Digi-Cadence Portfolio Management Platform - API Documentation

Version: 1.0
Date: December 2024
Author: Manus AI
Document Type: API Reference Guide

Table of Contents

API Overview

The Digi-Cadence API provides comprehensive programmatic access to all platform functionality, enabling seamless integration with existing enterprise systems and custom application development. The API follows RESTful principles and uses JSON for data exchange, making it easy to integrate with any programming language or platform.

API Architecture and Design Principles

The Digi-Cadence API is designed around RESTful principles with clear resource hierarchies and consistent HTTP methods for different operations. The API uses standard HTTP status codes for response indication and provides detailed error messages to facilitate debugging and integration.

Resource-based URLs follow a logical hierarchy that mirrors the platform's data model. For example, `/api/v1/organizations/{org_id}/projects/{project_id}/brands/{brand_id}` provides access to a specific brand within a project and organization. This hierarchical structure makes the API intuitive and predictable.

HTTP methods are used consistently throughout the API: GET for retrieving data, POST for creating new resources, PUT for updating existing resources, PATCH for partial updates, and DELETE for removing resources. This consistency makes the API easy to learn and use.

JSON is used exclusively for request and response payloads, with consistent field naming conventions and data types. All timestamps use ISO 8601 format, and all monetary values include currency information to avoid ambiguity.

Versioning is implemented through URL paths (e.g., `/api/v1/`) to ensure backward compatibility as the API evolves. Version deprecation follows a clear timeline with advance notice to allow for smooth transitions.

Base URL and Endpoints

The API base URL varies depending on your deployment environment. For production deployments, the base URL typically follows the pattern `https://your-domain.com/api/v1/`. All API endpoints are relative to this base URL.

The current API version is v1, and all endpoints are prefixed with `/api/v1/`. Future API versions will use different version numbers while maintaining backward compatibility for existing integrations.

All API communication must use HTTPS to ensure data security and integrity. HTTP requests will be automatically redirected to HTTPS, but clients should use HTTPS directly to avoid potential security issues.

Content-Type headers must be set to `application/json` for all requests that include a request body. The API will return a 400 Bad Request error if the Content-Type header is missing or incorrect.

Request and Response Format

All API requests and responses use JSON format with consistent structure and naming conventions. Request payloads should be well-formed JSON with appropriate data types for each field.

Response format includes both the requested data and metadata about the response. Successful responses include a `data` field containing the requested information and a `meta` field containing response metadata such as pagination information.

Error responses include an `error` field with detailed error information including error code, message, and additional context to help with debugging. Error responses follow RFC 7807 Problem Details for HTTP APIs standard.

Pagination is implemented for endpoints that return multiple items, using cursor-based pagination for optimal performance. Pagination metadata is included in the response to facilitate navigation through large result sets.

Field naming follows camelCase convention for consistency with JavaScript and other modern APIs. Date and time fields use ISO 8601 format with timezone information included.

Authentication Overview

The Digi-Cadence API uses JWT (JSON Web Token) based authentication for secure, stateless authentication that scales well across multiple application servers. All API requests must include a valid JWT token in the Authorization header.

Authentication tokens are obtained through the authentication endpoint using username and password credentials. Tokens have a configurable expiration time (typically 1 hour) and must be refreshed using refresh tokens for continued access.

Role-based access control is enforced at the API level, ensuring that users can only access data and perform actions appropriate to their assigned roles. API responses are filtered based on user permissions to prevent unauthorized data access.

Multi-factor authentication is supported and can be required for sensitive operations. When MFA is enabled, certain API operations may require additional authentication steps.

API keys are available for server-to-server integrations where user-based authentication is not appropriate. API keys have specific permissions and can be restricted to specific IP addresses for enhanced security.

Authentication

Authentication is the foundation of API security, ensuring that only authorized users and systems can access the Digi-Cadence platform. The authentication system is designed to be secure, scalable, and easy to integrate with existing enterprise authentication systems.

JWT Token Authentication

JWT (JSON Web Token) authentication provides secure, stateless authentication that scales well across distributed systems. JWT tokens contain user identity, role information, and expiration times, and are cryptographically signed to prevent tampering.

