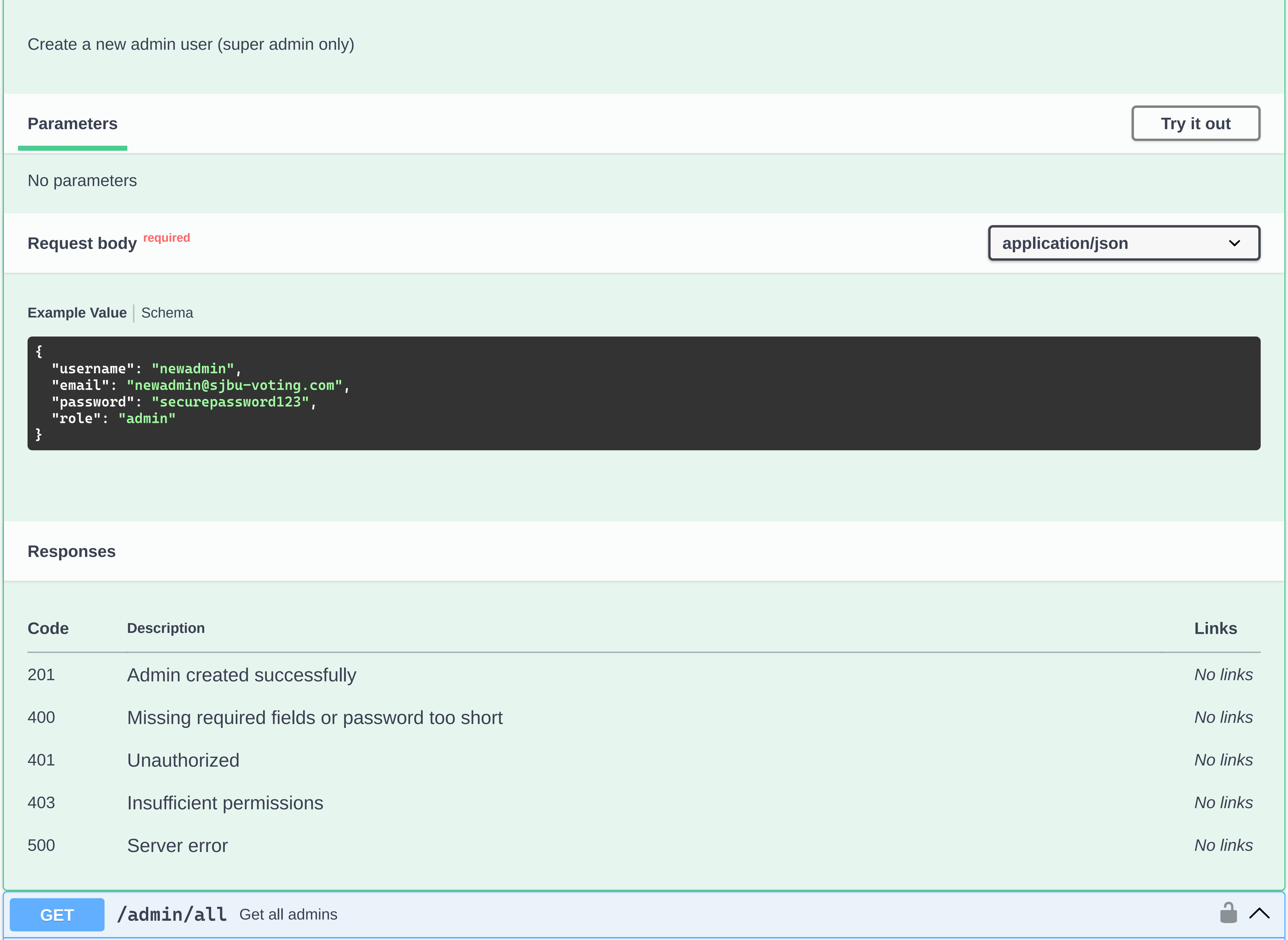
Example Value | Schema

```
{
  "currentPassword": "string",
  "newPassword": "string"
}
```

Responses

Code	Description	Links
200	Password changed successfully	*No links*
400	Missing passwords or new password too short	*No links*
401	Unauthorized	*No links*
500	Server error	*No links*

POST **/admin/create** Create new admin



Create a new admin user (super admin only)

No parameters

```
{
  "username": "newadmin",
  "email": "newadmin@sjbu-voting.com",
  "password": "securepassword123",
  "role": "admin"
}
```

Code	Description	Links
201	Admin created successfully	*No links*
400	Missing required fields or password too short	*No links*
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

`/admin/all` Get all admins

Retrieve list of all admin users (admin and super_admin only)

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	List of admins retrieved successfully	*No links*

Media type

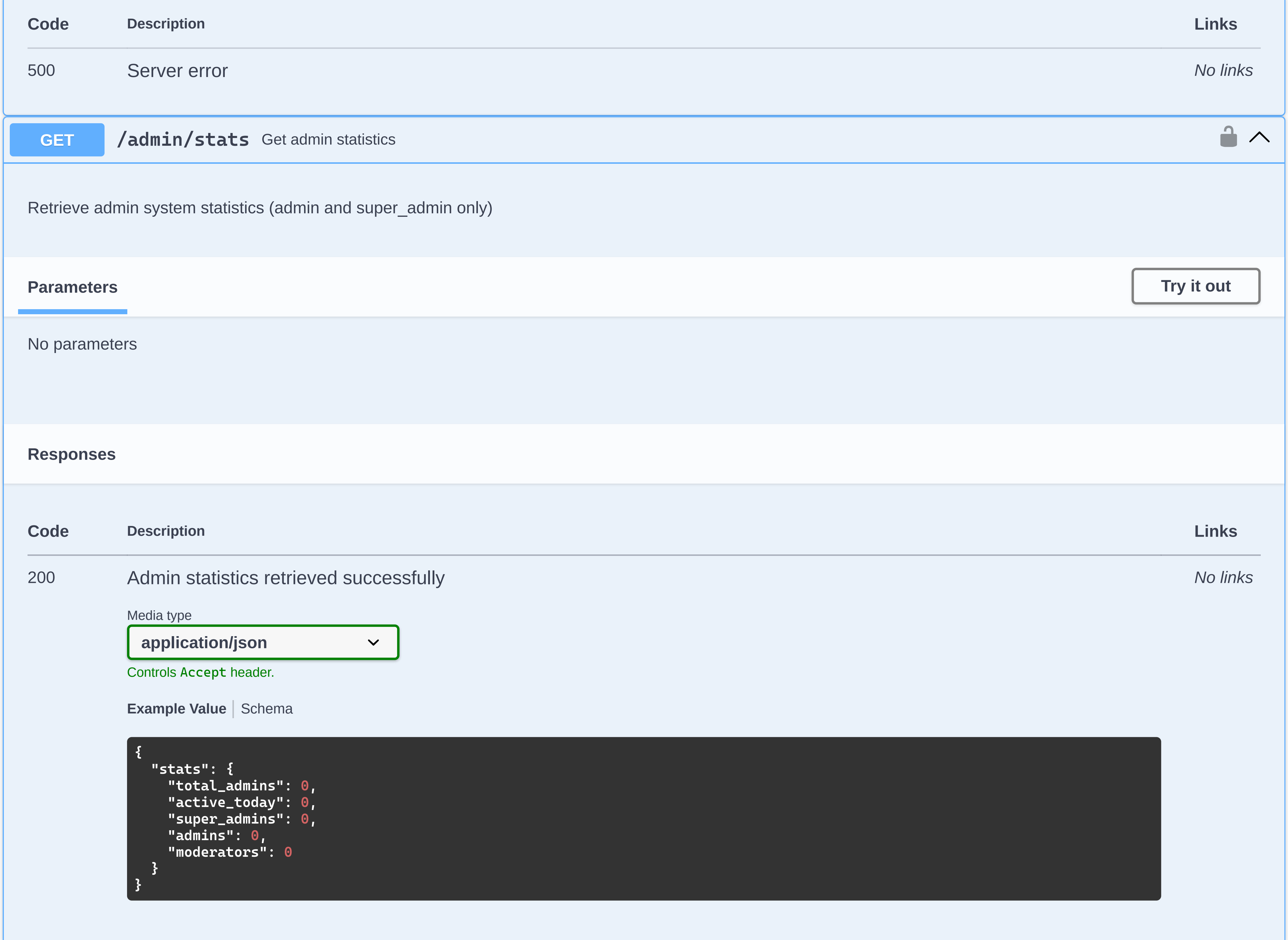
application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "admins": [
    {
      "id": 0,
      "username": "string",
      "email": "string",
      "role": "super_admin",
      "isActive": true,
      "lastLogin": "2025-10-18T15:19:56.599Z",
      "createdAt": "2025-10-18T15:19:56.599Z"
    }
  ]
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*



Code	Description	Links
500	Server error	*No links*

/admin/stats Get admin statistics

Retrieve admin system statistics (admin and super_admin only)

Parameters

No parameters

Responses

Code	Description	Links
200	Admin statistics retrieved successfully Media type application/json Controls `Accept` header. **Example Value** | Schema	*No links*

```
{
  "stats": {
    "total_admins": 0,
    "active_today": 0,
    "super_admins": 0,
    "admins": 0,
    "moderators": 0
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

PUT `/admin/{adminId}/role` Update admin role

Update an admin's role (super admin only)

Parameters

Try it out

Name	Description
adminId * required `integer` *(path)*	Admin user ID adminId

Request body required

application/json

Example Value | Schema

