RESELLER DEVELOPER DOCUMENTATION

Domains API

Integration guide and examples

A complete, tested guide to connecting your own systems to the reseller.co.tz Domains API: check availability, register, renew, transfer, update contacts and nameservers, and protect domains with a transfer lock or Whois privacy.

Audience	Reseller partners and their developers or system integrators
Product	Domains API v1 (REST, JSON over HTTPS)
Base URL	`https://reseller.co.tz/api/v1`
Live reference	https://reseller.co.tz/api-docs
Version	1.0
Last updated	25 October 2026

Scope of this guide

Covers domain lifecycle and management only: availability, registration, renewal, transfers, contacts/Whois, nameservers and child nameservers, transfer lock, Whois privacy and Auth/EPP codes. DNS record management (A, AAAA, CNAME, MX, TXT and similar records) is **not** part of this guide and is not required for the workflows described here. SMS sending uses a separate API and token.

Contents

No.	Section	What it covers
1	About this guide	Who it is for, what it covers and how to use it
2	API at a glance	Capabilities, base URL, integration boundary
3	Before you integrate	Access checklist, API key, IP whitelist, data model
4	Request and response conventions	Headers, envelope, identifiers, prices, errors, retries
5	Data reference	Domain object, contact object, status and price fields
6	Integration examples	cURL, PHP/Laravel, Node.js and plain PHP clients
7	Endpoint reference	Every domain endpoint with parameters and examples
8	Workflow guidance	Step-by-step registration, renewal, transfer, contacts and protection
9	Testing and go-live	Test cases and production launch checklist
10	Troubleshooting and support	Common errors and what to send to support
11	Document information	Versioning, maintenance and sharing

Tip: the live reference at reseller.co.tz/api-docs is updated with every API release and is the authority for parameter lists. This guide adds the integration guidance, end-to-end examples and operational rules that your team needs to plan, build and launch.

1 About this guide

Use this guide to connect a reseller system to the reseller.co.tz Domains API through a secure backend service. It is written for developers and technical partners who will sell, renew and manage domains through your account.

1.1 What you can build

- **Domain shop or checkout** inside your own website, application or Point of Sale system.
- **Client or membership portal** where each customer sees, renews and updates their own domains.
- **Billing and invoicing integration** that renews domains automatically before expiry.
- **Bulk operations** for agencies managing many domains across many customers.
- **Accounting and CRM sync** that holds one record per domain next to your customer record.

1.2 What this guide covers

Domain search and availability, registration and re-registration, renewal, inbound transfers, registrant and admin contacts (Whois), nameservers and child nameservers, transfer lock, Whois privacy and Auth/EPP codes.

Not covered: DNS record management (creating or editing A, AAAA, CNAME, MX, TXT, SRV and similar records), hosting, email and SMS. If you sell hosting or email elsewhere, point the domain at your nameservers (section 8.6) and manage DNS in that system.

1.3 How to use it

1. Read sections 2 to 5 to understand the platform, access setup and the data you will store.
2. Copy the examples in section 6 to get a first successful request working in your stack.
3. Use section 7 as the endpoint reference while you build each screen or task.
4. Follow section 8 for the business rules that prevent duplicate charges and confused customers.
5. Complete the launch checklist in section 9 before switching on real customers.

2 API at a glance

2.1 Capabilities

Area	What your system can do
Find and inspect	Check availability and per-extension pricing; list all your domains with search and filters; read a single domain.
Register and renew	Register a new domain, re-register a cancelled one, and renew for 1 to 10 years.
Transfer	Bring an existing domain in with its Auth/EPP code and track it until it completes.
Contacts and Whois	Read and update the registrant and admin contacts held at the registry.
Nameservers	Point a domain at your DNS provider, including child nameservers (glue) for country extensions that support them.
Protection	Enable or disable the transfer lock, manage Whois privacy, and request or set an Auth/EPP code where the extension supports it.

2.2 Base URL and transport

Item	Value
Base URL	https://reseller.co.tz/api/v1
Protocol	HTTPS only. TLS is required on every request.
Request body	JSON (`Content-Type: application/json`) for POST and PUT requests.
Response body	JSON. Always send `Accept: application/json` so errors come back as JSON.
Authentication	`X-API-KEY` header, plus a whitelisted source IP address.
Currency	All prices are whole Tanzanian Shillings (TZS) unless stated otherwise.
Versioning	The path carries the version (`/api/v1`). Breaking changes will be introduced under a new version.

2.3 Integration boundary

Call the API from a server, never from the browser

Your API key and IP whitelist belong to a server-side backend. Do not call the API directly from browser JavaScript, a mobile app bundle, or any code a customer can inspect. Browser and mobile clients should call **your** backend, and your backend calls reseller.co.tz.

Good fit	Do not do this
A backend service with fixed outbound IP addresses	Browser JavaScript holding the API key
A queue or worker that retries and reconciles safely	Retrying a billable call the moment it times out
Server-side storage of the API key in a secret manager	Storing the key in source control, chat or a public page
General, customer-friendly error messages	Showing registry or provider raw error text to customers

3 Before you integrate

Most first-time failures come from account access, a missing IP whitelist entry or an incomplete contact object. Complete these checks before you write production code.

3.1 Access checklist

Check	Expected state	Where to manage it
Reseller account	Active reseller account with a positive balance for billable actions.	reseller.co.tz sign-in
API key	A key is assigned and shown on the API Access page (masked preview visible).	API Access page
IP whitelist	At least one entry. Requests from other IPs are rejected with 403.	API Access page
Blocked IP list	Does not contain your server's outbound IP.	API Access page
Prices	Prices exist for every extension you plan to sell.	Domain prices in your portal
Contact data	You have valid registrant details for each customer.	Your own system

Portal path: sign in at reseller.co.tz, then open **API Access** from the menu (or go to https://reseller.co.tz/api-access). The page shows the base URL, your API key and the IP whitelist.

3.2 Create and protect the API key

- Use **one key per production system** so a credential can be rotated without disrupting unrelated systems. If you run several brands, request a separate reseller account or coordinate rotation windows.
- Store the key in an environment variable, vault or secret manager. It must never appear in a repository, a client-side bundle, a spreadsheet or an email.
- Never log the `X-API-KEY` header, Auth/EPP codes, or full request bodies that contain customer contact data.
- The key is shown in full only on the API Access page. Use the masked preview to confirm which key is active.
- **Regenerating the key invalidates the old one immediately.** Every connected system stops working until it is updated. Plan the change: deploy the new key, verify a safe read request, then retire the old values from your configuration.
- If you suspect a key has leaked, regenerate it immediately, review your server logs and rotate your stored value.

3.3 Configure the IP whitelist

Every request is checked against the source IP address after the API key is verified. A valid key from an IP that is not whitelisted returns **403**.

