Boing Network — Boing Quality Gate rules

Operational catalog of every rule a deployment must satisfy before it can be approved for inclusion in a block.

The Boing Quality Gate screens deployments against transparent rules before they reach the ledger. Automated checks handle clear cases. A decentralized community pool reviews edge cases. Boing Observer is where those rules, deployments, and outcomes stay visible and auditable.

Transparent by default — observable rules, deployments and outcomes.

Designed to reduce malicious, deceptive and low-quality deployments through transparent, protocol-level checks. Boing introduces protocol-level deployment checks designed to identify and restrict deployments that fail transparent quality criteria before they are accepted on-chain.

Public testnet — mainnet not live. The live registry on the RPC you query is the authority for list contents.

Field	Value
Document	QA Gate Rules (enforced catalog)
Date	26 August 2026
Protocol crate	`boing-qa` (`check_contract_deploy_full_with_metadata`)
Design spec	`docs/QUALITY-ASSURANCE-NETWORK.md`
Live registry	JSON-RPC `boing_getQaRegistry`
Live pool	JSON-RPC `boing_qaPoolConfig`
Pre-flight	JSON-RPC `boing_qaCheck` · boing.observer/tools/qa-check

This document lists **what the protocol actually enforces today**. The longer design spec explains *why*. Governance can change list contents (blocklist hashes, scam byte patterns, always-review categories, content terms, max size) without a chain restart. **Hard rule IDs, evaluation order, opcode whitelist, purpose categories, and matching algorithms are code.**

Live vs intended lists (public testnet, 26 August 2026).
`boing_getQaRegistry` on `https://testnet-rpc.boing.network/` returned empty `blocklist`, `scam_patterns`, `always_review_categories`, and `content_blocklist`, with `max_bytecode_size` = **32768**.
The **intended public-testnet policy** (applied with `npm run apply-public-testnet-qa-policy` in the protocol repo) merges the 132 English terms in `docs/config/qa_content_blocklist.en.json` into `content_blocklist`. Those terms are listed in Appendix A. They **do not reject** a deploy until the live registry contains them. Always trust `boing_getQaRegistry` on the RPC you submit to.

1. Scope

QA runs on every contract-deploy payload:

Payload	Purpose	Name / symbol
`ContractDeploy`	none (allowed)	not sent → content policy does not apply
`ContractDeployWithPurpose`	required string (empty is treated as omitted)	not sent → content policy does not apply
`ContractDeployWithPurposeAndMetadata`	required string	optional `asset_name` (max 256 UTF-8 bytes), `asset_symbol` (max 32 UTF-8 bytes)

The same checker also runs on **in-contract** `CREATE2` (`0xF5`), using purpose category `dapp` . Nested creates must pass the same bytecode rules.

QA does **not** gate transfers, contract calls, bond/unbond, or faucet. Existing contracts deployed before a rule existed are **not** retroactively removed.

Defense in depth: the node runs QA at **mempool insert** and again at **execution** before code is written to state.

2. Outcomes

Every check returns exactly one of:

Outcome	RPC	Meaning
Allow	submit succeeds	Eligible for inclusion in a block.
Reject	`-32050`	Never enters a block. Response includes `rule_id` and `message`.
Unsure	`-32051`	Referred to the community QA pool. Not auto-included. Includes pending `tx_hash`.

Allow is the only automated path to deployment. Unsure becomes Allow only if the pool votes Allow (or, if governance set `default_on_expiry` to `allow`, when the review window expires — public-testnet default is **reject** on expiry).

3. Evaluation order (code)

`check_contract_deploy_full_with_metadata` applies rules **in this order**. First match wins.

