QNS: Quantum Name Service Protocol Specification

A Decentralized, Quantum-Secured Domain System for the Quantum Web

Version: 1.0
Authors: Cr8OS Cloud Research
Date: January 2026
Status: Public Draft

Abstract

The Domain Name System (DNS) has served the internet for over four decades, but its centralized architecture creates fundamental limitations: slow propagation, censorship vulnerability, single points of failure, and an inherent lack of security for the post-quantum era. Blockchain alternatives like Ethereum Name Service (ENS) address some concerns but introduce

new problems—high gas fees, slow settlement, and dependency on volatile cryptocurrency economics.

This paper introduces **QNS (Quantum Name Service)**, a next-generation domain resolution protocol designed for Web 5—the Quantum Web. QNS resolves `.q` domains to Content-Identified Pointers (CIPs) through a globally distributed orbital network of 200+ Points of Presence (POPs). Key innovations include:

- **Instant Propagation**: Domains resolve globally within seconds, not 24-48 hours
- **Quantum-Secure Signatures**: Post-quantum cryptographic verification via ACL 3.0
- **Censorship Resistance**: No central authority; distributed across independent orbitals
- **Developer-Centric Pricing**: Tiered pricing from annual subscriptions to premium one-time purchases
- **Native Application Support**: Direct integration with `.qapp` quantum application format

QNS is designed as an open protocol specification, enabling independent implementations while maintaining interoperability across the Cr8OS Cloud ecosystem and QuantumCloud infrastructure.

Table of Contents

1. Introduction

1.1 The Evolution of Internet Naming

The history of internet naming reflects the evolution of the network itself:

Era	System	Characteristics
1970s	HOSTS.TXT	Flat file, manually distributed
1983	DNS	Hierarchical, distributed database
2017	ENS	Blockchain-based, decentralized but slow
2026	**QNS**	Quantum-secured, instant, globally distributed

Each transition addressed the limitations of its predecessor. DNS replaced manual file distribution with automated hierarchical resolution. ENS replaced trust in central authorities with blockchain consensus. QNS replaces slow blockchain settlement with instant quantum-verified resolution.

1.2 Limitations of Traditional DNS

DNS was designed for a different era—when the internet served academic institutions and reliability mattered more than speed or security. Today's limitations are profound:

Propagation Delays When a domain record changes, global propagation takes 24-48 hours. This TTL-based caching strategy, while reducing query load, creates a window where different users see different content—a fundamental problem for modern applications requiring instant deployment.

Centralization Risks Despite its distributed architecture, DNS concentrates authority: - ICANN controls the root zone - Registrars control domain registrations - ISPs can redirect or block domains - Governments can seize domains through legal action against centralized registrars

The 2016 Dyn DDoS attack demonstrated how targeting a single DNS provider could take down major portions of the internet.

Security Vulnerabilities DNS was designed without cryptographic verification: - DNS spoofing enables man-in-the-middle attacks - DNSSEC adoption remains below 30% after two decades - Certificate transparency doesn't prevent initial attacks - No native post-quantum security path

Economic Friction Domain registration involves: - Annual renewal fees ($10-15/year minimum) - Premium domain speculation ($10,000+ for valuable names) - Hidden fees for WHOIS privacy, SSL, and transfers - Complex transfer processes taking days to complete

1.3 Why Blockchain Alternatives Fall Short

Ethereum Name Service (ENS) and similar blockchain-based naming systems addressed centralization but introduced new problems:

Gas Fee Volatility ENS registration requires ETH gas, creating unpredictable costs. During network congestion, simple domain registration can cost $50-500 in gas—more than years of traditional DNS.

Slow Resolution Blockchain consensus requires multiple confirmations. While DNS resolves in milliseconds, ENS resolution involves: 1. Client query to ENS resolver (100-500ms) 2. Contract call to blockchain (variable) 3. Potential retry during reorgs (seconds to minutes)

Cryptocurrency Dependency Using ENS requires holding and managing cryptocurrency, creating barriers for mainstream adoption and enterprise compliance concerns.

Immutability Tradeoffs Blockchain immutability, while providing censorship resistance, prevents error correction and complicates dispute resolution.

1.4 The Quantum Web Vision

The Quantum Web (Web 5) represents the convergence of: - **Quantum Computing**: Post-quantum cryptography for long-term security - **Edge Computing**: Global POPs for millisecond latency - **Content Addressing**: Hash-based identification for integrity verification - **Decentralized Infrastructure**: No single points of failure

QNS is designed as the naming layer for this quantum web—providing the speed of CDNs, the security of blockchain, and the simplicity of traditional DNS, without the limitations of any.