- Whitelist your backend's **actual outbound public IP**, not your office IP, a customer IP or a browser's IP.
- Matching is exact. Add every address your service can use; there are no wildcard ranges.
- Behind a load balancer, container platform or serverless function, confirm the outbound IP with your host. Many platforms share an egress pool, so add each address.
- The **blocked IP** list is checked before the whitelist. If a request is blocked, check both lists.
- Removing an IP takes effect on the next request. Coordinate removals with deployment windows.
- If your outbound IP can change (for example cloud autoscaling), move to fixed egress addresses, or plan a process to add new addresses before they are used.

How to confirm your server's outbound IP

Run a request to the API from your production server and read the 403 error if it appears, or ask your hosting provider for the fixed egress addresses. Add each one on the API Access page. Use a staging key for verification instead of the production key where possible.

3.4 Decide how your system will store domain state

The API is the source of truth for ownership, status, expiry and pricing. Your integration still needs its own order and reconciliation records so a timeout, a retry or a failed render cannot lose or double-charge an order.

Store	Why it matters
Your order ID and the customer ID	Ties every API action to your own invoice and customer record.
The action and amount you showed the customer	Lets you compare what the customer agreed with what was charged.
A request fingerprint (for example customer + domain + action + period)	Stops duplicate submission when a user double-clicks or a job retries.
The API `domain.id` after success	The stable identifier for future reads, updates and renewals.
The `transaction` result and charge amount	Reconciles your books with the wallet deduction.

Store	Why it matters
The last read of expiry, status and price	Drives renewal reminders and prevents surprise redemption charges.

Suggested order state model

State	Meaning	Allowed next state
pending	Customer submitted the order but nothing has been sent yet.	submitted, cancelled
submitted	Request sent to the API; the result is not yet known.	confirmed, failed, needs_review
confirmed	API returned success and a domain or change is recorded.	renewing (later), transferred, expired
failed	API returned a definite error such as 422 validation, low balance or unsupported extension.	pending (after correction)
needs_review	Timeout, 5xx or ambiguous result. Do not retry until reconciled.	confirmed, failed

4 Request and response conventions

Implement these rules once in a shared API client, then every endpoint behaves consistently.

4.1 Authentication headers

Header	Value	Required when
`X-API-KEY`	Your reseller domain API key.	Every request.
`Accept`	`application/json`.	Every request.
`Content-Type`	`application/json`.	Requests with a JSON body (POST, PUT).

The SMS API uses a separate bearer token. Do not mix it into the domain client.

4.2 Common JSON response

Success

```
{
  "success": true,
  "message": "Domain loaded successfully.",
  "data": {
    "domain": { "id": 123, "domain_name": "example.co.tz" }
  }
}
```

Error

```
{
  "success": false,
  "message": "The given data was invalid.",
  "errors": {
    "registrant.country": ["The registrant country is not a valid country."]
  }
}
```

Always branch on the HTTP status first, then on `success`. Treat `message` as text for your own team; show customers a short, friendly explanation and never print raw error text from the API.

4.3 Domain identifiers

Endpoints that act on an existing domain accept either the portal domain ID or the full domain name in the `{domain}` path segment:

```
GET /api/v1/domains/123
GET /api/v1/domains/example.co.tz
```

Prefer the portal ID after registration. A domain that is not owned by the authenticated reseller returns **404**, which prevents one reseller from reading or changing another reseller's domains.

4.4 Prices and currency

Prices are integer TZS amounts returned by the API. Read the current quote from the API before you charge a customer or show a renewal amount; do not hard-code prices.

- `GET /domains/{domain}` returns `renewal_price` and `redemption_price` so your confirmation screen shows the same amount the renewal will use.
- When `in_redemption` is `true`, the first renewed year uses `redemption_price` and any extra years use `renewal_price`.
- `in_redemption` is `true` only when the extension has a redemption price configured. When it is `false` or the price is `null`, the normal `renewal_price` applies.
- `POST /domains/check` returns `price.registration`, `price.renewal` and `price.transfer`. For a premium name, quote `price.registration`, which already reflects the premium first year.
- Transfer prices can be zero where the extension does not charge a transfer fee.
- Your reseller wallet is the funding source for billable actions. Top up from the portal before high-volume releases.

4.5 Rate limits

The API applies an account-level limit to every request and tighter limits to selected actions. The following applies when a limit is exceeded:

Response	What it means	What your client should do
429 Too Many Requests	You are sending more requests than the limit for this window.	Pause, queue the work and retry later with a backoff. Never retry in a tight loop.
403 Forbidden	The IP is not whitelisted or is blocked, or the account may not use the API.	Stop retrying. Fix configuration or contact support.
401 Unauthorized	The key is missing, invalid or revoked.	Stop retrying. Correct the key or regenerate it.

- Avoid parallel bursts for a single customer action.
- Cache read-only catalogue data such as prices instead of reading it on every page view.
- Run reconciliation and status polling in a background worker with a controlled interval (for example every 15 to 60 minutes), not on page load.
- Use `per_page` (up to 100) when listing domains instead of making many small requests.

4.6 Status codes

Code	Meaning	Recommended client behaviour

Code	Meaning	Recommended client behaviour
200	Request completed successfully.	Read `success`, `message` and `data`.
201	A resource was created or a billable action completed (for example registration).	Treat as success and store the returned domain.
401	The API key is missing, revoked or invalid.	Stop retries; correct the key or contact the account owner.
403	The key is valid but the calling IP is not whitelisted, or the account is not allowed to use the API.	Check the exact outbound IP and whitelist settings.
404	The domain or resource does not belong to this account (or does not exist).	Verify the identifier and ownership. Do not retry.
422	Validation failed, the balance is too low, the action is unsupported for this extension, or the registry could not complete the request.	Show a safe message; keep field errors server-side.
429	Too many requests.	Back off and retry later.
500	Unexpected server error.	Retry reads cautiously after a short delay. Reconcile billable actions before retrying.

4.7 Retry and idempotency rules

Billable actions are not idempotent

Registration, re-registration, renewal and transfer deduct from your wallet. A network timeout does **not** prove the action failed. A blind retry can create a duplicate charge or a second registry action.

Action type	Safe retry rule
Reads: list, get, whois, nameservers, lock, privacy	Safe to retry, with a short delay and backoff. Never retry 401, 403 or 404.
Availability check (`/domains/check`)	Not billable. Retry with backoff if the connection fails.
Registration and re-registration	Do not retry automatically. Read the domain first: if it is now owned by your account, treat it as successful.
Renewal	Do not retry automatically. Read the domain and compare the expiry date; if it advanced by the period you requested, treat it as successful.
Transfer	Do not resubmit while the domain shows `pending_transfer`. Poll `POST /domains/{domain}/sync` instead.
Whois, nameservers, lock, privacy, EPP	Not billable. Retry once after a short delay, then reconcile with a read.

