# Pulse API Reference

This document provides a comprehensive reference for all Pulse API endpoints and procedures.

## Overview

Pulse uses **tRPC** for type-safe remote procedure calls. All API endpoints are accessed through the `/api/trpc` gateway. The API uses JSON for request and response bodies.

## Authentication

All endpoints require authentication via OAuth2. The application automatically handles authentication through session cookies.

### Authentication Flow

1. User logs in via Manus OAuth
2. Session cookie is set with JWT token
3. All subsequent requests include the session cookie
4. Server validates the token and sets `ctx.user` context

## API Endpoints

### Authentication

#### Get Current User

```
trpc.auth.me.useQuery()
```

**Response**:

```
{
  id: number;
  openId: string;
  name: string | null;
  email: string | null;
  role: "user" | "admin";
  createdAt: Date;
  updatedAt: Date;
  lastSignedIn: Date;
}
```

### Logout

```
trpc.auth.logout.useMutation()
```

**Response**:

```
{
  success: boolean;
}
```

## Monitoring Targets

### List All Targets

```
trpc.targets.list.useQuery()
```

**Response**:

```
MonitoringTarget[]
```

**Example**:

```
const { data: targets, isLoading } = trpc.targets.list.useQuery();
```

### Get Target by ID

```
trpc.targets.get.useQuery({ id: number })
```

**Parameters**: - `id` (number): Target ID

**Response**:

```
 {
  id: number;
  userId: number;
  name: string;
  url: string;
  description: string | null;
  protocol: "http" | "https";
  method: "GET" | "POST" | "HEAD";
  checkInterval: number;
  timeout: number;
  expectedStatusCode: number;
  isActive: boolean;
  createdAt: Date;
  updatedAt: Date;
}
```

### Create Target

```
trpc.targets.create.useMutation()
```

### Input:

```
 {
  name: string;
  url: string;
  description?: string;
  protocol?: "http" | "https";
  method?: "GET" | "POST" | "HEAD";
  checkInterval?: number;
  timeout?: number;
  expectedStatusCode?: number;
}
```

### Example:

```
 const createMutation = trpc.targets.create.useMutation();

await createMutation.mutateAsync({
  name: "My Website",
  url: "example.com",
  protocol: "https",
  method: "GET",
  checkInterval: 60,
  timeout: 10,
  expectedStatusCode: 200,
});
```

### Update Target

```
trpc.targets.update.useMutation()
```

### Input:

```
{
  id: number;
  data: {
    name?: string;
    url?: string;
    description?: string;
    protocol?: "http" | "https";
    method?: "GET" | "POST" | "HEAD";
    checkInterval?: number;
    timeout?: number;
    expectedStatusCode?: number;
    isActive?: boolean;
  };
}
```

### Delete Target

```
trpc.targets.delete.useMutation()
```

### Input:

```
{
  id: number;
}
```

### Test Target Health Check

```
trpc.targets.testCheck.useMutation()
```

### Input:

```
{
  id: number;
}
```

### Response:

```
{
  targetId: number;
  statusCode?: number;
  responseTime: number;
  isSuccess: boolean;
  errorMessage?: string;
}
```

## Monitoring Checks

### List Recent Checks

```
trpc.checks.recent.useQuery({
  targetId: number;
  hours?: number;
})
```

**Parameters**: - `targetId` (number): Target ID - `hours` (number, optional): Hours to look back (default: 24)

**Response**:

```
{
  id: number;
  targetId: number;
  statusCode?: number;
  responseTime: number;
  isSuccess: boolean;
  errorMessage?: string;
  checkedAt: Date;
}[]
```

### List All Checks

```
trpc.checks.list.useQuery({
  targetId: number;
  limit?: number;
})
```

**Parameters**: - `targetId` (number): Target ID - `limit` (number, optional): Maximum results (default: 100)

## Alert Rules

### List All Alert Rules

```
trpc.alertRules.list.useQuery()
```

**Response**:

```
{
  id: number;
  targetId: number;
  userId: number;
  name: string;
  description?: string;
  ruleType: "consecutive_failures" | "uptime_percentage" | "response_time";
  threshold: number;
  notificationChannels: string;
  isActive: boolean;
  createdAt: Date;
  updatedAt: Date;
}[]
```

### List Rules for Target

```
trpc.alertRules.listByTarget.useQuery({
  targetId: number;
})
```

### Create Alert Rule

```
trpc.alertRules.create.useMutation()
```

### Input:

```
{
  targetId: number;
  name: string;
  description?: string;
  ruleType: "consecutive_failures" | "uptime_percentage" | "response_time";
  threshold: number;
  notificationChannels: ("email" | "slack" | "discord")[];
}
```

### Example:

```
const createRule = trpc.alertRules.create.useMutation();

await createRule.mutateAsync({
  targetId: 1,
  name: "High Failure Rate",
  ruleType: "consecutive_failures",
  threshold: 3,
  notificationChannels: ["email", "slack"],
});
```

### Update Alert Rule

```
trpc.alertRules.update.useMutation()
```

**Input**:

```
{
  id: number;
  data: {
    name?: string;
    description?: string;
    threshold?: number;
    notificationChannels?: ("email" | "slack" | "discord")[];
    isActive?: boolean;
  };
}
```