To obtain a JWT token, send a POST request to the authentication endpoint with valid credentials:

```
 POST /api/v1/auth/login
Content-Type: application/json

{
  "username": "user@example.com",
  "password": "secure_password",
  "organization_id": "org_123"
}
```

Successful authentication returns both an access token and a refresh token:

```
{
  "data": {
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "token_type": "Bearer",
    "expires_in": 3600,
    "user": {
      "id": "user_456",
      "username": "user@example.com",
      "role": "brand_manager",
      "organization_id": "org_123"
    }
  }
}
```

Include the access token in the Authorization header for all subsequent API requests:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Token Refresh

Access tokens have a limited lifetime (typically 1 hour) for security purposes. Use the refresh token to obtain new access tokens without requiring the user to re-authenticate:

```
POST /api/v1/auth/refresh
Content-Type: application/json

{
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

The response includes a new access token and optionally a new refresh token:

```
{
  "data": {
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "token_type": "Bearer",
    "expires_in": 3600
  }
}
```

Multi-Factor Authentication

Multi-factor authentication (MFA) provides additional security for sensitive operations. When MFA is enabled, certain API operations require additional authentication steps.

Initiate MFA challenge for sensitive operations:

```
POST /api/v1/auth/mfa/challenge
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "operation": "delete_brand",
  "resource_id": "brand_789"
}
```

Complete MFA challenge with verification code:

```
POST /api/v1/auth/mfa/verify
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "challenge_id": "challenge_123",
  "verification_code": "123456"
}
```

API Key Authentication

API keys provide authentication for server-to-server integrations where user-based authentication is not appropriate. API keys have specific permissions and can be restricted to specific IP addresses.

Create API key (requires administrative privileges):

```
POST /api/v1/auth/api-keys
Authorization: Bearer {admin_access_token}
Content-Type: application/json

{
  "name": "Integration API Key",
  "permissions": ["read_portfolio", "read_analytics"],
  "ip_restrictions": ["192.168.1.100", "10.0.0.0/8"],
  "expires_at": "2025-12-31T23:59:59Z"
}
```

Use API key in requests:

```
GET /api/v1/organizations/org_123/portfolio
X-API-Key: api_key_abc123def456
```

Role-Based Access Control

The API enforces role-based access control (RBAC) to ensure users can only access data and perform actions appropriate to their role. The platform includes six predefined roles with different permission levels.

System Administrator: Full access to all platform functionality including user management, system configuration, and all data across all organizations.

Organization Administrator: Full access within their organization including user management, organization settings, and all portfolio data.

Portfolio Manager: Access to portfolio-level data and analytics across all projects and brands within their organization.

Brand Manager: Access to specific brands and projects they are assigned to, with full management capabilities for those brands.

Analyst: Read-only access to analytics data and reports for brands and projects they are assigned to.

Viewer: Read-only access to basic portfolio and brand information for brands they are assigned to.

Permission checking is performed at the API level, and responses are filtered based on user permissions. Attempting to access unauthorized resources returns a 403 Forbidden error.

Session Management

Session management ensures that user sessions are secure and properly managed throughout their lifecycle. Sessions are stored securely and include protection against common session-based attacks.

Session creation occurs during authentication and includes generation of secure session tokens with appropriate expiration times. Sessions are stored in Redis with encryption and include user identity and permission information.

Session validation occurs on every API request to ensure that sessions are still valid and that user permissions have not changed. Invalid or expired sessions result in 401 Unauthorized responses.

Session termination can be initiated by the user through logout or automatically when sessions expire. Logout invalidates both access and refresh tokens:

```
POST /api/v1/auth/logout
Authorization: Bearer {access_token}
```

Session monitoring tracks active sessions and can detect suspicious activity such as concurrent sessions from different locations. Administrators can view and terminate active sessions for security purposes.

Portfolio Management APIs

The Portfolio Management APIs provide comprehensive access to portfolio data and management functionality, enabling programmatic management of organizations, projects, brands, and their relationships. These APIs are essential for integrating Digi-Cadence with existing enterprise systems and building custom portfolio management applications.

Organization Management

Organization APIs provide access to organization-level data and settings. Organizations are the top-level container for all portfolio data and provide multi-tenant data isolation.

Get Organization Details