Example retry guard

```
def needs_review_on_timeout(action):
    # Billable actions (register, renew, transfer) can succeed even when the
    # connection drops, so never resubmit them blindly.
    return action in {"register", "reregister", "renew", "transfer"}

def reconcile(order):
    domain = api.get_domain(order.domain_name)
    if order.action == "renew":
        return "confirmed" if domain["expiry_date"] > order.previous_expiry else "failed"
    if order.action in {"register", "reregister"}:
        return "confirmed" if domain["id"] == order.domain_id_after else "needs_review"
    return "needs_review"  # transfer and other ambiguous cases: wait for sync
```

5 Data reference

These structures appear across several endpoints. Model them as typed objects in your integration rather than loose maps.

5.1 Domain object

Returned by `GET /domains`, `GET /domains/{domain}`, `POST /domains/register` and the action endpoints.

Field	Type	Meaning
`id`	integer	Portal domain ID. Use it as the stable identifier.
`domain_name`	string	Full domain name in lower case, for example `example.co.tz`.
`status`	string	Lifecycle status: `active`, `expired`, `pending`, `pending_transfer` or `cancelled`.
`registry_status`	string or null	Latest registry status text. Treat as informational; branch on `status`.
`redemption_state`	string	`grace` (normal renewal) or `redemption` (restoration window).
`in_redemption`	boolean	`true` when the first renewed year costs the redemption price.
`renewal_price`	number or null	Price of one renewal year in TZS for this extension.
`redemption_price`	number or null	Restoration price for the first year, or `null` if not configured.
`registration_date`	date or null	Registration date (`YYYY-MM-DD`).
`expiry_date`	date or null	Expiry date (`YYYY-MM-DD`). Drives renewal reminders.
`lock_status`	boolean	Transfer lock state as last synced.
`privacy_status`	boolean	Whois privacy state as last synced.
`last_synced_at`	ISO 8601 or null	When the registry details were last refreshed.

Example domain object

```
{
  "id": 123,
  "domain_name": "example.co.tz",
  "status": "active",
  "registry_status": "active",
  "redemption_state": "grace",
  "in_redemption": false,
  "renewal_price": 20000,
  "redemption_price": null,
  "registration_date": "2026-05-28",
  "expiry_date": "2027-05-28",
  "lock_status": false,
  "privacy_status": false,
  "last_synced_at": "2026-05-28T12:00:00.000000Z"
}
```

5.2 Contact object

Used for the registrant and admin contacts on registration and Whois updates.

Field	Requirement
`name`	required string, max 120

Field	Requirement
`organization`	required string, max 120
`address1`	required string, max 255
`city`	required string, max 120
`state`	required string, max 120
`postcode`	required string, max 30
`country`	required ISO 3166-1 alpha-2 code. `TZ` = Tanzania, `ZA` = South Africa, `KE` = Kenya
`phone`	required string, max 40. Include the country code, for example `+255.754123456`
`email`	required valid email address, max 255

`address2` is not accepted as a separate field. Include a second address line inside `address1`.

Example contact object

```
{
  "name": "Jane Doe",
  "organization": "Jane Doe Ltd",
  "address1": "123 Registry Road",
  "city": "Dar es Salaam",
  "state": "Dar es Salaam",
  "postcode": "11101",
  "country": "TZ",
  "phone": "+255.754123456",
  "email": "jane@example.co.tz"
}
```

One contact for both roles

When your form collects a single contact, send `same_contact: true` and provide the `registrant` object. The API copies the registrant onto the admin contact. If you untick one form, send a complete `admin` object as well.

```
{ "same_contact": true, "registrant": { ... } }
```

5.3 Availability result

Returned by `POST /domains/check` inside `data`.

Field	Type	Meaning
`domain_name`	string	Normalised name that was checked.
`available`	boolean	`true` when the name can be ordered now.
`bookable`	boolean	Same as `available`: the name can be ordered at the returned price.
`reserved`	boolean	`true` when the registry reserves the name. It can never be registered.
`reserved_category`	string or null	Category of the reserved name when applicable.
`premium`	boolean	`true` for a registry premium name that is priced per name.
`premium_sellable`	boolean	`true` when the premium name can be ordered at a configured price.
`status`	string	`available`, `unavailable`, `reserved` or `premium`.
`message`	string	Customer-safe explanation for the result.
`price.registration`	number	First-year price to quote (premium price when applicable).

Field	Type	Meaning
`price.renewal`	number	Renewal price per year.
`price.transfer`	number	Transfer price; may be 0.

6 Integration examples

The examples use placeholder domains, contact data and an API key read from the server environment. Replace only the placeholder values in your own code.

6.1 Quick start: your first successful request

Work through these five steps in a test environment before wiring the API into customer flows.

1. Sign in to reseller.co.tz and open **API Access**. Copy the base URL and key, and add your server's outbound IP to the whitelist.
2. Store the key as `RESELLER_API_KEY` in your server environment and restart the service.
3. Send the availability request in section 6.2 from your backend. A 200 response with `success: true` proves connectivity, credentials and whitelist are correct.
4. Read one of your existing domains with `GET /domains` and compare the result with the portal.
5. Build the registration flow using section 8.1 and the endpoint pages in section 7.

6.2 cURL: a first request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/check" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"domain_name": "example.co.tz"}'
```

A successful response returns `success: true` and a `data` object with `available` and `price`. Replace `$RESELLER_API_KEY` with the value from your server environment. Section 8.1 shows the full registration sequence step by step.

6.3 PHP and Laravel backend client

This example provides one reusable client for your application. It reads the key from the environment, adds the required headers, and retries only safe reads.

```php
<?php

use Illuminate\Http\Client\ConnectionException;
use Illuminate\Support\Facades\Http;

final class ResellerDomainsApi
{
    private string $base;
    private string $key;

    public function __construct()
    {
        $this->base = rtrim((string) env('RESELLER_API_BASE_URL', 'https://reseller.co.tz/api/v1'), '/');
        $this->key = (string) env('RESELLER_API_KEY');
    }

    /** Never retry billable calls: a timeout does not prove the action failed. */
    private function request(string $method, string $path, array $payload = [], bool $retry = false)
    {
        $client = Http::baseUrl($this->base)
            ->withHeaders(['X-API-KEY' => $this->key, 'Accept' => 'application/json'])
            ->timeout(30);

        $client = $retry ? $client->retry(2, 500) : $client;

        return $client->send($method, $path, ['json' => $payload]);
    }

    public function checkAvailability(string $domain)
    {
        return $this->request('POST', '/domains/check', ['domain_name' => $domain], retry: true);
    }

    public function domain(string $identifier)
    {
        return $this->request('GET', '/domains/'.rawurlencode($identifier), retry: true);
    }

    public function register(array $order)
    {
        return $this->request('POST', '/domains/register', $order); // billable: no retry
    }

    public function renew(string $identifier, int $period = 1)
    {
        return $this->request('POST', '/domains/'.rawurlencode($identifier).'/renew', ['period' => $period]);
    }
}
```

Using the client

```
$api = new ResellerDomainsApi();