```
{
  "role": "super_admin"
}
```

Responses

Code	Description	Links
200	Role updated successfully	*No links*
400	Invalid role or missing admin ID	*No links*
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

DELETE `/admin/{adminId}/deactivate` Deactivate admin

Deactivate an admin account (super admin only)

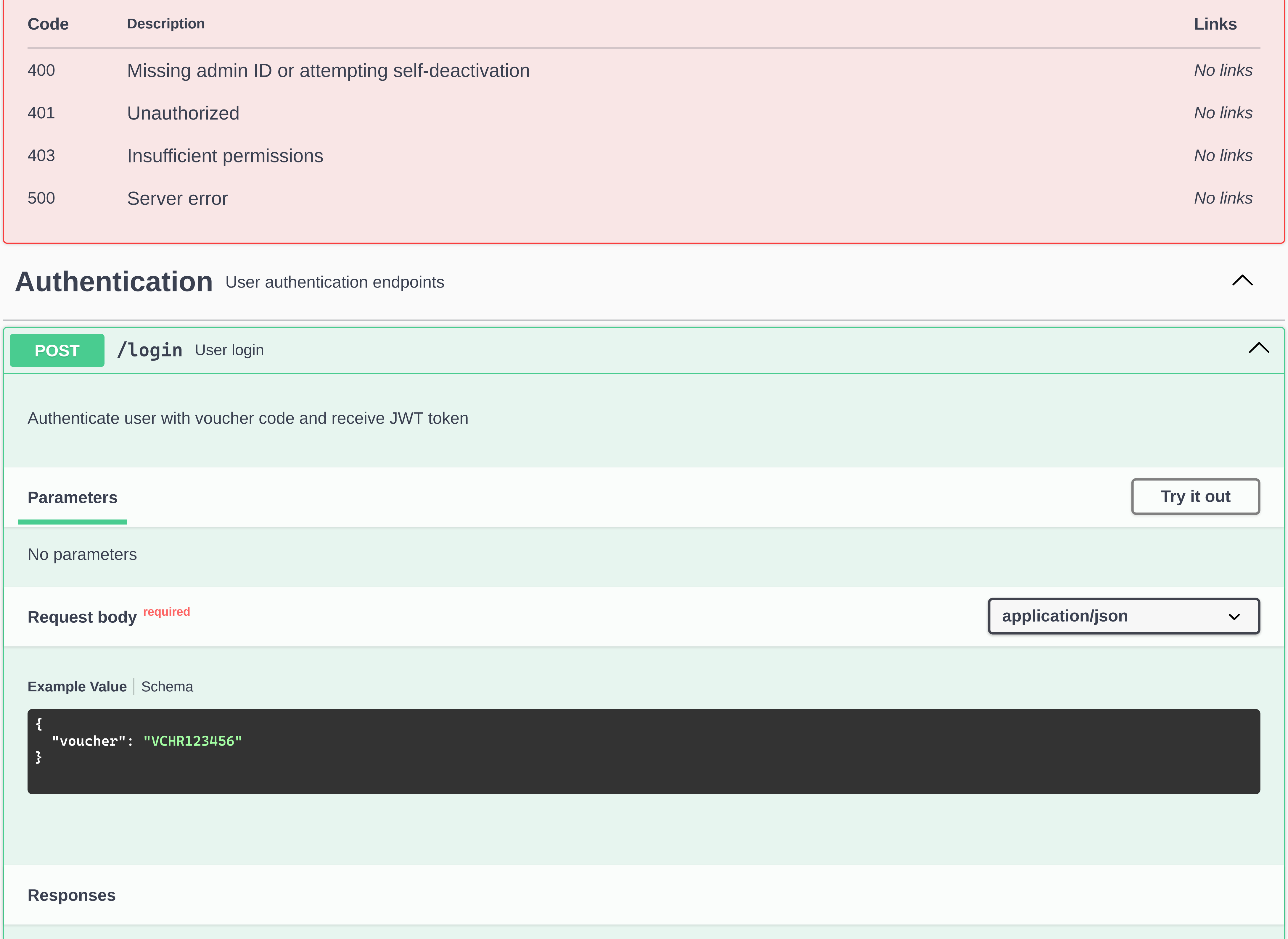
Parameters

Try it out

Name	Description
adminId * required `integer` *(path)*	Admin user ID to deactivate adminId

Responses

Code	Description	Links
200	Admin deactivated successfully	*No links*



Code	Description	Links
400	Missing admin ID or attempting self-deactivation	*No links*
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

Authentication

User authentication endpoints

POST /login

User login

Authenticate user with voucher code and receive JWT token

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "voucher": "VCHR123456"
}
```

Responses

Code	Description	Links
200	Login successful Media type application/json Controls `Accept` header. **Example Value** | Schema	*No links*
400	Missing voucher	*No links*
401	Invalid voucher	*No links*
429	Too many login attempts	*No links*

```
{
  "message": "Login successful",
  "user": {
    "id": 0,
    "voucher": "string"
  }
}
```

POST `/logout` User logout

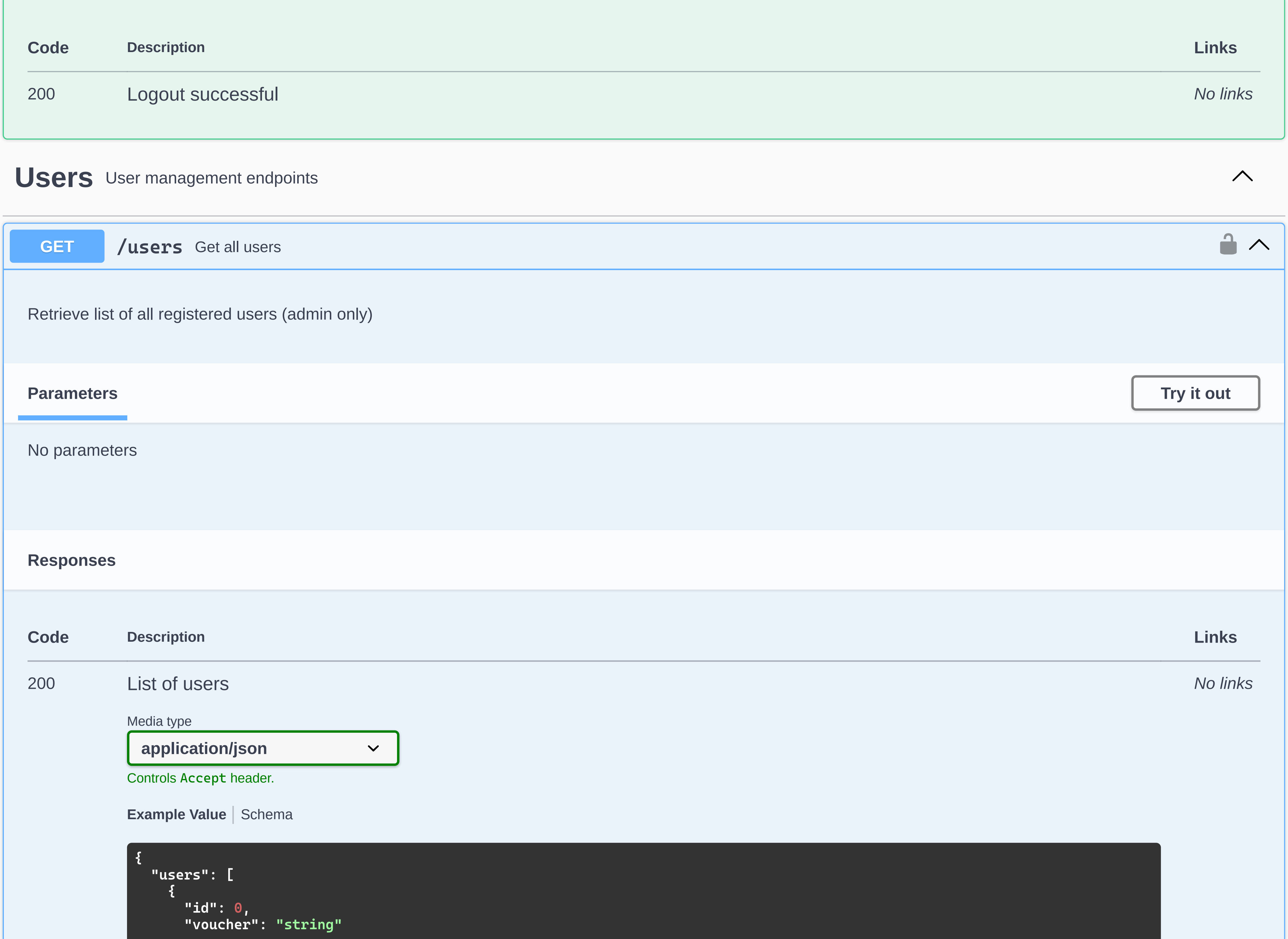
Clear authentication cookie

Parameters

Try it out

No parameters

Responses



Code	Description	Links
200	Logout successful	*No links*

Users User management endpoints

GET **/users** Get all users

Retrieve list of all registered users (admin only)

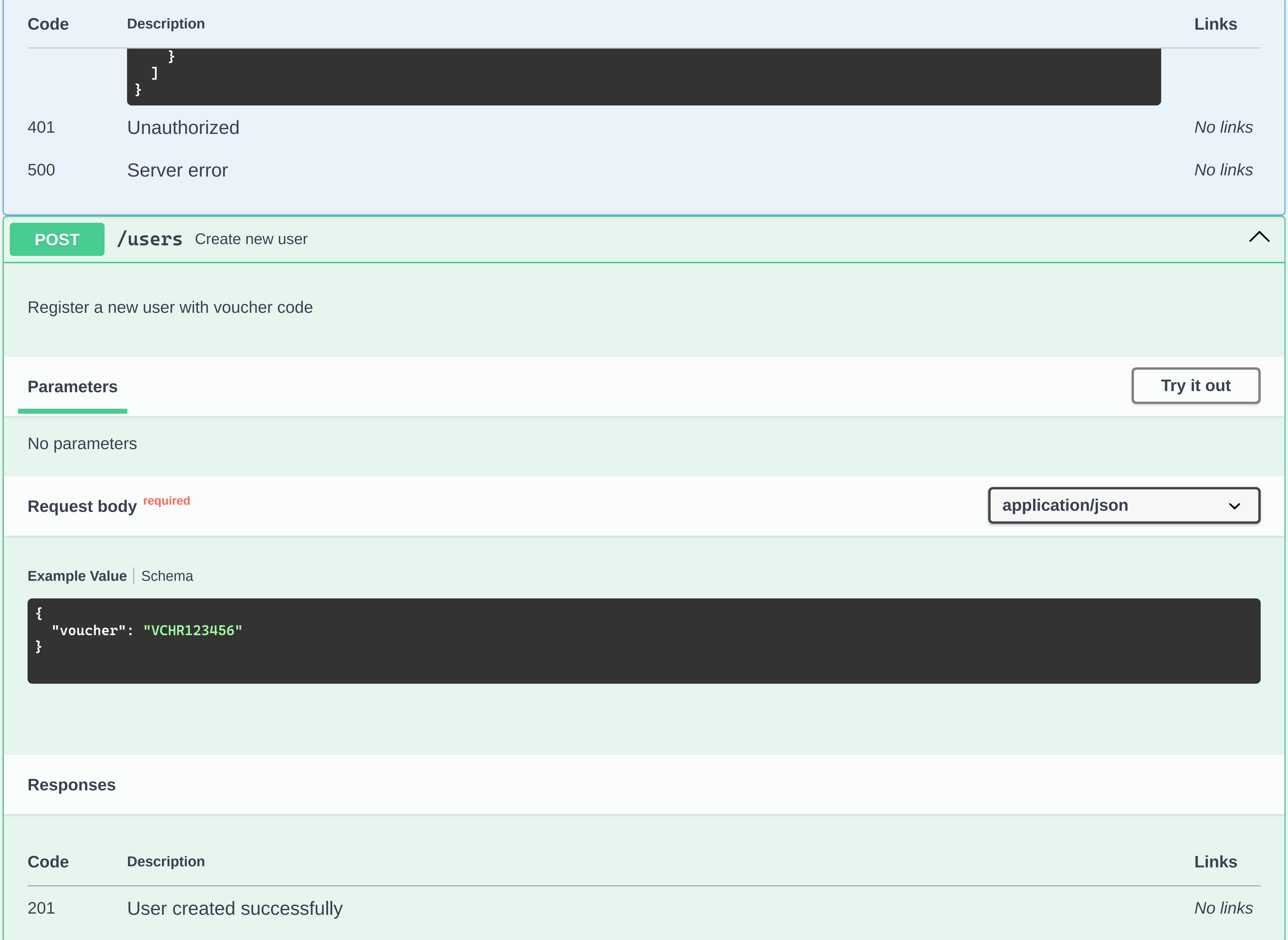
Parameters

No parameters

Responses

Code	Description	Links
200	List of users Media type application/json Controls `Accept` header. **Example Value** | Schema	*No links*

```
{
  "users": [
    {
      "id": 0,
      "voucher": "string"
```