```
GET /api/v1/organizations/{organization_id}
Authorization: Bearer {access_token}
```

Response:

```
{
  "data": {
    "id": "org_123",
    "name": "Acme Corporation",
    "description": "Global consumer goods company",
    "industry": "Consumer Goods",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-12-01T14:22:00Z",
    "settings": {
      "default_currency": "USD",
      "timezone": "America/New_York",
      "fiscal_year_start": "01-01"
    },
    "subscription": {
      "plan": "enterprise",
      "max_brands": 1000,
      "max_users": 500,
      "features": ["advanced_analytics", "ai_agents", "custom_reports"]
    }
  }
}
```

Update Organization Settings

```
PUT /api/v1/organizations/{organization_id}
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "name": "Acme Corporation",
  "description": "Global consumer goods company",
  "settings": {
    "default_currency": "USD",
    "timezone": "America/New_York",
    "fiscal_year_start": "01-01"
  }
}
```

Project Management

Project APIs manage marketing projects within an organization. Projects group related brands and marketing initiatives for better organization and analysis.

List Projects

```
GET /api/v1/organizations/{organization_id}/projects
Authorization: Bearer {access_token}
```

Query parameters: - `page`: Page number for pagination (default: 1) - `limit`: Number of items per page (default: 20, max: 100) - `status`: Filter by project status (active, inactive, completed) - `sort`: Sort field (name, created_at, updated_at) - `order`: Sort order (asc, desc)

Response:

```json
{
  "data": [
    {
      "id": "proj_456",
      "name": "Q1 2024 Product Launch",
      "description": "Launch campaign for new product line",
      "status": "active",
      "start_date": "2024-01-01",
      "end_date": "2024-03-31",
      "budget": {
        "total": 1000000,
        "allocated": 750000,
        "spent": 500000,
        "currency": "USD"
      },
      "brand_count": 3,
      "created_at": "2023-12-01T10:00:00Z",
      "updated_at": "2024-12-01T15:30:00Z"
    }
  ],
  "meta": {
    "page": 1,
    "limit": 20,
    "total": 15,
    "total_pages": 1
  }
}
```

Create Project

```
POST /api/v1/organizations/{organization_id}/projects
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "name": "Q2 2024 Brand Refresh",
  "description": "Comprehensive brand refresh initiative",
  "status": "active",
  "start_date": "2024-04-01",
  "end_date": "2024-06-30",
  "budget": {
    "total": 2000000,
    "currency": "USD"
  },
  "objectives": [
    "Increase brand awareness by 25%",
    "Improve brand sentiment score",
    "Launch in 3 new markets"
  ]
}
```

Get Project Details

```
 GET /api/v1/organizations/{organization_id}/projects/{project_id}
Authorization: Bearer {access_token}
```

Update Project

```
 PUT /api/v1/organizations/{organization_id}/projects/{project_id}
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "name": "Q2 2024 Brand Refresh - Updated",
  "description": "Comprehensive brand refresh initiative with expanded scope",
  "status": "active",
  "budget": {
    "total": 2500000,
    "currency": "USD"
  }
}
```

Brand Management

Brand APIs provide comprehensive management of individual brands within projects. Brands are the core entities that the platform analyzes and optimizes.

List Brands

```
 GET /api/v1/organizations/{organization_id}/projects/{project_id}/brands
Authorization: Bearer {access_token}
```

Query parameters: - `page`: Page number for pagination - `limit`: Number of items per page - `category`: Filter by brand category - `status`: Filter by brand status - `performance_tier`: Filter by performance tier (high, medium, low)

Response:

```json
{
  "data": [
    {
      "id": "brand_789",
      "name": "Premium Brand A",
      "description": "Luxury consumer product brand",
      "category": "Luxury Goods",
      "status": "active",
      "launch_date": "2020-03-15",
      "target_demographics": {
        "age_range": "25-45",
        "income_level": "high",
        "geographic_focus": ["North America", "Europe"]
      },
      "performance_metrics": {
        "brand_equity_score": 8.5,
        "market_share": 12.3,
        "customer_satisfaction": 4.2,
        "revenue_ytd": 15000000,
        "growth_rate": 8.7
      },
      "created_at": "2020-03-01T09:00:00Z",
      "updated_at": "2024-12-01T16:45:00Z"
    }
  ],
  "meta": {
    "page": 1,
    "limit": 20,
    "total": 8,
    "total_pages": 1
  }
}
```

Create Brand