$check = $api->checkAvailability('example.co.tz');

if (! $check->successful() || ! ($check->json('data.available') ?? false)) {
    // Show the safe message and stop the order.
    return back()->withErrors(['domain' => $check->json('message', 'This name is not available.')]);
}

$order = [
    'domain_name'  => 'example.co.tz',
    'period'       => 1,
    'nameservers'  => ['ns1.example.com', 'ns2.example.com'],
    'same_contact' => true,
    'registrant'   => [
        'name' => 'Jane Doe', 'organization' => 'Jane Doe Ltd',
        'address1' => '123 Registry Road', 'city' => 'Dar es Salaam',
        'state' => 'Dar es Salaam', 'postcode' => '11101', 'country' => 'TZ',
        'phone' => '+255.754123456', 'email' => 'jane@example.co.tz',
    ],
];

try {
    $response = $api->register($order);
} catch (ConnectionException $e) {
    // Ambiguous result: mark for manual review, then read the domain to reconcile.
    report($e);
    return redirect()->route('orders.review', $orderId);
}

if ($response->status() === 201) {
    $domain = $response->json('data.domain');
    // Store $domain['id'], $domain['expiry_date'] and $response->json('data.amount_charged').
}

if ($response->status() === 422) {
    // Validation, low balance or registry rejection. Keep field errors server-side.
}
```

6.4 Node.js backend request

Node 18 and later include `fetch`. This example shows the same safe-retry policy.

```
const BASE_URL = process.env.RESELLER_API_BASE_URL || 'https://reseller.co.tz/api/v1';
const API_KEY = process.env.RESELLER_API_KEY;

async function callApi(method, path, body) {
  const res = await fetch(`${BASE_URL}${path}`, {
    method,
    headers: {
      'X-API-KEY': API_KEY,
      'Accept': 'application/json',
      ...(body ? { 'Content-Type': 'application/json' } : {}),
    },
    body: body ? JSON.stringify(body) : undefined,
    signal: AbortSignal.timeout(30000),
  });

  const json = await res.json().catch(() => ({}));
  return { status: res.status, ok: res.ok, ...json };
}

// Safe to retry: availability checks are never billable.
async function checkDomain(domainName) {
  for (let attempt = 0; attempt < 3; attempt++) {
    try {
      return await callApi('POST', '/domains/check', { domain_name: domainName });
    } catch (err) {
      if (attempt === 2) throw err;
      await new Promise((r) => setTimeout(r, 500 * 2 ** attempt));
    }
  }
}

// Billable: call once. On failure, read the domain and reconcile before retrying.
async function renewDomain(domainName, period = 1) {
  return callApi('POST', `/domains/${encodeURIComponent(domainName)}/renew`, { period });
}

export { checkDomain, renewDomain };
```

6.5 Plain PHP without a framework

If you use WordPress, a legacy PHP site or any server that is not Laravel, use cURL directly.

```php
<?php
function reseller_api(string $method, string $path, ?array $body = null): array
{
    $ch = curl_init('https://reseller.co.tz/api/v1'.$path);
    curl_setopt_array($ch, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_CUSTOMREQUEST  => $method,
        CURLOPT_TIMEOUT        => 30,
        CURLOPT_HTTPHEADER     => array_filter([
            'X-API-KEY: '.getenv('RESELLER_API_KEY'),
            'Accept: application/json',
            $body !== null ? 'Content-Type: application/json' : null,
        ]),
        CURLOPT_POSTFIELDS     => $body !== null ? json_encode($body) : null,
    ]);

    $raw    = curl_exec($ch);
    $status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    $error  = curl_error($ch);
    curl_close($ch);

    return ['status' => $status, 'error' => $error, 'data' => json_decode((string) $raw, true)];
}

$result = reseller_api('POST', '/domains/check', ['domain_name' => 'example.co.tz']);
if ($result['status'] === 200 && ($result['data']['data']['available'] ?? false)) {
    // Offer the registration at $result['data']['data']['price']['registration'].
}
```

6.6 Environment variables

```
RESELLER_API_BASE_URL=https://reseller.co.tz/api/v1
RESELLER_API_KEY=your-server-side-key   # from the API Access page
```

Keep the API key out of version control. Add the variable to your deployment platform's secret store.

7 Endpoint reference

Every domain endpoint in API version 1.0. Each entry lists the method, path, billing classification, parameters, implementation notes and a copy-ready example.

7.1 Endpoint index

Group	Method	Path	Billable
Discover	GET	`/domains`	No
Discover	GET	`/domains/{domain}`	No
Discover	POST	`/domains/check`	No
Register	POST	`/domains/register`	Yes
Register	POST	`/domains/{domain}/register`	Yes
Register	POST	`/domains/{domain}/renew`	Yes
Register	POST	`/domains/{domain}/sync`	No
Transfer	POST	`/domains/{domain}/transfer`	Yes
Contacts	GET	`/domains/{domain}/whois`	No
Contacts	POST	`/domains/{domain}/whois`	No
Nameservers	GET	`/domains/{domain}/nameservers`	No
Nameservers	POST	`/domains/{domain}/nameservers`	No
Nameservers	GET	`/domains/{domain}/child-nameservers`	No
Nameservers	POST	`/domains/{domain}/child-nameservers`	No
Security	POST	`/domains/{domain}/epp-code`	No
Security	PUT	`/domains/{domain}/epp-code`	No
Security	GET	`/domains/{domain}/lock`	No
Security	PUT	`/domains/{domain}/lock`	No
Security	GET	`/domains/{domain}/privacy`	No
Security	PUT	`/domains/{domain}/privacy`	No

7 Find and inspect domains

Search availability, list the domains you own and read a single domain record.

GET /domains

METHOD GET	BILLING No charge	SUCCESS 200

List the domains owned by your reseller account with search, status filter and pagination.

Parameters

Name	Location and requirement
`q`	query (optional): text matched against the domain name
`status`	query (optional): `active`, `expired`, `pending` or `pending_transfer`. Omit or send `all` for every status
`per_page`	query (optional): integer 1-100, default 25
`page`	query (optional): page number

Implementation notes

- Only domains owned by the API key account are returned.
- `status=active` excludes expired domains. Use `expired` to find domains that need renewal.
- Results are ordered by expiry date, nearest first, so this endpoint also powers renewal reminders.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains?status=active&per_page=25" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "Domains loaded successfully.",
  "data": {
    "domains": [
      {
        "id": 123,
        "domain_name": "example.co.tz",
        "status": "active",
        "registry_status": "active",
        "redemption_state": "grace",
        "in_redemption": false,
        "renewal_price": 20000,
        "redemption_price": null,
        "registration_date": "2026-05-28",
        "expiry_date": "2027-05-28",
        "lock_status": false,
        "privacy_status": false,
        "last_synced_at": "2026-05-28T12:00:00.000000Z"
      }
    ],
    "pagination": {
      "current_page": 1,
      "per_page": 25,
      "total": 1,
      "last_page": 1
    }
  }
}
```


