

# Gateway Domain-Centric Routing

# GDCR

*The Semantic Domain Gateway of the SDIA Ecosystem*

**Ricardo Luz Holanda Viana**

Independent Solo Researcher | Enterprise Integration Architect  
SAP BTP Integration Suite Expert | SAP Press e-Bite Author — Enterprise Messaging (2021)  
rhviana@gmail.com · ORCID: 0009-0009-9549-5862

**DEIP Ecosystem DOI: 10.5281/zenodo.19004802 · CC BY 4.0 International**

Webpage: [www.domain-intent.com](http://www.domain-intent.com) (under construction) · GitHub: [github.com/rhviana/deip](https://github.com/rhviana/deip) (under reformulation)

*Edition: April 2026 — Warsaw, Poland*

*IP Note: Architectural patterns, governance framework, and naming specifications disclosed herein are publicly available prior art via timestamped Zenodo publication. This disclosure may serve as prior-art evidence under 35 U.S.C. § 102. This document does not constitute legal advice.*

*Author Note: The concepts, architectural models, governance principles, and naming structures presented in this work were conceived and developed by Ricardo Luz Holanda Viana (February / March 2026). Generative AI tools (Claude — Anthropic, Gemini, Genspark) were used solely as assistive instruments for drafting, language refinement, formatting, images, and structural organization. All substantive ideas, technical decisions, and final validation remain the sole responsibility of the author.*

## v8.0 Edition — Scope Disclaimer

GDCR and DDCR form the original foundation of this ecosystem. The first concepts (DCRP, PDCP) were published on SAP SCN from February 1 to 6, 2026 (see Appendix A). Every subsequent layer (ODCP, EDCP, DDCP, SDIA, DEIP) was built on top of the invariants first established here at the API management layer.

This edition is scoped to three artifacts owned by GDCR: **the Semantic URL, the Proxy DNA, and the Metadata Container DNA**. The routing engine logic (7-stage pipeline, action normalization, key construction, reference implementations) has been relocated to **DDCR v3.0**, where it belongs. GDCR governs the facade. DDCR governs the resolution that keeps the facade immutable.

This validation did not include direct consumption of vendor-standard ERP APIs (e.g., SAP ERP/S/4HANA OData or SOAP APIs, Workday APIs) exposed behind the gateway without an intermediate orchestration layer; therefore, conclusions regarding such scenarios are architectural inferences rather than empirically validated results.

## Abstract

GDCR represents the **semantic gateway layer (Layer 1)** of the SDIA ecosystem. It is a **vendor-neutral, metadata-driven governance model** in which **domain intent**, rather than backend system identity, is the primary key of the API facade. Its core invariant is simple: one proxy per domain, immutable gateway artifacts, semantic URLs, and backend variability externalized to a metadata control plane, with optional granularity in artifact naming.

In this specification, GDCR is limited to the gateway facade and its handoff to DDCR as the downstream resolution engine. GDCR receives the consumer's semantic request, validates it against PVRL Level 1 grammar, and forwards valid requests to DDCR (Layer 2). Its sole responsibility is to enforce the semantic gateway contract and naming convention based on domain intent under SDIA governance. The baseline validated URL format is **https://<apim>/domain/entity/action/target**, with supplementary context carried through custom headers. Optional granularity segments such as sector, division, and region are defined in the grammar as an extensibility surface but were not exercised in the validated scope. GDCR therefore provides a reference architecture for semantic gateway governance, not a prescriptive product template.

Validation was conducted across five gateway platforms — SAP API Management, AWS API Gateway, Azure API Management, Kong Gateway, and Netflix Zuul — using the same architectural setup and DDCR resolution logic, with SAP CPI as the reference backend execution platform. The validated scope comprised four domain proxies across four business domains — Sales, Procurement, Logistics, and Finance — with forty-two distinct endpoints and forty-two SAP CPI iFlows behind the facade. Complete plug-and-play artifacts for **SAP APIM and SAP CPI** are publicly available in the **GitHub repository**, while validation outputs and reference materials for the other gateway platforms are available under the **gdcr-proven folder**.

This edition intentionally excludes broader orchestration details, as well as IoT and LLM scopes, to maintain focus on the GDCR gateway contract and its deterministic metadata-driven handoff to DDCR at Layer 2. Prior to GDCR, API surfaces are indexed by technology, backend systems, or local implementation needs. With GDCR, the facade is indexed by domain intent: technology moves into the background, backend volatility is absorbed by metadata, and governance becomes measurable, scalable, and structurally stable.

## Companion Documents — DEIP / SDIA Ecosystem

Component	Full Name	Scope	DOI
DEIP	Domain Enterprise Integration Pattern	Decision model — platform binding	zenodo.19004802
SDIA	Semantic Domain Integration Architecture	Umbrella architecture (5 layers)	zenodo.18877635
<b>GDCR</b>	<b>Gateway Domain-Centric Routing</b>	<b>Layer 1 — Gateway (this document)</b>	<b>zenodo.18582492</b>
DDCR	Domain Driven-Centric Router	Layer 2 — Resolution engine	zenodo.18864832

Component	Full Name	Scope	DOI
ODCP	Orchestration Domain-Centric Pattern	Layer 3 — Orchestration	zenodo.18876593
EDCP	Event Domain-Centric Pattern	Layer 4 — Events / Messaging / Streaming	zenodo.19068766
DDCP	Data Domain-Centric Pattern	Layer 5 — Data	forthcoming

## Independent Research Disclaimer & Legal Notice

Independent research from first principles. No vendor docs. Validation in trial/sandbox only — no production data. Trademarks belong to respective owners. CC BY 4.0 — reuse with attribution.

### 1.0 The Problem — Integration Indexed by Technology

Modern enterprises think in business domains — Sales, Finance, Logistics, Procurement — but build their API gateway layer indexed by systems, vendors, and protocols. This structural mismatch produces compounding failure patterns.

Anti-Pattern	Manifestation	Consequence
Proxy Sprawl	One API proxy per backend system — 41 proxies for 4 domains. Rigid OpenAPI contract per proxy. Individual SLAs and dashboards per artifact.	90% of governance effort absorbed by proxy maintenance. One backend change cascades across unrelated proxies. Discovery becomes archaeology.
URL Topology Exposure	/salesforce-prod-347/customers/v2 — the URL reveals the backend, the environment, and the version.	Every vendor replacement breaks the consumer contract. Security surface leaks infrastructure. Governance operates on implementation, not intent.

These are not tooling problems. They are structural problems caused by a single root cause: business semantics are coupled to technical artifacts. Every new vendor, every new project adds complexity multiplicatively because the organizing principle is technology, and technology changes constantly. The domain does not.

#### 1.1 GDCR does not optimize this lifecycle. It replaces the organizing principle.

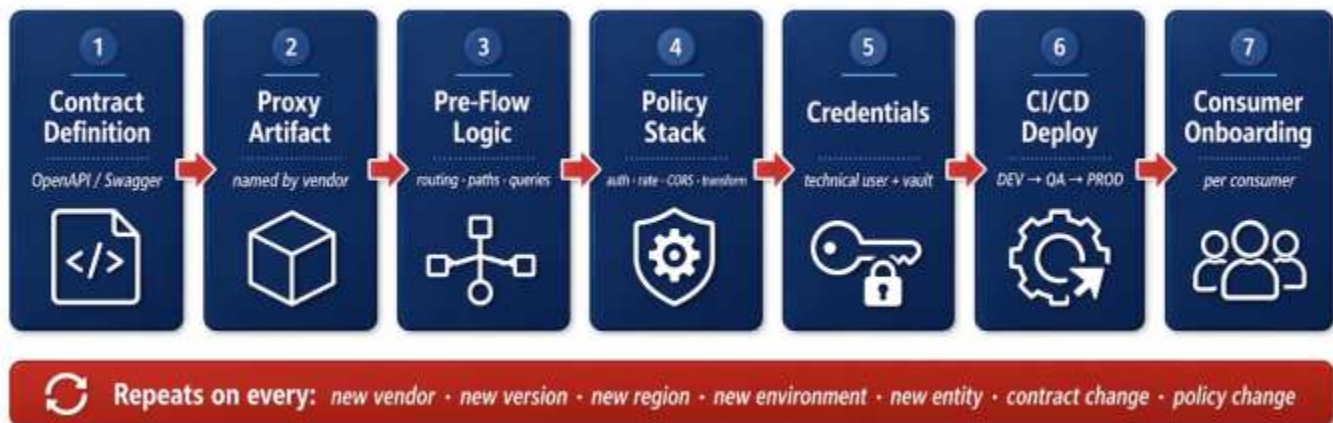


Figure 1 — The As-Is API Delivery Lifecycle

The seven sequential steps — contract, proxy, pre-flow, policies, credentials, CI/CD pipeline, and consumer notification — collapse into one action: append a new target URL to the metadata container attached to the domain proxy. No new artifact is created. No contract is rewritten. No pipeline is invoked. The proxy that was deployed months or years ago remains unchanged; only the metadata it reads at runtime is extended.

**GDCR compresses this lifecycle to a single metadata-only operation.**

## 1.2 Demonstrative Proxy Sprawl — SAP BTP API Management

**Develop**  
Create APIs, Products, Import Policy Templates and view Subscriptions here.

APIs (23)   Products (2)   Subscriptions (2)   Policy Templates (0)








Name	Title
 ProcurementOrderService	ProcurementOrderService
 SAPAribaPO	SAPAribaPO
 Order-Fulfillment-Service	Order-Fulfillment-Service
 AZ-CUST-PROFILE-V2	AZ-CUST-PROFILE-V2
 MS-SalesCustomer	MS-SalesCustomer
 SFDC-SALES-ORDER-CREATE	SFDC-SALES-ORDER-CREATE
 SIEMENS-MANUFACTURING-BOM-SYNCHRONIZATION-V1	SIEMENS-MANUFACTURING-BOM-SYNCHRONIZATI...

Figure 2 — Demonstrative sample of API Proxy Sprawl

Variability stops living in proxies and moves into metadata, while DDCR — implemented as an APIM policy or plugin preserves facade immutability.

