EnvSpec Naming

An open standard for semantic naming of IT resources.

1. Purpose of the Standard

The standard establishes a single coordinate system for an organization's IT resources: the hierarchical model "environment → perimeter → instance → slot → node", the rules for building network names (DNS/FQDN), cryptographic identifiers (SPIFFE), identifiers for flat environments (Kubernetes Namespaces, cloud projects, DBMS objects), and mandatory inventory tags.

The standard addresses the following tasks:

- defines a single semantics of concepts for business, development, operations, and information security
- makes names and tags machine-readable for inventory (CMDB), cost accounting (FinOps), and security access policies
- prohibits encoding mutable properties (versions, SLA, sites) in resource names
- defines the position of platform services, employee workplaces, and external systems in the trust model

The standard does not define access approval processes, the composition of security policies, or the order of their enforcement.

2. Terms and Definitions

> **KEY WORDS**
>
> The key words **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** are to be interpreted as described in RFC 2119.

- **Digital Asset** — an information system, service, or platform component registered in the corporate registry under a unique mnemonic code.

- **Environment** — a top-level trust mode defined by two attributes: *who the consumers are* and *what data is processed*. The nature of the data is unambiguously determined by the environment.
- **Perimeter** — a named collection of instances within one environment, joined for end-to-end interaction under a specific business or technical purpose (a product line, a compliance perimeter, a pilot, platform services). A perimeter is a unit of management.
- **Instance** — a deployed copy of a digital asset inside a perimeter, including its executable part and its own resources.
- **Slot** — a functional part of an instance. A logical layer or a release group (`api`, `worker`, `db`, `ui`, `v2`). Used to separate node roles and to deploy versions in parallel (blue-green, canary releases).
- **Node** — a unit of compute infrastructure running an instance's slot: a virtual machine, a physical server, a cluster worker node.
- **Site** — the physical location of a node: a data center, a cloud region, an availability zone. Not part of the hierarchy.
- **Segment** — a technical means of resource isolation at the network (VLAN, subnet, firewall), platform (Network Namespace, Service Mesh), or logical (Kubernetes Namespace, access tag) level. A segment implements the boundaries of an environment, perimeter, instance, slot, or node but does not define their logic. Not part of the hierarchy. Co-locating resources in one segment does not make them one instance, slot, or perimeter.
- **Subject** — an employee or an automated process accessing digital assets. Identified by a corporate directory account. Not part of the hierarchy.
- **Workplace** — a device or session from which an employee accesses digital assets: a workstation, a laptop, a mobile device, a VDI session.
- **Access Gateway** — an instance through which subjects gain access to an environment's digital assets: a published user interface, an API gateway, bastion/PAM, a VDI broker, VPN, a ZTNA proxy.
- **External System** — a digital asset outside the organization's zone of control: SaaS, a counterparty's system, a government system.

If components of a single information system must be placed in different trust modes (for example, a DMZ and the internal perimeter), such components MUST be registered in the corporate registry as separate digital assets with their own unique mnemonic codes (for example, `{system}-ui` and `{system}-api`).

3. Hierarchical Model

The standard is based on a strict five-level hierarchy that follows the "from specific to general" principle:

```
Environment (prod)                          ← trust mode and nature of
data
└── Perimeter (dmz)                         ← logical perimeter for a
purpose
    └── Instance (api-management)           ← copy of a digital asset
        └── Slot (websocket-ingress)        ← role or release slot
inside the instance
            └── Node (ingress-01)           ← compute unit
```

The hierarchy is strict: a node belongs to exactly one slot, a slot to exactly one instance, an instance to exactly one perimeter, a perimeter to exactly one environment. Site, subject, and workplace are not levels of the hierarchy.

3.1. Environments

The standard fixes six environment identifiers: `dev`, `test`, `stage`, `prod`, `infrastructure`, `workplace`. The list of environments MUST NOT be extended: the need for a special environment (a pilot, a sandbox, load testing, an archive, acceptance) MUST be expressed as a perimeter inside one of the environments.