GET /domains/{domain}

METHOD GET	BILLING No charge	SUCCESS 200

Read one owned domain, including expiry, lock, privacy and the exact renewal price.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Implementation notes

- Use this before any renewal to display the same amount the API will charge.
- `in_redemption` is `true` when the registry has moved the domain into its restoration window and a redemption price exists.
- When `in_redemption` is `true`, show `redemption_price` for the first year; extra years use `renewal_price`.
- Returns 404 when the domain is not owned by your account.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.co.tz" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "Domain loaded successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.co.tz",
      "status": "active",
      "registry_status": "active",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2027-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    }
  }
}
```


POST /domains/check

METHOD	BILLING	SUCCESS
POST	**No charge**	**200**

Check availability and return catalogue pricing plus reserved, premium and bookable flags.

Body parameters

Field	Requirement
`domain_name`	required full domain name including the extension, max 253 characters

Implementation notes

- `available` and `bookable` are normally the same. `reserved` means the registry blocks the name.
- `premium` means the registry prices the name per name. Quote `price.registration` and only offer it when `available` is `true`.
- A domain already managed in the portal is reported as unavailable.
- This endpoint is not billable, but it does call the registry, so cache results briefly and avoid polling per keystroke.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/check" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"domain_name": "example.co.tz"}'
```

Example response


```
{
  "success": true,
  "message": "Domain availability checked successfully.",
  "data": {
    "domain_name": "example.co.tz",
    "status": "available",
    "available": true,
    "bookable": true,
    "reserved": false,
    "reserved_category": null,
    "premium": false,
    "premium_sellable": false,
    "price": {
      "registration": 20000,
      "renewal": 20000,
      "transfer": 0,
      "premium_registration": null,
      "premium_sellable": false
    }
  }
}
```

7 Register, re-register and renew

Billable actions that create ownership or extend an expiry date. Your wallet is charged only after the registry accepts the order.

POST /domains/register

METHOD POST	BILLING Billable	SUCCESS 201

Register a new domain, create the portal ownership record and charge the wallet after registry success.

Body parameters

Field	Requirement
`domain_name`	required full domain name that is not already in the portal
`period`	required integer 1-10 years
`nameservers`	required array of 2 to 10 hostnames
`registrant`	required contact object (section 5.2)
`admin`	required contact object, unless `same_contact` is `true`
`same_contact`	optional boolean; `true` copies the registrant onto the admin contact
`provision_dns`	optional boolean; leave out unless you use portal DNS management

Implementation notes

- The wallet balance is checked before the registry call and charged only after the registry confirms success.
- Some extensions have registry rules. For example, `.co.za` requires a South African (`ZA`) registrant country.
- Reserved names are always rejected. Premium names are rejected unless the extension has a sellable premium price.
- `same_contact` is mirrored server-side, so a ticked single-contact order is always complete.
- Non-idempotent: add your own order lock and reconciliation (section 4.7).

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/register" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"domain_name": "example.co.tz", "period": 1, "nameservers": ["ns1.example.com", "ns2.example.com"],
"same_contact": true, "registrant": {"name": "Jane Doe", "organization": "Jane Doe Ltd", "address1": "123
Registry Road", "city": "Dar es Salaam", "state": "Dar es Salaam", "postcode": "11101", "country": "TZ",
"phone": "+255.754123456", "email": "jane@example.co.tz"}}'
```

Example response


```
{
  "success": true,
  "message": "Domain registered successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.co.tz",
      "status": "active",
      "registry_status": "active",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2027-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    },
    "dns": null,
    "amount_charged": 20000
  }
}
```


POST /domains/{domain}/register

METHOD **POST**	BILLING **Billable**	SUCCESS **201**

Register an owned cancelled domain again, reusing the existing portal domain record.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name with `status = cancelled`

Body parameters

Field	Requirement
`period`	required integer 1-10 years
`nameservers`	required array of 2 to 10 hostnames
`registrant`	required contact object
`admin`	required contact object, unless `same_contact` is `true`
`same_contact`	optional boolean
`provision_dns`	optional boolean

Implementation notes

- Only a domain your account already owns with status `cancelled` can be re-registered.
- A cancelled domain is registered at the normal registration price, not the restoration price.
- Sending this for any other status returns 422.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/register" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"period": 1, "nameservers": ["ns1.example.com", "ns2.example.com"], "same_contact": true,
"registrant": {"name": "Jane Doe", "organization": "Jane Doe Ltd", "address1": "123 Registry Road", "city":
"Dar es Salaam", "state": "Dar es Salaam", "postcode": "11101", "country": "TZ", "phone": "+255.754123456",
"email": "jane@example.co.tz"}}'
```

Example response

```
{
  "success": true,
  "message": "Domain registered successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.co.tz",
      "status": "active",
      "registry_status": "active",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2027-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    },
    "dns": null,
    "amount_charged": 20000
  }
}
```

POST /domains/{domain}/renew

METHOD **POST**	BILLING **Billable**	SUCCESS **200**

Renew an owned domain for one or more years and charge the wallet after registry success.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`period`	required integer 1-10 years

Implementation notes

- Domains in the restoration window cost `redemption_price` for the first year; extra years use `renewal_price`.
- Read the domain first to show the same amount this endpoint will charge.
- If the registry cannot be reached, the API returns a retry message and no charge is made.
- After a timeout, read the domain and compare the expiry date before trying again.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/renew" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"period": 1}'
```

Example response


```
{
  "success": true,
  "message": "Domain renewed successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.co.tz",
      "status": "active",
      "registry_status": "active",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2028-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    }
  }
}
```


POST /domains/{domain}/sync

METHOD	BILLING	SUCCESS
POST	**No charge**	**200**

Refresh local status, expiry, lock, privacy and safe registry metadata from the registry.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Implementation notes

- Never billable. Use it after a transfer completes or when a customer disputes the expiry date.
- The response returns the refreshed domain object.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/sync" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "Domain synced successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.co.tz",
      "status": "active",
      "registry_status": "active",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2027-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    }
  }
}
```

7 Transfer a domain in

POST /domains/{domain}/transfer

METHOD POST	BILLING Billable	SUCCESS 200

Submit a domain transfer with the Auth/EPP code and charge the transfer price after registry success.