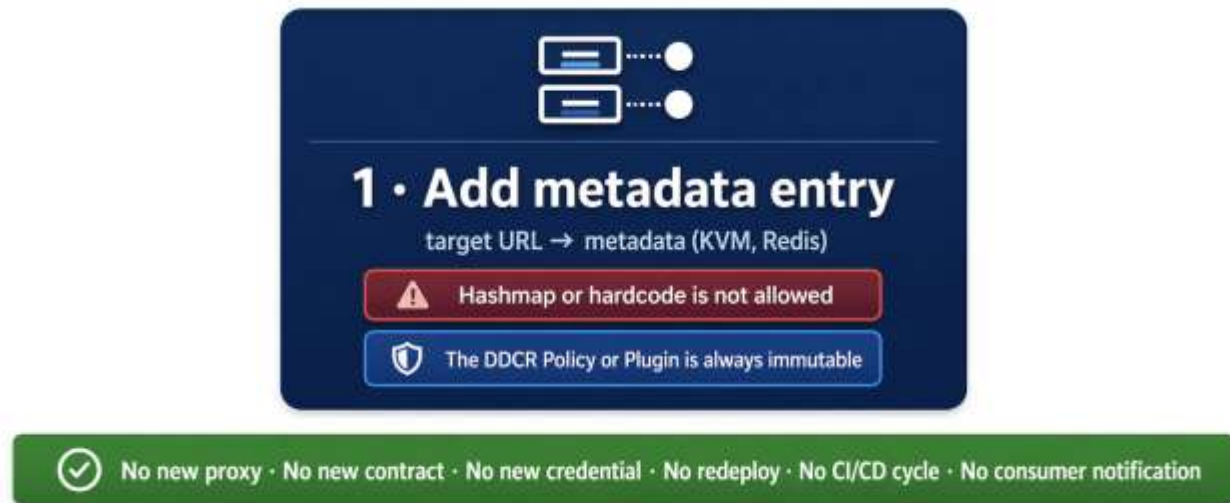


Figure 3 — The To-Be API Delivery Lifecycle

## 1.3 Observed Structural Impact (Representative Landscape)

Metric	Before GDCR	After GDCR	Improvement
API Proxies (4-domain model)	41 (one per backend)	4 (one per domain)	↓ 90%
Vendor / Backend Onboarding Time	Hours to days	< 30 seconds	↓ 99% - zero redeploy

Metric	Before GDCR	After GDCR	Improvement
Proxy Deployment Time (4 domains)	273 minutes	14.5 minutes	↓ 95%
Backend URL Exposure	Full topology visible	Hidden behind semantic abstraction	100% hidden

**Scope note:** figures derive from controlled PoC scenarios within the SAP BTP Trial environment and equivalent sandbox environments on SAP API Management, AWS API Gateway, Azure API Management, Kong Gateway, and Netflix Zuul. They illustrate structural differentials between artifact-centric and metadata-centric governance, not production SLA commitments.

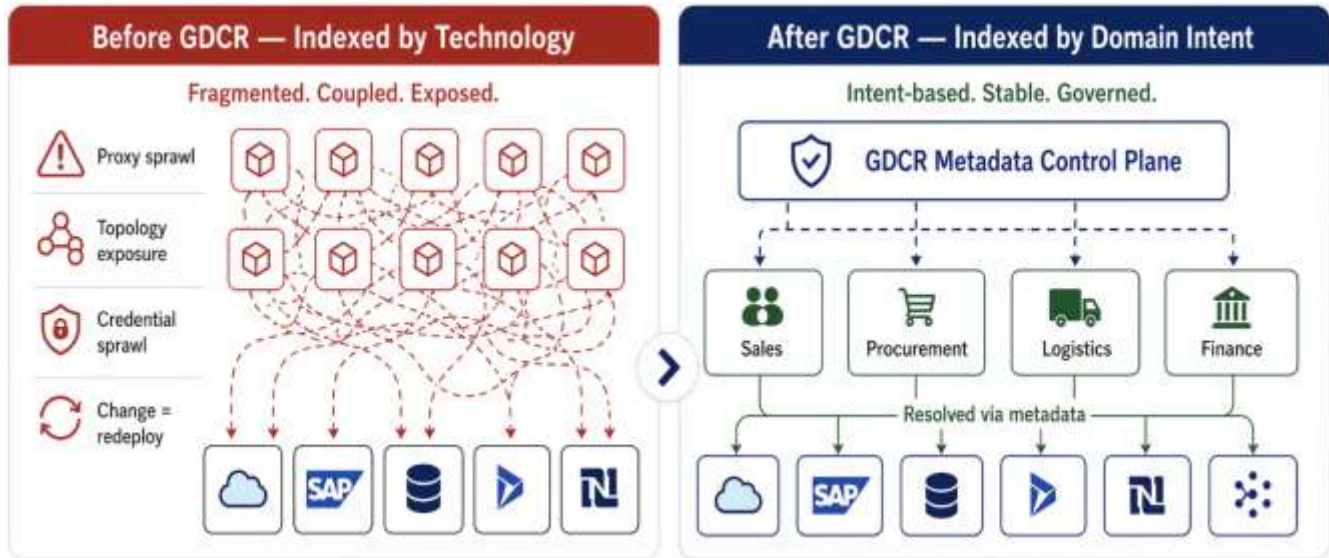


Figure 4 — Proxy Sprawl vs. Domain Consolidation: the GDCR re-indexing of the gateway layer

## 2. The Paradigm Shift

GDCR is not an optimization of the proxy-per-backend model. It is a different organizing principle: from indexing by technology to indexing by business intent.

Before GDCR — Indexed by Technology	After GDCR — Indexed by Domain Intent
41 proxies for 4 domains — one per backend system	4 proxies for 4 domains — one per business domain — immutable
Proxy names encode vendor and version: salesforce-prod-v2	Proxy names encode domain and process: acme.sales.o2c.proxy
URL reveals infrastructure topology	→ URL reveals business intent and a controlled target identifier /sales/orders/create/salesforce
Routing logic embedded in proxy config — redeployment required	Routing externalized to metadata control plane — zero redeployment
Vendor onboarding = new proxy + new credential	Vendor / backend onboarding = 1 metadata entry < 30 seconds
Business process invisible in the architecture	Business process becomes the architecture — /sales/ /finance/ /logistics/
Every version spawns a new proxy artifact	One proxy per domain — backend variability handled by metadata



Before GDCR — Indexed by Technology	After GDCR — Indexed by Domain Intent
OpenAPI contracts drift per vendor, per version	for enterprise-owned semantic facades, one domain-scoped contract may cover multiple backend vendors; this was not empirically validated for direct consumption of vendor-standard ERP APIs.

**Technology changes constantly. The domain does not.** GDCR makes the invariant the architecture, and treats the variant — which vendor, which protocol, which version — as metadata.

### 3. GDCR within the SDIA Ecosystem

GDCR is Layer 1 of the SDIA ecosystem — the semantic gateway, the point of entry for the consumer into the stack. Within the broader ecosystem, DEIP provides the decision model and SDIA provides the umbrella architecture. GDCR receives from DEIP the decision of platform binding (which gateway platform to use) and governs the semantic contract of the gateway.

#### 3.1 The Two-Layer SDIA Stack

Layer	Component	Responsibility
1 — Gateway	GDCR (this document)	Semantic gateway contract · 1 proxy per domain · immutable gateway artifacts · metadata control plane
2 — Resolution	DDCR – APIM Policy or Plugin	Deterministic 7-stage pipeline · 247 action variants → 15 canonical codes · Metadata lookup · fail-fast

#### 3.2 GDCR Scope — What It Does and Forwards

Phase	GDCR Responsibility
Receives	The consumer's semantic HTTP request (e.g., POST /sales/orders/create/salesforce). The consumer sees domain intent and a controlled target identifier, but never infrastructure topology, environment, or backend host details.
Validates	The request against the unified PVRL Level 1 grammar (§5). Invalid addresses are rejected at ingress with a structured semantic error. Valid requests proceed.
Annotates	Observability context: domain, entity, action, target — attached to the request for downstream tracing, metrics, and audit without altering the payload.
Forwards	The validated semantic request to DDCR — the deterministic resolution engine (Layer 2) — which computes the routing key and resolves the backend target.
Returns	The backend response to the consumer with infrastructure identity hidden behind the semantic facade. The consumer interacts with a domain-level contract rather than backend topology or implementation detail.

#### 3.3 The Semantic Chain — End-to-End Traceability

The domain token set at the GDCR layer is the same token carried — unchanged — through every downstream layer. This is the semantic continuity invariant of SDIA. It is later formalized as rule R-04. GDCR's role is to set and preserve the token, not to transform it.

SDIA Layer	Artifact	Example (domain = sales)
1 — GDCR	Semantic URL	/sales/orders/create/salesforce
1 — GDCR	Proxy DNA	acme.sales.o2c.proxy
1 — GDCR	Metadata Container	acme.sales.o2c.routing

SDIA Layer	Artifact	Example (domain = sales)
2 — DDCR	Routing Key (reference)	gdcrcorderscsalesforceid01:http
3 — ODCP	iFlow DNA	id01.o2c.salesforce.order.s4hana.c.in.sync
4 — EDCP	Event Channel	sales.orders.created.v1

The domain token “sales” appears — in identical form — at every layer. GDCR specifies only the Layer 1 artifacts shown in the first three rows; the remaining rows are included solely to illustrate semantic continuity across the SDIA stack.

## 4. Architectural Invariants and Plane Separation

GDCR is governed by four invariants. Every GDCR-compliant implementation must enforce all four.

#	Invariant	Specification
I-1	One Proxy Per Domain	Each business domain maps to exactly one API proxy — not one per vendor, system, or version. Multiple vendors, versions, and protocols within a domain are handled entirely through metadata, never through additional proxy artifacts.
I-2	Immutable Gateway	The proxy artifact is structurally frozen. New vendors, versions, and entities do not trigger proxy modification — they trigger metadata entries. The gateway stops being a deployment surface and becomes a routing contract.
I-3	Semantic URL Contract	The consumer address expresses business intent, not backend topology. /sales/orders/create/salesforce describes what the consumer wants, not the backend location or deployment topology, not where Salesforce lives. The URL is permanent — it survives vendor replacement, backend migration, and version evolution.
I-4	Metadata-Driven Resolution	All routing decisions are resolved at runtime from the metadata control plane. The consumer never specifies a backend URL. The backend URL is constructed by the DDCR runtime policy from metadata, never extracted or inferred from client input. This makes GDCR a whitelist system — only metadata-defined routes execute.