Code	Description	Links

	`    }`
	`  ]`
	`}`	

401	Unauthorized	*No links*
500	Server error	*No links*

POST `/users` Create new user

Register a new user with voucher code

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "voucher": "VCHR123456"
}
```

Responses

Code	Description	Links
201	User created successfully	*No links*

Code	Description	Links
400	Voucher is required	*No links*
500	Server error	*No links*

Positions Position management endpoints

GET `/positions` Get all positions

Retrieve list of all voting positions

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	List of positions	*No links*

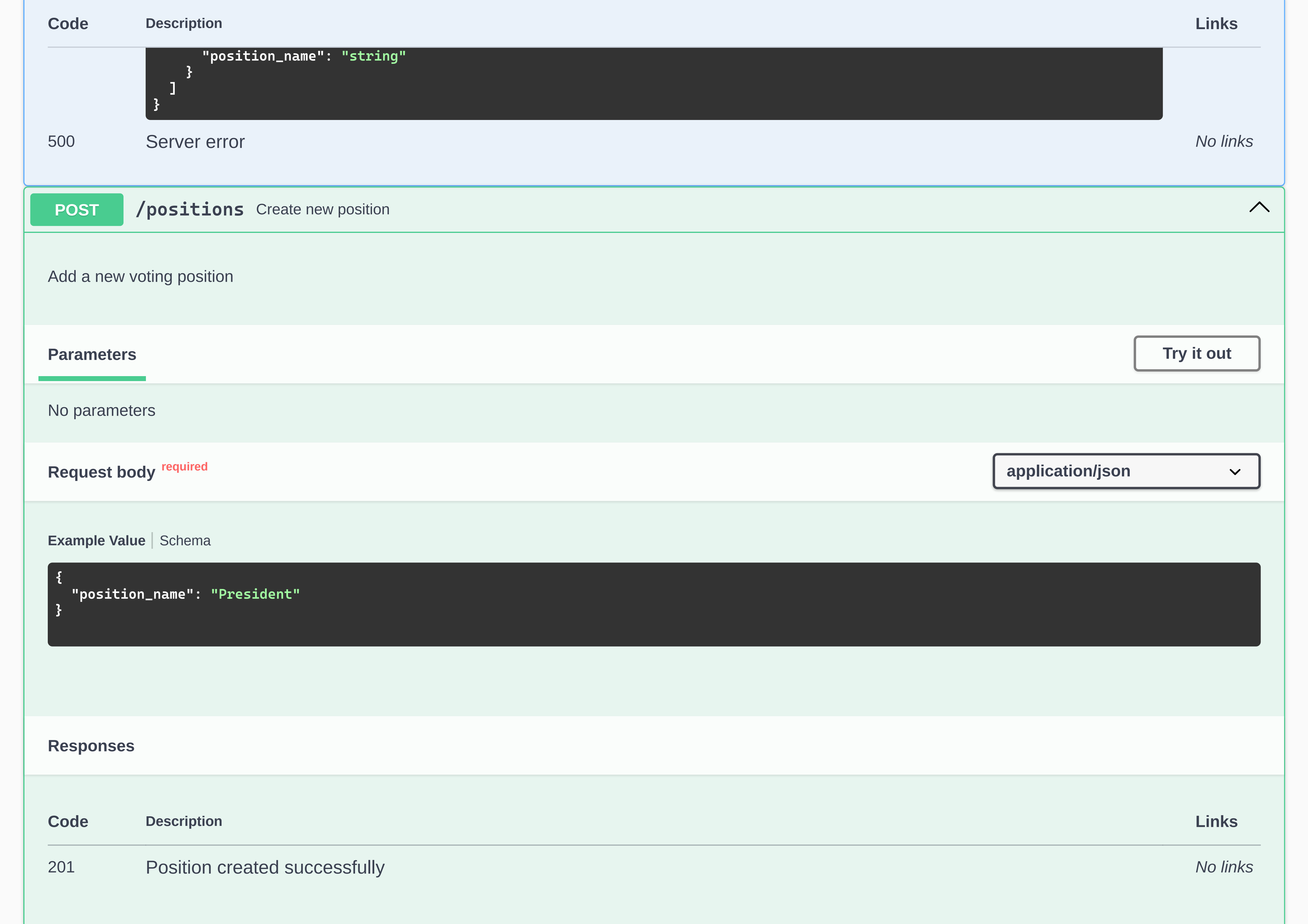
Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "data": [
    {
      "id": 0,
```



Code	Description	Links

	`"position_name": "string"`
	`}`
	`]`
	`}`	

500	Server error	*No links*

`/positions` Create new position

Add a new voting position

Parameters

No parameters

Request body required

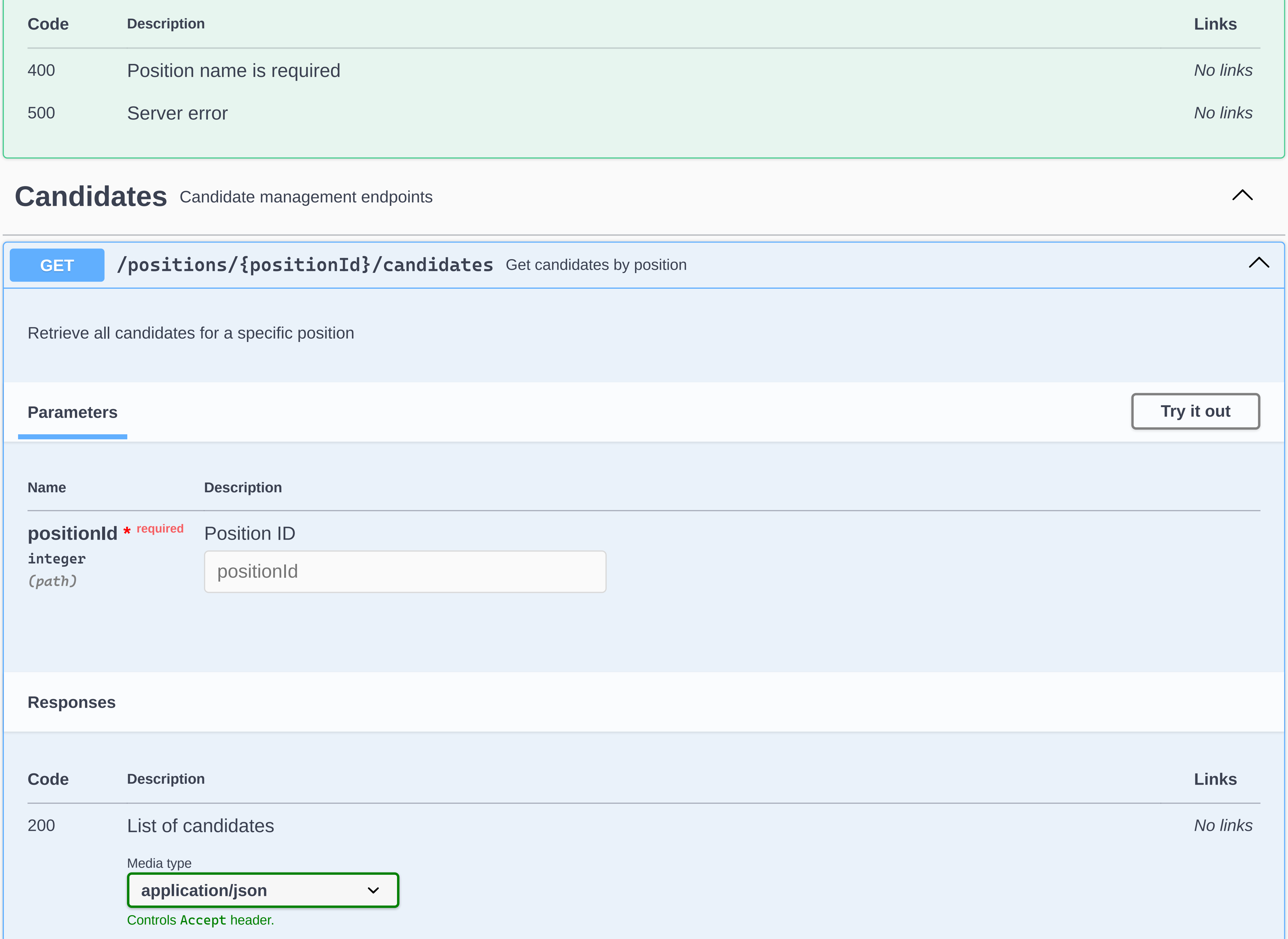
application/json

Example Value | Schema

```
{
  "position_name": "President"
}
```

Responses

Code	Description	Links
201	Position created successfully	*No links*



Code	Description	Links
400	Position name is required	*No links*
500	Server error	*No links*

Candidates

Candidate management endpoints

`/positions/{positionId}/candidates` Get candidates by position

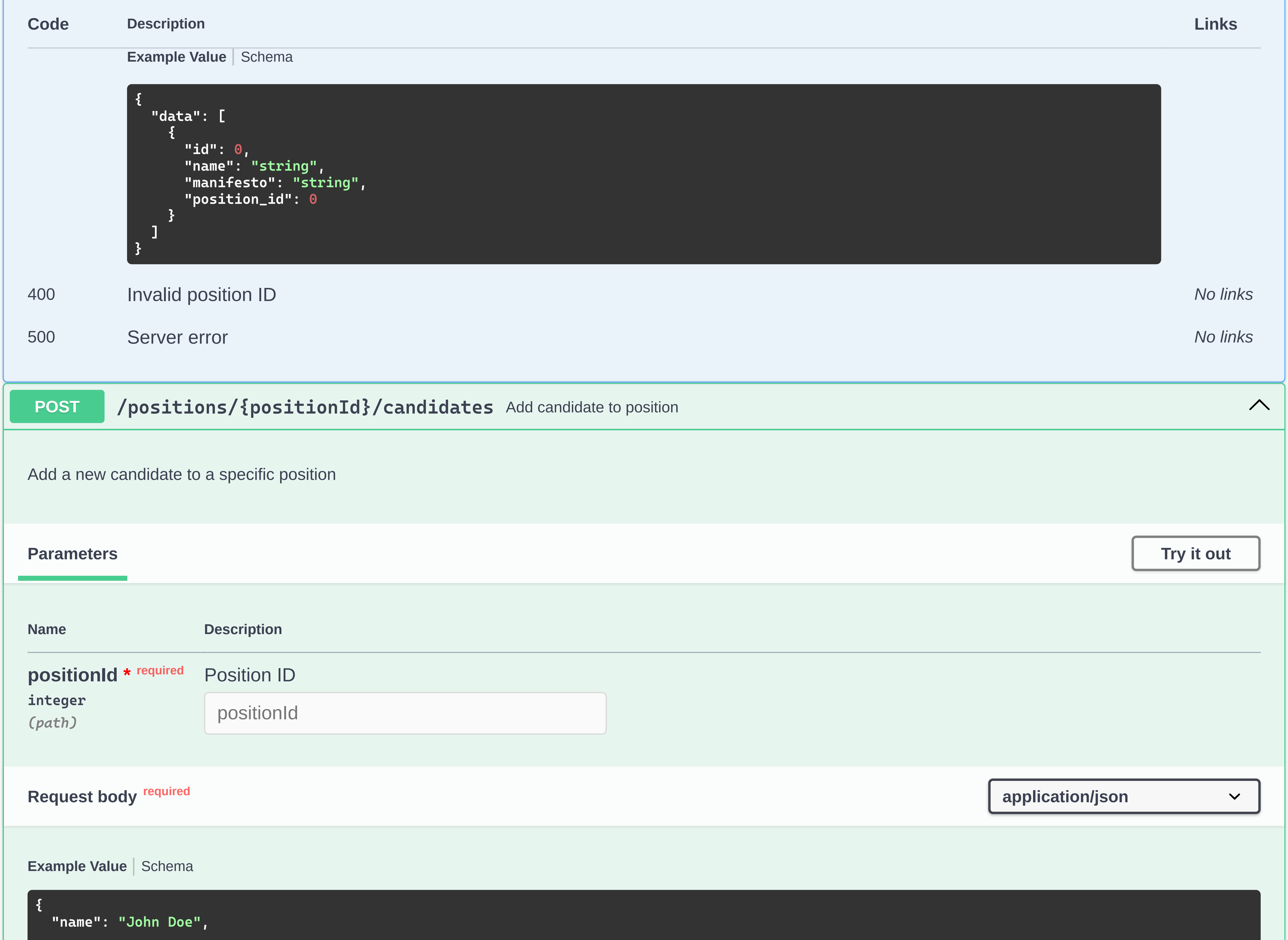
Retrieve all candidates for a specific position