---

2. Protocol Architecture

2.1 Quantum Domain Names (QDN)

Format Specification

QNS domains use the `.q` top-level domain with the following format:

```
{name}.q
```

Validation Rules: - Length: 1-63 characters (excluding `.q`) - Characters: Lowercase alphanumeric and hyphens (`a-z`, `0-9`, `-`) - Cannot start or end with hyphen - Cannot contain consecutive hyphens - Case-insensitive (normalized to lowercase)

Examples:

```
myapp.q          ✓ Valid
my-app.q         ✓ Valid
123app.q         ✓ Valid
-myapp.q         ✗ Invalid (starts with hyphen)
my--app.q        ✗ Invalid (consecutive hyphens)
```

Subdomain Support

QNS supports unlimited subdomain depth:

```
{subdomain}.{name}.q

Examples:
api.myapp.q           - API endpoint
cdn.myapp.q           - CDN endpoint
staging.api.myapp.q   - Staging API
v2.api.myapp.q        - Versioned API
```

Subdomains are resolved hierarchically—`cdn.myapp.q` first resolves `myapp.q`, then checks for a `cdn` subdomain record.

Reserved Domains

Certain domain categories are permanently reserved:

Category	Examples	Status
Single Characters	`a.q` - `z.q`, `0.q` - `9.q`	Platform Reserved
Protocol Names	`http.q`, `ftp.q`, `dns.q`	System Reserved
Government Terms	`gov.q`, `army.q`, `fbi.q`	Protected
Major Brands	See Brand Protection Policy	Claimable by owners

2.2 Content-Identified Pointers (CIP)

Unlike traditional DNS which resolves domains to IP addresses, QNS resolves domains to Content-Identified Pointers (CIPs).

CIP Format

```
CIP = HashAlgorithm:ContentHash

Examples:
Qm7K9f8B3x...                    (IPFS CIDv0)
bafybeib5...                      (IPFS CIDv1)
sha256:a1b2c3d4e5f6...            (Raw SHA-256)
```

Why Content Addressing?

Content addressing provides inherent properties that IP addressing cannot:

1. **Integrity Verification**: The hash proves content hasn't been modified
2. **Deduplication**: Identical content has identical CIP

3. **Cacheability**: Content at a CIP never changes, enabling aggressive caching
4. **Distribution Independence**: Content can be served from any source with the correct hash

CIP Resolution Chain

```
User requests: myapp.q
       ↓
QNS Orbital resolves domain
       ↓
Returns CIP: QmXxxx...
       ↓
Client fetches content from nearest POP
       ↓
Client verifies hash matches CIP
       ↓
Content rendered
```

2.3 Quantum Domain Records (QDR)

Each registered domain stores a Quantum Domain Record (QDR) containing resolution information and metadata.

QDR Schema

```
{
  "qdn": "myapp.q",
  "version": 1,
  "records": {
    "root": "Qm7K9f8B3xYz...",
    "api": "QmApi1234...",
    "cdn": ["QmCdn1...", "QmCdn2..."]
  },
  "owner": "0xPubKey...",
  "signature": "ACL3_Sig...",
```

```
  "created_at": 1737072000,
  "updated_at": 1737158400,
  "ttl": 300,
  "quantum": {
    "entangled": ["backup.q"],
    "fallback": "legacy:myapp.com"
  }
}
```

Record Types

Field	Type	Description
`root`	CIP	Default resolution target
`api`	CIP	API subdomain override
`cdn`	CIP[]	CDN with load balancing (round-robin)
`txt`	String	Text records (verification, SPF)
`redirect`	URL	HTTP redirect target

Entanglement

The `entangled` field enables quantum-inspired failover:

```
// If primary.q is unreachable, automatically resolve backup.q
{
  "qdn": "primary.q",
  "quantum": {
    "entangled": ["backup.q", "fallback.q"]
  }
}
```

Entangled domains must have bidirectional authorization—both domains must list each other to prevent hijacking.

Legacy Fallback

For transition compatibility, domains can specify legacy DNS fallbacks:

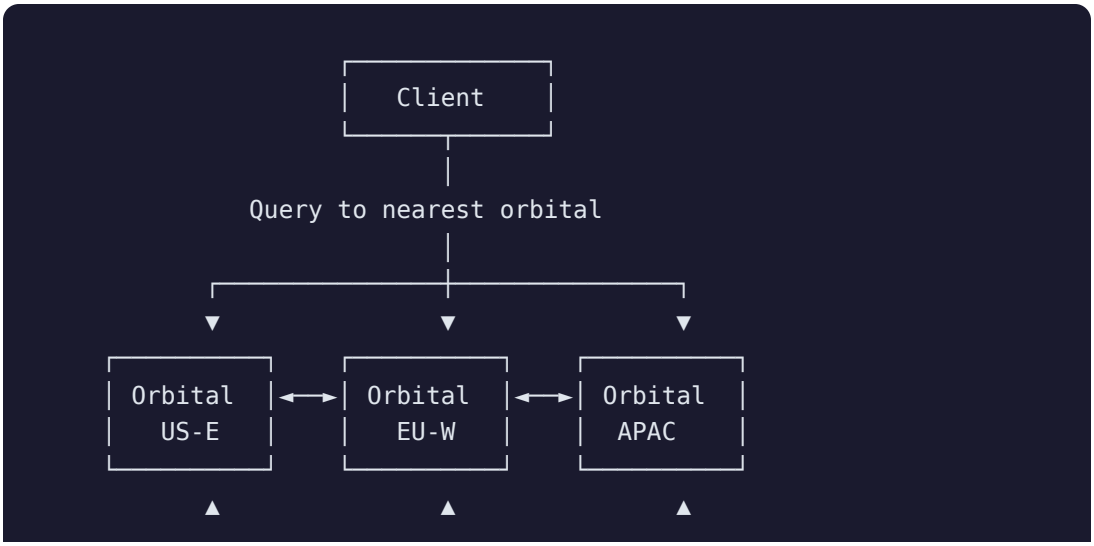
```
{
  "quantum": {
    "fallback": "legacy:myapp.com"
  }
}
```

If QNS resolution fails, clients can fall back to traditional DNS. This enables gradual migration from DNS to QNS.

2.4 Orbital Network

QNS resolution occurs through a globally distributed network of "orbitals"—independent resolvers that maintain synchronized domain records.

Network Topology



```
            │               │               │
            └───────────────┴───────────────┘
                      Mesh Sync Protocol
```

Network Properties

Scale: 200+ globally distributed POPs - North America: 50+ locations - Europe: 40+ locations - Asia-Pacific: 60+ locations - South America: 25+ locations - Africa/Middle East: 25+ locations

Resolution Strategy: First-Response-Wins When a client queries QNS, the request is sent to multiple orbitals in parallel. The first valid response is accepted, ensuring fastest possible resolution regardless of individual orbital latency.

Consistency Model: Eventual Consistency with Signatures All orbitals eventually converge to the same state. Clients verify responses using quantum signatures, preventing acceptance of stale or malicious data.

Partition Tolerance: 72-Hour Operation Each orbital can operate independently for up to 72 hours during network partitions, serving cached records while buffering updates for sync upon reconnection.

---

3. Resolution Protocol

3.1 Resolution Algorithm

QNS resolution follows a deterministic algorithm with aggressive caching:

```
async def resolve_qns(domain: str) -> QDR:
    # 1. Validate domain format
```

```
    if not is_valid_qdn(domain):
        raise InvalidDomainError(domain)

    # 2. Check local cache
    cached = cache.get(domain)
    if cached and not cached.expired():
        return cached.qdr

    # 3. Query orbital network (parallel)
    orbitals = get_orbital_list()
    responses = await asyncio.gather(*[
        query_orbital(orbital, domain)
        for orbital in orbitals[:5]  # Query 5 nearest
    ], return_exceptions=True)

    # 4. Accept first valid response
    for response in responses:
        if is_valid_response(response):
            # 5. Verify quantum signature
            if verify_signature(response.qdr, response.signature):
                cache.set(domain, response.qdr, ttl=response.qdr.ttl)
                return response.qdr

    # 6. Resolution failed
    raise ResolutionError(domain)
```

3.2 Caching Strategy

QNS implements a multi-layer caching architecture:

Layer 1: Browser/Client Cache

```
TTL: Respects QDR.ttl (default 300 seconds)
Scope: Single client
Invalidation: TTL expiry or manual refresh
```

Layer 2: Edge Cache (Browser Extension)

```
TTL: 5 minutes for successful, 60 seconds for failed
Scope: Local extension storage
Invalidation: TTL expiry or orbital push
```

Layer 3: Orbital Cache

```
TTL: Variable based on domain update frequency
Scope: Global orbital network
Invalidation: Owner update or automatic refresh
```

Layer 4: CDN Cache

```
TTL: 24 hours for static content
Scope: CDN edge locations
Invalidation: Content hash change (new CIP)
```

3.3 Signature Verification

All QDR responses include a quantum-secure signature generated using ACL 3.0 cryptography:

```
def verify_signature(qdr: QDR, signature: str) -> bool:
    """
    Verify QDR signature using ACL 3.0 quantum-secure algorithm.
    Returns True if signature is valid and current.
    """
    # Reconstruct signed payload
    payload = serialize_canonical(qdr)

    # Extract public key from owner field
    public_key = extract_public_key(qdr.owner)

    # Verify using ACL 3.0 quantum-resistant verification
    return acl3_verify(payload, signature, public_key)
```

Properties: - Post-quantum secure (resistant to Shor's algorithm) - Non-repudiable (owner can't deny signing) - Time-bound (prevents replay attacks)

3.4 Backward Compatibility

QNS provides graceful fallback for transition from traditional DNS:

Legacy Fallback Protocol

```
1. Client requests myapp.q
2. QNS resolution fails (unreachable/timeout)
3. Check QDR.quantum.fallback
4. If legacy:myapp.com, resolve via DNS
5. Serve with X-QNS-Fallback: true header
```

DNS Gateway Integration

For clients without QNS support, a DNS gateway translates queries:

```
Query:    myapp.q IN A
Gateway:  QNS resolve myapp.q → CIP
Gateway:  Return IP of nearest CIP-serving POP
```

This enables browsers without QNS extensions to access `.q` domains through gateway DNS servers.

