# readme.md for @apiclient.xyz/namecheap

A comprehensive TypeScript client for the Namecheap API, providing a clean, type-safe interface for domain management, DNS configuration, and domain transfers.

# Features

- **Complete API Coverage**: Supports all major Namecheap API endpoints
- **Type Safety**: Full TypeScript definitions for all API requests and responses
- **Error Handling**: Comprehensive error handling with detailed error messages
- **Sandbox Support**: Test your integration with the Namecheap sandbox environment
- **Modular Design**: Organized into logical modules for different API functionalities

# Installation

```
# Using npm
npm install @apiclient.xyz/namecheap

# Using yarn
yarn add @apiclient.xyz/namecheap

# Using pnpm
pnpm add @apiclient.xyz/namecheap
```

# Quick Start

```
import { NamecheapClient } from '@apiclient.xyz/namecheap';

// Create a new client instance
```

```
const client = new NamecheapClient({
  apiUser: 'your-api-username',
  apiKey: 'your-api-key',
  userName: 'your-username', // Often the same as apiUser
  clientIp: 'your-ip-address'
}, true); // true for sandbox mode, false for production

// Check if a domain is available
const availability = await client.domains.check('example.com');
console.log(`Domain is ${availability[0].available ? 'available' : 'not available'}`);

// Get a list of domains in your account
const domainList = await client.domains.getList();
console.log(`You have ${domainList.domains.length} domains`);
```

# API Documentation

## Client Initialization

```
// Create a client for production use
const productionClient = new NamecheapClient({
  apiUser: 'your-api-username',
  apiKey: 'your-api-key',
  userName: 'your-username',
  clientIp: 'your-ip-address'
});

// Create a client for sandbox testing
const sandboxClient = new NamecheapClient({
  apiUser: 'your-api-username',
  apiKey: 'your-api-key',
  userName: 'your-username',
  clientIp: 'your-ip-address'
}, true);

// Switch between sandbox and production
client.enableSandbox(); // Switch to sandbox
```

```
client.disableSandbox(); // Switch to production

// Set request timeout (in milliseconds)
client.setTimeout(30000); // 30 seconds
```

# Domain Management

## Check Domain Availability

```
// Check a single domain
const singleResult = await client.domains.check('example.com');
console.log(`Domain is ${singleResult[0].available ? 'available' : 'not available'}`);

// Check multiple domains
const multipleResults = await client.domains.check(['example.com', 'example.org',
'example.net']);
multipleResults.forEach(result => {
  console.log(`${result.domain} is ${result.available ? 'available' : 'not available'}`);

  if (result.isPremium) {
    console.log(`Premium domain: Registration price: ${result.premiumRegistrationPrice}`);
  }
});
```

## Get Domain List

```
// Get all domains
const allDomains = await client.domains.getList();

// Get with pagination
const page2 = await client.domains.getList({
  Page: 2,
  PageSize: 20
});

// Filter domains
const expiringDomains = await client.domains.getList({
  ListType: 'EXPIRING'
});
```

```
// Search domains
const searchResults = await client.domains.getList({
  SearchTerm: 'example'
});

// Sort domains
const sortedDomains = await client.domains.getList({
  SortBy: 'EXPIREDATE_DESC'
});
```

## Get Domain Information

```
// Get detailed information about a domain
const domainInfo = await client.domains.getInfo('example.com');

console.log(`Domain: ${domainInfo.domainName}`);
console.log(`Created: ${domainInfo.createdDate}`);
console.log(`Expires: ${domainInfo.expiredDate}`);
console.log(`WhoisGuard: ${domainInfo.whoisGuard.enabled ? 'Enabled' : 'Disabled'}`);
```

## Domain Contact Management

```
// Get contact information for a domain
const contacts = await client.domains.getContacts('example.com');

console.log('Registrant:', contacts.registrant);
console.log('Technical Contact:', contacts.tech);
console.log('Admin Contact:', contacts.admin);
console.log('Billing Contact:', contacts.auxBilling);

// Update contact information
const updatedContacts = {
  registrant: {
    FirstName: 'John',
    LastName: 'Doe',
    Address1: '123 Main St',
    City: 'Anytown',
    StateProvince: 'CA',
    PostalCode: '12345',
```

```
    Country: 'US',
    Phone: '+1.5555555555',
    EmailAddress: 'john.doe@example.com'
  },
  // You can update any or all contact types
  tech: { /* ... */ },
  admin: { /* ... */ },
  auxBilling: { /* ... */ }
};

const success = await client.domains.setContacts('example.com', updatedContacts);
```