Parameters

Name	Location and requirement
`domain`	path (required): full domain name to transfer

Body parameters

Field	Requirement
`auth_code`	required string 6-128 characters. Never log or store it
`registrant`	optional contact object, required by some extensions
`admin`	optional contact object

Implementation notes

- Auth codes are never echoed back in API responses.
- A transfer only completes when the current registrar releases the domain. Poll `POST /domains/{domain}/sync`.
- The domain shows status `pending_transfer` until the registry finishes the move.
- If the domain is already managed by another portal account, the API returns 422.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.com/transfer" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"auth_code": "sample-transfer-code", "registrant": {"name": "Jane Doe", "organization": "Jane Doe
Ltd", "address1": "123 Registry Road", "city": "Dar es Salaam", "state": "Dar es Salaam", "postcode":
"11101", "country": "TZ", "phone": "+255.754123456", "email": "jane@example.co.tz"}, "admin": {"name": "Jane
Doe", "organization": "Jane Doe Ltd", "address1": "123 Registry Road", "city": "Dar es Salaam", "state": "Dar
es Salaam", "postcode": "11101", "country": "TZ", "phone": "+255.754123456", "email": "jane@example.co.tz"}}'
```

Example response


```
{
  "success": true,
  "message": "Domain transfer submitted successfully.",
  "data": {
    "domain": {
      "id": 123,
      "domain_name": "example.com",
      "status": "pending_transfer",
      "registry_status": "pending_transfer",
      "redemption_state": "grace",
      "in_redemption": false,
      "renewal_price": 20000,
      "redemption_price": null,
      "registration_date": "2026-05-28",
      "expiry_date": "2027-05-28",
      "lock_status": false,
      "privacy_status": false,
      "last_synced_at": "2026-05-28T12:00:00.000000Z"
    }
  }
}
```

7 Contacts and Whois

Read and update the registrant and admin contacts held at the registry.

GET /domains/{domain}/whois

METHOD GET	BILLING No charge	SUCCESS 200

Read the registrant and admin contact details currently held at the registry.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Implementation notes

- Some extensions omit `state` or `organization`. Send the returned values back unchanged when updating.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.co.tz/whois" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response

```
{
  "success": true,
  "message": "Whois contact details loaded successfully.",
  "data": {
    "registrant": {
      "name": "Jane Doe",
      "organization": "Jane Doe Ltd",
      "address1": "123 Registry Road",
      "city": "Dar es Salaam",
      "state": "Dar es Salaam",
      "postcode": "11101",
      "country": "TZ",
      "phone": "+255.754123456",
      "email": "jane@example.co.tz"
    },
    "admin": {
      "name": "Jane Doe",
      "organization": "Jane Doe Ltd",
      "address1": "123 Registry Road",
      "city": "Dar es Salaam",
      "state": "Dar es Salaam",
      "postcode": "11101",
      "country": "TZ",
      "phone": "+255.754123456",
      "email": "jane@example.co.tz"
    }
  }
}
```

POST /domains/{domain}/whois

METHOD POST	BILLING No charge	SUCCESS 200

Update the registrant and admin contact details at the registry.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`registrant`	required contact object (section 5.2)
`admin`	required contact object

Implementation notes

- The contact country must be an ISO 3166-1 alpha-2 code (`TZ`, `ZA`, `KE`). A full country name is accepted and stored as its code.
- Extensions with registry rules reject some changes. For example, `.co.za` requires a South African registrant.
- Send every field. There is no partial update.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/whois" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"registrant": {"name": "Jane Doe", "organization": "Jane Doe Ltd", "address1": "123 Registry Road",
"city": "Dar es Salaam", "state": "Dar es Salaam", "postcode": "11101", "country": "TZ", "phone":
"+255.754123456", "email": "jane@example.co.tz"}, "admin": {"name": "Jane Doe", "organization": "Jane Doe
Ltd", "address1": "123 Registry Road", "city": "Dar es Salaam", "state": "Dar es Salaam", "postcode":
"11101", "country": "TZ", "phone": "+255.754123456", "email": "jane@example.co.tz"}}'
```

Example response

```
{
  "success": true,
  "message": "Whois contact details updated successfully.",
  "data": {
    "registrant": {
      "name": "Jane Doe",
      "organization": "Jane Doe Ltd",
      "address1": "123 Registry Road",
      "city": "Dar es Salaam",
      "state": "Dar es Salaam",
      "postcode": "11101",
      "country": "TZ",
      "phone": "+255.754123456",
      "email": "jane@example.co.tz"
    },
    "admin": {
      "name": "Jane Doe",
      "organization": "Jane Doe Ltd",
      "address1": "123 Registry Road",
      "city": "Dar es Salaam",
      "state": "Dar es Salaam",
      "postcode": "11101",
      "country": "TZ",
      "phone": "+255.754123456",
      "email": "jane@example.co.tz"
    }
  }
}
```

7 Nameservers and child nameservers

Point a domain at your DNS provider and manage child nameservers (glue records) where the extension supports them. DNS records themselves are managed at your DNS provider, not through this API.

GET /domains/{domain}/nameservers

METHOD **GET** | BILLING **No charge** | SUCCESS **200**

Read the nameservers currently set for an owned domain.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.co.tz/nameservers" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response

```
{
  "success": true,
  "message": "Nameservers loaded successfully.",
  "data": {
    "nameservers": [
      "ns1.example.com",
      "ns2.example.com"
    ]
  }
}
```

POST /domains/{domain}/nameservers

METHOD **POST** | BILLING **No charge** | SUCCESS **200**

Replace the nameservers of an owned domain.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`nameservers`	required array of 2 to 10 hostnames
`provision_dns`	optional boolean; only for portal DNS management

Implementation notes

- Submit the full replacement set, not an addition to the current list.

- The allowed number depends on the domain extension. A list that is too short or too long is rejected with 422.
- Read the nameservers back after updating, and allow time for registry propagation.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/nameservers" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"nameservers": ["ns1.example.com", "ns2.example.com"]}'
```

Example response

```
{
  "success": true,
  "message": "Nameservers updated successfully.",
  "data": {
    "nameservers": [
      "ns1.example.com",
      "ns2.example.com"
    ]
  }
}
```

GET /domains/{domain}/child-nameservers

METHOD GET	BILLING No charge	SUCCESS 200

Read saved child nameservers (glue records) for extensions that support them.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name
`child_nsset_id`	query (optional): a saved child nameserver set ID

Implementation notes

- When no set has been saved, the response returns an empty `nameservers` list and `child_nsset_id: null`.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.co.tz/child-nameservers?child_nsset_id=NSSET123" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response


```
{
  "success": true,
  "message": "Child nameservers loaded successfully.",
  "data": {
    "child_nsset_id": "NSSET123",
    "nameservers": [
      {
        "label": "ns1",
        "name": "ns1.example.co.tz",
        "addresses": [
          "192.0.2.10"
        ]
      },
      {
        "label": "ns2",
        "name": "ns2.example.co.tz",
        "addresses": [
          "192.0.2.11"
        ]
      }
    ]
  }
}
```


POST /domains/{domain}/child-nameservers

METHOD **POST** | BILLING **No charge** | SUCCESS **200**

Save child nameservers with glue addresses and optionally apply them as the active nameservers.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`child_nameserver_action`	optional `save_child_nameservers` (default) or `use_child_nameservers`
`child_nameservers`	required array of 2 to 5 objects
`child_nameservers[].label`	required label such as `ns1`, or a full hostname inside the same domain
`child_nameservers[].addresses`	required IPv4 and/or IPv6 glue addresses, comma or space separated

Implementation notes

- Use `use_child_nameservers` to save the set and switch the domain to it in one request.
- Only extensions that support glue records accept this action; others return a 422 unsupported-action message.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/child-nameservers" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"child_nameserver_action": "use_child_nameservers", "child_nameservers": [{"label": "ns1",
"addresses": "192.0.2.10, 2001:db8::10"}, {"label": "ns2", "addresses": "192.0.2.11"}]}'
```

Example response


```
{
  "success": true,
  "message": "Child nameservers saved and applied successfully.",
  "data": {
    "child_nsset_id": "NSSET123",
    "nameservers": [
      {
        "label": "ns1",
        "name": "ns1.example.co.tz",
        "addresses": [
          "192.0.2.10",
          "2001:db8::10"
        ]
      },
      {
        "label": "ns2",
        "name": "ns2.example.co.tz",
        "addresses": [
          "192.0.2.11"
        ]
      }
    ]
  }
}
```

7 Transfer lock, Whois privacy and EPP codes

Protect a domain from unauthorised transfers and manage Auth/EPP codes and Whois privacy where the extension supports it.

