TechCorp Customer API v3.2 - Integration Guide

API Version: 3.2.1

Documentation Version: 1.8

Last Updated: February 2025

Maintained By: Platform Engineering Team

Table of Contents

---

1. Introduction

The TechCorp Customer API allows partners and internal systems to programmatically interact with customer data, orders, and analytics. This RESTful API uses JSON for request and response payloads.

1.1 Key Features

- Customer profile management (CRUD operations)
- Order creation and status tracking
- Real-time inventory checks
- Analytics and reporting data
- Webhook notifications for events

1.2 API Changelog

- **v3.2.1** (Feb 2025): Added support for bulk customer updates
- **v3.2.0** (Jan 2025): Introduced webhook retry logic
- **v3.1.0** (Nov 2024): Enhanced search filters for orders
- **v3.0.0** (Aug 2024): Major version - deprecated v2 endpoints

---

2. Authentication

2.1 API Keys

All API requests require authentication using API keys passed in the request header.

Header Format:

```
Authorization: Bearer YOUR_API_KEY
X-API-Version: 3.2
```

Obtaining API Keys:

1. Log into the TechCorp Developer Portal: https://developers.techcorp.com (https://developers.techcorp.com)
2. Navigate to "API Credentials"
3. Click "Generate New Key"
4. Select appropriate scopes (read, write, admin)
5. Store the key securely - it will only be shown once

Key Types:

- **Production Keys:** Rate limit of 10,000 requests/hour
- **Sandbox Keys:** Rate limit of 1,000 requests/hour, test data only
- **Internal Keys:** No rate limit, IP-restricted

2.2 OAuth 2.0 (Enterprise Only)

For enterprise customers requiring user-specific authentication:

Authorization Flow:

```
1. Redirect user to: https://auth.techcorp.com/oauth/authorize
   Parameters: client_id, redirect_uri, scope, state

2. User approves access

3. Redirect back with authorization code

4. Exchange code for access token:
   POST https://auth.techcorp.com/oauth/token
   Body: {
     "grant_type": "authorization_code",
     "code": "AUTH_CODE",
     "client_id": "YOUR_CLIENT_ID",
     "client_secret": "YOUR_CLIENT_SECRET",
     "redirect_uri": "YOUR_REDIRECT_URI"
   }

5. Use access token in API requests:
   Authorization: Bearer ACCESS_TOKEN
```

Token Expiration:

- Access tokens: 1 hour
- Refresh tokens: 30 days
- Use refresh token to obtain new access token without re-authentication

---

3. Base URLs

3.1 Environments

Environment	Base URL	Purpose
Production	`https://api.techcorp.com/v3`	Live customer data
Sandbox	`https://sandbox-api.techcorp.com/v3`	Testing and development
Staging	`https://staging-api.techcorp.com/v3`	Pre-production validation

3.2 Regional Endpoints

For improved latency, use regional endpoints:

- **US East:** `https://us-east.api.techcorp.com/v3`
- **EU West:** `https://eu-west.api.techcorp.com/v3`
- **Asia Pacific:** `https://apac.api.techcorp.com/v3`

---

4. Rate Limiting

4.1 Default Limits

Account Type	Requests/Hour	Burst Limit
Free Tier	1,000	50/min
Professional	10,000	200/min
Enterprise	100,000	1,000/min
Internal	Unlimited	2,000/min

4.2 Rate Limit Headers

Every response includes rate limit information:

```
X-RateLimit-Limit: 10000
X-RateLimit-Remaining: 9847
X-RateLimit-Reset: 1706889600
```

4.3 Handling Rate Limit Errors

When rate limit is exceeded (HTTP 429):

```
{
  "error": {
    "code": "rate_limit_exceeded",
    "message": "API rate limit exceeded. Try again in 847 seconds.",
    "retry_after": 847
  }
}
```

Best Practice: Implement exponential backoff:

1. Wait for `retry_after` seconds
2. If not specified, wait 60 seconds
3. Retry request
4. If fails again, double wait time up to 15 minutes max

---

5. Core Endpoints

5.1 Customers

5.1.1 Get Customer by ID

```
GET /customers/{customer_id}
```

Parameters:

- `customer_id` (path, required): UUID of the customer

Response 200 OK:

```
{
  "id": "cust_1a2b3c4d5e6f",
  "email": "john.doe@example.com",
  "first_name": "John",
  "last_name": "Doe",
  "phone": "+1-555-0123",
  "status": "active",
  "created_at": "2024-01-15T10:30:00Z",
  "updated_at": "2025-02-01T14:22:00Z",
  "metadata": {
    "tier": "premium",
    "source": "web_signup"
  },
  "addresses": [
    {
      "id": "addr_9z8y7x6w",
      "type": "shipping",
      "street": "123 Main St",
      "city": "San Francisco",
      "state": "CA",
      "postal_code": "94102",
      "country": "US",
      "is_default": true
    }
  ],
  "lifetime_value": 15420.50
}
```