### Delete Alert Rule

```
trpc.alertRules.delete.useMutation()
```

**Input**:

```
{
  id: number;
}
```

## Alerts

### List All Alerts

```
trpc.alerts.list.useQuery({
  limit?: number;
})
```

**Parameters**: - `limit` (number, optional): Maximum results (default: 50)

**Response**:

```
{
  id: number;
  ruleId: number;
  targetId: number;
  userId: number;
  status: "triggered" | "acknowledged" | "resolved";
  message: string;
  severity: "low" | "medium" | "high" | "critical";
  triggeredAt: Date;
  acknowledgedAt?: Date;
  resolvedAt?: Date;
  createdAt: Date;
  updatedAt: Date;
}[]
```

### Get Active Alerts

```
trpc.alerts.active.useQuery()
```

**Response**: Same as list, but only triggered alerts

### Update Alert Status

```
trpc.alerts.updateStatus.useMutation()
```

**Input**:

```
{
  id: number;
  status: "triggered" | "acknowledged" | "resolved";
}
```

## Notification Settings

### Get Notification Settings

```
trpc.notificationSettings.get.useQuery()
```

**Response**:

```
{
  id: number;
  userId: number;
  emailEnabled: boolean;
  slackWebhookUrl?: string;
  discordWebhookUrl?: string;
  createdAt: Date;
  updatedAt: Date;
}
```

### Update Notification Settings

```
trpc.notificationSettings.update.useMutation()
```

**Input**:

```
{
  emailEnabled?: boolean;
  slackWebhookUrl?: string;
  discordWebhookUrl?: string;
}
```

## Statistics

### Get Uptime Statistics

```
trpc.statistics.uptime.useQuery({
  targetId: number;
  period: "daily" | "weekly" | "monthly";
  days?: number;
})
```

**Parameters**: - `targetId` (number): Target ID - `period` (string): Statistics period - `days` (number, optional): Days to look back (default: 30)

**Response**:

```
{
  id: number;
  targetId: number;
  period: "daily" | "weekly" | "monthly";
  date: string;
  totalChecks: number;
  successfulChecks: number;
  uptimePercentage: number;
  averageResponseTime?: number;
  createdAt: Date;
  updatedAt: Date;
}[]
```

### Get Target Summary

```
trpc.statistics.summary.useQuery({
  targetId: number;
})
```

**Response**:

```
{
  uptime: number;
  avgResponseTime: number;
  totalChecks: number;
  successfulChecks: number;
  lastCheck: MonitoringCheck;
}
```

### Audit Logs

#### List Audit Logs

```
trpc.auditLogs.list.useQuery({
  limit?: number;
})
```

**Parameters**: - `limit` (number, optional): Maximum results (default: 100)

**Response**:

```
{
  id: number;
  userId: number;
  action: string;
  entityType: string;
  entityId?: number;
  details?: string;
  createdAt: Date;
}[]
```

## Error Handling

All API errors return a standardized error response:

```
{
  code: string;
  message: string;
  data?: {
    code: string;
    httpStatus: number;
    path: string;
  };
}
```

### Common Error Codes

| Code | HTTP Status | Description |
|---|---|---|
| `UNAUTHORIZED` | 401 | User not authenticated |
| `FORBIDDEN` | 403 | User lacks required permissions |
| `NOT_FOUND` | 404 | Resource not found |
| `BAD_REQUEST` | 400 | Invalid input parameters |
| `INTERNAL_SERVER_ERROR` | 500 | Server error |

## Rate Limiting

API endpoints are rate-limited to prevent abuse:

- **Default**: 100 requests per minute per user
- **Burst**: 10 requests per second

Rate limit headers are included in responses:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1635789600
```

## Pagination

List endpoints support pagination through limit and offset parameters:

```
trpc.alerts.list.useQuery({
  limit: 20,
  offset: 0,
})
```

## Filtering

Some list endpoints support filtering:

```
trpc.targets.list.useQuery({
  isActive: true,
  protocol: "https",
})
```

## Sorting

List endpoints support sorting:

```
trpc.alerts.list.useQuery({
  sortBy: "triggeredAt",
  sortOrder: "desc",
})
```

## WebSocket Support (Future)

Real-time updates will be available via WebSocket:

```
const socket = io('http://localhost:3000');

socket.on('alert:triggered', (alert) => {
  console.log('New alert:', alert);
});

socket.on('check:completed', (check) => {
  console.log('Check completed:', check);
});
```

# SDK Usage

## React Hook Usage

```
import { trpc } from '@/lib/trpc';

function MyComponent() {
  // Query
  const { data: targets, isLoading } = trpc.targets.list.useQuery();

  // Mutation
  const createMutation = trpc.targets.create.useMutation({
    onSuccess: () => {
      console.log('Target created!');
    },
    onError: (error) => {
      console.error('Error:', error);
    },
  });

  return (
    <div>
      {isLoading ? 'Loading...' : targets?.length}
      <button onClick={() => createMutation.mutate({...})}>
        Create
      </button>
    </div>
  );
}
```

## Error Handling

```
const createMutation = trpc.targets.create.useMutation();

try {
  await createMutation.mutateAsync({
    name: "My Target",
    url: "example.com",
  });
} catch (error) {
  if (error.code === 'BAD_REQUEST') {
    console.error('Invalid input:', error.message);
  } else if (error.code === 'UNAUTHORIZED') {
    console.error('Please log in');
  }
}
```

---

**Version**: 1.0.0

**Last Updated**: 2024