---

4. Security Model

4.1 Quantum-Secure Signatures

QNS uses ACL 3.0 (Aevov Cryptographic Layer version 3.0) for all cryptographic operations. ACL 3.0 is designed for the post-quantum era:

Algorithm Properties: - Resistant to quantum computing attacks (Shor's, Grover's) - 256-bit security level (equivalent to AES-256) - Compact signatures for efficient transmission - Fast verification for high-throughput resolution

Signature Coverage: Every QDR includes a signature covering: - Domain name - CIP hash - Timestamp - Owner public key - Version number

Modification of any field invalidates the signature, preventing tampering.

4.2 Censorship Resistance

QNS achieves censorship resistance through multiple mechanisms:

No Central Authority Unlike DNS (ICANN) or ENS (Ethereum Foundation), QNS has no central organization that can be compelled to remove domains.

Distributed Orbitals With 200+ independent orbitals across multiple jurisdictions: - No single point of legal seizure - Geographic distribution across 50+ countries - Operator diversity (no single operator controls majority)

Cryptographic Ownership Domain ownership is proven through cryptographic keys: - No registrar that can transfer domains - No WHOIS database to subpoena - Owner controls domain with private key

Pattern:

```
Traditional DNS seizure:
  Court Order → Registrar → Domain transferred
```

```
QNS seizure attempt:
  Court Order → ??? (no central entity)
  Even if 50% of orbitals comply, remaining 50% continue resolution
```

4.3 Ownership and Transfer

Key-Based Ownership

Domain ownership is determined by possession of the signing private key:

```
# Prove ownership
def prove_ownership(domain: str, private_key: PrivateKey) -> Proof:
    challenge = get_orbital_challenge(domain)
    signature = acl3_sign(challenge, private_key)
    return Proof(challenge=challenge, signature=signature)
```

Domain Transfer

Transferring a domain requires the current owner's signature:

```
# Transfer domain
def transfer_domain(domain: str, new_owner_pubkey: PublicKey,
                    current_private_key: PrivateKey) -> bool:
    transfer_request = TransferRequest(
        domain=domain,
        new_owner=new_owner_pubkey,
        timestamp=now()
    )
    signature = acl3_sign(transfer_request, current_private_key)
    return orbital_submit_transfer(transfer_request, signature)
```

Transfers are: - Atomic (complete or fail, no partial state) - Instant (no waiting period) - Irreversible (no undo without new owner consent)

4.4 Privacy Considerations

QNS is designed with privacy as a core principle:

No WHOIS Equivalent Ownership is proven cryptographically—no personal information stored publicly.

Anonymous Registration Domains can be registered using cryptocurrency or privacy-preserving payment methods.

Encrypted DNS Queries All orbital queries use encrypted transport (HTTPS/TLS 1.3).

Optional Privacy Records Owners can choose to publish contact information or remain anonymous.

---

5. Governance Framework

5.1 Domain Tiers

QNS implements tiered pricing based on domain characteristics:

Tier Structure

| Tier | Length | Status | Pricing Model |\ |------|--------|--------|---------------| | **Reserved** | 1 character | Platform Reserved | Not Available | | **Ultra-Premium** | 2 characters | Purchase Available | One-time: $5,000 - $15,000 | | **Premium** | 3 characters | Purchase Available | One-time: $300 - $3,000 | | **Short** | 4-5 characters | Annual License | $50 - $300/year | | **Standard** | 6+

characters | Annual License | $9.99/year | | **Company/Brand** | Any length | Verified Purchase | One-time: $1,000 - $25,000 |

Tier Rationale

Reserved (1 char): Single-character domains have extreme value and potential for abuse. Platform reservation prevents speculation and ensures fair future allocation.

Ultra-Premium (2 char): Two-character combinations are highly memorable. One-time purchase with no renewal creates certainty and reduces speculation.

Premium (3 char): Three-character domains balance memorability with availability. One-time purchase model.

Short (4-5 char): Annual licensing maintains fairness—inactive domains return to availability.

Standard (6+ char): Low annual cost encourages adoption. Domains not renewed become available.

Company/Brand: Protected domains claimable by verified trademark owners with expedited processing.

