# **Logo Builder Frontend - API Documentation**

## **Overview**

This document provides comprehensive documentation for all API interactions within the Logo Builder Frontend application. The application follows a RESTful API architecture with JWT-based authentication and standardized request/response formats.

## **Base Configuration**

### **API Gateway Configuration**

* **Production Base URL:** <https://api-gw.builderservices.io>
* **UAT Base URL:** <https://api-gw.uat.builderservices.io>
* **Default API Name:** logos
* **Content Type:** application/json
* **Authentication:** Bearer Token (JWT)

### **Environment Detection**

const getBaseUrl = () => {  
 const env = getRuntimeConfig().\_env ?? window?.shellRuntime?.runtime?.\_env ?? 'prod'  
 return env === 'prod' || env === 'production'  
 ? '<https://api-gw.builderservices.io>'  
 : '<https://api-gw.uat.builderservices.io>'  
}

## **Authentication**

### **JWT Token Management**

**Implementation:** src/helpers/auth-ujwt-helper.js

#### **Token Acquisition**

const getToken = async () => {  
 // Token retrieval logic  
 return await AuthUjwtHelper.getToken()  
}

#### **Authorization Header Format**

Authorization: Bearer <JWT\_TOKEN>

### **Authentication Endpoints**

#### **POST /v1.0/token/me**

**Purpose:** Validate current user token and retrieve user information

**Implementation:** src/helpers/authentication-helper.js

**Authentication:** Required

**Request Headers:**

Authorization: Bearer <token>  
Content-Type: application/json

**Response:**

{  
 "user": {  
 "id": "user-123",  
 "email": "[user@example.com](mailto:user@example.com)",  
 "name": "John Doe",  
 "accountType": "premium",  
 "permissions": ["logo:create", "logo:edit", "logo:download"]  
 },  
 "token": {  
 "expiresAt": "2024-12-31T23:59:59Z",  
 "refreshToken": "refresh-token-here"  
 }  
}

## **Logo Management API**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/logos/v1.0/

**Implementation:** src/clients/logo.client.js

### **Logo CRUD Operations**

#### **GET /v1.0/logos/list**

**Purpose:** Retrieve user's logo collection

**Authentication:** Required

**Implementation:** LogoClient.getLogos()

**Query Parameters:**

* include\_logo (boolean) - Include full logo template data
* page (number, optional) - Page number for pagination
* limit (number, optional) - Number of items per page
* sort (string, optional) - Sort order: 'created\_desc', 'created\_asc', 'name\_asc', 'name\_desc'

**Request Example:**

GET /v1.0/logos/list?include\_logo=true&page=1&limit=10  
Authorization: Bearer <token>

**Response:**

{  
 "logos": [  
 {  
 "id": "logo-123",  
 "name": "My Company Logo",  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"Company\"}}",  
 "createdAt": "2024-01-15T10:30:00Z",  
 "updatedAt": "2024-01-15T14:20:00Z",  
 "downloadCount": 5,  
 "isPublic": false,  
 "metadata": {  
 "industry": "technology",  
 "style": "modern",  
 "colors": ["#FF6B6B", "#4ECDC4"],  
 "fonts": ["Roboto", "Open Sans"]  
 }  
 }  
 ],  
 "pagination": {  
 "total": 25,  
 "page": 1,  
 "limit": 10,  
 "totalPages": 3  
 }  
}

#### **GET /v1.0/logos/{logoId}/load**

**Purpose:** Load specific logo by ID

**Authentication:** Required

**Implementation:** LogoClient.getLogo(logoId)

**Path Parameters:**

* logoId (string) - Unique logo identifier

**Request Example:**

GET /v1.0/logos/logo-123/load  
Authorization: Bearer <token>

**Response:**

{  
 "id": "logo-123",  
 "name": "My Company Logo",  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"Company\",\"slogan\":\"Innovation First\"},\"font\":{\"brand1\":{\"id\":\"roboto\",\"size\":1.2,\"color\":\"#333333\"}},\"layout\":{\"decoration\":{\"style\":\"modern\"},\"symbol\":{\"position\":\"left\"}},\"colors\":[\"#FF6B6B\",\"#4ECDC4\"]}",  
 "userId": "user-456",  
 "createdAt": "2024-01-15T10:30:00Z",  
 "updatedAt": "2024-01-15T14:20:00Z",  
 "downloadCount": 5,  
 "isPublic": false,  
 "metadata": {  
 "industry": "technology",  
 "style": "modern",  
 "dimensions": {  
 "width": 800,  
 "height": 600  
 }  
 }  
}