The environments `dev`, `test`, `stage`, `prod` form a linear scale of increasing trust mode:

```
dev < test < stage < prod
```

The environments `infrastructure` (3.2) and `workplace` (3.3) are not part of the linear scale.

Environment membership criterion. Membership is determined by consumers and the nature of the data, NOT by hardware capacity, site, cost, or the operational stage of the information system:

Identifier	Primary consumers	Nature of data
`dev`	Developers	Synthetic data
`test`	Testing teams (QA), adjacent integration teams	Synthetic or anonymized data
`stage`	Release engineers, acceptance teams working on anonymized data	Configuration mirrors `prod`. Synthetic or anonymized data only
`prod`	End users, automated business processes, authorized personnel with access rights to real data (acceptance, migrations, support)	Real (production) data
`infrastructure`	Engineering personnel (DevOps, support, security), automated delivery, observability, and access management processes	Technical data of all environments: secrets, artifacts, logs, metrics, backups, accounts. Protection mode — not lower than `prod`
`workplace`	Employees of all roles	Stores no landscape data of its own. Environment data is accessible only through access gateways (3.5)

- The nature of data is a function of the environment. A perimeter processing real data belongs to `prod` by definition, whatever it is called and whoever it is created for: this applies equally to acceptance (UAT), trial operation, and technical migrations on real data. In particular, a pilot that reads real data without the right to modify it MUST run in a separate `prod` perimeter, not in `stage` "as an exception".
- Real data MUST NOT enter `dev`, `test`, and `stage` without prior anonymization.
- The environment does not define criticality, SLA, capacity, or redundancy composition. A `prod` perimeter with low criticality and a single replica is acceptable. Weakening the protection measures for real data, which are uniform across all `prod` perimeters, is unacceptable.

3.2. The `infrastructure` Environment

The environment of platform digital assets serving all environments of the landscape: CI/CD, artifact registries, secret stores, corporate directories (IdP/AD), monitoring and log collection, backup, configuration management, workplace management.

- The trust mode of `infrastructure` is not lower than `prod`: a compromise of a platform instance is equivalent to a compromise of `prod`. Protection measures for `infrastructure` instances MUST NOT be weaker than those for `prod`.
- An `infrastructure` instance MUST NOT serve as a data transfer channel between environments of the linear scale (for example, `prod` → `test` through a shared broker, registry, or log store). Data of different environments inside an `infrastructure` instance MUST be separated by means of the asset itself (tenants, indexes, buckets, projects) while preserving the source environment attribute.
- An `infrastructure` instance MAY host support systems and information security systems (SIEM, EDR, DLP, PAM). Collection, processing, and storage of security audit events MUST be performed in `prod` by the environment's digital assets (including platform ones).
- Copies of platform systems created for developing and testing those systems themselves are placed as regular instances in `dev`, `test`, `stage` (`vault.secrets.test`), not in `infrastructure`.
- For platform automation and data storage systems (backup, log collection, object storage), an architectural split is allowed: management components (Control Plane) are placed in `infrastructure`, while components storing production data (Data Plane / Repositories) are placed in perimeters of the corresponding target environments (for example, `prod`) as part of their infrastructure.

3.3. The `workplace` Environment

The environment of employee workplaces: workstations, laptops, mobile devices, VDI sessions, as well as the infrastructure that directly serves them (VDI brokers, terminal farms, VPN concentrators).

- A workplace is an untrusted device. It MUST NOT be granted trust based on its network location and MUST NOT be a storage location for data of the linear-scale environments and `infrastructure`.