Parameters

Name	Description
positionId * required `integer` *(path)*	Position ID positionId

Responses

Code	Description	Links
200	List of candidates Media type application/json Controls `Accept` header.	*No links*



Code	Description	Links
	Example Value | Schema	
400	Invalid position ID	*No links*
500	Server error	*No links*

```
{
  "data": [
    {
      "id": 0,
      "name": "string",
      "manifesto": "string",
      "position_id": 0
    }
  ]
}
```

POST `/positions/{positionId}/candidates` Add candidate to position

Add a new candidate to a specific position

Parameters

Name	Description
positionId * required `integer` *(path)*	Position ID positionId

Request body required

Example Value | Schema

```
{
  "name": "John Doe",
  "manifesto": "I will lead with integrity ... "
}
```

Responses

Code	Description	Links
201	Candidate created successfully	*No links*
400	Invalid position ID or missing name	*No links*
500	Server error	*No links*

Voting Voting endpoints

POST `/vote` Cast a vote

Submit a vote for a candidate in a position (rate limited)

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "voucher": "VCHR123456",
  "candidateId": 1,
  "positionId": 1
}
```

Responses

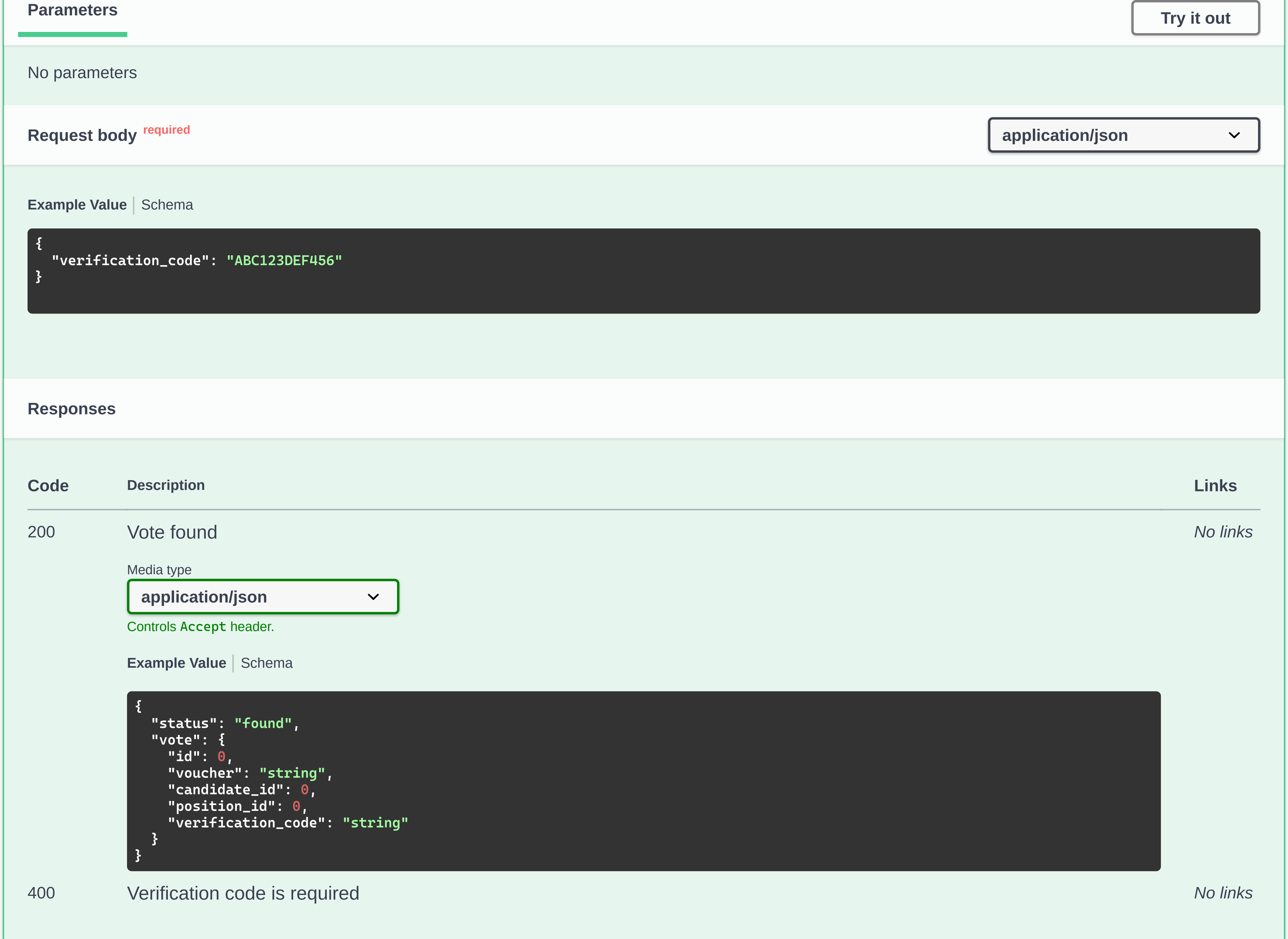
Code	Description	Links
201	Vote cast successfully Media type application/json Controls `Accept` header. **Example Value** | Schema	*No links*
400	Missing required fields	*No links*
401	Unauthorized	*No links*
429	Too many voting attempts	*No links*
500	Server error	*No links*

Example value for 201:

```
{
  "data": {
    "id": 0,
    "voucher": "string",
    "candidate_id": 0,
    "position_id": 0,
    "verification_code": "string"
  }
}
```

POST `/verify` Verify a vote

Verify a vote using verification code



Parameters

No parameters

Request body required

Example Value | Schema

```
{
  "verification_code": "ABC123DEF456"
}
```

Responses

Code | Description | Links

200 Vote found *No links*

Media type

Controls `Accept` header.

Example Value | Schema

```
{
  "status": "found",
  "vote": {
    "id": 0,
    "voucher": "string",
    "candidate_id": 0,
    "position_id": 0,
    "verification_code": "string"
  }
}
```

400 Verification code is required *No links*

Code	Description	Links
404	Vote not found	*No links*
500	Server error	*No links*

Audit Vote auditing and fraud detection endpoints (admin only)

GET `/audit/vote/{voteId}` Audit single vote

Perform detailed audit on a specific vote to check for irregularities

Parameters

Try it out

Name	Description
voteId * required `integer` *(path)*	Vote ID to audit voteId

Responses

Code	Description	Links
200	Vote audit completed successfully Media type application/json Controls `Accept` header.	*No links*

Code Description Links

Example Value | Schema

```
{
  "success": true,
  "data": {
    "voteId": 0,
    "voucher": "string",
    "verificationCode": "string",
    "candidateId": 0,
    "positionId": 0,
    "timestamp": "2025-10-18T15:19:56.625Z",
    "isValid": true,
    "issues": [
      "string"
    ],
    "riskScore": 0
  }
}
```

400	Missing vote ID	*No links*
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

POST `/audit/full` Full system audit

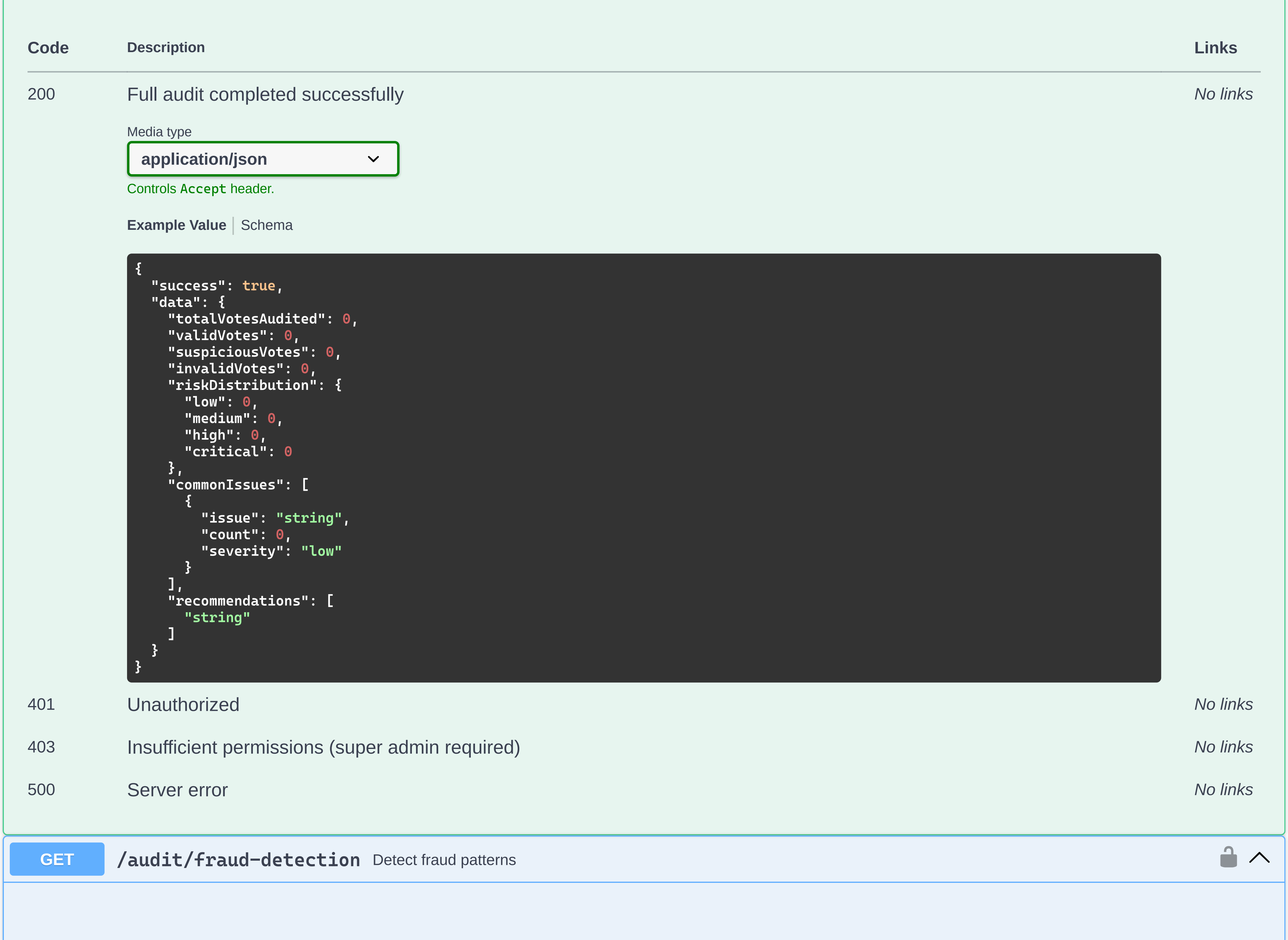
Perform comprehensive audit of all votes in the system (super admin only)

Parameters

Try it out

No parameters

Responses



Code	Description	Links
200	Full audit completed successfully	*No links*

Media type

application/json ⌄

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "totalVotesAudited": 0,
    "validVotes": 0,
    "suspiciousVotes": 0,
    "invalidVotes": 0,
    "riskDistribution": {
      "low": 0,
      "medium": 0,
      "high": 0,
      "critical": 0
    },
    "commonIssues": [
      {
        "issue": "string",
        "count": 0,
        "severity": "low"
      }
    ],
    "recommendations": [
      "string"
    ]
  }
}
```