## Register a Domain

```
// Register a new domain
const registrationResult = await client.domains.create({
  domainName: 'example.com',
  years: 1,
  contacts: {
    registrant: {
      FirstName: 'John',
      LastName: 'Doe',
      Address1: '123 Main St',
      City: 'Anytown',
      StateProvince: 'CA',
      PostalCode: '12345',
      Country: 'US',
      Phone: '+1.5555555555',
      EmailAddress: 'john.doe@example.com'
    },
    tech: { /* Same structure as registrant */ },
    admin: { /* Same structure as registrant */ },
    auxBilling: { /* Same structure as registrant */ }
  },
  nameservers: ['dns1.namecheaphosting.com', 'dns2.namecheaphosting.com'],
  addFreeWhoisguard: true,
  whoisguardPrivacy: true
});

console.log(`Domain registered: ${registrationResult.domain}`);
```

```
console.log(`Order ID: ${registrationResult.orderId}`);
```

## Renew a Domain

```
// Renew a domain registration
const renewalResult = await client.domains.renew('example.com', 1);

console.log(`Domain renewed: ${renewalResult.domainName}`);
console.log(`New expiry date: ${renewalResult.expireDate}`);
```

## Reactivate an Expired Domain

```
// Reactivate an expired domain
const reactivationResult = await client.domains.reactivate('example.com');

console.log(`Domain reactivated: ${reactivationResult.domain}`);
console.log(`Order ID: ${reactivationResult.orderId}`);
```

## Registrar Lock Management

```
// Get the registrar lock status
const isLocked = await client.domains.getRegistrarLock('example.com');
console.log(`Domain is ${isLocked ? 'locked' : 'unlocked'}`);

// Set the registrar lock status
await client.domains.setRegistrarLock('example.com', true); // Lock the domain
await client.domains.setRegistrarLock('example.com', false); // Unlock the domain
```

## Get Available TLDs

```
// Get a list of available TLDs
const tlds = await client.domains.getTldList();
console.log(`Available TLDs: ${tlds.join(', ')}`);
```

# DNS Management

## Get DNS Host Records

```
// Get all DNS records for a domain
const hostRecords = await client.dns.getHosts('example.com');

hostRecords.forEach(record => {
  console.log(`${record.name} (${record.type}): ${record.address} (TTL: ${record.ttl})`);
});
```

## Set DNS Host Records

```
// Set DNS records for a domain
const newRecords = [
  {
    hostName: '@',
    recordType: 'A',
    address: '192.0.2.1',
    ttl: 300
  },
  {
    hostName: 'www',
    recordType: 'CNAME',
    address: '@',
    ttl: 300
  },
  {
    hostName: 'mail',
    recordType: 'MX',
    address: 'mail.example.com',
    mxPref: 10,
    ttl: 300
  }
];

const success = await client.dns.setHosts('example.com', newRecords);
```

## Set Custom Nameservers

```
// Set custom nameservers for a domain
const nameservers = [
  'ns1.example.com',
```

```
  'ns2.example.com'
];

const success = await client.dns.setCustom('example.com', nameservers);
```

## Email Forwarding

```
// Get email forwarding settings
const forwardings = await client.dns.getEmailForwarding('example.com');

forwardings.forEach(forward => {
  console.log(`${forward.from}@example.com → ${forward.to}`);
});

// Set email forwarding
const newForwardings = [
  {
    from: 'info',
    to: 'your-email@gmail.com'
  },
  {
    from: 'sales',
    to: 'sales@company.com'
  }
];

const success = await client.dns.setEmailForwarding('example.com', newForwardings);
```

## Get DNS Servers

```
// Get a list of DNS servers for a domain
const dnsServers = await client.dns.getList('example', 'com');
console.log(`DNS Servers: ${dnsServers.join(', ')}`);
```