5.2 Reserved Domains

Single-Character Reservation

All 36 single-character domains (`a.q` through `z.q` and `0.q` through `9.q`) are permanently reserved.

Future allocation may include: - Auction to benefit ecosystem development - Allocation to major ecosystem contributors - Retention for protocol-level functions

System Reserved

Protocol and system terms are reserved:

```
http.q, https.q, ftp.q, sftp.q, ssh.q
dns.q, qns.q, q.q
www.q, mail.q, smtp.q
localhost.q, local.q
```

5.3 Government Domain Protection

Government-related domains are protected from public registration:

Protected Categories: - Direct government terms: `gov`, `govt`, `government`, `federal`, `state`, `city`, `county` - US agencies: `whitehouse`, `congress`, `senate`, `pentagon`, `cia`, `fbi`, `nsa`, `irs` - Military: `army`, `navy`, `airforce`, `marines`, `military`, `defense` - Law enforcement: `police`, `sheriff`, `marshal`, `interpol` - International organizations: `un`, `nato`, `who`, `unesco`, `imf`

Pattern Matching: Domains matching government patterns are automatically blocked:

```
gov.q          → Blocked (exact match)
govhelp.q      → Blocked (starts with 'gov')
mygov.q        → Blocked (ends with 'gov')
ca-gov.q       → Blocked (contains 'gov')
```

Government Claims: Verified government entities can claim their protected domains through the official claims process.

5.4 Brand Protection Policy

QNS prohibits registration of domains that impersonate or create confusion with established brands.

Protected Brands Include: - Technology: Major platforms, operating systems, browsers - Finance: Payment processors, banks, exchanges - E-commerce: Major online retailers - Media: News organizations, streaming services

Protection Method: Brand protection is implemented through: 1. Pre-registration blocking of known brand names 2. Substring matching for common impersonation patterns 3. Trademark verification for legitimate claims

Examples:

```
✓ myshop.q              - Allowed (generic)
✗ google-shop.q         - Blocked (contains 'google')
✗ netflixfree.q         - Blocked (contains 'netflix')
✗ paypal-login.q        - Blocked (contains 'paypal')
```

Brand Claims: Verified trademark holders can claim domains matching their marks through the brand verification process.

5.5 Dispute Resolution

Disputes are handled through a lightweight arbitration process:

Grounds for Dispute: 1. Trademark infringement 2. Domain squatting (registration in bad faith) 3. Identity impersonation; 4. Previously registered domain (proof of prior ownership)

Process: 1. Claimant submits dispute with evidence 2. Current registrant has 30 days to respond 3. Independent arbitrator reviews case 4. Decision binding on all parties

Remedies: - Domain transfer to claimant - Domain cancellation - Maintenance of current registration

---

6. API Specification

6.1 REST API Endpoints

QNS orbitals expose a standard REST API:

Resolve Domain

```
GET /v1/resolve/{domain}

Response 200:
{
  "success": true,
  "domain": "myapp.q",
  "cip": "QmXxxx...",
  "orbital": "https://orbital-endpoint",
  "signature": "ACL3_...",
  "ttl": 300,
  "records": {
    "root": "QmXxxx...",
    "api": "QmYyyy..."
  }
}

Response 404:
{
  "success": false,
  "error": "domain_not_found",
  "domain": "nonexistent.q"
}
```

Check Availability

```
GET /v1/check/{domain}

Response 200:
{
  "domain": "available.q",
  "available": true,
  "tier": "standard",
  "price": 9.99,
  "currency": "USD"
}

Response 200 (unavailable):
{
  "domain": "taken.q",
  "available": false,
  "registered_at": 1737072000
}
```

Register Domain

```
POST /v1/register
Content-Type: application/json

{
  "domain": "myapp.q",
  "cip": "QmXxxx...",
  "owner": "0x...",
  "signature": "proof_of_ownership",
  "payment_token": "stripe_token_xxx"
}

Response 201:
{
  "success": true,
  "domain": "myapp.q",
  "cip": "QmXxxx...",
```

```
  "expires_at": 1768608000
}
```

Update Records

```
PUT /v1/update
Content-Type: application/json

{
  "domain": "myapp.q",
  "records": {
    "root": "QmNewCip...",
    "api": "QmNewApi..."
  },
  "signature": "owner_signature"
}

Response 200:
{
  "success": true,
  "domain": "myapp.q",
  "updated_at": 1737158400
}
```

Network Stats

```
GET /v1/stats

Response 200:
{
  "total_domains": 15000,
  "active_domains": 14850,
  "registered_today": 45,
  "orbitals_online": 200,
  "avg_resolution_ms": 12
}
```