1. `METADATA_TOO_LONG` — `asset_name` > 256 UTF-8 bytes, or `asset_symbol` > 32 UTF-8 bytes.
2. **Empty bytecode** → `MALFORMED_BYTECODE`.
3. **Init-code marker only** — payload is `0xFD` with no bytes after it → `MALFORMED_BYTECODE`.
4. `MAX_BYTECODE_SIZE` — full payload length (including optional `0xFD` prefix) exceeds the registry max (default **32 768** bytes).
5. **Opcode / well-formedness** on the bytes **after** an optional leading `0xFD`:
   - unknown opcode byte → `INVALID_OPCODE`
   - truncated `PUSH` immediate, or stream length mismatch → `MALFORMED_BYTECODE`
6. `BLOCKLIST_MATCH` — BLAKE3-256 of the **full** bytecode (prefix included) equals a registry hash.
7. `PURPOSE_DECLARATION_INVALID` — purpose provided, non-empty after trim, and not in the valid set (case-insensitive).
8. `SCAM_PATTERN_MATCH` — any registry byte sequence is a contiguous substring of bytecode.
9. `CONTENT_POLICY_VIOLATION` — `asset_name` or `asset_symbol` matches a content-blocklist term (see §7).
0. **Always-review category** → **Unsure** (no `rule_id`; pool referral).
1. **Soft rule** — purpose `other` and `description_hash` length **< 4 bytes** (missing counts as 0) → **Unsure**.
2. Else **Allow**.

4. Hard rules (must pass)

These are binary. Failure is **Reject**. Passing all of them is **necessary** for Allow, not always **sufficient** (Unsure can still fire in steps 10–11).

R1 — Bytecode must not be empty

rule_id	`MALFORMED_BYTECODE`
Message	`Bytecode must not be empty`
Test	`bytecode.len() == 0`

R2 — Init-code prefix must have a body

rule_id	`MALFORMED_BYTECODE`
Message	`Init-code marker present but no init bytecode after prefix`
Test	First byte is `CONTRACT_DEPLOY_INIT_CODE_MARKER` (`0xFD`) and there are no following bytes.
Notes	Opcode checks apply to bytes **after** `0xFD` . A payload that is only `0xFD` is malformed. `0xFD` + `STOP` (`0x00`) is valid (empty runtime).

R3 — Bytecode size cap

rule_id	`MAX_BYTECODE_SIZE`
Default limit	**32 768 bytes** (32 KiB)
Message	`Bytecode size {n} exceeds maximum {max}`
Governance	`RuleRegistry.max_bytecode_size`

R4 — Opcode whitelist

rule_id	`INVALID_OPCODE`
Message	`Invalid opcode at offset {n}`
Test	Every instruction byte (not `PUSH` immediates) must be in the table in §5.

`PUSH` immediate bytes are **data**, not opcodes, and are not whitelist-checked.

R5 — Well-formed instruction stream

rule_id	`MALFORMED_BYTECODE`
Messages	`Truncated PUSH immediate at offset {n}` · `Trailing bytes or malformed instruction stream at offset {n}`
Test	Linear scan: each `PUSH1` – `PUSH32` consumes exactly 1–32 following bytes; scan ends exactly at `len`.

Not checked at QA time: jump targets landing on instruction boundaries, "no jump into PUSH data," or runtime stack safety. Those may fault in the VM later; they are **not** deploy-time rejects today.

R6 — Bytecode hash blocklist

rule_id	`BLOCKLIST_MATCH`
Message	`Bytecode matches blocklist (known scam/malware)`
Test	BLAKE3-256 of the full payload ∈ `registry.blocklist`
Canonical / live default	empty list

R7 — Purpose category, when provided

rule_id	`PURPOSE_DECLARATION_INVALID`
Message	`Invalid purpose category '{cat}'; valid: dApp, token, NFT, meme, community, entertainment, tooling, other`
Test	If purpose is `Some` and trim is non-empty, it must match a valid category **case-insensitively**.
Missing purpose	**Allowed.** Omitted purpose is not a reject.

Valid categories (canonical spellings; matching is lowercase):

`dapp` / `dApp` · `token` · `nft` / `NFT` · `meme` · `community` · `entertainment` · `tooling` · `other`

Meme, community, and entertainment are first-class. "No traditional utility" is **not** a reject and **not** Unsure.

There is no `scam` or `malware` category; declaring one is invalid → Reject.