- An employee's role (development, testing, DevOps, support, security, internal user of information systems) MUST NOT be encoded in the environment, perimeter, or workplace name. A subject's rights are determined by their account and roles in the corporate directory (an `infrastructure` instance), not by the workplace.
- Access from a workplace to digital assets of any environment, including `infrastructure`, is performed only through that environment's access gateway (3.5).
- A workplace's `{perimeter}` is a group of workplaces by management and connection method (for example `office`, `remote`, `vdi`, `kiosk`); `{system}` is the workplace class code from the digital asset registry (for example `arm`, `laptop`, `vdi`). The FQDN template (4.3) MAY be applied to workplaces; tagging (section 6) is mandatory.
- An exception to the untrusted-workplace rule is dedicated technology management workstations (network engineers, security administrators) and Air-Gap terminals. Such devices MUST be separated into dedicated perimeters of the `workplace` environment (for example, `net-ops`, `sec-ops`, `airgap`). Network access to critical infrastructure (network equipment, system cores) MUST be restricted at the firewall level based on membership in these technology perimeters.
- For workstations located in physically isolated segments (Air-Gap) with no access to the corporate directory service and DNS, the canonical FQDN template is applied locally, replacing the corporate suffix `{domain}` with a local one (for example, `.local`). At the same time, tagging with the mandatory inventory tags in the corporate CMDB is strictly required, and the `envspec.io/site` tag MUST contain the exact physical location (building, room, secure area).

3.4. Reserved Perimeters `platform` and `external`

Two perimeter identifiers are reserved in every environment. They MUST NOT be used for other purposes.

- **`platform`** — the perimeter of the environment's platform digital assets serving several perimeters of the same environment: shared DBMS clusters, message brokers, ESB and API gateways, orchestration clusters, application authentication services, access gateways, security systems, and so on. A digital asset serving several perimeters is placed once in `platform`; instances of other perimeters access it as clients. Consumer separation is performed by means of the asset itself (schemas, topics, namespaces), not by splitting the instance across perimeters.

- **`external`** — the perimeter representing external systems coupled with the given environment. An external system receives the identifier `{system}.external.{env}` in the environment whose data it exchanges: a counterparty's production API — in `external.prod`, its sandbox — in `external.test`. The identifier serves local DNS aliases, gateway rules, tags, and declarations; it is not the external system's own name. The organization's instances MUST NOT be placed in the `external` perimeter.

3.5. Access Gateways

- Subjects at workplaces and external systems access an environment's digital assets only through access gateways. Direct network connectivity between workplaces or external systems and instance nodes, bypassing the gateway, is PROHIBITED.
- An access gateway is an instance of the environment it opens access to, placed in its `platform` perimeter or in the perimeter of the served system: `bastion.platform.prod`, `ingress.platform.prod`, `apigw.platform.prod`, `sso.platform.infrastructure`. VPN concentrators and VDI brokers extend the `workplace` environment to remote devices and belong to `platform.workplace`.
- A gateway MUST authenticate the subject by their corporate directory account. The matrix "subject role → environment → digital asset" is a matter of the organization's policy and is not defined by the standard; the standard fixes the entry point (the gateway) and the source of identity (the directory).

3.6. Trust Rules Between Environments

1. Instances of different linear-scale environments (`dev`, `test`, `stage`, `prod`) MUST NOT interact directly.
2. `infrastructure` is the only environment whose instances are allowed to interact with instances of any environment. Two classes of interaction are allowed: **management** (`infrastructure` → environment: delivery of artifacts and configuration, issuance of secrets, management of accounts and workplaces) and **telemetry** (environment → `infrastructure`: logs, metrics, traces). Transit restriction — per 3.2.
3. `workplace` → any environment, including `infrastructure`: only through the target environment's access gateway (3.5).

4. `external.{env}` ↔ instances of `{env}`: only through the access gateway (ingress/egress) of environment `{env}`. Interaction of `external.{env}` with instances of other environments is PROHIBITED: a `prod` instance MUST NOT call `*.external.test`, and vice versa.
5. The procedure for formalizing and approving exceptions to rules 1–4 is outside the scope of the standard.

3.7. Orthogonality of Physical Placement

A site (data center, cloud region, availability zone) is an orthogonal "where" dimension. A node resides at exactly one site; a perimeter and an instance MAY span several. The site is recorded in inventory attributes (CMDB, tags). The site code MUST NOT be included in network names and resource identifiers. Moving a node between sites must not lead to a change of its name.