5.1.2 Create Customer

```
POST /customers
```

Request Body:

```
{
  "email": "jane.smith@example.com",
  "first_name": "Jane",
  "last_name": "Smith",
  "phone": "+1-555-0199",
  "metadata": {
    "tier": "standard",
    "source": "mobile_app"
  }
}
```

Response 201 Created:

```
{
  "id": "cust_7g8h9i0j1k2l",
  "email": "jane.smith@example.com",
  "first_name": "Jane",
  "last_name": "Smith",
  "phone": "+1-555-0199",
  "status": "active",
  "created_at": "2025-02-12T08:15:30Z",
  "updated_at": "2025-02-12T08:15:30Z",
  "metadata": {
    "tier": "standard",
    "source": "mobile_app"
  },
  "addresses": [],
  "lifetime_value": 0.00
}
```

5.1.3 Update Customer

```
PATCH /customers/{customer_id}
```

Request Body (partial updates allowed):

```
{
  "first_name": "Jane",
  "metadata": {
    "tier": "premium"
  }
}
```

5.1.4 Search Customers

```
GET /customers/search
```

Query Parameters:

- `q` (string): Search query (searches name, email)
- `status` (string): Filter by status (active, inactive, suspended)
- `tier` (string): Filter by tier from metadata
- `created_after` (datetime): ISO 8601 format
- `created_before` (datetime): ISO 8601 format
- `limit` (integer): Results per page (max 100, default 20)
- `offset` (integer): Pagination offset

Example:

```
GET /customers/search?q=john&status=active&limit=50
```

5.2 Orders

5.2.1 Create Order

```
POST /orders
```

Request Body:

```
{
  "customer_id": "cust_1a2b3c4d5e6f",
  "items": [
    {
      "product_id": "prod_abc123",
      "quantity": 2,
      "unit_price": 49.99
    },
    {
      "product_id": "prod_def456",
      "quantity": 1,
      "unit_price": 129.99
    }
  ],
  "shipping_address_id": "addr_9z8y7x6w",
  "payment_method": "credit_card",
  "shipping_method": "standard",
  "notes": "Gift wrap requested"
}
```

Response 201 Created:

```
{
  "id": "order_x1y2z3a4b5",
  "customer_id": "cust_1a2b3c4d5e6f",
  "status": "pending",
  "subtotal": 229.97,
  "tax": 20.70,
  "shipping": 8.99,
  "total": 259.66,
  "currency": "USD",
  "items": [
    {
      "id": "item_111",
      "product_id": "prod_abc123",
      "product_name": "Widget Pro",
      "quantity": 2,
      "unit_price": 49.99,
      "total": 99.98
    },
    {
      "id": "item_222",
      "product_id": "prod_def456",
      "product_name": "Gadget Plus",
      "quantity": 1,
      "unit_price": 129.99,
      "total": 129.99
    }
  ],
  "created_at": "2025-02-12T09:30:00Z",
  "updated_at": "2025-02-12T09:30:00Z",
  "estimated_delivery": "2025-02-18T00:00:00Z"
}
```

5.2.2 Get Order Status

```
GET /orders/{order_id}
```

Response 200 OK:

```
{
  "id": "order_x1y2z3a4b5",
  "status": "shipped",
  "tracking_number": "1Z999AA10123456784",
  "carrier": "UPS",
  "shipped_at": "2025-02-13T14:20:00Z",
  "estimated_delivery": "2025-02-18T00:00:00Z"
}
```

Order Status Values:

- `pending`: Order created, awaiting payment
- `processing`: Payment confirmed, preparing shipment
- `shipped`: Order dispatched to carrier
- `delivered`: Order successfully delivered
- `cancelled`: Order cancelled by customer or system
- `refunded`: Payment refunded to customer

5.3 Inventory

5.3.1 Check Product Availability

```
GET /inventory/{product_id}
```

Response 200 OK:

```
{
  "product_id": "prod_abc123",
  "sku": "WGT-PRO-001",
  "available_quantity": 1247,
  "reserved_quantity": 53,
  "warehouses": [
    {
      "location": "US-WEST-1",
      "quantity": 842
    },
    {
      "location": "US-EAST-1",
      "quantity": 405
    }
  ],
  "restock_date": "2025-02-20T00:00:00Z",
  "status": "in_stock"
}
```

5.3.2 Bulk Inventory Check

```
POST /inventory/bulk
```

Request Body:

```
{
  "product_ids": ["prod_abc123", "prod_def456", "prod_ghi789"]
}
```

---

6. Error Handling

6.1 Error Response Format

All errors follow this structure:

```
{
  "error": {
    "code": "validation_error",
    "message": "Invalid email format",
    "details": [
      {
        "field": "email",
        "issue": "Must be a valid email address"
      }
    ],
    "request_id": "req_abc123xyz789"
  }
}
```

6.2 HTTP Status Codes

Code	Meaning	Common Causes
200	Success	Request completed successfully
201	Created	Resource created successfully
400	Bad Request	Invalid parameters or request body
401	Unauthorized	Missing or invalid API key
403	Forbidden	API key lacks required permissions
404	Not Found	Resource doesn't exist
409	Conflict	Resource already exists (duplicate)
422	Unprocessable	Validation failed
429	Rate Limited	Too many requests
500	Server Error	Internal server error
503	Unavailable	Service temporarily down