# Nameserver Management

## Create a Nameserver

```
// Create a new nameserver
const success = await client.ns.create(
  'example', // SLD (Second-Level Domain)
  'com',     // TLD (Top-Level Domain)
  'ns1.example.com', // Nameserver hostname
  '192.0.2.1' // IP address
);
```

## Delete a Nameserver

```
// Delete a nameserver
const success = await client.ns.delete(
  'example', // SLD
  'com',     // TLD
  'ns1.example.com' // Nameserver hostname
);
```

## Get Nameserver Information

```
// Get information about a nameserver
const nsInfo = await client.ns.getInfo(
  'example', // SLD
  'com',     // TLD
  'ns1.example.com' // Nameserver hostname
);

console.log(`Nameserver: ${nsInfo.nameserver}`);
console.log(`IP: ${nsInfo.ip}`);
console.log(`Statuses: ${nsInfo.statuses.join(', ')}`);
```

## Update a Nameserver

```
// Update a nameserver's IP address
const success = await client.ns.update(
  'example', // SLD
  'com',     // TLD
  'ns1.example.com', // Nameserver hostname
  '192.0.2.1', // Old IP address
  '192.0.2.2'  // New IP address
```

```
);
```

# Domain Transfers

## Get Transfer List

```
// Get a list of domain transfers
const transfers = await client.transfer.getList();

transfers.transfers.forEach(transfer => {
  console.log(`${transfer.domainName} (Status: ${transfer.status})`);
});

// With pagination
const page2 = await client.transfer.getList(2, 20);

// Filter by status
const inProgress = await client.transfer.getList(1, 20, 'INPROGRESS');
```

## Get Transfer Status

```
// Get the status of a specific transfer
const status = await client.transfer.getStatus(12345); // Transfer ID

console.log(`Domain: ${status.domainName}`);
console.log(`Status: ${status.status}`);
console.log(`Description: ${status.statusDescription}`);
```

## Create a Transfer

```
// Initiate a domain transfer to Namecheap
const transferResult = await client.transfer.create(
  'example.com', // Domain name
  1, // Number of years to renew for
  'AUTH_CODE', // Authorization code from current registrar
  {
    addFreeWhoisguard: true,
    whoisguardEnable: true
  }
```

```
);

console.log(`Transfer initiated: ${transferResult.transferId}`);
console.log(`Status: ${transferResult.transferStatus}`);
```

## Update Transfer Status

```
// Update the status of a transfer (e.g., to resubmit)
const success = await client.transfer.updateStatus(12345, true); // Transfer ID, resubmit flag
```

## Get Transfer Information

```
// Get detailed information about a transfer
const transferInfo = await client.transfer.getInfo(12345); // Transfer ID

console.log(`Domain: ${transferInfo.domainName}`);
console.log(`Status: ${transferInfo.status}`);
console.log(`Order Date: ${transferInfo.orderDate}`);
console.log(`WhoisGuard Status: ${transferInfo.whoisguardStatus}`);
```

# Error Handling

The client provides detailed error messages for API errors:

```
try {
  await client.domains.getInfo('example.com');
} catch (error) {
  console.error('API Error:', error.message);

  // For more detailed error information
  if (error.errors) {
    error.errors.forEach(err => console.error(' - ', err));
  }
}
```

# Sandbox Testing

Namecheap provides a sandbox environment for testing API integrations. To use it:

1. Register for a sandbox account at https://www.sandbox.namecheap.com/
2. Enable API access in your sandbox account
3. Get your API key from the profile settings
4. Whitelist your IP address in the API settings
5. Create a client with sandbox mode enabled:

```
const client = new NamecheapClient({
  apiUser: 'your-sandbox-username',
  apiKey: 'your-sandbox-api-key',
  userName: 'your-sandbox-username',
  clientIp: 'your-whitelisted-ip'
}, true); // true enables sandbox mode
```

# License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

**Please note:** The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

# Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

# Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

---

Revision #6
Created 2026-03-28 10:48:21 UTC by foss.global Team
Updated 2026-03-28 12:14:15 UTC by foss.global Team