4. Naming Convention

4.1. Format of Name Components

- Every name component MUST match the pattern `^[a-z0-9]([a-z0-9-]*[a-z0-9])?$`: lowercase Latin letters, digits, hyphens inside the label.
- The underscore character (`_`), uppercase letters, and special characters in components are PROHIBITED (the exception is the flat-environment separator per section 5).
- The length of a single component MUST NOT exceed 63 characters.
- `{perimeter}` and `{system}` MUST be no longer than 13 characters: this guarantees that all flat forms of section 5 fit within the 63-character limit. For platforms with stricter limits, the organization MUST set a smaller limit.
- A decomposed `{system}` mnemonic code (for example, `{system}-ui`, `{system}-api` per section 2) MAY reach 16 characters as long as the flat forms of section 5 still fit within the 63-character limit.
- Hyphens SHOULD NOT be used in `{perimeter}` and `{system}`: this way, flat forms with a hyphen separator (5.2, 5.3) decompose into components unambiguously.
- `{env}` is strictly one of the identifiers: `dev`, `test`, `stage`, `prod`, `infrastructure`, `workplace`.

- Uniqueness: `{perimeter}` is unique within an environment, `{system}` within the digital asset registry, `{slot}` within an instance, `{node}` within a slot.
- Components strictly follow the hierarchical order **from specific to general**; the suffix always determines the environment.

4.2. Prohibition of Mutable Properties

Encoding mutable operational or organizational characteristics in resource names is PROHIBITED. For example, SLA parameters, criticality classes (~~tier1~~, ~~prod-critical~~), lifecycle statuses (~~legacy~~, ~~old~~), hardware technical parameters, site, employee role.

A change of criticality, topology, or site MUST NOT lead to renaming a resource or changing its DNS record. These properties are maintained in tags and inventory, not in names.

Major versions and release groups (blue-green, canary releases) MUST be expressed exclusively through the `{slot}` component (`app-01.v2.antifraud.…`) and MUST NOT be glued to the system or perimeter code (~~antifraud-v2~~).

4.3. Network Names (DNS / FQDN)

The canonical FQDN template of an infrastructure node:

```
{node}.{slot}.{system}.{perimeter}.{env}.{domain}
```

- `{node}` — node name: functional role and sequence number (`app-01`, `db-02`, `worker-11`).
- `{slot}` — functional component or release layer of the instance (`api`, `worker`, `db`, `ui`, `v2`). If the instance requires no internal division, the component MUST take the value `main`. The number of name segments is fixed.
- `{system}` — mnemonic code of the digital asset from the corporate registry.
- `{perimeter}` — perimeter identifier, including the reserved `platform` and `external` (3.4).
- `{env}` — environment identifier (4.1).
- `{domain}` — base corporate DNS zone (`example.ru`).

Reference examples:

FQDN	Reads as
`db-01.main.billing.payments.prod.example.ru`	DBMS node #1 of the `billing` instance in the `payments` perimeter of the `prod` environment
`app-02.v2.antifraud.risk.test.example.ru`	second node of release slot `v2` of the `antifraud` instance in the `risk` perimeter of the `test` environment
`runner-24.main.jenkins.cicd.infrastructure.example.ru`	runner #24 of the `jenkins` instance in the `cicd` perimeter of the `infrastructure` environment
`pg-01.main.pgcluster.platform.prod.example.ru`	node #1 of the shared PostgreSQL cluster in the platform perimeter of `prod`
`bastion-01.main.pam.platform.prod.example.ru`	privileged access gateway in `prod`
`api.main.companyid.external.prod.example.ru`	local alias of the integration point of a counterparty's production API coupled with `prod`
`mgmt-arm-01.main.arm.net-ops.workplace.example.ru`	dedicated workstation #1 in slot `main` for network management (`mgmt-arm`) in the network engineers' perimeter (`net-ops`) of the workplace environment
`term-02.main.terminal.airgap.workplace.local`	isolated Air-Gap terminal #2 in the `airgap` perimeter of a