#### **POST /v1.0/logos/save**

**Purpose:** Create new logo

**Authentication:** Required

**Implementation:** LogoClient.saveLogo(logo)

**Request Body:**

{  
 "name": "New Company Logo",  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"New Company\"},\"font\":{\"brand1\":{\"id\":\"arial\",\"size\":1.0}}}",  
 "metadata": {  
 "industry": "healthcare",  
 "style": "professional",  
 "colors": ["#2E86AB", "#A23B72"],  
 "fonts": ["Arial"]  
 }  
}

**Response:**

{  
 "id": "logo-456",  
 "name": "New Company Logo",  
 "createdAt": "2024-01-16T09:15:00Z",  
 "message": "Logo created successfully"  
}

#### **POST /v1.0/logos/{logoId}/save**

**Purpose:** Update existing logo

**Authentication:** Required

**Implementation:** LogoClient.updateLogo(logoId, logo)

**Path Parameters:**

* logoId (string) - Logo identifier to update

**Request Body:**

{  
 "name": "Updated Logo Name",  
 "logo": "{\"logoVersion\":2,\"text\":{\"brandName\":\"Updated Company\",\"slogan\":\"New Slogan\"}}",  
 "metadata": {  
 "industry": "technology",  
 "style": "modern"  
 }  
}

**Response:**

{  
 "id": "logo-123",  
 "name": "Updated Logo Name",  
 "updatedAt": "2024-01-16T11:30:00Z",  
 "message": "Logo updated successfully"  
}

#### **DELETE /v1.0/logos/{logoId}/delete**

**Purpose:** Delete logo

**Authentication:** Required

**Implementation:** LogoClient.deleteLogo(logoId)

**Path Parameters:**

* logoId (string) - Logo identifier to delete

**Request Example:**

DELETE /v1.0/logos/logo-123/delete  
Authorization: Bearer <token>

**Response:**

{  
 "message": "Logo deleted successfully",  
 "deletedAt": "2024-01-16T12:00:00Z"  
}

#### **POST /v1.0/logos/{logoId}/duplicate**

**Purpose:** Create duplicate of existing logo

**Authentication:** Required

**Implementation:** LogoClient.duplicateLogo(logoId)

**Path Parameters:**

* logoId (string) - Logo identifier to duplicate

**Request Example:**

POST /v1.0/logos/logo-123/duplicate  
Authorization: Bearer <token>

**Response:**

{  
 "id": "logo-789",  
 "originalId": "logo-123",  
 "name": "Copy of My Company Logo",  
 "createdAt": "2024-01-16T13:45:00Z",  
 "message": "Logo duplicated successfully"  
}

#### **POST /v1.0/logos/{logoId}/downloaded**

**Purpose:** Mark logo as downloaded (for analytics)

**Authentication:** Required

**Implementation:** LogoClient.markLogoDownloaded(logoId)

**Path Parameters:**

* logoId (string) - Logo identifier

**Request Example:**

POST /v1.0/logos/logo-123/downloaded  
Authorization: Bearer <token>

**Response:**

{  
 "message": "Download recorded successfully",  
 "downloadCount": 6,  
 "timestamp": "2024-01-16T14:00:00Z"  
}

## **Temporary Logo API**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/logos/v1.0/temporary-logos/

**Implementation:** src/clients/temporary-logo.client.js

### **Temporary Logo Operations**

#### **GET /v1.0/temporary-logos/{logoId}/load**

**Purpose:** Load temporary logo (no authentication required)

**Authentication:** None

**Implementation:** TemporaryLogoClient.getLogo(logoId)

**Path Parameters:**

* logoId (string) - Temporary logo identifier

**Request Example:**

GET /v1.0/temporary-logos/temp-logo-123/load

**Response:**

{  
 "id": "temp-logo-123",  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"Guest Logo\"}}",  
 "createdAt": "2024-01-16T10:00:00Z",  
 "expiresAt": "2024-01-23T10:00:00Z",  
 "sessionId": "session-456"  
}

#### **POST /v1.0/temporary-logos/save**

**Purpose:** Save temporary logo for guest users

**Authentication:** None

**Implementation:** TemporaryLogoClient.saveLogo(logo)

**Request Body:**

{  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"Guest Company\"}}",  
 "sessionId": "session-789"  
}