### 4.1 Why the Proxy Is Immutable — Control Plane / Data Plane Separation

**The Data Plane** is the proxy artifact itself — the deployed gateway object, the Semantic URL contract, the PVRL grammar. It carries what is invariant. Deployed once, structurally frozen.

**The Control Plane** is the metadata container attached to the proxy — the KVM, Redis, or equivalent key-value store. It carries what changes: backend URLs, vendor identifiers, regional endpoints, version mappings.

The separation is strict:

- The proxy never contains a backend URL. It contains only the grammar describing which semantic URLs it accepts.
- The metadata never contains a consumer contract. It contains only backend resolution data.
- The metadata evolves independently; the proxy does not.

**This is what makes the proxy immutable.**

Onboarding a new vendor does not modify the proxy — it adds an entry to metadata. Replacing a backend does not modify the proxy — it updates an entry. The proxy artifact has no reason to change unless the business domain itself changes.

Violating the separation breaks the pattern. A proxy that hardcodes a backend URL is no longer a GDCR proxy. A metadata entry that carries consumer contract information is no longer metadata.

**The control plane evolves. The data plane stays.**



Figure 5 — The data plane carries the invariant contract; the control plane carries the variable metadata

## 4.2 Derived Properties

From these four invariants and the plane separation, the following properties follow as consequences, not as separate design goals:

- Zero-redeployment vendor onboarding
- Injection-safe routing (URL not concatenated with client input)
- Fail-fast rejection (unregistered routes rejected before backend contact)
- Backend interchangeability (Salesforce → Dynamics is a metadata change)
- Domain-level observability
- Semantic firewall behavior

## 4.3 Semantic URL Fakery — The Security Contract of the Facade

The Semantic URL is a security boundary. It exposes domain intent and a controlled target identifier, while all execution details are resolved through the metadata control plane.

### Never exposed:

- Infrastructure topology, backend hostnames, environment details, version mappings, orchestration logic, and credentials.

### Enforced structurally:

- Whitelist-only execution — unregistered combinations rejected before backend contact.
- Fail-fast rejection — invalid URLs are rejected with an HTTP error before backend contact. No probing, no fuzzy matching, no fallback.
- No path concatenation — backend URL resolved from metadata, never from client input. Injection and path traversal structurally impossible.

## Comparison — Traditional Exposure vs. GDCR Semantic Abstraction:

Aspect	Traditional Vendor-Named URL	GDCR Semantic URL
Example	/internal-host-prod/salesforce-order-creation-v2-final	/sales/orders/create/salesforce
Backend hostname visible	<b>Yes — internal-cpi-host exposed</b>	<b>No — hidden behind facade</b>
Technology stack visible	<b>Yes — CPI reveals the platform</b>	<b>No — technology is background</b>
Environment visible	<b>Yes — prod in path</b>	<b>No — environment is metadata</b>
Version visible	<b>Yes — v2-final in path</b>	<b>No — version mapping is handled in metadata</b>



Aspect	Traditional Vendor-Named URL	GDCR Semantic URL
Attack surface	<b>Full topology exposed to unauthenticated discovery</b>	<b>Topology hidden; only whitelisted routes discoverable</b>

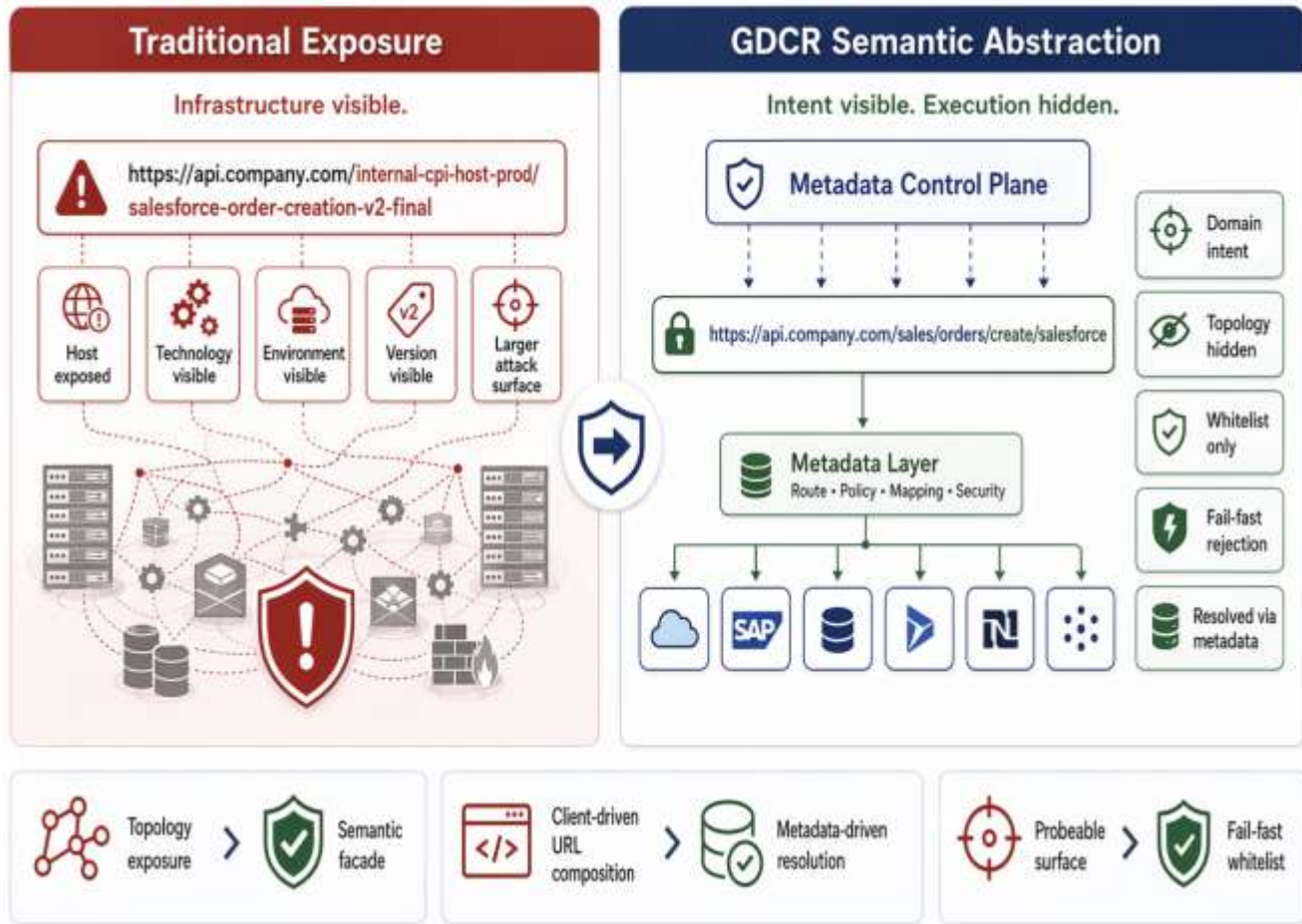


Figure 6 — URL Fakery: Traditional Exposure vs. GDCR Semantic Abstraction

Security positioning. Not security through obscurity. The consumer never needed that infrastructure detail — only business intent. The metadata control plane enforces whitelist semantics: no URL outside the registered set resolves.

**The facade is a semantic firewall. What is not registered cannot execute.**

#### 4.4 Scaling Dimensions — Horizontal and Vertical

Mode	Trigger	Change Surface	Time	Proxy Impact
Horizontal	New vendor / new backend	1 metadata entry	< 30 sec	None — immutable
Vertical	New business process within domain	Additive semantic contract extension + metadata	Minutes	Controlled additive extension

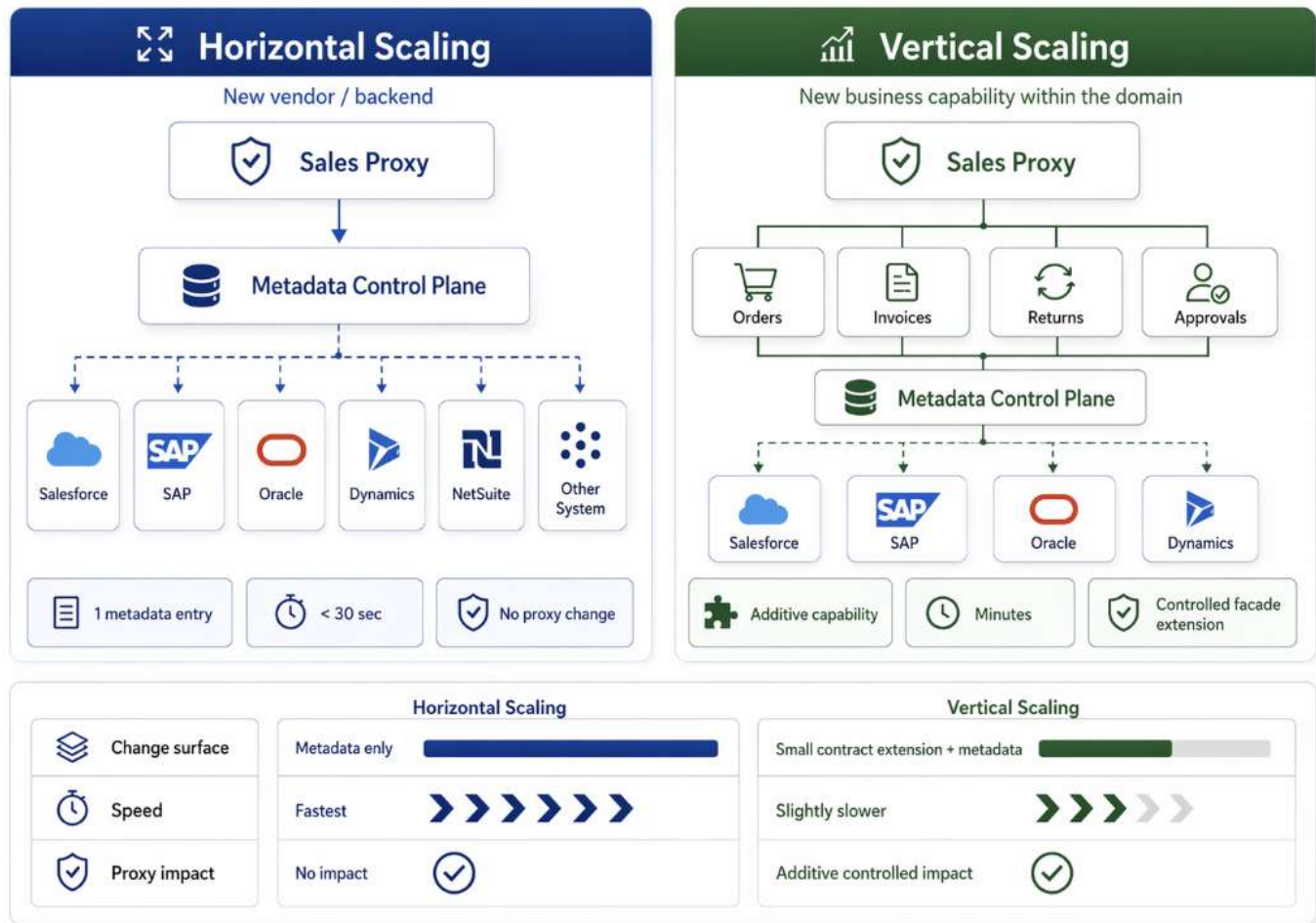


Figure 7 — Two-Level Scaling Strategy: Horizontal Vendor Onboarding and Vertical Process Expansion