POST /domains/{domain}/epp-code

METHOD **POST** | BILLING **No charge** | SUCCESS **200**

Request the current Auth/EPP code so a customer can move the domain away.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Implementation notes

- Only some extensions expose an Auth/EPP code. Others return an unsupported-action message.
- Never store or log the code, and never display it on a public page. Deliver it through a protected channel to the authorised domain holder.

Example request

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.com/epp-code" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response

```
{
  "success": true,
  "message": "EPP code requested successfully.",
  "data": {
    "epp_code": "sample-auth-code"
  }
}
```

PUT /domains/{domain}/epp-code

METHOD **PUT** | BILLING **No charge** | SUCCESS **200**

Set a new Auth/EPP code when the extension supports it.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`epp_code`	required string 8-128 characters

Example request

```
curl -X PUT "https://reseller.co.tz/api/v1/domains/example.com/epp-code" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"epp_code": "a-strong-new-auth-code"}'
```

Example response

```
{
  "success": true,
  "message": "EPP code updated successfully.",
  "data": {}
}
```

GET /domains/{domain}/lock

METHOD **GET** | BILLING **No charge** | SUCCESS **200**

Read the transfer lock status where the extension supports a client lock.

Parameters

Name	Location and requirement
domain	path (required): portal domain ID or full domain name

Implementation notes

- Returns 422 when the extension does not support a transfer lock.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.co.tz/lock" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response

```
{
  "success": true,
  "message": "Domain lock status loaded successfully.",
  "data": {
    "locked": true
  }
}
```

PUT /domains/{domain}/lock

METHOD **PUT** | BILLING **No charge** | SUCCESS **200**

Enable or disable the transfer lock where the extension supports it.

Parameters

Name	Location and requirement
domain	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`locked`	required boolean

Implementation notes

- A locked domain cannot be transferred away until the lock is removed.

Example request

```
curl -X PUT "https://reseller.co.tz/api/v1/domains/example.co.tz/lock" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"locked": true}'
```

Example response

```
{
  "success": true,
  "message": "Domain lock updated successfully.",
  "data": {
    "locked": true
  }
}
```

GET /domains/{domain}/privacy

METHOD **GET** | BILLING **No charge** | SUCCESS **200**

Read Whois privacy status where the extension supports it.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Implementation notes

- Returns 422 when the extension does not support Whois privacy.

Example request

```
curl -X GET "https://reseller.co.tz/api/v1/domains/example.com/privacy" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Accept: application/json"
```

Example response

```
{
  "success": true,
  "message": "Whois privacy status loaded successfully.",
  "data": {
    "enabled": false
  }
}
```

PUT /domains/{domain}/privacy

METHOD PUT	BILLING No charge	SUCCESS 200

Enable or disable Whois privacy where the extension supports it.

Parameters

Name	Location and requirement
`domain`	path (required): portal domain ID or full domain name

Body parameters

Field	Requirement
`enabled`	required boolean

Implementation notes

- Many country extensions, including `.tz` domains, do not offer Whois privacy.

Example request

```
curl -X PUT "https://reseller.co.tz/api/v1/domains/example.com/privacy" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"enabled": true}'
```

Example response

```
{
  "success": true,
  "message": "Whois privacy updated successfully.",
  "data": {
    "enabled": true
  }
}
```

8 Workflow guidance

These sequences keep the customer experience predictable and reduce the risk of duplicate charges.

8.1 cURL walkthrough: register a domain end to end

This sequence is the safest path for a new order.

Step 1 - check availability

```
curl -X POST "https://reseller.co.tz/api/v1/domains/check" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"domain_name": "example.co.tz"}'
```

Step 2 - read the price before charging

Use `price.registration` for the first year and `price.renewal` for additional years. Stop if `available` is `false`, or if `premium` is `true` and `premium_sellable` is `false`.

Step 3 - register

```
curl -X POST "https://reseller.co.tz/api/v1/domains/register" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"domain_name": "example.co.tz", "period": 1, "nameservers": ["ns1.example.com", "ns2.example.com"],
"same_contact": true, "registrant": {"name": "Jane Doe", "organization": "Jane Doe Ltd", "address1": "123
Registry Road", "city": "Dar es Salaam", "state": "Dar es Salaam", "postcode": "11101", "country": "TZ",
"phone": "+255.754123456", "email": "jane@example.co.tz"}}'
```

Step 4 - store the result and confirm

Store `data.domain.id`, `data.domain.expiry_date` and `data.amount_charged`. Show the customer the domain name and expiry date, and confirm the amount deducted from your wallet.

Step 5 - renew later

```
curl -X POST "https://reseller.co.tz/api/v1/domains/example.co.tz/renew" \
  -H "X-API-KEY: $RESELLER_API_KEY" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d '{"period": 1}'
```

8.2 Register a domain

1. Check availability with `POST /domains/check` and store the quoted price.
2. Validate the customer's contact details in your own form before sending them.
3. Create an order in state `pending` with a lock on the customer and domain.
4. Send `POST /domains/register` once. Do not retry automatically.
5. On 201, store `data.domain.id`, `expiry_date` and `amount_charged`, and show the confirmation.
6. On 422, show a safe message and mark the order `failed`. Keep field errors server-side.
7. On a timeout or 500, mark `needs_review` and read the domain before doing anything else.

```
Register
  -> check availability and show price
  -> validate contacts locally
  -> POST /domains/register (once)
     -> 201: store domain, confirm to customer
     -> 422: show safe message, mark failed
     -> timeout / 500: mark needs_review, then GET /domains/{name}
```

8.3 Renew a domain

- Read `GET /domains/{domain}` immediately before the confirmation screen.
- Display `renewal_price` normally. If `in_redemption` is `true`, display `redemption_price` for the first year.
- Submit `POST /domains/{domain}/renew` with a `period` from 1 to 10 years.
- After a timeout, read the domain and compare its expiry date before trying again.