**Response:**

{  
 "id": "temp-logo-456",  
 "createdAt": "2024-01-16T11:00:00Z",  
 "expiresAt": "2024-01-23T11:00:00Z",  
 "message": "Temporary logo saved successfully"  
}

#### **PUT /v1.0/temporary-logos/{logoId}/save**

**Purpose:** Update existing temporary logo

**Authentication:** None

**Implementation:** TemporaryLogoClient.updateLogo(logoId, logo)

**Path Parameters:**

* logoId (string) - Temporary logo identifier

**Request Body:**

{  
 "logo": "{\"logoVersion\":2,\"text\":{\"brandName\":\"Updated Guest Logo\"}}",  
 "sessionId": "session-789"  
}

**Response:**

{  
 "id": "temp-logo-456",  
 "updatedAt": "2024-01-16T12:00:00Z",  
 "message": "Temporary logo updated successfully"  
}

#### **POST /v1.0/temporary-logos/transfer/{logoId}**

**Purpose:** Transfer temporary logo to permanent user account

**Authentication:** Required

**Implementation:** TemporaryLogoClient.transferLogo(logoId)

**Path Parameters:**

* logoId (string) - Temporary logo identifier

**Request Example:**

POST /v1.0/temporary-logos/transfer/temp-logo-456  
Authorization: Bearer <token>

**Response:**

{  
 "id": "logo-999",  
 "originalTempId": "temp-logo-456",  
 "name": "Transferred Logo",  
 "transferredAt": "2024-01-16T13:00:00Z",  
 "message": "Logo transferred successfully"  
}

## **Account Limitations API**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/logos/v1.0/limitation/

**Implementation:** src/clients/limitations.client.js

### **Limitation Endpoints**

#### **GET /v1.0/limitation**

**Purpose:** Get logo-specific limitations

**Authentication:** Required

**Implementation:** LimitationsClient.getLogomakerLimitations(logoId)

**Query Parameters:**

* resourceId (string) - Logo ID to check limitations for

**Request Example:**

GET /v1.0/limitation?resourceId=logo-123  
Authorization: Bearer <token>

**Response:**

{  
 "resourceId": "logo-123",  
 "limitations": {  
 "canEdit": true,  
 "canDownload": true,  
 "canDelete": true,  
 "downloadFormats": ["png", "jpg"],  
 "maxResolution": "1920x1080",  
 "commercialUse": false  
 },  
 "accountType": "free",  
 "upgradeRequired": false  
}

#### **GET /v2.0/limitation**

**Purpose:** Get account-level limitations

**Authentication:** Required

**Implementation:** LimitationsClient.getLogomakerAccountLimitations()

**Request Example:**

GET /v2.0/limitation  
Authorization: Bearer <token>

**Response:**

{  
 "accountType": "premium",  
 "sku": "premium-monthly",  
 "limitations": {  
 "maxLogos": 50,  
 "currentLogoCount": 12,  
 "downloadFormats": ["png", "jpg", "svg", "pdf"],  
 "maxResolution": "4096x4096",  
 "commercialUse": true,  
 "advancedFeatures": true,  
 "prioritySupport": true  
 },  
 "subscription": {  
 "status": "active",  
 "renewsAt": "2024-02-15T00:00:00Z",  
 "cancelledAt": null  
 },  
 "usage": {  
 "logosCreated": 12,  
 "downloadsThisMonth": 45,  
 "storageUsed": "125MB"  
 }  
}

#### **GET /v1.0/accounts/info**

**Purpose:** Get account information

**Authentication:** Required

**Implementation:** LimitationsClient.getAccountInfo()

**Request Example:**

GET /v1.0/accounts/info  
Authorization: Bearer <token>

**Response:**

{  
 "user": {  
 "id": "user-123",  
 "email": "[user@example.com](mailto:user@example.com)",  
 "name": "John Doe",  
 "createdAt": "2023-06-15T10:00:00Z"  
 },  
 "account": {  
 "type": "premium",  
 "status": "active",  
 "billingCycle": "monthly",  
 "nextBillingDate": "2024-02-15T00:00:00Z"  
 },  
 "preferences": {  
 "defaultIndustry": "technology",  
 "preferredStyle": "modern",  
 "notifications": {  
 "email": true,  
 "browser": false  
 }  
 }  
}