**Architectural Implication:** the domain facade remains stable while metadata evolves continuously. Horizontal scaling requires metadata only; vertical scaling extends the semantic contract in a controlled domain-scoped manner. Neither scaling mode reintroduces proxy sprawl, credential sprawl, or URL drift.

## 5. The Three Artifacts and Their Grammar

The GDCR grammar governs three artifacts simultaneously:

- Semantic URL — consumer-facing address
- Proxy DNA — immutable gateway object name
- Metadata Container name — KVM / store identifier attached to proxy

**All three share the same domain invariant and lexical constraints.**



Figure 8 — PVRL Level 1: Three Artifacts, One Domain Invariant

## 5.1 Semantic URL Grammar (ABNF)

The optional granularity segments (sub-domain, sector, division, region) define an extensibility surface, not a mandatory classification model. Enterprises may adopt only the segments that are useful for their governance needs. Their semantic interpretation is organization-defined and must be governed consistently once selected; GDCR specifies the syntactic order, not a universal taxonomy.

; GDCR Semantic URL Grammar — PVRL Level 1

; RFC 5234 ABNF notation

; Baseline

; semantic-url = "/" domain [ "/" sub-domain ] [ "/" sector ] [ "/" division ] [ "/" region ] "/" entity "/" action "/" target

; domain = 1\*ALPHA-LOWER

; business domain — never abbreviated

; sub-domain = 1\*(ALPHA-LOWER / DIGIT)

; optional · e.g, o2c, r2r, s2p

; sector = 1\*ALPHA-LOWER

; optional · industry sector

; division = 1\*ALPHA-LOWER

; optional · business division

; region = 1\*ALPHA-LOWER

; optional · geography

; entity = 1\*ALPHA-LOWER

; entity = 1\*ALPHA-LOWER ; lexical form only; plural-noun enforcement is defined by validation rules

; action = action-token ; present-tense canonical verb

; target = 1\*(ALPHA-LOWER / DIGIT) ; vendor / system identifier / receiver identifier

; action = action-token ; canonical action token = "create" / "read" / "update" / "delete" / "process" / "validate" / "submit" / "approve" / "reject" / "cancel" / "post" / "query" / "sync" / "notify" / "transfer"

; illustrative reference set of canonical action tokens

; in the validated DDCR implementation, 247 raw action variants were normalized to this reference set

; enterprises may adapt the canonical vocabulary according to their own semantic model and governance requirements

ALPHA-LOWER = %x61-7A

; a-z only — no uppercase, no underscores, lowercase letters

; Examples — baseline form (validated end-to-end)

; /sales/orders/create/salesforce

; /finance/invoices/post/s4hana

; Examples — extended form (optional segments, not exercised in validation)

; /sales/o2c/orders/create/salesforce

; /sales/o2c/emea/orders/create/salesforce

## 5.2 Proxy DNA Grammar (ABNF)

; GDCR Proxy DNA Grammar — PVRL Level 1

; RFC 5234 ABNF notation

; Baseline

; proxy-dna = company [ "." qualifier ] "." domain "." sub-domain ".proxy"

; company = 1\*ALPHA-LOWER

; organizational tenant prefix

; qualifier = 1\*ALPHA-LOWER

; optional · division or sector

; domain = 1\*ALPHA-LOWER

; business domain — never abbreviated

; sub-domain = process-token

; canonical business process

; process-token = "o2c" / "r2r" / "s2p" / "p2p" / "h2r" / "i2p" / "b2b" / "le" / "scm" / "custom"

; illustrative examples of canonical business process codes

; enterprise-specific process tokens may be adapted according to organizational needs

; ALPHA-LOWER = %x61-7A

; lowercase letters a-z only

**; Examples — baseline form**

```
; acme.sales.o2c.proxy
; acme.finance.r2r.proxy
```

**; Examples — extended form**

```
; acme.retail.sales.o2c.proxy
; acme.consumer.sales.o2c.proxy
```

**5.3 Metadata Container DNA Grammar (ABNF)**

The Metadata Container is the KVM, Redis, or equivalent key-value store/ configuration namespace attached to the proxy. Its name is structurally aligned with the Proxy DNA, with .routing replacing .proxy as the terminal segment.

**; GDCR Metadata Container DNA Grammar — PVRL Level 1****; RFC 5234 ABNF notation****; Baseline**

```
; metadata-container-dna = company [ "." qualifier ] "." domain "." sub-domain ".routing"

; company = 1*ALPHA-LOWER      ; organizational tenant prefix
; qualifier = 1*ALPHA-LOWER    ; optional · division or sector
; domain = 1*ALPHA-LOWER      ; business domain — never abbreviated
; sub-domain = process-token   ; canonical business process

; process-token = "o2c" / "r2r" / "s2p" / "p2p" / "h2r" / "i2p" / "b2b" / "le" / "scm" / "custom"

; illustrative examples of canonical business process codes
; enterprise-specific process tokens may be adapted according to organizational needs

; ALPHA-LOWER = %x61-7A ; lowercase letters a-z only
```

**; Examples — baseline form (validated end-to-end)**

```
; acme.sales.o2c.routing
; acme.finance.r2r.routing
```

**; Examples — extended form (optional qualifier, not exercised in validation)**

```
; acme.retail.sales.o2c.routing
; acme.consumer.sales.o2c.routing
```

**; Alignment invariant (R-08):**

**; Metadata Container DNA and Proxy DNA must share all segments except the terminal suffix.**

```
; acme.sales.o2c.proxy ↔ acme.sales.o2c.routing
; acme.retail.sales.o2c.proxy ↔ acme.retail.sales.o2c.routing
```

The Metadata Key format (the key stored inside the container) is owned by DDCR (Layer 2), not by GDCR. Its normative specification lives in DDCR v3.0.

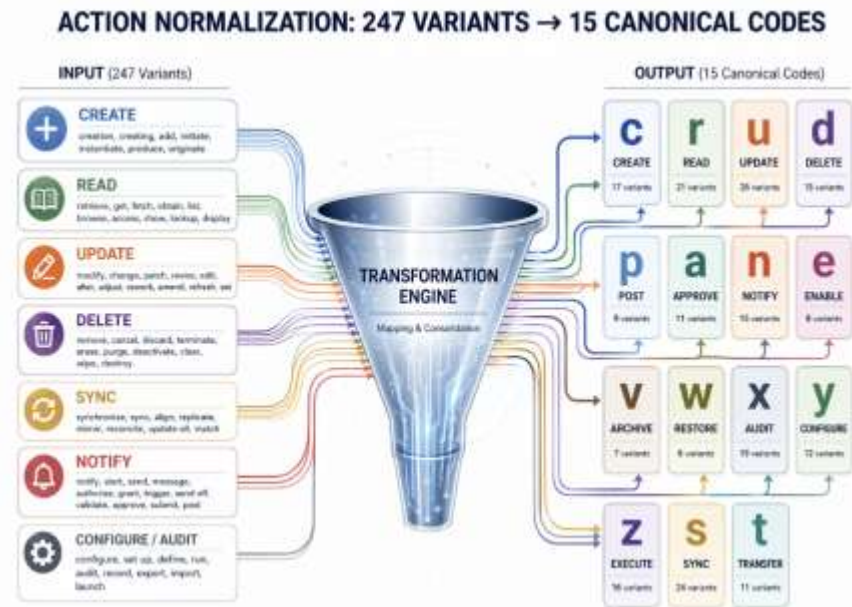


Figure 8 - Total: 247 action variants → 15 canonical codes. The gateway resolves diversity; the domain remains canonical.

## 5.4 Valid and Invalid Examples

Semantic URL	Verdict	Rationale
/sales/orders/create/salesforce	VALID	All segments lowercase; entity plural; action canonical; target alphanumeric.
/finance/invoices/post/s4hana	VALID	Canonical post action; s4hana is alphanumeric.
/sales/o2c/emea/orders/create/salesforce	VALID	Valid extended example illustrating optional granularity under enterprise-defined semantics; domain continuity is preserved.
/Sales/Orders/Create/Salesforce	INVALID	Uppercase rejected (R-01).
/sales/orders/create/salesforce-prod-v2-eu-west	INVALID	Hyphens in target rejected (R-01). Environment, region, and version qualifiers belong in metadata — not in the URL.
/sales/orders/buy/salesforce	INVALID	Action not in canonical vocabulary (R-03).
//orders/create/salesforce	INVALID	Empty domain segment rejected (R-06).

## 6. PVRL Validation Rules and Enforcement Points

Eight normative validation rules define PVRL Level 1. A grammar violation is a hard rejection.