FQDN	Reads as
	local landscape

Scope of application. The template is mandatory for infrastructure nodes (virtual machines, physical servers, cluster worker nodes). Ephemeral workloads of orchestrated platforms (pods, jobs) do not receive an FQDN by the template: their hierarchy membership is carried by the platform's namespace and labels (section 5) and the SPIFFE ID (4.4). A name in the `external` perimeter is a local alias (a CNAME or a gateway address), not the external system's name. Geographic DNS zones, if needed, are implemented as technical aliases on top of the canonical name, not as part of it.

4.4. Cryptographic Names (SPIFFE / Workload Identity)

When mutual authentication (mTLS) or security tokens (JWT) are used, the name hierarchy MUST be translated into SPIFFE identifiers:

```
spiffe://{trust-
domain}/env/{env}/perimeter/{perimeter}/system/{system}/slot/{slot}
```

`{trust-domain}` is the organization's trust domain (`example.ru`); the remaining components strictly correspond to the components of the DNS name.

Encoding the identifier into security artifacts:

- **X.509-SVID:** the identifier is written into the Subject Alternative Name (SAN) extension of type URI; the Subject Common Name (CN) field duplicates the node's canonical FQDN. Example: CN `db-01.main.billing.payments.prod.example.ru`, SAN URI `spiffe://example.ru/env/prod/perimeter/payments/system/billing/slot/main`.
- **JWT-SVID:** the identifier is written into the mandatory `sub` field. Example: `"sub": "spiffe://example.ru/env/prod/perimeter/payments/system/billing/slot/main"`.

Issuance rules:

- The identity issuer MUST NOT issue a SPIFFE ID based solely on data from the workload. The `env`, `perimeter`, `system` components MUST be confirmed by the hosting platform's metadata (mandatory tags, Kubernetes namespace and labels). A resource tagged `env: test` cannot receive an identifier with `/env/prod/`.
- External systems (`external`) MUST NOT be issued identifiers of the organization's trust domain.
- Workplaces (`workplace`) MUST NOT be issued identifiers by the template. The subject's identity is a corporate directory account.

Special platform tooling (Service Mesh) is not required to apply the rule. The identifier is a string assembled by the existing certificate issuer (HashiCorp Vault, a local PKI) and verified by a regular expression in application code or web server configuration.

Identification of workplaces (`workplace`) and subjects using X.509 cryptographic certificates (including hardware tokens and smart cards) follows these rules:

- Employees' personal certificates identify the subject, are bound to the corporate directory (owner CN, UserPrincipalName fields), and do not use the infrastructure's hierarchical template.
- Device certificates (machine certificates) for workstations and terminals MUST contain the workplace's canonical FQDN built by the rules of 4.3 (for example, `CN=mgmt-arm-01.main.arm.net-ops.workplace.example.ru`).
- Access gateways MUST validate device machine certificates and match their perimeter (`{perimeter}`) against the subject's access rights. Access to infrastructure management from devices whose certificates do not belong to trusted technology perimeters (for example, `net-ops` or `sec-ops`) MUST be automatically blocked at the gateway.

5. Projection onto Flat Execution Environments

If the target platform does not support a hierarchical dotted structure (Kubernetes Namespaces, cloud projects, DBMS objects, resource tags), the dot separator is replaced with `-` or `_`. The component order **from specific to general** MUST be preserved. Gluing components together and changing their order (moving the environment to the front) is PROHIBITED.

When translating into flat containers, the `{node}` and `{slot}` components are omitted: the platform manages objects inside its own perimeter on its own.