R8 — Known-bad byte patterns

rule_id	`SCAM_PATTERN_MATCH`
Message	`Bytecode contains known scam/malware pattern`
Test	Any non-empty `registry.scam_patterns` entry is a contiguous window of bytecode
Canonical / live default	empty list

There is **no** fuzzy similarity score. Only exact hash (R6) and exact byte-sequence (R8) matches reject for “known malware.”

R9 — Metadata length

rule_id	`METADATA_TOO_LONG`
Limits	`asset_name` ≤ **256** UTF-8 bytes · `asset_symbol` ≤ **32** UTF-8 bytes
When	Only if those fields are present on `ContractDeployWithPurposeAndMetadata`

R10 — Content policy (name / symbol)

rule_id	`CONTENT_POLICY_VIOLATION`
Message	`Deployment metadata contains governance-forbidden content (vulgarity/offensiveness policy)`
When	`asset_name` and/or `asset_symbol` present **and** `content_blocklist` is non-empty
Intended public-testnet list	132 English terms (Appendix A)
Live public RPC on 26 Aug 2026	`content_blocklist: []` — this rule is **inert** until the registry is applied

Matching algorithm (§7) is part of the rule, not a guideline.

5. Allowed opcodes

Any other opcode byte is **Reject** (`INVALID_OPCODE`). `PUSH1` – `PUSH32` are the contiguous range `0x60` – `0x7F` .

Byte	Name	Byte	Name
`0x00`	STOP	`0x01`	ADD
`0x02`	SUB	`0x03`	MUL
`0x04`	DIV	`0x06`	MOD
`0x08`	ADDMOD	`0x09`	MULMOD
`0x10`	LT	`0x11`	GT
`0x14`	EQ	`0x15`	ISZERO
`0x16`	AND	`0x17`	OR
`0x18`	XOR	`0x19`	NOT
`0x1B`	SHL	`0x1C`	SHR
`0x1D`	SAR	`0x30`	ADDRESS
`0x33`	CALLER	`0x40`	BLOCKHEIGHT
`0x41`	TIMESTAMP	`0x51`	MLOAD
`0x52`	MSTORE	`0x54`	SLOAD
`0x55`	SSTORE	`0x56`	JUMP
`0x57`	JUMPI	`0x60` – `0x7F`	PUSH1–PUSH32
`0x80`	DUP1	`0xA0` – `0xA4`	LOG0–LOG4
`0xF1`	CALL	`0xF3`	RETURN
`0xF5`	CREATE2		

6. Soft / policy rules (Unsure, not Reject)

These never auto-Allow and never auto-Reject. They send the signed deploy to the **community QA pool**.

U1 — Always-review category

If the (trimmed, lowercased) purpose is in `registry.always_review_categories` → **Unsure**.

Canonical and live default: **empty**. Governance may add high-stakes categories. **meme, token, NFT, dApp, tooling, community, entertainment are not always-review** unless governance puts them there.

U2 — `other` with almost no description

If purpose is `other` and `description_hash` length is **less than 4 bytes** (including omitted) → **Unsure**.

If `other` is paired with a description hash of **4 or more bytes** → this soft rule passes (Allow, provided all hard rules pass).

Meme / community / entertainment with a short or empty description is **Allow**, not Unsure.

7. Content-policy matching (R10)

Inputs: `asset_name` and `asset_symbol` (each checked independently). Comparison is **case-insensitive** after trim. Empty blocklist → skip.

For each forbidden term `t`:

1. If `t` is empty, skip.
2. If `t` contains **any non-alphanumeric character** (spaces, punctuation): **substring** match. Examples: `kill yourself`, `child porn`, `go die`.
3. If `t` is alphanumeric **and length ≥ 4**: **substring** match. Example: `shit` matches `ShitCoin`.
4. If `t` is alphanumeric **and length < 4**: **whole-token** match only (tokens split on non-alphanumeric). Example: `ass` does **not** match `Classic`; it would match the token `ass`.

The reject message does **not** name which term matched.

Content policy **does not** scan bytecode, purpose text, or off-chain NFT URIs. Names stored only in contract storage after deploy are **not** seen at gate time.