### 6.1 Validation Rules R-01 through R-08

Rule	Name	Specification	Scope
R-01	Lowercase Invariant	All labels must be lowercase. Digits are permitted only where explicitly allowed by the grammar (for example, target identifiers and process tokens such as o2c, r2r, and s2p). Uppercase, underscores, and spaces are forbidden. Hyphens are not permitted in GDCR Level 1 segments.	All artifacts
R-02	Entity Plural Form	The entity token in the Semantic URL must be a plural noun. Singular forms are rejected at ingress.	Semantic URL



Rule	Name	Specification	Scope
R-03	Action Canonical Vocabulary	The action token must resolve to one of the 15 canonical action codes. Raw verbs outside this vocabulary are rejected by DDCR Stage 3 normalization (247 → 15).	Semantic URL
R-04	Domain Continuity	The domain token from the Semantic URL must be present — unchanged — in the Proxy DNA, Metadata Container name, DDCR routing key, ODCP iFlow DNA, and EDCP event channel. Token drift across layers is a semantic violation.	All artifacts · all layers
R-05	Optional Segment Order	When optional granularity segments are used, they must appear in canonical order after the domain or sub-domain anchor, as defined by the artifact grammar. Any other order is a grammar violation.	All artifacts
R-06	No Empty Segments	No segment in any artifact may be empty. /sales//create/salesforce is rejected at ingress. The parser must reject consecutive separators.	All artifacts
R-07	Explicit Target Declaration	The target segment in the Semantic URL must be explicitly declared. Omission is a grammar violation. Default-target resolution is a DDCR policy concern, not a GDCR convenience.	Semantic URL
R-08	Artifact Name Alignment	The Proxy DNA and the Metadata Container name must share all segments except the terminal suffix (.proxy vs .routing). Divergence is rejected at deployment time.	Proxy DNA ↔ Container

## 6.2 Enforcement Points

#	Point	Phase	Rules	Action on Violation
1	Ingress Parser	Runtime — request arrival	R-01, R-02, R-03, R-05, R-06, R-07	HTTP 400 with structured semantic-error descriptor
2	Proxy DNA Validator	Deployment — CI/CD pipeline	R-01, R-05, R-08	Deployment rejected; artifact refused
3	Metadata Container Validator	Deployment — KVM provisioning	R-01, R-05, R-08	Provisioning rejected; container not created
4	DDCR Handoff Validator	Runtime — forward to Layer 2	R-04 (domain continuity)	HTTP 500 with cross-layer token-drift error
5	Observability Tagger	Runtime — metrics and audit	R-04 (tag consistency)	Alert on divergence; metrics labelled with violation

The ingress parser is the most important enforcement point — it runs on every request, before any backend is contacted. A malformed URL is not a candidate for correction by the gateway. It is a contract violation by the consumer. Silent normalization would defeat the purpose of the grammar.

## 7. Platform Implementations and Empirical Validation

GDCR is a governance model, not a gateway product. The same invariants — one proxy per domain, immutable gateway artifacts, semantic URL, metadata-driven resolution, PVRL Level 1 grammar — apply unchanged across every enterprise API gateway platform. This section documents the validated mappings and the empirical results.

### 7.1 Platform Equivalence Matrix

Platform	Policy Language	Metadata Store	Status
SAP API Management	JavaScript (Nashorn / V8)	KVM (KeyValueMap)	<b>PROVEN — Phantom v12 E2E CPI</b>
Kong Gateway (Docker)	Lua (custom plugin)	Redis / in-memory	<b>PROVEN — mock</b>
Kong Gateway (Kubernetes)	Lua (custom plugin)	Redis	<b>PROVEN — E2E CPI</b>



Platform	Policy Language	Metadata Store	Status
AWS API Gateway	Python Lambda + DynamoDB	DynamoDB	<b>PROVEN — E2E CPI</b>
Azure API Management	C# Policy Expressions	Redis / Named Values	<b>PROVEN — E2E CPI</b>
Netflix Zuul	Java (Filter)	HashMap or external store	<b>PROVEN — Phantom v12</b>
Google Apigee	JavaScript / Node target	KVM (built-in)	<b>PORTABLE</b>
MuleSoft Anypoint	DataWeave / Java	Object Store v2	<b>PORTABLE</b>

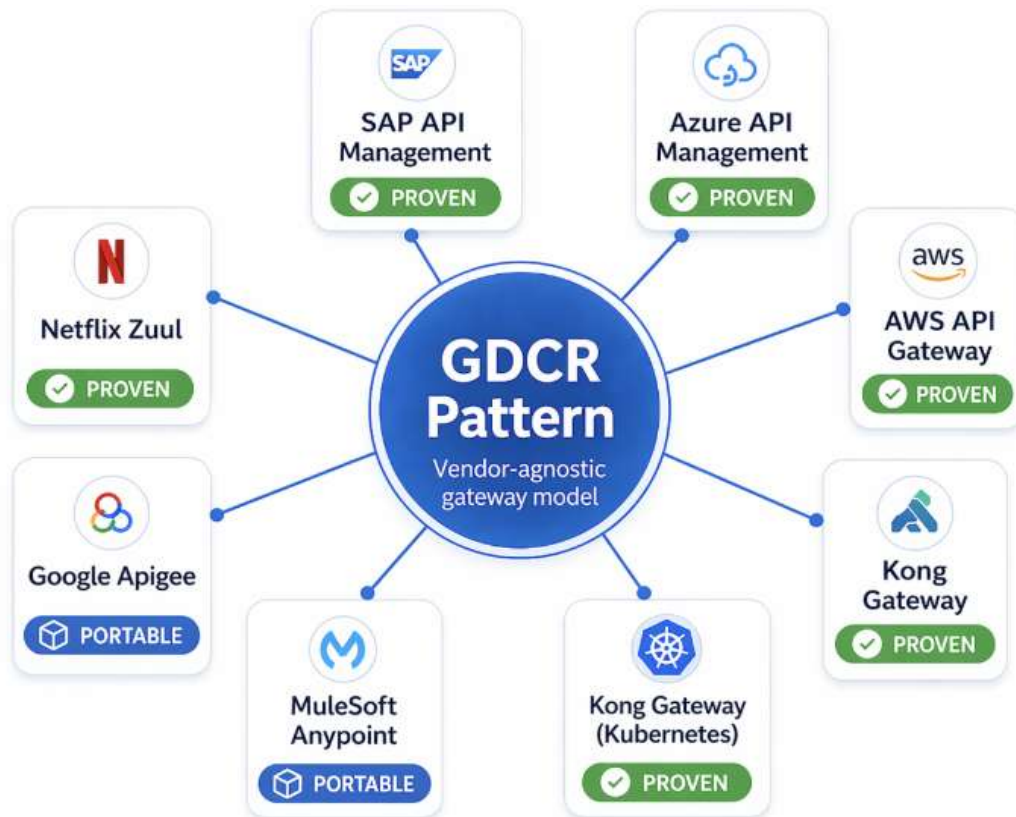


Figure 9 — GDCR implemented across 8 gateway platforms. Same pattern, different technologies

## 7.2 SDIA Ecosystem Scope — 9 validated configurations (5 gateway platform validations + 4 additional ecosystem scenarios)

Scope	Configurations	Requests	Success Rate	Routing Failures
APIM Reference Scope	5	~1,599,871	99.99%	0
SDIA Ecosystem	9 (5 + 4)	~2,067,904	99.99%	0

Honesty note: Kong Docker runs used mock backends local to the container and therefore measure the routing plugin's throughput under ideal conditions — not end-to-end latency against enterprise backend systems. The SAP APIM, AWS, Azure, and Kong K8s runs targeted SAP Cloud Integration in Europe from a Newman client in Warsaw, Poland — end-to-end transatlantic or intra-EU paths with real backend processing.

158 failures total across all configurations were network-layer events only (ECONNRESET / ETIMEDOUT) over sustained transatlantic runs from Warsaw, Poland to US/EU cloud infrastructure. Zero routing-logic failures across all platforms and all configurations.

Validation boundary. The empirical runs reported in **§7.2** cover semantic facades resolved by DDCR and executed through the validated backend orchestration paths documented in this work, primarily SAP Cloud Integration. They do not cover direct gateway-to-ERP consumption of vendor-standard APIs such as SAP ERP/S/4HANA standard OData/SOAP services or Workday-delivered APIs.

### 7.3 Scope Boundary — Standard Enterprise APIs

GDCR was validated against custom APIs where the facade must remain immutable across backend changes. This is where GDCR's value is proven: one domain-scoped proxy, metadata-driven resolution, zero-redeployment vendor onboarding.

What was NOT tested: Standard packaged enterprise APIs — SAP S/4HANA OData/SOAP, SuccessFactors, Oracle EBS, Dynamics 365, and similar vendor-delivered surfaces.

**Why not tested:** the routing invariant is expected to hold (domain invariant, vendor variant), but practical application is not necessarily justified. Standard enterprise APIs are typically consumed through platform-native proxy patterns where the Pre-Flow already encapsulates substantial routing logic (path construction, query assembly, OData filters, conditional transformations). Adding a GDCR facade above would be architecturally valid but operationally redundant — the vendor dictates the backend contract, and vendor replacement at that level is rare.

Where GDCR's value is consolidated and testable:

- Custom APIs built for internal consumers
- Multi-vendor landscapes requiring backend substitutability (e.g., Salesforce ↔ Dynamics ↔ S/4HANA for same Sales domain)
- Composite APIs aggregating multiple backends behind a single domain surface
- Any facade that must remain stable while backends evolve

In these cases, GDCR's invariants render the facade structurally immutable. Against standard packaged APIs, the invariants still hold, but the architectural ROI is lower.

#### 7.3.1 Direct Standard API Consumption (SAP ERP / Workday)

This work did not empirically validate direct gateway-to-ERP routing over vendor-standard APIs such as SAP ERP/S/4HANA OData/SOAP services or Workday-delivered APIs. In such scenarios, the GDCR facade remains architecturally possible, but the economics of abstraction change: the backend contract is already standardized by the vendor, and the gateway may become a semantic wrapper over a contract whose structure, query semantics, and lifecycle are externally dictated.

#### 7.3.2 Architectural Trade-Offs

In direct standard-API scenarios, the main question is not whether DDCR can resolve targets, but whether introducing a semantic facade above an already standardized vendor contract produces sufficient governance value to justify the additional abstraction layer. The trade-off is explicit: pass-through preserves the vendor contract and minimizes mediation effort, while canonical domain-level abstraction improves facade stability but typically reintroduces transformation, mapping, and orchestration concerns beyond the strict gateway scope of GDCR.

## 8. Governance Requirements and When Not to Use GDCR

### 8.1 Governance Requirements

GDCR is not a product that ships with governance included. It is a governance model that requires specific organizational commitments to function. The following requirements are normative for a GDCR-compliant implementation.