Redemption pricing example

```
1 year  = redemption_price
3 years = redemption_price + (2 x renewal_price)
```

Build renewal reminders from `expiry_date`. A typical schedule is 30, 14, 7 and 1 day before expiry, and follow-up once the domain shows `expired`.

Renewal failure message

If the registry cannot complete a renewal, the API returns a professional retry message with 422 and no charge is made. Show that message as-is in your confirmation UI and log the order for review; do not attempt an immediate second renewal.

8.4 Transfer a domain in

- Collect the Auth/EPP code securely and send it only in the transfer request body.
- Do not store the code in your order table, logs, analytics or customer-facing history.
- Confirm the domain appears as `pending_transfer` after submission.
- Poll `POST /domains/{domain}/sync` at a controlled interval (for example every 30 to 60 minutes). Transfer completion depends on release by the current registrar.
- Do not resubmit the transfer simply because the status remains pending.

8.5 Update contacts

1. Read `GET /domains/{domain}/whois` and show the current values.
2. Let the customer edit every field in the contact object shown in section 5.2.
3. Send the complete object with `POST /domains/{domain}/whois`.
4. Read the contacts back and confirm the change.

Keep the country as an ISO code so the registry accepts it. Remember that some extensions enforce contact rules and may reject a change.

8.6 Configure nameservers

Point the domain at your DNS provider's nameservers. Nameserver limits depend on the extension, so submit the full replacement set, confirm every hostname is valid, and read the nameservers back. If your service hosts the DNS zone itself, this is where you finish the configuration; DNS records are then managed in your own control panel.

8.7 Protect a domain

- Keep the transfer lock enabled unless the domain holder has explicitly requested an outbound transfer.
- Request an Auth/EPP code only for an authorised outbound transfer and deliver it through a protected channel.
- Read lock and privacy state before offering a toggle so your UI reflects extension support and the current registry state.
- Expect some country extensions to reject Whois privacy or EPP changes. Present the returned unsupported-action message as an extension limitation, not an error in your system.

8.8 Background reconciliation job

Run one scheduled job that keeps your records aligned with the API.

Job step	API call	What to do with the result
Refresh owned domains	GET /domains?per_page=100 (page through)	Update status, expiry and prices in your database.
Poll pending transfers	POST /domains/{domain}/sync	Move completed transfers to `active`; keep polling pending ones.
Detect expiring domains	GET /domains?status=active	Queue renewal reminders and reports.
Resolve `needs_review` orders	GET /domains/{domain}	Confirm or fail the order; never re-submit billable orders automatically.

9 Testing and go-live

9.1 Minimum test cases before release

Test case	Expected result
Missing API key	401 with `success: false`; no retry loop.
Valid key from a non-whitelisted IP	403 with an IP whitelist message.
Availability for an available name	`available: true` and a price to quote.
Availability for a reserved name	`reserved: true`, `available: false`.
Registration with an invalid country	422 with a field error on the contact.
Registration with insufficient balance	422 with a balance message; no domain created.
Duplicate submission (double click)	Only one order is sent; the second is blocked by your order lock.
Renewal amount shown equals amount charged	Confirmation matches the wallet deduction.
Renewal timeout handling	Order moves to `needs_review`; expiry is checked before any retry.
Transfer submission	Status becomes `pending_transfer`; polling does not resubmit.
Nameservers outside the extension limit	422 and a clear message.
Lock or privacy on an unsupported extension	422 unsupported-action message shown as an extension limitation.
429 handling	Client backs off and retries later without losing the order.

9.2 Production launch checklist

Control	Evidence of completion
API key stored in the secret manager and confirmed on the API Access page	Key present, masked preview matches.
All production egress IPs whitelisted	403 disappears from the server for every instance.
Shared API client used for every call	One place adds headers and handles errors.
Order lock and state model implemented	Double submissions are blocked in testing.
Billable retries disabled; reconciliation job scheduled	Job runs and resolves `needs_review` orders.
Customer-facing messages reviewed	No raw registry or provider text is displayed.
Auth codes and keys excluded from logs	Log sample contains no `auth_code` or `X-API-KEY`.
Pagination handled for large accounts	Listing above 100 domains works.
Key rotation runbook written	Documented owner and steps.
Support contact and escalation path agreed	Ticket process and response times documented.

10 Troubleshooting and support

10.1 Common errors

Symptom	Likely cause	Next check
401 Unauthorized	Key missing, mistyped, revoked or regenerated.	Read the key from environment configuration, not a copy from memory.
403 Forbidden	Calling IP is not whitelisted or is blocked.	Compare the server's outbound IP with the API Access lists.
404 Not Found on a domain	Domain is not owned by this account, or the identifier is wrong.	List domains with `q` search to confirm the exact name or ID.
422 validation error	A required field is missing or has the wrong format.	Check the `errors` keys; country is usually an ISO code.
422 low balance	Wallet balance is below the quoted total.	Top up the account and retry the order.
422 unsupported action	The extension does not support that action.	Offer the action only where your catalogue says it is supported.
429 Too Many Requests	Too many requests in the window.	Back off, queue the work and cache reads.
Domain reserved	The registry reserves the name.	Never attempt registration; show the reserved message.
Renewal says retry in 30 minutes	The registry could not complete the action at this moment.	No charge was made; wait and retry later.
Transfer stuck as `pending_transfer`	The current registrar has not released the domain.	Keep polling `sync`; contact the losing registrar if it remains pending.

10.2 Information to collect before requesting support

- The UTC timestamp, HTTP method, endpoint path and HTTP status.
- Your internal order or correlation ID, without the API key or an Auth/EPP code.
- The response `message` and the safe error field names, with secrets and contact data removed.
- The affected domain name and portal domain ID (`domain.id`).
- Whether the request was sent once or retried, and the result of any reconciliation read.

10.3 Support channels

Channel	Details
Live API reference	https://reseller.co.tz/api-docs
Machine-readable reference	https://reseller.co.tz/api-docs/ai-instructions.json
Account and integration help	Open a support ticket from your reseller portal.

Protect credentials in support requests

Never attach your API key, an Auth/EPP code or full customer contact details to a ticket or an email. Support can investigate with the domain name, the timestamp and the HTTP status.

11 Document information

Item	Value
Document	Domains API integration guide and examples
Version	1.0
API version	1.0 (`/api/v1`)
Last updated	25 October 2026
Owner	reseller.co.tz
Live reference	https://reseller.co.tz/api-docs

11.1 Maintenance rule

This guide must stay aligned with the live API documentation. When an endpoint or parameter changes, update the source API description first, then revise this guide in the same release. Change the version number and the last-updated date on every published revision.

11.2 Share notice

This guide is intended for reseller partners and their authorised technical teams. It contains public API instructions only: no registry account names, provider credentials, internal endpoints or customer data. Share it with the developers or integrators that need it, and avoid forwarding it to public forums.

11.3 Version history

Version	Date	Change
1.0	25 October 2026	First reseller release for domain management (DNS record management intentionally excluded).