5.1. Translation Matrix

Platform / resource type	Separator	Canonical format
Network names (DNS / FQDN)	`.`	`{node}.{slot}.{system}.{perimeter}.{env}.{domain}`
Cryptographic identifier (SPIFFE ID)	`/`	`spiffe://{trust-domain}/env/{env}/perimeter/{perimeter}/system/{sy`
Kubernetes Namespace	`-`	`{system}-{perimeter}-{env}`
Cloud project / Resource Group	`-`	`{perimeter}-{env}`
Database / DBMS schema	`_`	`{system}_{perimeter}_{env}`
DBMS user / role	`_`	`user_{system}_{perimeter}_{env}`

5.2. Kubernetes Namespaces

A namespace is the projection of an instance: `{system}-{perimeter}-{env}`.

- `billing-payments-prod`
- `antifraud-risk-test`
- `vault-secrets-infrastructure`

- `pgcluster-platform-prod`

Inverted forms (~~`prod-payments-billing`~~) are PROHIBITED.

An orchestration cluster is a digital asset in its own right and an instance of the `platform` perimeter of its environment (`kube1.platform.prod`); its worker nodes are named per 4.3 (`worker-11.main.kube1.platform.prod.example.ru`). Consumer namespaces are instances of other perimeters hosted on the cluster. The "namespace → cluster" relationship is maintained by the platform inventory and MUST NOT be included in the namespace name. A namespace MUST NOT contain instances of different environments. A cluster MUST NOT host instances of different environments without platform-level isolation.

5.3. Cloud Projects / Resource Groups

A cloud provider's logical resource container is the projection of a perimeter: `{perimeter}-{env}`.

- `payments-prod`
- `pilot-stage`
- `cicd-infrastructure`

5.4. Databases and DBMS Users

Names of logical databases (schemas) and DBMS system accounts are assembled with the `_` separator: `{system}_{perimeter}_{env}` and `user_{system}_{perimeter}_{env}`.

- `billing_payments_prod`
- `antifraud_risk_test`
- `user_billing_payments_prod`

6. Identification Attributes

Every IT resource, regardless of hosting platform (virtual machines, cloud resources, Kubernetes pods and namespaces, workplaces), MUST be tagged with three mandatory

tags (metadata keys) that duplicate the name's semantics for inventory, metrics collection, and FinOps:

Tag	Value
`envspec.io/env`	Environment identifier: `dev`, `test`, `stage`, `prod`, `infrastructure`, `workplace`
`envspec.io/perimeter`	Perimeter identifier
`envspec.io/system`	Mnemonic code of the digital asset from the corporate registry

Optional tags: `envspec.io/slot` — slot; `envspec.io/site` — site code (3.7).

If the platform does not allow the `/` and `.` characters in keys, the keys MUST be written as `envspec_env`, `envspec_perimeter`, `envspec_system` (`envspec_slot`, `envspec_site`).

The tag values of a resource whose name is built per 4.3 or section 5 MUST match the corresponding name components. A mismatch is a violation of the standard.

> ⚠ **ON COMPATIBILITY WITH LEGACY SYSTEMS**
>
> For legacy and architecturally complex systems whose network names cannot be changed without breaking the landscape, tagging takes priority over renaming:
>
> 1. Renaming the existing network names (DNS/FQDN) of legacy hosts is NOT a requirement of the standard. Names may remain unchanged until planned decommissioning or migration.
> 2. To bring such a system into compliance with the standard, it is SUFFICIENT to tag it with the three mandatory tags at the virtualization platform or cloud level.
> 3. Correct tags make the system transparent to automation and inventory and fully compensate for the legacy structure of its DNS name.

7. Compliance Criteria

To claim compliance with the **EnvSpec Naming 1.0.x** specification, an organization's IT infrastructure or its isolated landscape MUST satisfy the following automatically verifiable criteria:

1. **Tagging completeness.** 100% of active compute resources, workplaces, and container spaces (Namespaces) have the mandatory `envspec.io/*` tags filled in.
2. **Identifier validity.** 0 resources with an `envspec.io/env` value outside the list in 3.1. The perimeter identifiers `platform` and `external` are used only per 3.4.
3. **DNS validity for new resources.** 100% of internal DNS zone records created after the standard's adoption date and pointing to infrastructure nodes match the template in 4.3, including the mandatory `{slot}` component.
4. **Legacy system audit.** All systems whose DNS names do not match the template in 4.3 have 100% coverage by the mandatory tags (section 6). This is recognized as full compliance with the standard.
5. **Name cleanliness.** Names of new resources contain no software version markers (other than the `{slot}` component), SLA parameters, criticality classes, site codes, or employee roles.
6. **Layer consistency.** For 100% of resources that simultaneously have an FQDN per 4.3, a SPIFFE ID per 4.4, and the mandatory tags, the `env`, `perimeter`, `system` components match across all three layers.