Requirement	Specification
Domain Ownership	Each domain must have a single, named owner with authority over the domain proxy, the domain metadata container, and the domain entity whitelist. Ownership is domain-level, not vendor-level.
Metadata Change Control	Metadata container entries are production-impacting configuration. Changes require versioned, peer-reviewed, and auditable change control. GDCR's zero-redeployment property is a technical capability, not a license to bypass review.
Grammar Enforcement at Build Time	PVRL Level 1 grammar (§5) and rules R-01 through R-08 (§6) must be enforced by the CI/CD pipeline, not only by runtime parsers. A proxy or container that violates the grammar must not reach production.
Observability Contract	Every request traversing GDCR must be emitted with domain, entity, action, and target as first-class tags. Metrics bound only to proxy identifiers are insufficient — they reproduce the topology-coupling GDCR aims to remove.
Failure Mode Resolution	Documented runbooks for: missing metadata key (DDCR fail-fast), grammar violation at ingress (HTTP 400), proxy/container name drift (R-08, deployment rejection), cross-layer token drift (R-04), metadata store outage (HA + DR), namespace collision (domain-boundary review).

## 8.2 When NOT to Use GDCR

Condition	Reason
<b>Low governance maturity</b>	Environments lacking defined domain boundaries, naming standards, or change-control will not benefit from GDCR and may experience higher operational overhead than with a conventional proxy-per-service approach.
<b>Rapidly changing domain models</b>	If the business domain model itself is unstable, the Semantic URL contract will require corresponding updates at multiple layers. A simpler proxy-per-service model is more appropriate until the domain model stabilizes.
<b>Single-vendor / single-backend landscapes</b>	The primary value of GDCR — vendor-agnostic routing, zero-redeployment vendor onboarding, domain-level governance — is most pronounced in multi-vendor landscapes. In single-backend landscapes, governance overhead may exceed operational benefit.
<b>Experimental / short-lived APIs</b>	GDCR's overhead — domain definition, metadata registration, PVRL enforcement — is justified for multi-year integration lifecycles. For throwaway prototypes, a direct proxy is simpler and lower cost.
<b>Direct consumption of vendor-standard ERP APIs with low substitution probability</b>	When the backend contract is already standardized and operationally accepted as the external contract, a GDCR facade may add governance consistency but limited practical ROI relative to its abstraction cost.

## 8.3 Authentication Patterns Compatible with the GDCR Facade

GDCR does not specify an authentication mechanism. It specifies a facade that is compatible with standard enterprise authentication patterns and, for high-throughput machine-to-machine (M2M) scenarios, compatible with metadata-based authorization performed directly at the facade layer.

The table below documents the four authentication patterns exercised in the validated reference implementations. GDCR imposes no constraints on which pattern is used; the choice depends on the consumer profile, the trust zone, and the latency envelope required by the integration scenario.

Pattern	Use Case	Implementation	Latency Profile
OAuth 2.0 / OIDC	User-facing flows, consent-required scenarios, audit-heavy integrations	Remote token introspection against the identity provider	High — external dependency per call
Basic Auth	Legacy system integration, internal trust zones, back-office M2M	KVM-backed credential validation at the facade	Sub-5 ms
Header Key	High-throughput M2M, API key validation	KVM key lookup at the facade	Sub-3 ms
Semantic Fail-Fast	GDCR-native M2M routing authorization — the registered route is itself the authorization surface	In-memory metadata validation at the facade	Sub-1 ms

**Security Layering.** OAuth 2.0/OIDC governs who the caller is. The metadata control plane governs which route the caller may resolve. Orthogonal and complementary.

**Metadata as Authorization Surface.** Under Semantic Fail-Fast, registered metadata entries form an implicit whitelist. A caller authenticated at identity layer still cannot reach a backend not registered in metadata — rejected at resolution, before backend contact. Structural consequence of Invariant I-4.

**Scope Boundary.** Token validation, credential rotation, key lifecycle, IdP integration are operational concerns of enterprise security, not of GDCR. GDCR specifies where authentication is checked (facade, before resolution) and what guarantees it provides (whitelist-only, fail-fast). The choice of mechanism is governed by the organization's security architecture.

#### 8.4 Semantic Context Propagation — Facade Headers

The GDCR facade SHOULD inject semantic headers into the forwarded request so that downstream layers (resolution, orchestration, events) and backend systems receive the resolved domain intent without re-parsing the URL. This eliminates redundant parsing, enables consistent cross-layer tracing, and preserves semantic context throughout the request lifecycle.

The following headers are exercised in the validated reference implementation (DDCR Phantom v12 on SAP APIM, JavaScript policy). They represent the proof-of-concept set; additional headers may be added or removed at implementation time based on organizational observability, tracing, and audit requirements.

Header	Source	Purpose
dcrp.routing.success	Policy variable	Boolean resolution outcome — true on success, false on any fail-fast rejection
target.url	Resolved from KVM value	Backend target URL — constructed from metadata, never from client input
x-dcrp-process	Dynamically extracted from KVM key	Business process identifier (o2c, r2r, s2p, p2p, etc.)
x-dcrp-adapter	KVM value (after colon)	Protocol adapter selected for the backend (http, cxf, etc.)
x-dcrp-version	Policy constant	Engine / policy version identifier for operational traceability

Header	Source	Purpose
x-dcrp-key	KVM key (original)	Full lookup key used during resolution — preserved for audit
x-packagename	KVM secondary variable	ODCP package identifier — cross-reference to the orchestration layer
x-sapprocess	KVM secondary variable	Platform-specific process tag — reference implementation (SAP CPI)
x-senderid	Parsed via runtime custom headers	Vendor / backend identifier as declared in the semantic URL target segment
x-correlationid	messageid (platform-generated)	Cross-layer trace identifier for end-to-end observability
x-idinterface	KVM value (iflowid segment)	iFlow identifier — cross-reference to the ODCP orchestration artifact

**Implementation note:** Headers are proof-of-concept. Each enterprise adapts to its own observability and audit needs. Headers **MUST** come from resolved metadata — never from client input. Custom headers **SHOULD** follow a consistent prefix (e.g., x-dcrp-\*).

**Companion reference implementation:** DDCR Phantom v12 (CC BY 4.0) — DOI 10.5281/zenodo.18619641 · [github.com/rhviana/deip](https://github.com/rhviana/deip). JavaScript and Java implementations are the most mature in the current reference set; Lua, Python, and C# are maintained as functional baseline implementations. GDCR specifies the facade contract; runtime belongs to DDCR v3.0.

## 9. API Contracts, Versioning, and OpenAPI Governance

GDCR changes API governance structurally. The proxy is immutable. The domain surface is stable. Contracts are organized by domain, not vendor. Backend evolution does not touch the facade.

### 9.1 Versioning Without Proxy Multiplication

In the traditional model, each version typically results in a new proxy artifact (v1, v2, v3). Duplicated policies, monitoring, lifecycle.

GDCR: proxy is a stateless semantic router, not a version container. Versioning is governed through metadata and contract evolution, not through proxy multiplication.

Dimension	Traditional Versioning	GDCR Versioning
Proxy per version	<b>New proxy created per version</b>	<b>Single proxy — immutable across versions</b>
Deployment event	<b>Redeployment required per version</b>	<b>Metadata update only — zero redeployment</b>
Infrastructure surface	<b>Multiplies linearly with versions</b>	<b>Constant</b>
Policy duplication	<b>Policies re-authored per version</b>	<b>Policies inherited — no duplication</b>
Monitoring	<b>Fragmented across version proxies</b>	<b>Unified at the domain level</b>

### 9.2 OpenAPI: Contract, Not Structure

**GDCR uses OpenAPI (OAS 3.x) only as a contract artifact, decoupled from runtime routing.** The specification describes the Semantic URL facade, not backend resolution, vendor targets, or metadata. **Runtime resolution belongs exclusively to the metadata control plane.**

Dimension	Traditional OpenAPI	GDCR OpenAPI
Specification scope	One spec per vendor or system	For enterprise-owned semantic facades, one domain-scoped spec may cover multiple backend vendors. (e.g., sales-domain-v2.yaml)
API portal	Fragmented — one catalog entry per vendor	Unified — one catalog entry per domain
Spec redeployment	Required on vendor change	Spec independent of backend — no redeployment
Contract coupling	Contract coupled to backend infrastructure	Contract decoupled — describes the semantic facade only
Version evolution	New proxy + new spec per version	New metadata entries + spec revision; proxy unchanged

## OpenAPI Governance + DDCR Runtime Hybrid

OpenAPI governs the contract. DDCR governs the runtime resolution.

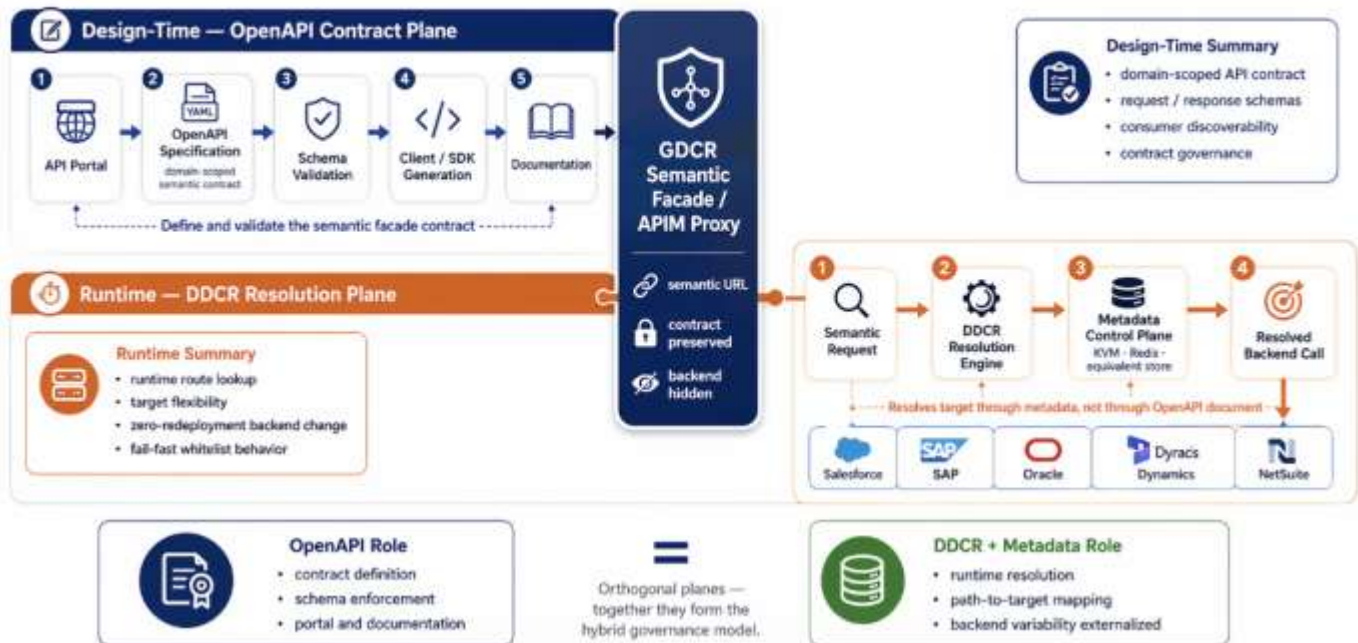


Figure 10 — OpenAPI Contract Plane vs. DDCR Runtime Resolution Plane

This domain-scoped OpenAPI claim applies most directly to custom semantic facades owned by the enterprise. Where the backend surface is a vendor-standard API, the enterprise must choose between exposing the vendor contract through the facade with minimal mediation, or defining a canonical domain contract above it. The former reduces abstraction value; the latter increases mediation responsibility.