401	Unauthorized	*No links*
403	Insufficient permissions (super admin required)	*No links*
500	Server error	*No links*

GET `/audit/fraud-detection` Detect fraud patterns

Analyze voting patterns to detect potential fraud or irregularities

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Fraud detection analysis completed	*No links*

Media type

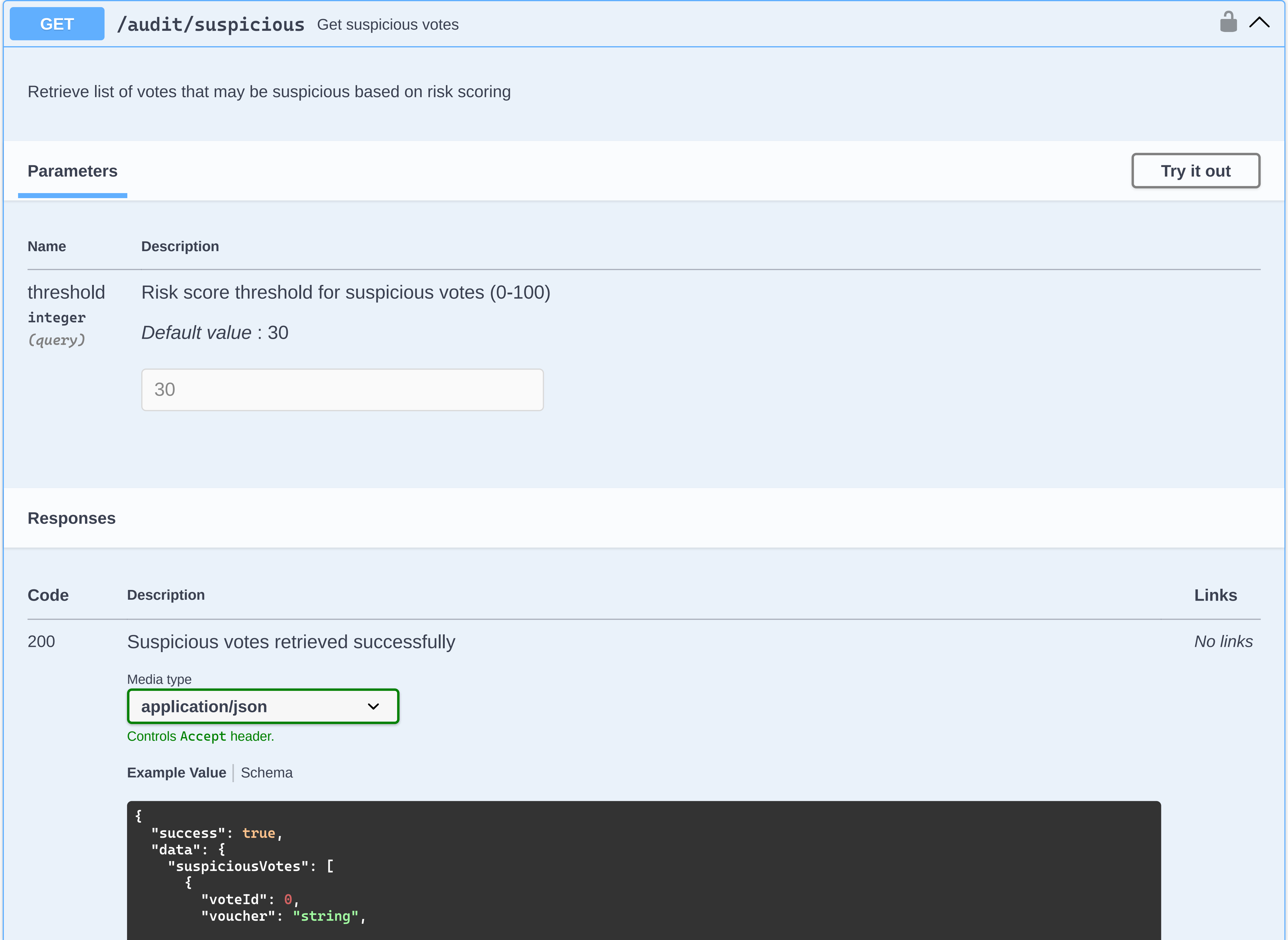
application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "isSuspicious": true,
    "riskLevel": "low",
    "riskFactors": [
      "string"
    ],
    "confidence": 0,
    "details": {}
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*



`/audit/suspicious` Get suspicious votes

Retrieve list of votes that may be suspicious based on risk scoring

Parameters

Name	Description
threshold `integer` *(query)*	Risk score threshold for suspicious votes (0-100) *Default value* : 30 30

Responses

Code	Description	Links
200	Suspicious votes retrieved successfully	*No links*

Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "suspiciousVotes": [
      {
        "voteId": 0,
        "voucher": "string",
```

Code	Description	Links

```
        "candidateName": "string",
        "positionName": "string",
        "timestamp": "2025-10-18T15:19:56.638Z",
        "issues": [
          "string"
        ],
        "riskScore": 0
      }
    ],
    "totalCount": 0,
    "threshold": 0
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

GET `/audit/logs` Get audit logs

Retrieve system audit logs and activity history

Parameters

Try it out

Name	Description
limit `integer` *(query)*	Number of log entries to return *Default value* : 50 50
offset `integer` *(query)*	Number of log entries to skip *Default value* : 0

Name	Description
	0

Responses

Code	Description	Links
200	Audit logs retrieved successfully	*No links*

Media type

application/json

Controls `Accept` header.