8. What is required for Allow

All of the following:

1. Every hard rule R1–R10 that applies to the payload **passes**.
2. Purpose is omitted, empty, or a valid category (R7).
3. Purpose is **not** in the always-review set (U1).
4. If purpose is `other`, `description_hash` is at least 4 bytes (U2).
5. Registry lists that are empty (typical defaults) simply do not fire R6, R8, R10, or U1.

Passing QA **Allow** means the tx may enter the mempool and, if included, execution will install code. It is not a review of token economics, off-chain websites, or later storage writes.

9. Community pool (Unsure → Allow or Reject)

Public-testnet file: `docs/config/qa_pool_config.public-testnet.json`.
Production-style file: `docs/config/qa_pool_config.canonical.json`.

Live `boing_qaPoolConfig` on 26 August 2026:

Knob	Live public testnet
Accepts new pending	true
Max pending items	256
Max pending per deployer	16
Review window	604 800 s (7 days)
Quorum fraction	0.5
Allow threshold	2/3 of counted Allow+Reject votes
Reject threshold	2/3 of counted Allow+Reject votes
Default on expiry	**reject**
Dev open voting	**true**
Governance admins	0

Production defaults (`QaPoolGovernanceConfig::production_default`) differ: `public_membership: true`, `dev_open_voting: false`, `max_pending_items: 32`, `max_pending_per_deployer: 2`, `min_quorum_votes: 3`, `reward_per_counted_vote: 1` BOING from `PROTOCOL_TREASURY`.

Voting rules (code):

- Votes: Allow, Reject, Abstain. Later vote from the same account **overwrites**.
- **Deployer cannot vote on their own item.**
- Abstain does **not** count toward quorum and does **not** pay unless `pay_abstain` is true (default false).
- Public membership: any 32-byte account may vote, subject to optional `min_voter_stake` (0 = off). Electorate size for quorum math is `min_quorum_votes` (default 3), not “everyone on earth.”
- Quorum: `(allow + reject) / electorate ≥ quorum_fraction`, then Allow if `allow / (allow+reject) ≥ 2/3`, else Reject if reject ratio ≥ 2/3, else still pending.
- On **Allow**: stored bytecode is installed; counted voters may be paid from the treasury.
- On **Reject** or **expiry with default reject**: pending record dropped; no contract created.

- Pool full → new Unsure submissions refused (`-32055` global, `-32056` per deployer). `max_pending_items = 0` disables the pool (`-32054`).

Pool bar (Appendix B of the design spec). Designed to reduce malicious, deceptive and low-quality deployments through transparent, protocol-level checks. Reviewers should reject phishing, rug-pulls, malware, impersonation, deceptive naming, Ponzi patterns, and spam/abuse. **Not a reason to reject:** meme, experimental, unconventional culture, failed/unpopular projects. When in doubt, pool guidance is **reject**. Discriminating against memes as a class is contrary to protocol policy. Outcomes stay visible on Boing Observer.

10. RPC error codes (QA)

Code	Meaning
`-32050`	Rejected by protocol QA (`rule_id` + `message`)
`-32051`	Unsure — referred to the pool (`tx_hash`)
`-32052`	No pending pool item for that hash
`-32053`	Voter ineligible (not a member / deployer conflict / stake)
`-32054`	Pool disabled
`-32055`	Global pending capacity reached
`-32056`	Per-deployer pending cap reached
`-32057`	Operator RPC auth (`X-Boing-Operator`)

Pre-flight: `boing_qaCheck` params `[hex_bytecode, purpose_category?, description_hash?, asset_name?, asset_symbol?]`.