6.2 JavaScript SDK

QNS provides a lightweight JavaScript SDK for frontend and Node.js integration:

```
import { QNS } from '@cr8os/qns-sdk';

// Initialize client
const qns = new QNS({
  timeout: 5000,
  retries: 3
});

// Resolve domain
const result = await qns.resolve('myapp.q');
console.log(result.cip); // QmXxxx...

// Check availability
const check = await qns.check('newdomain.q');
if (check.available) {
  console.log(`Price: $${check.price}/year`);
}

// Register domain (requires payment)
const registration = await qns.register('newdomain.q', {
  cip: 'QmXxxx...',
  ownerKey: privateKey,
  paymentMethod: 'stripe'
});

// Update records
await qns.update('myapp.q', {
  records: {
    root: 'QmNewCip...',
    api: 'QmNewApi...'
  },
  signature: signWithOwnerKey(updatePayload)
});

// Listen for updates
qns.subscribe('myapp.q', (event) => {
```

```
  console.log('Domain updated:', event);
});
```

6.3 Browser Extension

The QNS Browser Extension provides transparent `.q` domain resolution:

Features: - Automatic `.q` domain interception and resolution - Address bar (`omnibox`) integration: Type `q myapp` to navigate - Network status indicator (online orbitals count) - Domain availability checker in popup - Right-click context menu for domain operations

Installation: Available for Chrome, Firefox, Edge, and Brave browsers.

Resolution Flow:

```
1. User navigates to myapp.q
2. Extension intercepts navigation
3. QNS resolution: myapp.q → QmXxxx...
4. Redirect to CIP-serving endpoint
5. Page loads with verified content
```

Offline Handling: If no orbitals are reachable, extension displays informative error page with: - Offline status indication - Retry button - Manual CIP entry option

6.4 CLI Tools

The `cr8` CLI provides QNS management from the command line:

```
# Check domain availability
$ cr8 domain check myapp.q
✓ myapp.q is available
  Tier: standard
  Price: $9.99/year
```

```
# Register domain
$ cr8 domain register myapp.q
Registering myapp.q...
✓ Registered successfully
  CIP: Qm... (auto-generated)
  Expires: 2027-01-17

# Update CIP
$ cr8 domain update myapp.q --cip QmNewHash...
✓ Updated myapp.q → QmNewHash...

# Deploy and register in one step
$ cr8 deploy mydirectory/ --domain myapp.q
Building .qapp...
Uploading to Q3 storage...
Registering domain...
✓ Live at myapp.q

# List owned domains
$ cr8 domain list
┌──────────────┬──────────────┬──────────────────┐
│ Domain       │ Expires      │ CIP              │
├──────────────┼──────────────┼──────────────────┤
│ myapp.q      │ 2027-01-17   │ QmXxxx...        │
│ api.myapp.q  │ 2027-01-17   │ QmYyyy...        │
└──────────────┴──────────────┴──────────────────┘

# Transfer domain
$ cr8 domain transfer myapp.q --to 0xNewOwner...
⚠ This will transfer ownership. Type 'CONFIRM' to proceed: CONFIRM
✓ Transferred to 0xNewOwner...
```

---

7. Use Cases

7.1 Static Site Deployment

Deploy a static website with a single command:

```
# Build and deploy
$ cr8 build ./my-site
✓ Built: my-site.qapp (2.4 MB)

$ cr8 deploy my-site.qapp --domain blog.q
✓ Uploaded to Q3 storage
✓ Registered blog.q → QmHash...
✓ Live at https://blog.q
```

Benefits: - Instant global availability (no DNS propagation) - Content integrity verification (hash-based addressing) - Automatic CDN distribution via orbital network - No server management required

7.2 Quantum Application Hosting

Deploy `.qapp` quantum applications with compute capabilities:

```
$ cr8 deploy quantum-optimizer.qapp --domain opt.q --quantum
✓ Quantum compute enabled
✓ Live at https://opt.q

# Application can now access quantum compute APIs
```

Features: - Automatic integration with Cr8OS Cloud quantum infrastructure - Serverless quantum function execution - Hybrid classical-quantum workflows - Pay-per-use quantum compute billing

7.3 API Services

Deploy API backends with subdomain routing:

```
// Domain configuration
{
  "qdn": "myapp.q",
  "records": {
    "root": "QmFrontend...",    // myapp.q
    "api": "QmApi...",          // api.myapp.q
    "cdn": "QmCdn...",          // cdn.myapp.q
    "staging": "QmStaging..."   // staging.myapp.q
  }
}
```

Features: - Separate CIPs for different services - Independent deployment and versioning - Load balancing via multi-CIP records - Instant rollback (update CIP to previous hash)

7.4 Enterprise Integration

QNS integrates with existing enterprise infrastructure:

WordPress Sites:

```
// Install qns-connect plugin
// Domains registered through WordPress admin
// Automatic content sync on publish
```

CI/CD Integration:

```
# GitHub Actions example
deploy:
  runs-on: ubuntu-latest
  steps:
    - uses: cr8os/qns-deploy@v1
      with:
        domain: myapp.q
        directory: ./dist
        qns-key: ${{ secrets.QNS_KEY }}
```