### 9.3 Metadata and OpenAPI — Orthogonal Planes



OpenAPI = design-time contract (schemas, docs). Metadata = runtime routing (backend endpoints). Together: contract discoverable, routing evolvable — no spec change when backend changes.

## 9.4 SLA and Domain-Scoped Governance

SLAs bind to business capability, not vendors or versions. Rate limits at domain level (e.g., /sales/orders). Version upgrades require no gateway reconfiguration. Vendor replacement = consumer sees no change.

# 10. Positioning, Conclusion, and References

## 10.1 Positioning vs. Related Work

GDCR does not invent new components. It composes existing principles — bounded contexts, metadata-driven routing, control/data plane separation, formal grammar — into a coherent governance model that applies specifically to the API gateway layer.

Prior Work	What It Provides	GDCR's Additional Contribution
Domain-Driven Design (Evans, 2003)	Bounded contexts and domain semantics for application modeling	Extends the bounded context principle into the API gateway layer — domain boundaries structurally enforced at the gateway artifact
Enterprise Integration Patterns (Hohpe & Woolf, 2003)	Message routing, transformation, and endpoint patterns inside the integration platform	Stabilizes the external semantic facade above EIP — gateway surface immutable regardless of internal EIP evolution
REST (Fielding, 2000)	Uniform interface constraint and stateless communication	Enforces the uniform interface at the governance layer — Semantic URL as the permanent, grammar-validated contract
Netflix Zuul (2013)	Dynamic metadata-driven routing for microservices at scale	Applies equivalent principles to enterprise integration with heterogeneous vendor backends and cross-platform portability
SDN Control / Data Plane	Network architecture with control plane (metadata) distinct from data plane (forwarding)	Applies the control/data plane separation to enterprise API governance — metadata evolves independently of proxy artifacts
ABNF / RFC 5234	Formal grammar notation	PVRL Level 1 uses ABNF to make semantic identifiers machine-verifiable, not merely conventional

## 10.2 Conclusion

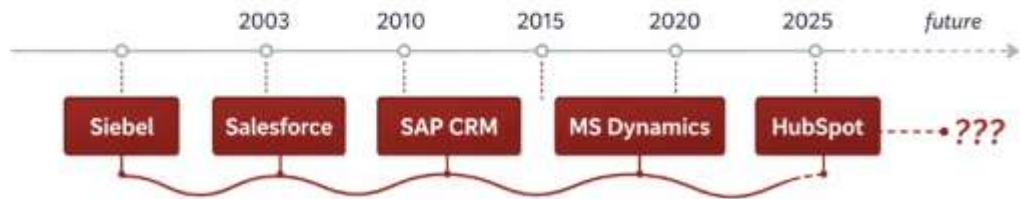
GDCR v8.0 makes the domain the primary key of the API gateway: one proxy per domain, immutable facade, semantic URLs, metadata-driven resolution, and PVRL grammar enabling strict rejection of semantic violations. Validated across five gateway platforms, with nine validated configurations in the broader SDIA ecosystem and zero routing-logic failures over ~2M requests.

# The Domain Never Lies

*technology changes constantly · the business domain does not*

## VOLATILE

*Vendors change ·  
contracts churn ·  
platforms migrate*



## THE GATEWAY LINE

**GDCR — Layer 1 of SDIA — re-indexes the gateway by domain, not by vendor**

## STABLE



*Domain stays the same · contract preserved · governance measurable.*

Figure 11 — The Domain Never Lies: vendor volatility above the gateway, domain stability below

## 10.3 References

- [1] R. L. H. Viana, "Gateway Domain-Centric Routing (GDCR) — v8.0," Zenodo, April 2026. DOI: 10.5281/zenodo.18582492
- [2] R. L. H. Viana, "Domain Driven-Centric Router (DDCR) — v3.0," Zenodo, 2026. DOI: 10.5281/zenodo.18864832
- [3] R. L. H. Viana, "Orchestration Domain-Centric Pattern (ODCP) — v2.0," Zenodo, 2026. DOI: 10.5281/zenodo.18876593
- [4] R. L. H. Viana, "Event Domain-Centric Pattern (EDCP) — v1.0," Zenodo, 2026. DOI: 10.5281/zenodo.19068766
- [5] R. L. H. Viana, "Semantic Domain Integration Architecture (SDIA) — v2.0," Zenodo, March 2026. DOI: 10.5281/zenodo.18877635
- [6] R. L. H. Viana, "Domain Enterprise Integration Pattern (DEIP) — v2.0," Zenodo, 2026. DOI: 10.5281/zenodo.19004802
- [7] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003.
- [8] G. Hohpe and B. Woolf, Enterprise Integration Patterns. Addison-Wesley, 2003.
- [9] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, UC Irvine, 2000.
- [10] O. Zimmermann et al., Patterns for API Design. Addison-Wesley, 2022.
- [11] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," RFC 5234, IETF, January 2008.
- [12] Netflix Technology Blog, "Announcing Zuul: Edge Service in the Cloud," 2013.
- [13] R. L. H. Viana, SAP Press e-Bite — Enterprise Messaging. SAP Press, 2021.

## Appendix A — Historical Context: From DCRP and PDCP to GDCR

Patterns originated Feb 2026 on SAP SCN (two blog posts → GDCR + ODCP). Archived Feb 6, 2026 on Internet Archive — prior art under 35 U.S.C. § 102.

### A.1 DCRP — Domain-Centric Routing Pattern (February 2026)

The first blog (DCRP) proposed the core thesis: one API proxy per domain, metadata-driven routing externalized from the proxy, domain as primary key of the gateway facade. Scoped to SAP APIM + CPI.

DCRP → GDCR when generalized across AWS, Azure, Kong, Netflix Zuul — proving invariants hold regardless of vendor. The structural core survives unchanged in GDCR v8.0: domain-as-primary-key, metadata-driven routing, immutable gateway artifact.

**Archived (Internet Archive — Whitepaper Machine, 6 Feb 2026):**

<https://web.archive.org/web/20260206121858/https://community.sap.com/t5/technology-blog-posts-by-members/sap-btp-apim-domain-centric-routing-pattern-dcrp-governing-apis-via-cpi/ba-p/14312788>

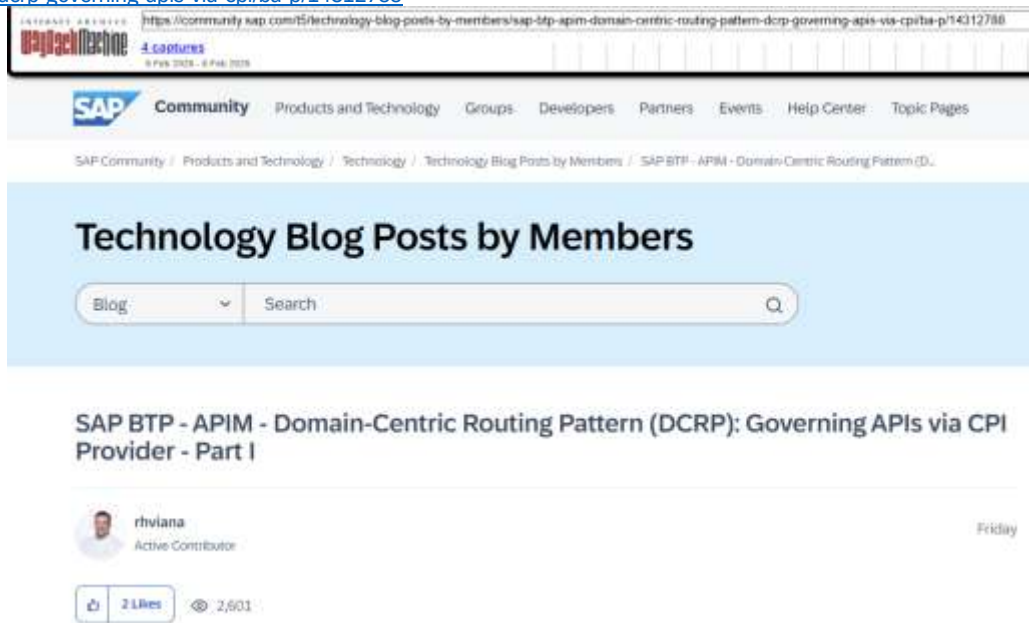


Figure A.1 — DCRP original publication, SAP Community, February 2026

### A.2 PDCP — Package Domain-Centric Pattern (February 2026)

The second blog (PDCP) addressed package sprawl: iFlows proliferating per vendor, version, environment. Proposed one package per domain process with metadata-governed sub-flows. PDCP was later renamed ODCP — Layer 3 of SDIA. The iFlow DNA naming convention became the normative grammar of ODCP v2.0.