11. Explicitly not required / not enforced at the gate

The following are **not** reasons to Reject or Unsure in shipped automation:

- Missing purpose category.
- Meme / community / entertainment with little or no description.
- Novel bytecode that does not match a blocklist hash or scam pattern (new patterns **Allow** if hard rules pass).
- Jump-target alignment, stack-depth, or “looks suspicious” heuristics (not implemented).
- Token/NFT ABI layout (`BOING-REFERENCE-TOKEN.md` / `BOING-REFERENCE-NFT.md`) — interoperability convention, not a deploy reject. NFT template **v3** ships `mint_batch` (`0x06` , n≤500); that is calldata/bytecode convention, not a QA rule. Official NFT↔fungible **linked pairs** are a separate on-chain registry (`BOING-LINKED-NFT-TOKEN.md`); QA does not validate or mint those links at deploy time.
- Off-chain metadata, websites, social accounts.
- Vulgarity **inside bytecode** or storage (only deploy-time name/symbol vs `content_blocklist`).
- Upgrade-proxy *product* pattern of a hub that `CALL` s a separately QA’d implementation (permitted). Opaque patterns meant to hide what users execute may be judged by the pool under malice rules, not a dedicated opcode ban.

12. Deployer checklist

1. Keep bytecode ≤ 32 KiB; use only the opcode table in §5; do not truncate `PUSH` immediates.
2. Prefer a valid purpose (`meme` is fine). If you use `other`, attach a description hash of at least 4 bytes.
3. To have names checked (and later indexed cleanly), use `ContractDeployWithPurposeAndMetadata` with `asset_name` / `asset_symbol` within length limits and not matching Appendix A once that list is live.
4. Call `boing_qaCheck` before `boing_submitTransaction`.
5. If you receive Unsure, the item is on boing.observer/qa until the pool or the 7-day window resolves it.

Appendix A — English content blocklist (intended public-testnet policy)

Source: `docs/config/qa_content_blocklist.en.json` (132 terms). Merged into the registry by `npm run apply-public-testnet-qa-policy`. **Live RPC may still have an empty list** — confirm with `boing_getQaRegistry`.

Matching uses §7. Terms with spaces use substring match. Alphanumeric terms of length ≥ 4 also match inside larger words (`shit` → `ShitCoin`). Short alphanumeric terms match whole tokens only.

a55hole, asshole, assholes, b1tch, badass, bastard, bastards, beaner, beaners, bestiality, bitch, bitches, bitchy, blowjob, bollocks, bukkake, bullshit, child porn, childporn, chink, chinks, clitoris, cocksucker, cocksuckers, cunt, cunts, dilf, dipshit, douche, douchebag, douchebags, dumbass, dyke, dykes, ejaculate, ejaculation, faggot, faggots, fagot, fck, fuck, fucked, fucker, fuckers, fuckface, fuckhead, fucking, fuckwit, fuk, fvck, gangbang, go die, gook, gooks, handjob, heil, hentai, hitler, jackass, kike, kikes, kill yourself, kys, masturbate, masturbation, mfucker, milf, motherfucker, motherfuckers, nazi, nazis, nigga, niggas, nigger, niggers, onlyfans, orgasm, paedophile, paedophilia, pedophile, pedophilia, penis, penises, phuck, porn, pornhub, porno, pornography, pussies, pussy, raghead, retard, retardation, retarded, rimjob, sandnigger, scrotum, sh1t, shit, shitbag, shitface, shithead, shits, shitshow, shitty, slut, sluts, slutty, spic, spics, swastika, testicle, testicles, towelhead, trannies, tranny, twat, twats, vagina, vaginas, vulva, wanker, wankers, wetback, wetbacks, whitepower, whore, whores, whoring, xvideos, xxx, zoophilia.

Appendix B — Canonical malice (pool voting bar)

Reject (or never Allow) assets that are malicious, deceptive, or low-quality: **phishing**, **rug-pulls**, **malware**, **impersonation**, **deceptive naming**, **Ponzi patterns**, or **spam/abuse**.

Not malice: **meme assets**, **experimental/novel** code that is not deceptive, **unconventional art or culture**, **failed or unpopular** projects.

Appendix C — Governance knobs

Mutable via proposal key `qa_registry` or operator `boing_operatorApplyQaPolicy`:

- `max_bytecode_size`
- `blocklist` (32-byte BLAKE3 hashes)
- `scam_patterns` (raw byte sequences)
- `always_review_categories`
- `content_blocklist`

Pool knobs: proposal key `qa_pool_config`.

Boing Observer hosts this catalog at boing.observer/qa/rules. The live registry on the RPC you trust is the authority for list contents.