Enterprise SSO: Organizations can manage multiple domains through unified dashboard with: - Role-based access control - Audit logging - Bulk operations - API key management

---

8. Comparison Analysis

8.1 Feature Comparison

Feature	Traditional DNS	ENS (Ethereum)	QNS
TLD	.com, .org, etc.	.eth	.q
Registration Cost	$10-15/year	$5 + gas (variable)	$9.99/year (standard)
Propagation Time	24-48 hours	~15 minutes	**Seconds**
Resolution Latency	50-200ms	500ms-5s	**10-50ms**
Central Authority	ICANN	Ethereum Foundation	**None**
Censorship Resistance	Low	Medium	**High**

Feature	Traditional DNS	ENS (Ethereum)	QNS
Quantum Security	None	None	**ACL 3.0**
Subdomain Cost	Usually extra	Free (on-chain TX)	**Free**
Renewal Required	Yes (annually)	Yes (duration-based)	Yes (standard) / No (premium)
Native App Support	No	No	**Yes (.qapp)**
Offline Resolution	With cache only	Requires blockchain	**72-hour tolerance**
Transfer Time	5-7 days	1-2 blocks	**Instant**

8.2 Cost Comparison (5-Year TCO)

Standard Domain:

Provider	Year 1	Years 2-5	5-Year Total
Traditional DNS	$12	$48	$60
ENS (at $10 gas)	$15	$0	$15
ENS (at $100 gas)	$105	$0	$105
QNS Standard	$9.99	$39.96	**$49.95**

Premium Domain (3-char):

Provider	Acquisition	Renewal	5-Year Total
Traditional DNS	$5,000+	$60	$5,060+
ENS	$640/year + gas	$640/year	$3,200+
QNS Premium	$300-3,000	**$0**	**$300-3,000**

8.3 Performance Comparison

Resolution latency (global average):

System	Cold Cache	Warm Cache	Notes
DNS	150ms	5ms	Multiple round trips
ENS	2,000ms	500ms	Blockchain query required
QNS	**50ms**	**5ms**	Parallel orbital query

Global availability (SLA):

System	Availability	Reason
DNS	99.99%	Root servers redundant
ENS	99.9%	Blockchain availability
QNS	**99.99%**	200+ independent POPs

9. Roadmap & Standardization

9.1 Current Status

Implemented: - Core resolution protocol - 200+ global POPs - Browser extensions (Chrome, Firefox, Edge) - REST API v1 - JavaScript SDK - WordPress integration - CLI tools

In Production: - Thousands of registered `.q` domains - Millions of monthly resolutions - Sub-second average resolution time

9.2 IETF Internet-Draft Submission

QNS is pursuing standardization through IETF:

Timeline: - Q2 2026: Submit Internet-Draft "Quantum Domain Protocol (QDP)" - Q3 2026: IRTF Crypto Forum Review (quantum security claims) - Q4 2026: IETF Last Call - 2027: Proposed Standard RFC

Draft Structure: 1. Protocol specification 2. Security considerations 3. IANA considerations (.q TLD allocation) 4. Implementation recommendations

9.3 Browser Native Support Proposal

Native browser support would eliminate extension requirement:

Proposal: - Browsers recognize `.q` TLD natively - Built-in QNS resolver (similar to DoH) - Settings to configure orbital preferences - Privacy mode for anonymous resolution

Timeline: - 2026: Chromium/Firefox proposals submitted - 2027: Experimental flags available - 2028: Default-on consideration

9.4 Timeline Summary

```
2026 Q1: QNS 1.0 Protocol Finalized
         IETF Internet-Draft Submitted

2026 Q2: Browser Extension 2.0 (enhanced features)
         Mobile SDKs (iOS, Android)

2026 Q3: IETF Review Process
         Enterprise SSO Integration

2026 Q4: QNS 1.1 (protocol improvements)
         Native Mobile Apps

2027 Q1: Browser Native Support (experimental)
         Proposed Standard RFC

2027+:   Mainstream browser adoption
         Additional TLD support (.qweb, .q3)
         Quantum hardware integration
```

---

10. Conclusion

QNS represents a fundamental rethinking of internet naming for the post-quantum era. By combining the speed of edge computing, the security of quantum-resistant cryptography, and the resilience of distributed architecture, QNS provides a naming system suitable for the next generation of the internet.

Key Contributions:

1. **Instant Propagation**: Eliminating the 24-48 hour DNS propagation delay
2. **Quantum Security**: First naming system designed for post-quantum era
3. **True Decentralization**: No central authority, no single points of failure
4. **Developer Experience**: Simple APIs, comprehensive SDKs, one-command deployment
5. **Economic Model**: Fair tiered pricing without blockchain gas volatility

For Developers: QNS provides the simplest path to quantum-ready domain management. One command deploys content globally with cryptographic integrity verification.