6.3 Common Error Codes

- `invalid_request`: Malformed request
- `authentication_failed`: Invalid credentials
- `insufficient_permissions`: API key lacks scope
- `resource_not_found`: Requested resource doesn't exist
- `validation_error`: Request data failed validation
- `rate_limit_exceeded`: Too many requests
- `duplicate_resource`: Resource already exists
- `internal_error`: Unexpected server error

---

7. Webhooks

7.1 Webhook Events

Subscribe to real-time events:

Event	Description
`customer.created`	New customer registered
`customer.updated`	Customer information changed
`customer.deleted`	Customer account deleted
`order.created`	New order placed
`order.updated`	Order status changed
`order.shipped`	Order dispatched
`order.delivered`	Order delivered
`order.cancelled`	Order cancelled
`payment.succeeded`	Payment processed successfully
`payment.failed`	Payment failed

7.2 Webhook Configuration

Configure webhooks in the Developer Portal:

1. Navigate to "Webhooks"
2. Click "Add Endpoint"
3. Enter your webhook URL (must be HTTPS)
4. Select events to subscribe to
5. Save and note your webhook secret

7.3 Webhook Payload

```
{
  "id": "evt_1a2b3c",
  "type": "order.shipped",
  "created_at": "2025-02-12T10:15:30Z",
  "data": {
    "object": {
      "id": "order_x1y2z3a4b5",
      "customer_id": "cust_1a2b3c4d5e6f",
      "status": "shipped",
      "tracking_number": "1Z999AA10123456784",
      "carrier": "UPS"
    }
  }
}
```

7.4 Webhook Signature Verification

Verify webhooks using HMAC SHA256:

```
import hmac
import hashlib

def verify_webhook(payload, signature, secret):
    expected = hmac.new(
        secret.encode('utf-8'),
        payload.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()
    return hmac.compare_digest(expected, signature)

# Usage
webhook_secret = "whsec_abc123xyz789"
received_signature = request.headers.get('X-Webhook-Signature')
is_valid = verify_webhook(request.body, received_signature, webhook_secret)
```

7.5 Retry Logic

If webhook delivery fails:

- Retry after 1 minute
- Retry after 5 minutes
- Retry after 30 minutes
- Retry after 2 hours
- Retry after 12 hours
- Endpoint disabled after 5 failures

---

8. Best Practices

8.1 Idempotency

Use idempotency keys for safe retries on POST/PATCH requests:

```
POST /orders
Idempotency-Key: order-20250212-user123-001
```

If request is retried with same key, existing resource is returned instead of creating duplicate.

8.2 Pagination

For large result sets, use cursor-based pagination:

```
{
  "data": [...],
  "pagination": {
    "next_cursor": "cursor_abc123",
    "has_more": true
  }
}
```

Next request:

```
GET /customers?cursor=cursor_abc123&limit=50
```

8.3 Filtering and Field Selection

Reduce payload size by requesting only needed fields:

```
GET /customers/{id}?fields=id,email,first_name,last_name
```

8.4 Caching

Respect cache headers:

```
Cache-Control: max-age=300, must-revalidate
ETag: "33a64df551425fcc55e4d42a148795d9"
```

Use conditional requests:

```
GET /customers/{id}
If-None-Match: "33a64df551425fcc55e4d42a148795d9"
```

Response: 304 Not Modified (if unchanged)

8.5 Error Handling Example

```
import requests
import time

def make_api_request(url, max_retries=3):
    for attempt in range(max_retries):
        try:
            response = requests.get(
                url,
                headers={"Authorization": f"Bearer {API_KEY}"},
                timeout=30
            )

            if response.status_code == 200:
                return response.json()

            elif response.status_code == 429:
                retry_after = int(response.headers.get('Retry-After', 60))
                time.sleep(retry_after)
                continue

            elif response.status_code >= 500:
                time.sleep(2 ** attempt)  # Exponential backoff
                continue

            else:
                response.raise_for_status()

        except requests.exceptions.Timeout:
            if attempt == max_retries - 1:
                raise
            time.sleep(2 ** attempt)

    raise Exception("Max retries exceeded")
```

---

9. Support and Resources

Developer Portal: https://developers.techcorp.com (https://developers.techcorp.com)

API Status: https://status.techcorp.com (https://status.techcorp.com)

Support Email: api-support@techcorp.com

Response Time: < 24 hours for standard, < 4 hours for enterprise

Slack Community: #techcorp-api-developers

GitHub Examples: https://github.com/techcorp/api-examples (https://github.com/techcorp/api-examples)

Document Version History:

- v1.8 (Feb 2025): Added bulk operations documentation
- v1.7 (Jan 2025): Updated webhook retry logic
- v1.6 (Nov 2024): Added OAuth 2.0 section
- v1.5 (Aug 2024): Initial v3 API documentation