Example Value | Schema

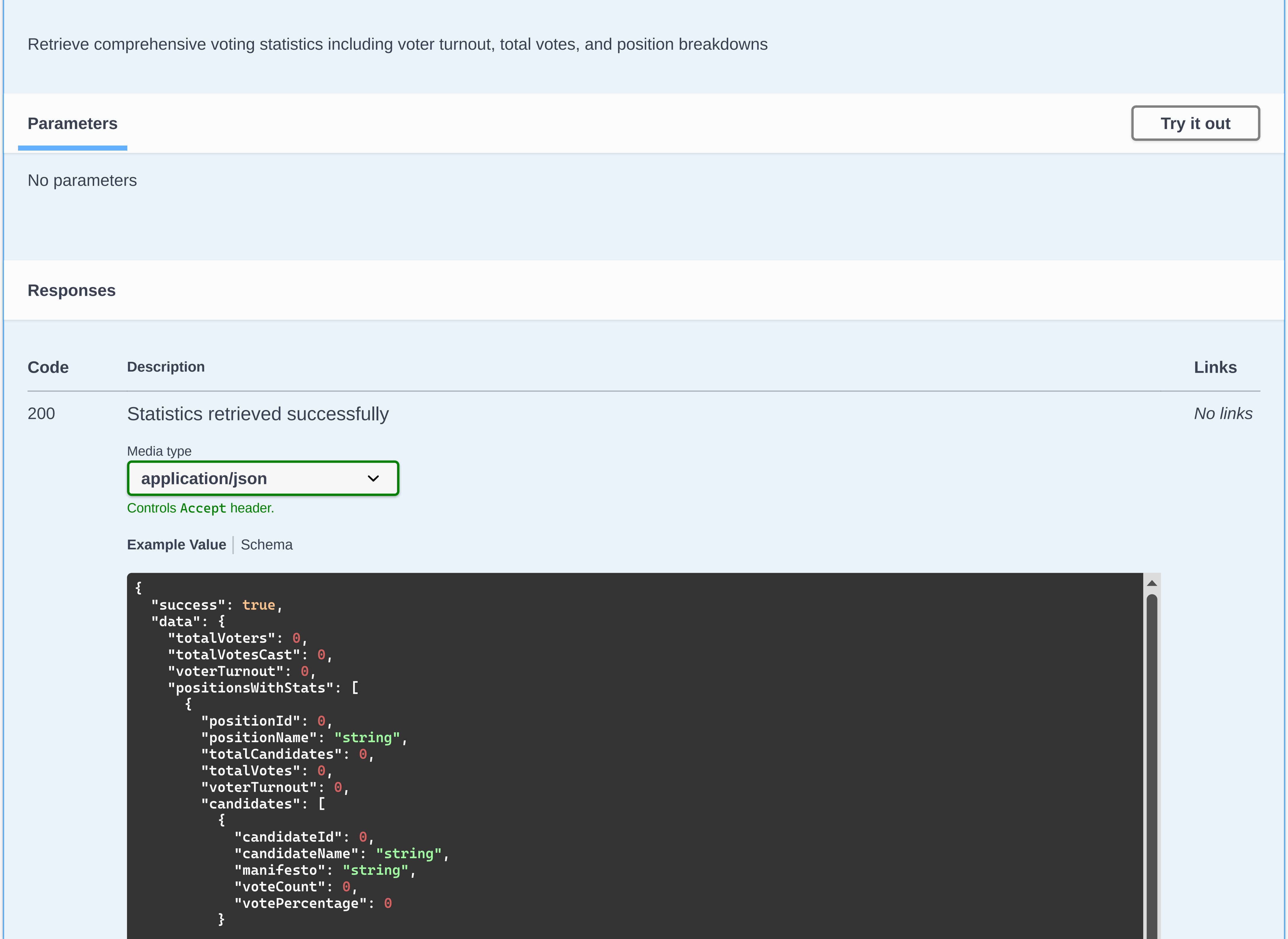
```
{
  "success": true,
  "data": {
    "totalAuditsPerformed": 0,
    "lastAuditDate": "2025-10-18T15:19:56.643Z"
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

Statistics

Voting statistics and analytics endpoints (admin only)

GET `/stats/overall` Get overall voting statistics



Retrieve comprehensive voting statistics including voter turnout, total votes, and position breakdowns

Parameters

No parameters

Responses

Code	Description	Links
200	Statistics retrieved successfully	*No links*

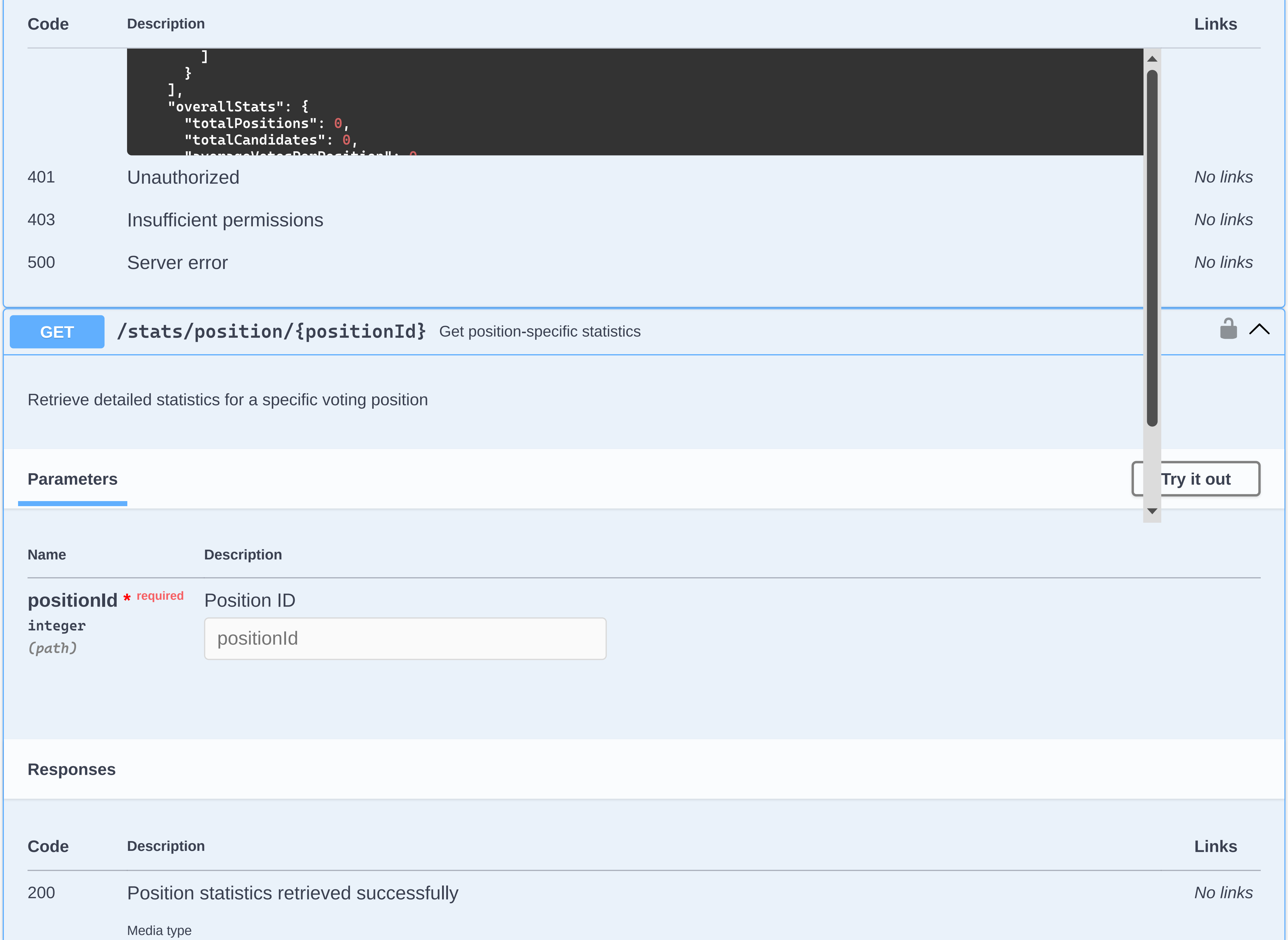
Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "totalVoters": 0,
    "totalVotesCast": 0,
    "voterTurnout": 0,
    "positionsWithStats": [
      {
        "positionId": 0,
        "positionName": "string",
        "totalCandidates": 0,
        "totalVotes": 0,
        "voterTurnout": 0,
        "candidates": [
          {
            "candidateId": 0,
            "candidateName": "string",
            "manifesto": "string",
            "voteCount": 0,
            "votePercentage": 0
          }
```



Code	Description	Links

```
        ]
      }
    ],
    "overallStats": {
      "totalPositions": 0,
      "totalCandidates": 0,
      "averageVotesPerPosition": 0
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

GET **/stats/position/{positionId}** Get position-specific statistics

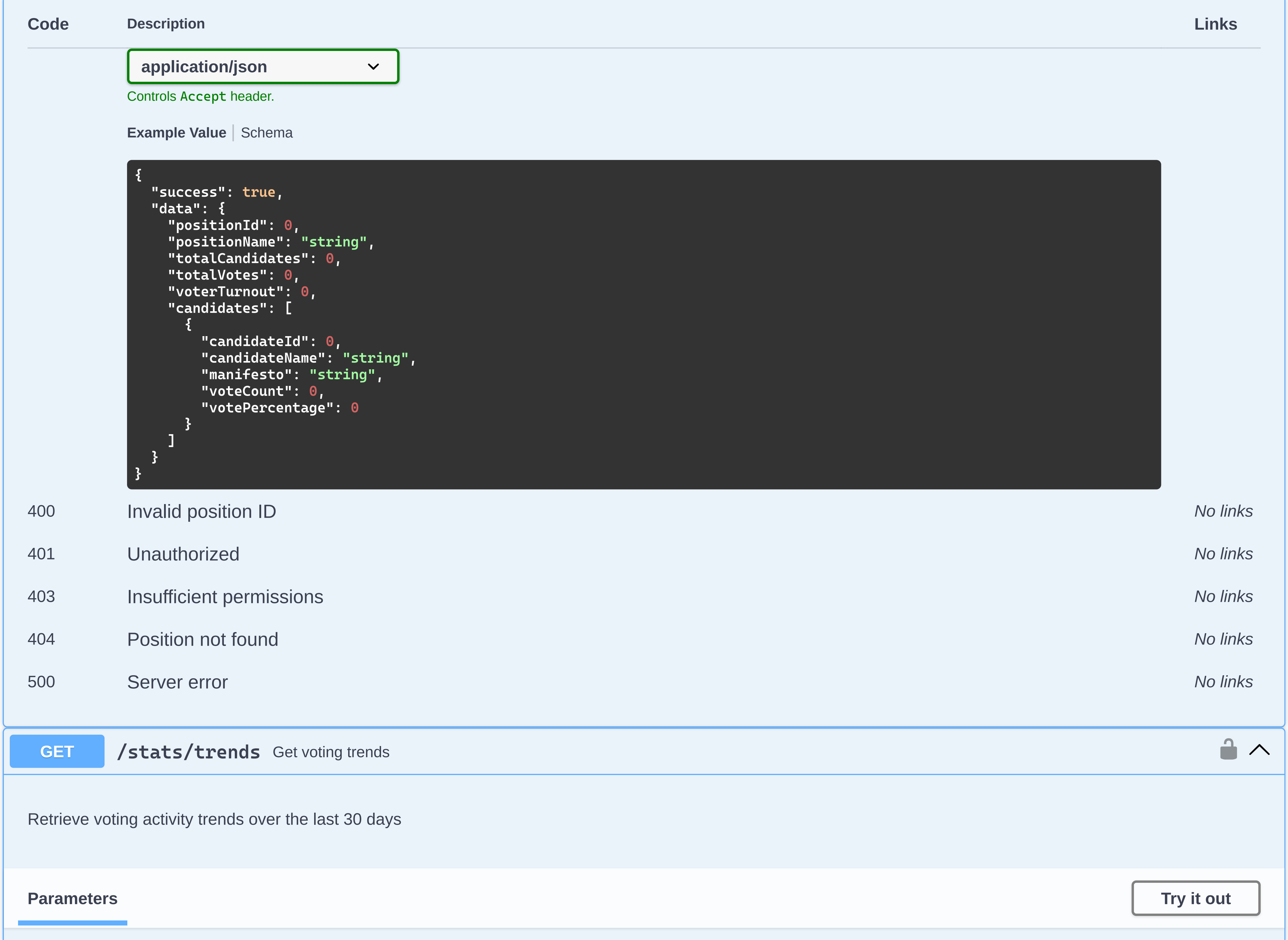
Retrieve detailed statistics for a specific voting position

Parameters

Name	Description
positionId * required `integer` *(path)*	Position ID positionId

Responses

Code	Description	Links
200	Position statistics retrieved successfully Media type	*No links*



Controls `Accept` header.