```
POST /api/v1/organizations/{organization_id}/projects/{project_id}/brands
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "name": "New Brand B",
  "description": "Innovative tech product brand",
  "category": "Technology",
  "status": "active",
  "launch_date": "2024-06-01",
  "target_demographics": {
    "age_range": "18-35",
    "income_level": "medium",
    "geographic_focus": ["North America"]
  },
  "positioning": {
    "value_proposition": "Affordable innovation for everyone",
    "key_differentiators": ["Price", "Ease of use", "Reliability"],
    "competitive_positioning": "Value leader"
  }
}
```

Get Brand Details

```
 GET
/api/v1/organizations/{organization_id}/projects/{project_id}/brands/{brand_id}
Authorization: Bearer {access_token}
```

Update Brand

```
 PUT
/api/v1/organizations/{organization_id}/projects/{project_id}/brands/{brand_id}
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "name": "Updated Brand Name",
  "description": "Updated brand description",
  "target_demographics": {
    "age_range": "20-40",
    "income_level": "medium-high",
    "geographic_focus": ["North America", "Asia Pacific"]
  }
}
```

Brand Relationships

Brand relationship APIs manage the complex relationships between brands in a portfolio, including hierarchies, partnerships, and competitive relationships.

List Brand Relationships

```
 GET /api/v1/organizations/{organization_id}/brand-relationships
Authorization: Bearer {access_token}
```

Create Brand Relationship

```
POST /api/v1/organizations/{organization_id}/brand-relationships
Authorization: Bearer {access_token}
Content-Type: application/json

{
  "source_brand_id": "brand_789",
  "target_brand_id": "brand_456",
  "relationship_type": "parent_child",
  "strength": 0.8,
  "description": "Premium Brand A is the parent brand of Brand B",
  "effective_date": "2024-01-01",
  "metadata": {
    "hierarchy_level": 1,
    "shared_resources": ["marketing_team", "distribution"]
  }
}
```

Portfolio Analytics

Portfolio analytics APIs provide access to portfolio-level metrics and analysis that consider the relationships between brands and projects.

Get Portfolio Overview

```
GET /api/v1/organizations/{organization_id}/portfolio/overview
Authorization: Bearer {access_token}
```

Query parameters: - `time_period`: Time period for analysis (7d, 30d, 90d, 1y) - `include_projections`: Include future projections (true, false)

Response:

```json
{
  "data": {
    "portfolio_value": {
      "total": 150000000,
      "currency": "USD",
      "growth_rate": 12.5,
      "vs_previous_period": 8.3
    },
    "brand_performance": {
      "total_brands": 25,
      "high_performers": 8,
      "medium_performers": 12,
      "low_performers": 5,
      "average_equity_score": 7.2
    },
    "market_position": {
      "total_market_share": 18.7,
      "market_rank": 3,
      "competitive_strength": "strong"
    },
    "synergy_analysis": {
      "synergy_score": 0.73,
      "positive_correlations": 15,
      "negative_correlations": 3,
      "optimization_opportunities": 7
    }
  }
}
```

Get Cross-Brand Correlation Analysis

```
GET /api/v1/organizations/{organization_id}/portfolio/correlations
Authorization: Bearer {access_token}
```

Query parameters: - `brand_ids`: Comma-separated list of brand IDs to analyze - `metrics`: Comma-separated list of metrics to correlate - `time_period`: Time period for analysis

Response:

```json
{
  "data": {
    "correlation_matrix": [
      {
        "brand_a": "brand_789",
        "brand_b": "brand_456",
        "correlation": 0.65,
        "significance": 0.001,
        "relationship_type": "positive",
        "metrics": {
          "revenue": 0.72,
          "market_share": 0.58,
          "brand_sentiment": 0.43
        }
      }
    ],
    "insights": [
      {
        "type": "synergy_opportunity",
        "description": "Strong positive correlation between Brand A and Brand B suggests cross-promotion opportunities",
        "confidence": 0.85,
        "potential_impact": "high"
      }
    ]
  }
}
```