Appendix A. Reference Regular Expressions

For linters, CI/CD checks, and admission policies (Validating Webhooks in Kubernetes):

Object	Regular expression
Name component (RFC 1123, ≤ 63)	`^[a-z0-9]([a-z0-9-]{0,61}[a-z0-9])?$`
Base suffix `{perimeter}`,	`^[a-z0-9]([a-z0-9-]{0,11}[a-z0-9])?$`

Object	Regular expression
`{system}` (≤ 13)	
Decomposed suffix `{system}` (with hyphen, ≤ 16)	`^[a-z0-9]([a-z0-9-]{0,14}[a-z0-9])?$`
`{env}`	`^(dev|test|stage|prod|infrastructure|workplace)$`
Node FQDN (4.3, including the zone)	`^[a-z0-9]([a-z0-9-]{0,61}[a-z0-9])?\.[a-z0-9]([a-z0-9-]{0,61}9])?\.[a-z0-9]([a-z0-9-]{0,14}[a-z0-9])?\.[a-z0-9]([a-z0-9-]{z0-9])?\.(dev|test|stage|prod|infrastructure|workplace)\.[a-z`
SPIFFE ID (4.4)	`^spiffe://[a-z0-9.-]+/env/(dev|test|stage|prod|infrastructure|workplace)/peri z0-9]([a-z0-9-]{0,11}[a-z0-9])?/system/[a-z0-9]([a-z0-9-]{0,19])?/slot/[a-z0-9]([a-z0-9-]{0,61}[a-z0-9])?$`

Object	Regular expression
Kubernetes Namespace (5.2)	`^[a-z0-9]([a-z0-9-]{0,14}[a-z0-9])?-[a-z0-9]([a-z0-9-]{0,11}-(dev|test|stage|prod|infrastructure|workplace)$`
Cloud project (5.3)	`^[a-z0-9]([a-z0-9-]{0,11}[a-z0-9])?-(dev|test|stage|prod|infrastructure|workplace)$`
Database / schema (5.4)	`^[a-z0-9]([a-z0-9-]{0,14}[a-z0-9])?_[a-z0-9]([a-z0-9-]{0,11}_(dev|test|stage|prod|infrastructure|workplace)$`
DBMS user (5.4)	`^user_[a-z0-9]([a-z0-9-]{0,14}[a-z0-9])?_[a-z0-9]([a-z0-9-]{0z0-9])?_(dev|test|stage|prod|infrastructure|workplace)$`

The FQDN and SPIFFE ID expressions check the format and lengths of components; the admissibility of specific `{perimeter}` and `{system}` values is checked against the digital asset registry and the perimeter registry.

About This Document

The standard is versioned per SemVer: MAJOR — incompatible changes to the model or rules, MINOR — new rules and identifiers, PATCH — clarifications of wording. The specification fixes the rules of naming, tagging, and cryptographic identity. Security policy management and access approval processes are outside its scope.

Author — Andrei Ganiushkin (Ганюшкин Андрей Александрович).

Restriction on the use of the name. It is prohibited to use the names "EnvSpec", "EnvSpec Naming", "EnvSpec Governance", "EnvSpec Security", and their derivatives for

public distribution of modified versions of this document. Public derivative documents MUST be fully renamed (for example, "Company X IT Resource Naming Standard"). The EnvSpec name in modified public documents is allowed only in the context of citing the original source: "Based on the EnvSpec methodology by Andrei Ganiushkin".