```
{
  "success": true,
  "data": {
    "positionId": 0,
    "positionName": "string",
    "totalCandidates": 0,
    "totalVotes": 0,
    "voterTurnout": 0,
    "candidates": [
      {
        "candidateId": 0,
        "candidateName": "string",
        "manifesto": "string",
        "voteCount": 0,
        "votePercentage": 0
      }
    ]
  }
}
```

Code	Description	Links
400	Invalid position ID	*No links*
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
404	Position not found	*No links*
500	Server error	*No links*

GET `/stats/trends` Get voting trends

Retrieve voting activity trends over the last 30 days

No parameters

Responses

Code	Description	Links
200	Voting trends retrieved successfully	*No links*

Media type

application/json ⌄

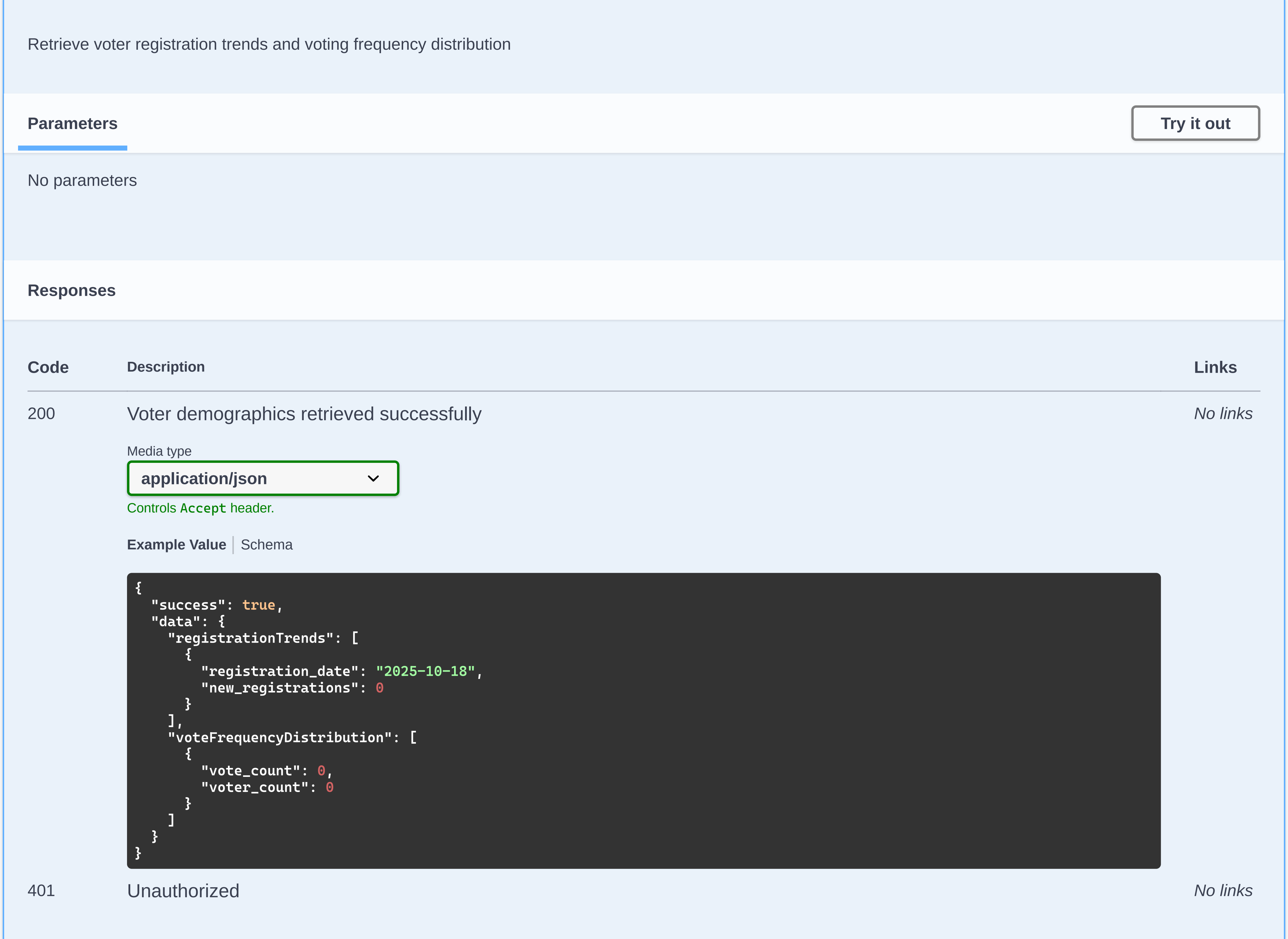
Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "dailyTrends": [
      {
        "vote_date": "2025-10-18",
        "votes_count": 0,
        "unique_voters": 0
      }
    ],
    "hourlyPattern": [
      {
        "hour": 0,
        "votes_count": 0
      }
    ]
  }
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

GET `/stats/demographics` Get voter demographics



Retrieve voter registration trends and voting frequency distribution

Parameters

No parameters

Responses

Code	Description	Links
200	Voter demographics retrieved successfully	*No links*
401	Unauthorized	*No links*

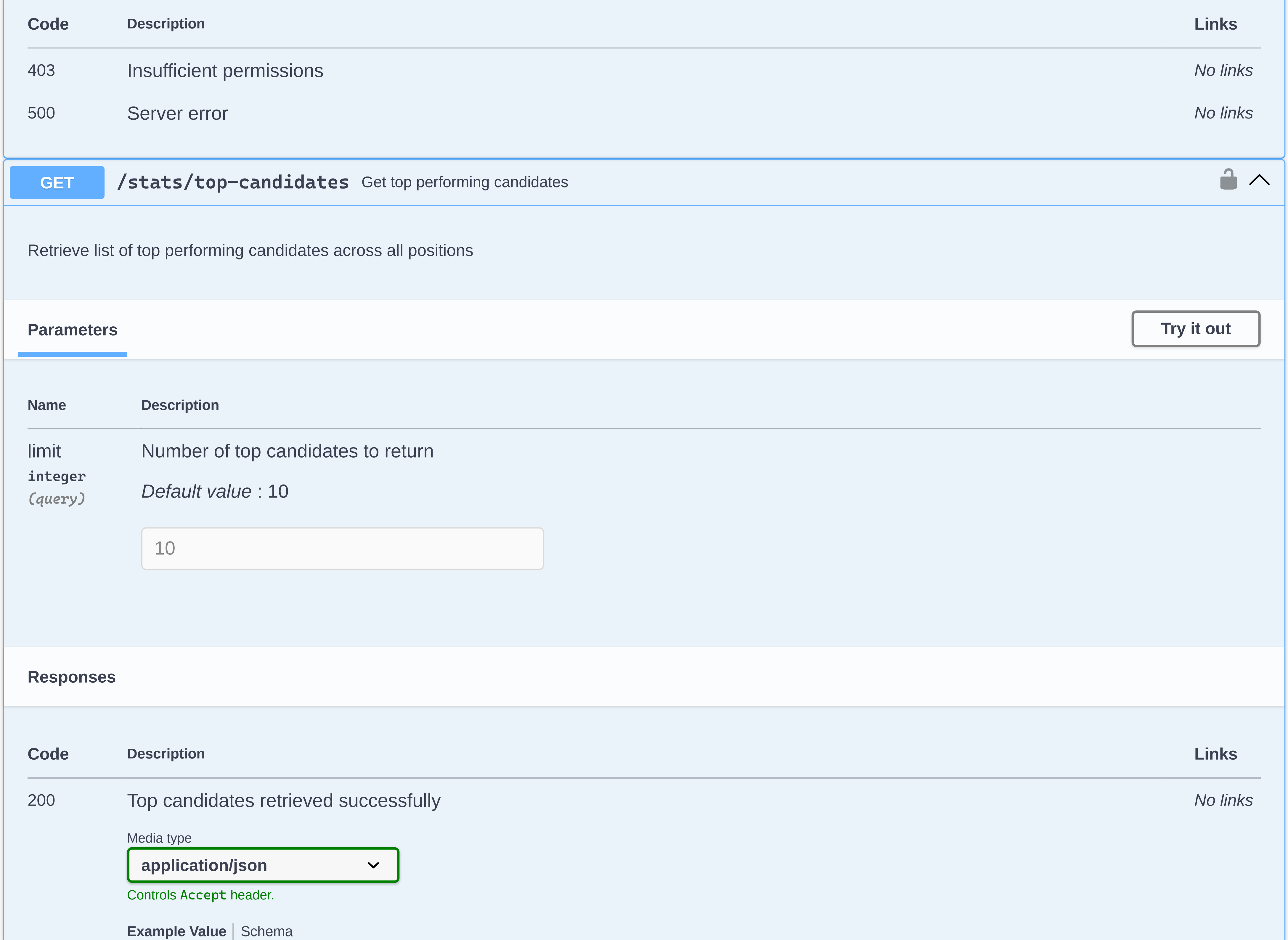
Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "registrationTrends": [
      {
        "registration_date": "2025-10-18",
        "new_registrations": 0
      }
    ],
    "voteFrequencyDistribution": [
      {
        "vote_count": 0,
        "voter_count": 0
      }
    ]
  }
}
```



Code	Description	Links
403	Insufficient permissions	*No links*
500	Server error	*No links*

GET `/stats/top-candidates` Get top performing candidates

Retrieve list of top performing candidates across all positions

Parameters

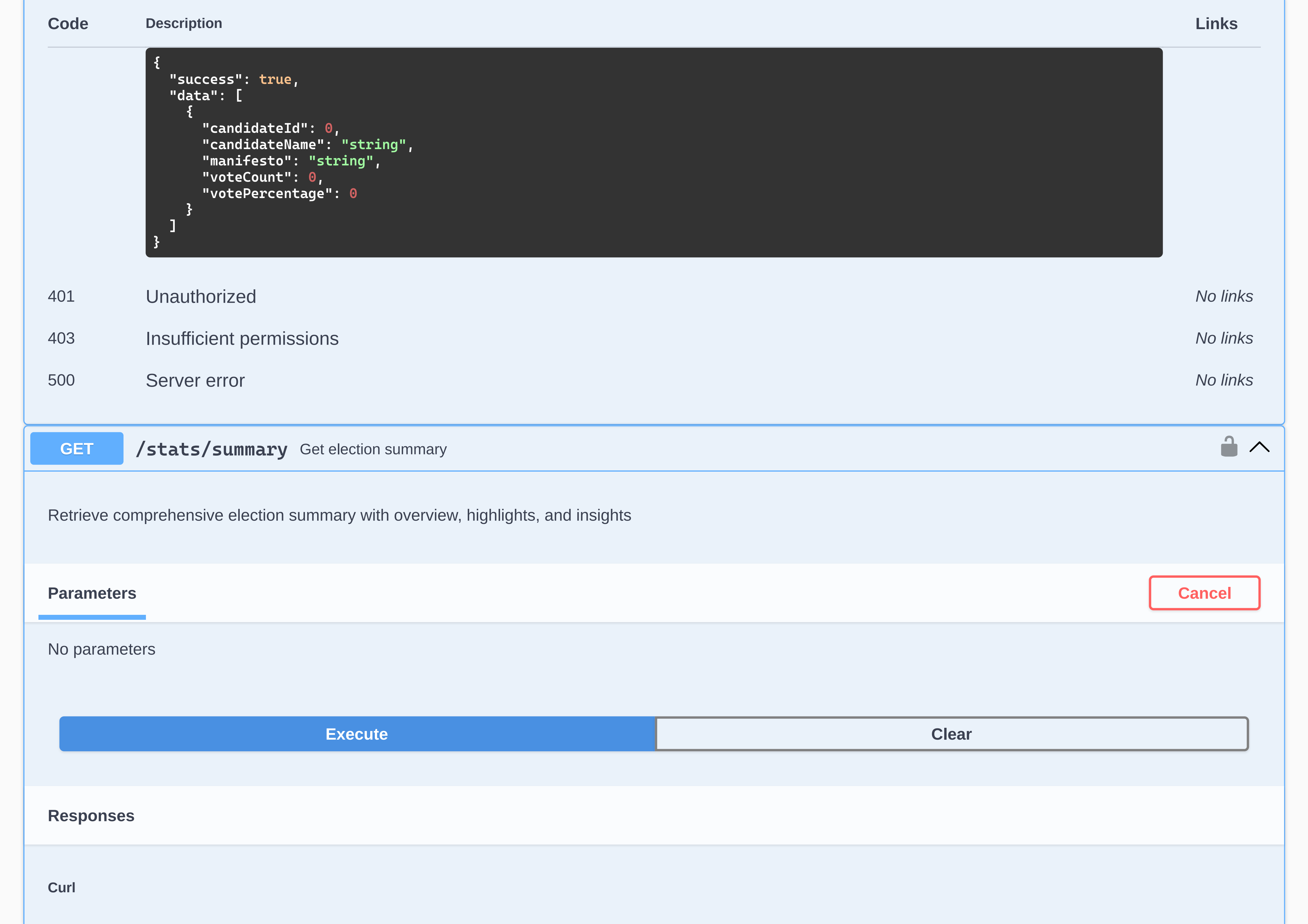
Name	Description

limit
`integer`
`(query)`
	Number of top candidates to return
	Default value : 10
	10

Responses

Code	Description	Links
200	Top candidates retrieved successfully Media type **application/json** Controls `Accept` header.	*No links*

Example Value | Schema



```json
{
  "success": true,
  "data": [
    {
      "candidateId": 0,
      "candidateName": "string",
      "manifesto": "string",
      "voteCount": 0,
      "votePercentage": 0
    }
  ]
}
```