#### **GET /v1.0/limitation/provisioning-status**

**Purpose:** Check account provisioning status

**Authentication:** Required

**Implementation:** LimitationsClient.getProvisioningStatus()

**Request Example:**

GET /v1.0/limitation/provisioning-status  
Authorization: Bearer <token>

**Response:**

{  
 "status": "completed",  
 "provisionedAt": "2024-01-15T08:30:00Z",  
 "services": {  
 "logoBuilder": "active",  
 "storage": "active",  
 "analytics": "active"  
 },  
 "pendingActions": []  
}

## **Onboarding API**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/logos/v1.0/onboarding/

**Implementation:** src/clients/onboarding-info.client.js

#### **GET /v1.0/onboarding/{onboardingGuid}**

**Purpose:** Retrieve onboarding information

**Authentication:** Optional

**Implementation:** OnboardingInfoClient.getOnboardingInfo(onboardingGuid)

**Path Parameters:**

* onboardingGuid (string) - Unique onboarding session identifier

**Request Example:**

GET /v1.0/onboarding/onboard-123-456-789

**Response:**

{  
 "guid": "onboard-123-456-789",  
 "step": "template-selection",  
 "progress": 60,  
 "data": {  
 "businessInfo": {  
 "name": "Tech Startup",  
 "industry": "technology",  
 "targetAudience": "professionals"  
 },  
 "preferences": {  
 "style": "modern",  
 "colors": ["#2E86AB", "#A23B72"],  
 "symbols": ["tech", "innovation"]  
 },  
 "selectedTemplate": "template-456"  
 },  
 "createdAt": "2024-01-16T09:00:00Z",  
 "expiresAt": "2024-01-16T21:00:00Z"  
}

## **Identity Server API**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/ids4-api/

**Implementation:** src/clients/ids4.client.js

#### **POST /ujwt**

**Purpose:** Get user JWT token

**Authentication:** Required

**Implementation:** Ids4Client.getUjwt()

**Request Example:**

POST /ujwt  
Authorization: Bearer <token>

**Response:**

{  
 "ujwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
 "expiresAt": "2024-01-16T18:00:00Z",  
 "scope": ["logo:read", "logo:write", "account:read"]  
}

## **AI API Integration**

### **Base Endpoint**

**URL Pattern:** {baseUrl}/ai-api/v1.0/

**Implementation:** src/api-config/index.js

#### **Configuration**

export const aiApi = (path = '', returnType = 'json', useCache = true) => {  
 return {  
 url: `${getBaseUrl()}/ai-api/v1.0/${path}`,  
 useCache,  
 returnType  
 }  
}

## **Error Handling**

### **Standard Error Response Format**

All API endpoints return errors in a consistent format:

{  
 "error": {  
 "code": "ERROR\_CODE",  
 "message": "Human-readable error message",  
 "details": {  
 "field": "specific field that caused error",  
 "value": "invalid value",  
 "constraint": "validation rule violated"  
 },  
 "timestamp": "2024-01-16T15:30:00Z",  
 "requestId": "req-123-456-789"  
 }  
}

### **HTTP Status Codes**

#### **Success Codes**

* **200 OK** - Request successful
* **201 Created** - Resource created successfully
* **204 No Content** - Request successful, no content returned

#### **Client Error Codes**

* **400 Bad Request** - Invalid request format or parameters
* **401 Unauthorized** - Authentication required or invalid
* **403 Forbidden** - Access denied due to permissions
* **404 Not Found** - Resource not found
* **409 Conflict** - Resource conflict (e.g., duplicate name)
* **422 Unprocessable Entity** - Validation errors
* **429 Too Many Requests** - Rate limit exceeded

#### **Server Error Codes**

* **500 Internal Server Error** - Unexpected server error
* **502 Bad Gateway** - Upstream service error
* **503 Service Unavailable** - Service temporarily unavailable
* **504 Gateway Timeout** - Request timeout

### **Common Error Codes**

#### **Authentication Errors**

* AUTH\_TOKEN\_MISSING - Authorization header missing
* AUTH\_TOKEN\_INVALID - Invalid or expired token
* AUTH\_TOKEN\_EXPIRED - Token has expired
* AUTH\_INSUFFICIENT\_PERMISSIONS - User lacks required permissions