**Archived (Internet Archive — Whitepaper Machine, 6 Feb 2026):**

<https://web.archive.org/web/20260206123644/https://community.sap.com/t5/technology-blog-posts-by-members/sap-btp-cpi-package-domain-centric-pattern-pdcp-solving-package-sprawl-at/ba-p/14318864>

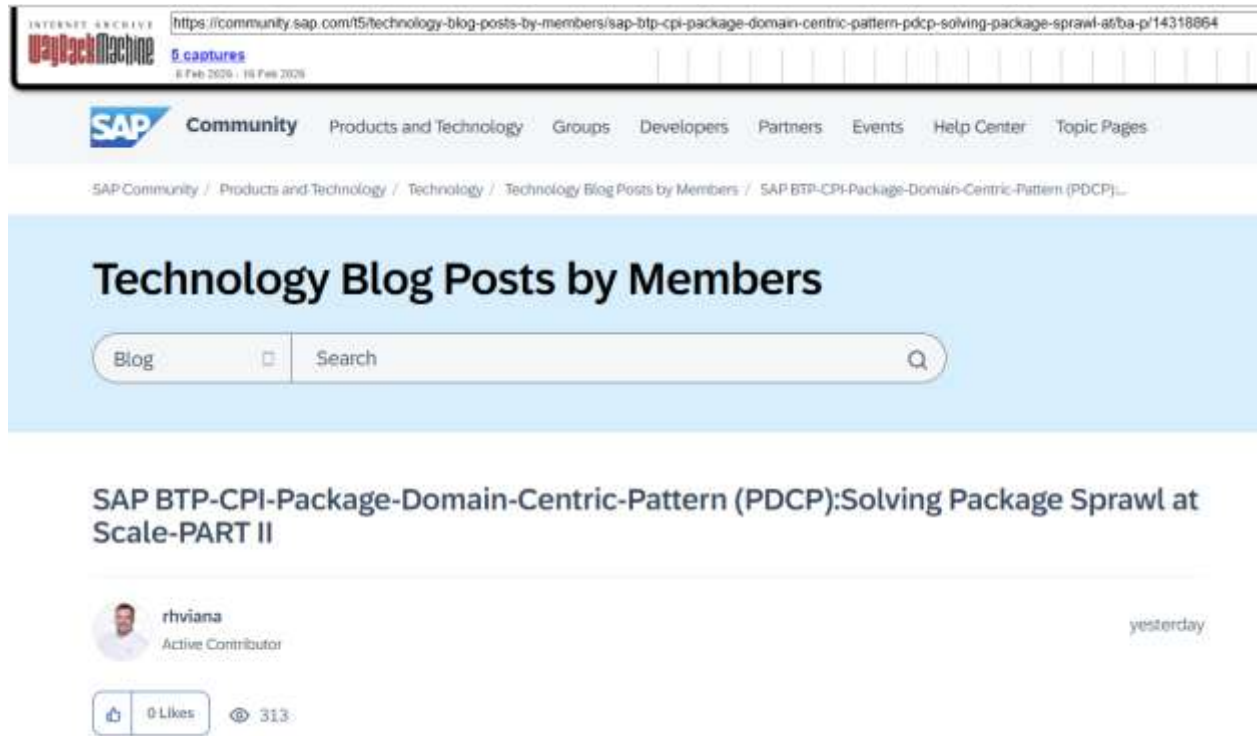


Figure A.2 — PDCP original publication, SAP Community, February 2026

### A.3 From Platform-Scoped Patterns to the SDIA Ecosystem

Between Feb and Apr 2026, DCRP and PDCP were extended from SAP to multiple platforms. The core invariants — domain-as-primary-key, metadata-driven resolution, immutable artifacts — held across vendors. This produced the five-layer SDIA ecosystem.

Layer	Current Name	Evolution
1 — Gateway	GDCR	Generalization of DCRP beyond SAP APIM to all enterprise gateway platforms
2 — Resolution	DDCR	The runtime resolution engine extracted from DCRP and formalized as an independent specification
3 — Orchestration	ODCP	Generalization of PDCP beyond SAP CPI to all enterprise orchestration platforms
4 — Events	EDCP	New pattern extending the domain-invariant principle to event channels, topics, and messaging
5 — Data	DDCP	New pattern extending the domain-invariant principle to data persistence contracts

DEIP governs platform binding — which gateway a domain bind to — before any specification layer.

Archived SAP SCN publications (Feb 6, 2026) serve as dated prior art, preceding Zenodo publications by ~2 months. Independently verifiable via Wayback Machine.

### A.4 Discovery Sequence — From HTTP 404 to the SDIA Ecosystem

GDCR emerged from a real structural problem: API landscapes accumulating proxies — one per vendor, project, team — until governance becomes impossible. Each step revealed the next.

Date	Discovery	Outcome
Jan 2026	HTTP 404 on SAP APIM. KVM discovery. TargetEndpoint redirect test.	The routing key concept. Domain as the organizing index.
Feb 6, 2026	First SCN blog: DCRP (Domain-Centric Routing Pattern). PDCP created.	Prior art chain begins. Wayback Machine captures Feb 6, 2026.
Feb 2026	Coffee moment: "It is just domain." The key turned.	Central insight: domain is the only valid primary key for routing.
Feb 7, 2026	First DOI published. 6 versions in 5 days. Newman running 20 hours.	GDCR v1.0. The pattern is proven empirically.
Feb 2026	Revelation: DCRP → GDCR, PDCP → ODCP, engine → DDCR.	The ecosystem naming stabilizes. Vendor-agnostic scope confirmed.
Mar 2, 2026	GDCR v6.0 published. Cross-platform validation across SAP, Kong, AWS, Azure.	Vendor-agnostic claim validated empirically.
Mar 4, 2026	DDCR v1.0 published with its own DOI.	The resolution engine gets its own specification — facade / engine separation.
Mar 5, 2026	ODCP v1.0 + SDIA v1.0 published.	Orchestration layer separated. Ecosystem architecture declared.
Mar 8–12	DEIP published — the decision layer above the stack.	The semantic control plane decision model.
Mar 13–17	EDCP — the event and fabric layer.	The complete ecosystem across 7 specifications in ~35 days of concentrated work.
Apr 2026	GDCR v8.0 — this edition — scope-refactored facade spec.	Three artifacts, ABNF grammar, plane separation, 9 configurations validated, ~2M requests.

The insight is simple: domains are stable, vendors are volatile. Route by domain, not by vendor. `/<domain>/<entity>/<action>/target` outlasts any backend. Technology changes quarterly. Business processes last decades.

## Appendix B — Glossary

Definitions of terms used normatively throughout this specification. Terms are listed alphabetically.

Term	Definition
Action	The canonical verb segment of the Semantic URL (e.g., create, post, approve). Must resolve to one of the 15 canonical action codes defined in §5.1. Enforced by R-03.
Company	The organizational tenant prefix in the Proxy DNA and Metadata Container name (e.g., acme). Identifies the owning enterprise entity.
Control Plane	The metadata container attached to the proxy. Carries all variable backend information (URLs, vendors, regions, versions). Evolves at runtime without modifying the proxy. See §4.1.
Data Plane	The proxy artifact itself — the deployed gateway object, the Semantic URL contract, and the PVRL grammar. Deployed once, structurally frozen thereafter. See §4.1.
DDCR	Domain Driven-Centric Router. Layer 2 of the SDIA ecosystem. The runtime policy or plugin (JavaScript, Lua, Python, C#, Java) that reads the metadata and resolves the backend target. Specified in DDCR v3.0.
DEIP	Domain Enterprise Integration Pattern. The decision model above the SDIA stack. Governs platform binding — which gateway platform a given domain should bind to.
Division	Optional granularity segment in the three artifacts (e.g., consumer, b2b). When present, appears after sector and before region in canonical order (R-05).

Term	Definition
Domain	The business domain token (e.g., sales, finance, logistics). The primary key of the gateway facade. The invariant preserved unchanged across all three GDCR artifacts and across every downstream SDIA layer (R-04).
Entity	The business entity segment of the Semantic URL (e.g., orders, invoices). Must be a plural noun (R-02).
Facade	The consumer-facing surface of the gateway — the Semantic URL contract exposed externally. GDCR governs the facade; DDCR governs the resolution behind it.
KVM	Key-Value Map. The generic term for the metadata store attached to the proxy — implemented as SAP APIM KeyValueCollection, Redis, DynamoDB, Azure Named Values, or similar depending on the platform.
Metadata Container	The named KVM, Redis, or equivalent key-value store attached to the domain proxy (e.g., acme.sales.o2c.routing). The control plane of GDCR. Its name is structurally aligned with the Proxy DNA (R-08).
Process	The business process code in the Proxy DNA and Metadata Container name (e.g., o2c, r2r, s2p, p2p). One of 10 canonical process tokens defined in §5.2.
Proxy DNA	The immutable name of the domain-scoped gateway proxy object (e.g., acme.sales.o2c.proxy). One per domain. Never modified after deployment. Governed by the ABNF grammar in §5.2.
PVRL	Probabilistic Vocabulary Restriction Language. The unified grammar family shared across the SDIA ecosystem. Level 1 governs GDCR artifacts; Levels 2–5 govern DDCR, ODCP, EDCP, and DDCP respectively.
Region	Optional granularity segment (e.g., emea, apac, euwest). When present, appears after division in canonical order (R-05).
Resolution	The runtime process of computing the backend target URL from the metadata container. Performed by the DDCR policy or plugin. Not performed by GDCR.
SDIA	Semantic Domain Integration Architecture. The umbrella architecture that composes the five layers (GDCR, DDCR, ODCP, EDCP, DDCP) into a single end-to-end domain semantic topology.
Sector	Optional granularity segment representing industry sector (e.g., retail, corporate). When present, appears first in the optional segment canonical order (R-05).
Semantic URL	The consumer-facing address exposed by the domain proxy (e.g., /sales/orders/create/salesforce). Expresses business intent, not backend topology. Permanent — survives vendor replacement. Governed by the ABNF grammar in §5.1.
Sender	The consumer identifier, transmitted as an HTTP header — not as a URL segment. The baseline validated URL form is /domain/entity/action/target with sender in the header. See Abstract.
Target	The backend vendor or system identifier in the Semantic URL (e.g., salesforce, s4hana, ariba). Must be explicitly declared (R-07). Alphanumeric lowercase only (R-01) — hyphens, environment suffixes, and version qualifiers belong in metadata, not in the URL.