Code	Description	Links
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

/stats/summary Get election summary

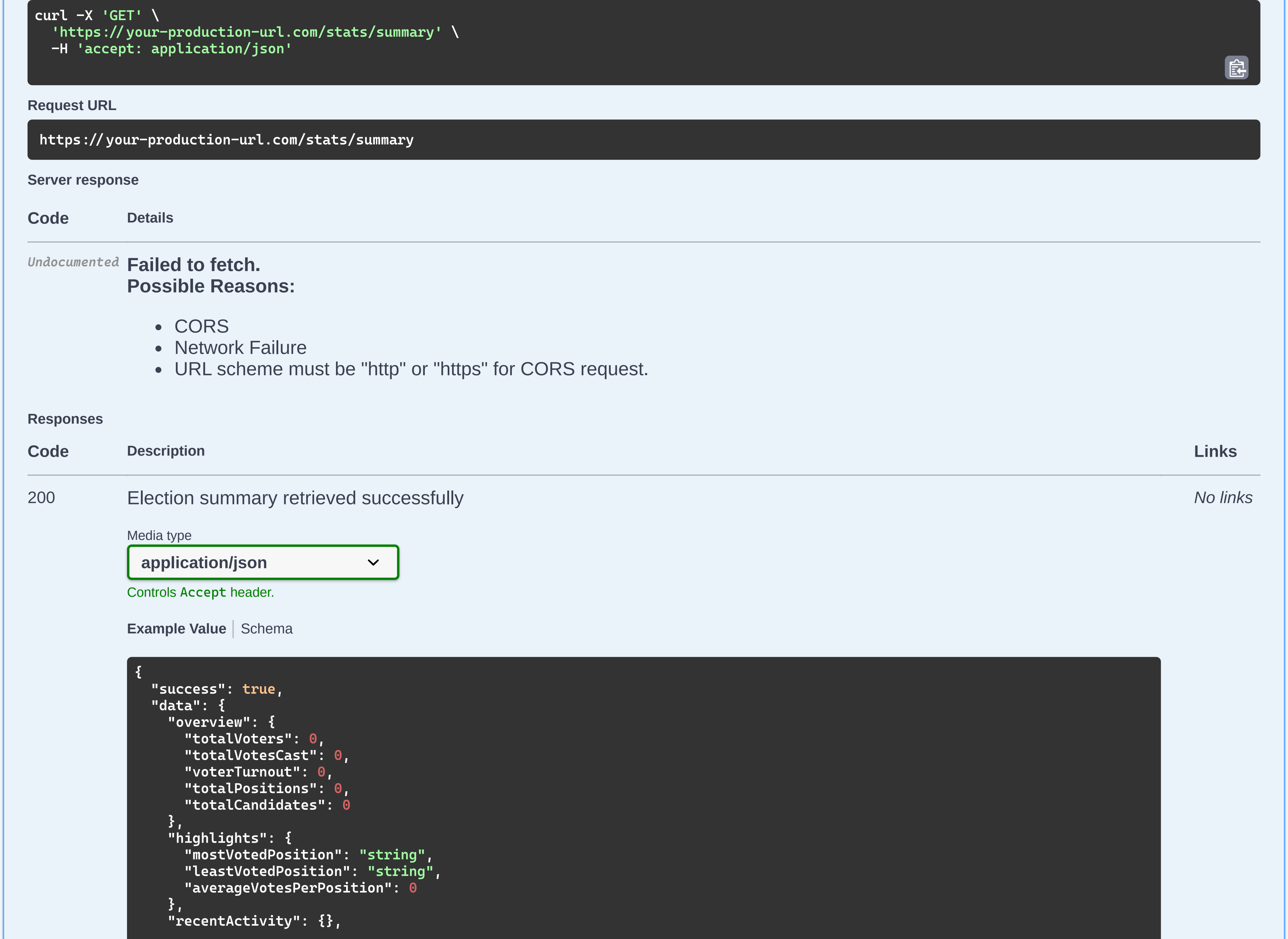
Retrieve comprehensive election summary with overview, highlights, and insights

Parameters

No parameters

Responses

Curl



```
curl -X 'GET' \
  'https://your-production-url.com/stats/summary' \
  -H 'accept: application/json'
```

Request URL

```
https://your-production-url.com/stats/summary
```

Server response

Code	Details
Undocumented	**Failed to fetch.** **Possible Reasons:** • CORS • Network Failure • URL scheme must be "http" or "https" for CORS request.

Responses

Code	Description	Links
200	Election summary retrieved successfully	*No links*

Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "success": true,
  "data": {
    "overview": {
      "totalVoters": 0,
      "totalVotesCast": 0,
      "voterTurnout": 0,
      "totalPositions": 0,
      "totalCandidates": 0
    },
    "highlights": {
      "mostVotedPosition": "string",
      "leastVotedPosition": "string",
      "averageVotesPerPosition": 0
    },
    "recentActivity": {},
```

Code	Description	Links
	```
	    "voterInsights": {}
	  }
	}
	```
401	Unauthorized	*No links*
403	Insufficient permissions	*No links*
500	Server error	*No links*

default

GET `/api-docs` API Documentation

Interactive API documentation and testing interface

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	API documentation interface	*No links*

Schemas

```
Error {
    error                    string
                             Error message

    message                  string
                             Detailed error message

}
```

```
User {
    id                       integer
                             User ID

    voucher                  string
                             User voucher code

}
```

```
Position {
    id                       integer
                             Position ID

    position_name            string
                             Name of the position

}
```

```
Candidate {
    id                    integer
                          Candidate ID

    name                  string
                          Candidate name

    manifesto             string
                          Candidate manifesto

    position_id           integer
                          Position ID

}
```

```
Vote {
    id                    integer
                          Vote ID

    voucher               string
                          User voucher

    candidate_id          integer
                          Candidate ID

    position_id           integer
                          Position ID

    verification_code     string
                          Vote verification code

}
```

```
AdminUser {
   id
                       integer
                      Admin user ID

   username
                       string
                      Admin username

   email
                       string
                      Admin email

   role
                       string
                      Admin role

                      Enum:
                       [ super_admin, admin, moderator ]
   isActive
                       [ ... ]
   lastLogin
                       string($date-time)
                      Last login timestamp

   createdAt
                       string($date-time)
                      Account creation timestamp

}
```

```
VotingStats ⌄ {
   totalVoters
                          ⌄ integer
                         Total number of registered voters

   totalVotesCast
                          ⌄ integer
                         Total number of votes cast

   voterTurnout
                          ⌄ number($float)
                         Voter turnout percentage

   positionsWithStats
                          ⌄ [PositionStats ⌄ {
                            positionId
                                                   ⌄ integer
                                                  Position ID

                            positionName
                                                   ⌄ string
                                                  Position name

                            totalCandidates
                                                   ⌄ integer
                                                  Number of candidates

                            totalVotes
                                                   ⌄ integer
                                                  Total votes for this position

                            voterTurnout
                                                   ⌄ number($float)
                                                  Turnout percentage for this position

                            candidates
                                                   ⌄ [CandidateStats ⌄ {
                                                     candidateId
                                                                            ⌄ integer
                                                                           Candidate ID

                                                     candidateName
                                                                            ⌄ string
                                                                           Candidate name

                                                     manifesto
                                                                            ⌄ string
                                                                           Candidate manifesto

                                                     voteCount
                                                                            ⌄ integer
                                                                           Number of votes received

                                                     votePercentage
                                                                            ⌄ number($float)
                                                                           Percentage of votes

                                                   }]
                          }]
   overallStats
                         OverallStats ⌄ {
                            totalPositions
```

```
                                   integer
                                   Total number of positions
    totalCandidates
                                   integer
                                   Total number of candidates
    averageVotesPerPosition
                                   number($float)
                                   Average votes per position
    mostVotedPosition
                                   string
                                   Name of most voted position
    leastVotedPosition
                                   string
                                   Name of least voted position

  }
}
```

```
PositionStats ⌄ {
   positionId
                           ⌄ integer
                           Position ID

   positionName
                           ⌄ string
                           Position name

   totalCandidates
                           ⌄ integer
                           Number of candidates

   totalVotes
                           ⌄ integer
                           Total votes for this position

   voterTurnout
                           ⌄ number($float)
                           Turnout percentage for this position

   candidates
                           ⌄ [CandidateStats ⌄ {
                              candidateId
                                                   ⌄ integer
                                                   Candidate ID

                              candidateName
                                                   ⌄ string
                                                   Candidate name

                              manifesto
                                                   ⌄ string
                                                   Candidate manifesto

                              voteCount
                                                   ⌄ integer
                                                   Number of votes received

                              votePercentage
                                                   ⌄ number($float)
                                                   Percentage of votes

                            }]
}
```

```
CandidateStats ⌄ {
    candidateId                  ⌄ integer
                                 Candidate ID

    candidateName                ⌄ string
                                 Candidate name

    manifesto                    ⌄ string
                                 Candidate manifesto

    voteCount                    ⌄ integer
                                 Number of votes received

    votePercentage               ⌄ number($float)
                                 Percentage of votes

}
```

```
OverallStats ⌄ {
    totalPositions                 ⌄ integer
                                   Total number of positions

    totalCandidates                ⌄ integer
                                   Total number of candidates

    averageVotesPerPosition        ⌄ number($float)
                                   Average votes per position

    mostVotedPosition              ⌄ string
                                   Name of most voted position

    leastVotedPosition             ⌄ string
                                   Name of least voted position

}
```

```
VoteAuditResult ⌄ {
    voteId
                            ⌄ integer
                            Vote ID

    voucher
                            ⌄ string
                            User voucher

    verificationCode
                            ⌄ string
                            Verification code

    candidateId
                            ⌄ integer
                            Candidate ID

    positionId
                            ⌄ integer
                            Position ID

    timestamp
                            ⌄ string($date-time)
                            Vote timestamp

    isValid
                            ⌄ boolean
                            Whether vote is valid

    issues
                            ⌄ [
                            List of issues found

                            > [...]]
    riskScore
                            ⌄ number($float)
                            Risk score (0-100)

}
```

```
AuditReport ⌄ {
   totalVotesAudited
                            ⌄ integer
                            Total votes audited

   validVotes
                            ⌄ integer
                            Number of valid votes

   suspiciousVotes
                            ⌄ integer
                            Number of suspicious votes

   invalidVotes
                            ⌄ integer
                            Number of invalid votes

   riskDistribution
                            ⌄ {
                               low
                                                      ⌄ integer
                               medium                 ⌄ integer
                               high                   ⌄ integer
                               critical               ⌄ integer
                            }
   commonIssues             ⌄ [ ⌄ {
                               issue
                                                      ⌄ string
                               count                  ⌄ integer
                               severity               ⌄ string
                                                     Enum:
                                                      ⌄ [ low, medium, high, critical ]
                             }]
   recommendations          ⌄ [ ⌄ string]
}
```

```
FraudDetectionResult {
    isSuspicious
                              boolean
                            Whether suspicious activity detected

    riskLevel
                              string
                            Risk level

                            Enum:
                              [ low, medium, high, critical ]
    riskFactors
                              [
                            List of risk factors

                              string]
    confidence
                              number($float)
                            Confidence level (0-100)

    details
                              {
                               description:           Detailed analysis results

                            }
}
```