#### **Logo Management Errors**

* LOGO\_NOT\_FOUND - Requested logo does not exist
* LOGO\_ACCESS\_DENIED - User cannot access this logo
* LOGO\_INVALID\_FORMAT - Logo data format is invalid
* LOGO\_NAME\_REQUIRED - Logo name is required
* LOGO\_SIZE\_EXCEEDED - Logo data exceeds size limits

#### **Account Limitation Errors**

* ACCOUNT\_LIMIT\_EXCEEDED - Account logo limit reached
* FEATURE\_NOT\_AVAILABLE - Feature requires upgrade
* DOWNLOAD\_LIMIT\_EXCEEDED - Monthly download limit reached
* RESOLUTION\_NOT\_ALLOWED - Requested resolution not available

## **Rate Limiting**

### **Rate Limit Headers**

All API responses include rate limiting information:

X-RateLimit-Limit: 1000  
X-RateLimit-Remaining: 999  
X-RateLimit-Reset: 1642348800  
X-RateLimit-Window: 3600

### **Rate Limit Policies**

* **Authenticated Users:** 1000 requests per hour
* **Guest Users:** 100 requests per hour
* **Logo Operations:** 50 saves per hour
* **Download Operations:** 20 downloads per hour

## **Request/Response Examples**

### **Complete Logo Creation Flow**

#### **1. Create Temporary Logo (Guest User)**

curl -X POST "<https://api-gw.builderservices.io/logos/v1.0/temporary-logos/save>" \  
 -H "Content-Type: application/json" \  
 -d '{  
 "logo": "{\"logoVersion\":1,\"text\":{\"brandName\":\"My Company\"}}",  
 "sessionId": "guest-session-123"  
 }'

#### **2. User Signs Up and Gets Token**

curl -X POST "<https://api-gw.builderservices.io/logos/v1.0/token/me>" \  
 -H "Authorization: Bearer <new-user-token>"

#### **3. Transfer Temporary Logo**

curl -X POST "<https://api-gw.builderservices.io/logos/v1.0/temporary-logos/transfer/temp-logo-123>" \  
 -H "Authorization: Bearer <user-token>"

#### **4. Update Transferred Logo**

curl -X POST "<https://api-gw.builderservices.io/logos/v1.0/logos/logo-456/save>" \  
 -H "Authorization: Bearer <user-token>" \  
 -H "Content-Type: application/json" \  
 -d '{  
 "name": "My Final Logo",  
 "logo": "{\"logoVersion\":2,\"text\":{\"brandName\":\"My Company\",\"slogan\":\"Excellence\"}}"  
 }'

#### **5. Download Logo**

curl -X POST "<https://api-gw.builderservices.io/logos/v1.0/logos/logo-456/downloaded>" \  
 -H "Authorization: Bearer <user-token>"

## **SDK and Client Libraries**

### **JavaScript Client Implementation**

The frontend uses a custom HTTP client wrapper:

// src/helpers/utils.js  
export const performFetch = async (url, options = {}) => {  
 const defaultHeaders = {  
 'Content-Type': 'application/json'  
 }  
  
 const mergedOptions = {  
 method: options.method || 'GET',  
 headers: {  
 ...defaultHeaders,  
 ...options.headers  
 },  
 ...(options.body && {   
 body: typeof options.body === 'string'   
 ? options.body   
 : JSON.stringify(options.body)   
 })  
 }  
  
 const response = await window.fetch(url, mergedOptions)  
 const data = await handleResponse(response)  
 return data  
}

### **API Client Pattern**

Each API domain has its own client module:

// src/clients/logo.client.js  
const LogoClient = {  
 getLogos: async () =>  
 performFetch(api('v1.0/logos/list?include\_logo=true'), {  
 headers: { Authorization: `Bearer ${await AuthUjwtHelper.getToken()}` }  
 }),  
  
 saveLogo: async logo =>  
 performFetch(api('v1.0/logos/save'), {  
 method: 'POST',  
 headers: { Authorization: `Bearer ${await AuthUjwtHelper.getToken()}` },  
 body: logo  
 })  
}

## **Conclusion**

This API documentation provides comprehensive coverage of all frontend API interactions within the Logo Builder application. The standardized request/response formats, consistent error handling, and detailed examples enable efficient development and integration with the backend services.

The modular client architecture and authentication patterns ensure secure, scalable communication between the frontend and backend systems while maintaining a clean separation of concerns.