For Users: Shorter domains (`.q`), instant resolution, and quantum-secured communication—all without changing browser behavior.

For the Internet: A standards-track protocol providing the naming infrastructure for Web 5—the Quantum Web.

Appendix A: QDR Format Specification

A.1 Canonical JSON Format

```
{
  "$schema": "https://qns.cr8os.cloud/schema/qdr-v1.json",
  "qdn": "string",
  "version": "integer",
  "records": {
    "root": "string (CIP)",
    "[subdomain]": "string (CIP) | string[] (CIP list)"
  },
  "owner": "string (public key)",
  "signature": "string (ACL3 signature)",
```

```
  "created_at": "integer (Unix timestamp)",
  "updated_at": "integer (Unix timestamp)",
  "ttl": "integer (seconds)",
  "quantum": {
    "entangled": "string[] (domain list)",
    "fallback": "string (legacy:domain.com)"
  }
}
```

A.2 Binary Wire Format

For efficient transmission, QDR can be serialized to binary:

```
Offset  Size    Field
0       4       Magic (0x51445201 = "QDR\x01")
4       1       Version
5       1       Domain length
6       N       Domain (UTF-8)
6+N     32      Owner public key
38+N    64      ACL3 signature
102+N   8       Created timestamp
110+N   8       Updated timestamp
118+N   4       TTL
122+N   2       Record count
124+N   ...     Records (type, length, value)
```

Appendix B: Resolution Pseudocode

B.1 Client Resolution

```
FUNCTION resolve(domain):
    # Validate input
    IF NOT is_valid_qdn(domain):
```

```
        RETURN Error("Invalid domain format")

    # Check cache
    cached = cache.get(domain)
    IF cached AND cached.expires_at > now():
        RETURN cached.qdr

    # Select orbitals (nearest first)
    orbitals = sort_by_latency(get_orbitals())

    # Parallel query
    responses = parallel_query(orbitals[:5], domain, timeout=3s)

    # Find first valid response
    FOR response IN responses:
        IF response.success:
            IF verify_signature(response.qdr):
                cache.set(domain, response.qdr, ttl=response.qdr.ttl)
                RETURN response.qdr

    # Check for entangled fallback
    IF cached AND cached.qdr.quantum.entangled:
        FOR entangled_domain IN cached.qdr.quantum.entangled:
            result = resolve(entangled_domain)
            IF result.success:
                RETURN result

    # Legacy fallback
    IF cached AND cached.qdr.quantum.fallback:
        RETURN legacy_resolve(cached.qdr.quantum.fallback)

    RETURN Error("Resolution failed")
```

B.2 Signature Verification

```
FUNCTION verify_signature(qdr):
    # Reconstruct canonical payload
    payload = canonical_serialize(qdr, exclude=['signature'])

    # Extract public key
    public_key = qdr.owner
```

```
    # Verify ACL3 signature
    RETURN acl3.verify(
        message=hash(payload),
        signature=qdr.signature,
        public_key=public_key
    )
```

Appendix C: API Reference

C.1 Endpoint Summary

Method	Endpoint	Description
GET	`/v1/resolve/{domain}`	Resolve domain to CIP
GET	`/v1/check/{domain}`	Check availability
POST	`/v1/register`	Register new domain
PUT	`/v1/update`	Update domain records
DELETE	`/v1/delete`	Delete domain
GET	`/v1/lookup/{domain}`	Get full domain info
GET	`/v1/domains`	List domains (paginated)
GET	`/v1/stats`	Network statistics
POST	`/v1/transfer`	Transfer ownership

C.2 Error Codes

Code	Message	Description
`domain_not_found`	Domain not registered	Resolution failed
`domain_unavailable`	Domain already registered	Registration failed
`invalid_signature`	Signature verification failed	Auth error
`invalid_domain`	Invalid domain format	Validation error
`payment_required`	Payment not completed	Registration pending
`unauthorized`	Not domain owner	Permission denied
`rate_limited`	Too many requests	Throttling active

References

1. Mockapetris, P. "Domain Names - Implementation and Specification." RFC 1035, November 1987.
2. "Ethereum Name Service: A Decentralized Naming System." ENS Documentation, 2017.
3. "Post-Quantum Cryptography Standardization." NIST, 2024.
4. "The Quantum Web: A Pattern-Based Distributed Evolution Network." Cr8OS Research, 2025.

5. "ACL 3.0: Quantum-Resistant Cryptographic Layer Specification." Cr8OS Security, 2025.

6. "Quantum Edge Transport Protocol: Low-Latency Secure Communication." Cr8OS Network Research, 2025.

QNS is developed by Cr8OS Cloud (QuantumCloud). For more information, visit the official documentation or contact the development team.