GambillData

Real-World Medallion Architecture Blueprint

A practical field guide for naming, ownership, access, governance, and when Platinum actually makes sense.

How to use this guide

- Use this as a design default, not a religion.
- Pressure-test naming, ownership, and access before you build a single production table.
- Default examples use a Databricks-style catalog.schema.table namespace, but the operating model generalizes to other lakehouse platforms.

What is inside

- Naming convention template
- Layer ownership matrix
- Access model checklist
- Governance checklist
- Guide for when Platinum makes sense

Operating model in one glance

Catalog: layer + environment

Bronze schema: source or vendor

Silver schema: business domain

Gold schema: trusted business-facing domain

Platinum schema: system-facing or AI-ready domain

Semantic: broad governed business consumption

Rule of thumb

If the data still thinks like the source system, it belongs in Bronze. If it thinks like the business, it belongs in Silver, Gold, or Platinum.

Example namespace ladder

Layer	Example namespace	Why it belongs there
Bronze	bronze_prod.salesforce.account	Source-aligned raw intake with load metadata and traceability
Silver	silver_prod.revops.customer	Conformed, reusable business-aligned entity
Gold	gold_prod.finance.mrr_monthly	Trusted business-facing metric table
Platinum	platinum_prod.ai.support_agent_context	System-facing or AI-ready serving contract
Semantic	semantic_prod.executive.kpi_model	Broad governed consumption layer for business users

1. Naming Convention Template

Build the hierarchy so each level answers one question.

The most common naming failure is trying to make one object name carry every idea at once. Keep each level narrow, useful, and boring in the best possible way.

Design rules

- One axis per level. Catalog tells you layer and environment. Schema tells you source or business domain. Table tells you entity, metric, event, or serving contract.
- Do not repeat information already expressed higher in the namespace.
- Source thinking belongs in Bronze. Business thinking begins in Silver and stays through Gold and Platinum.
- Avoid vague labels like final, curated, ready, latest, or v2_fixed_for_real.

Recommended naming template

Object level	Default pattern	What it should answer	Examples
Catalog	<layer>_<environment>	What layer is this, and in what environment?	bronze_dev, silver_test, gold_prod, platinum_prod, semantic_prod
Bronze schema	<source_or_vendor>	Where did this data come from?	salesforce, hubspot, stripe, netsuite, workday
Silver / Gold / Platinum schema	<business_domain>	What business area owns or consumes this?	finance, revops, sales, executive, ai, operations
Table	<entity_or_output>	What exactly is this object?	customer, invoice, pipeline_coverage, customer_profile_api

Good patterns vs bad patterns

Pattern	Examples	Keep or kill	Why
Layer + env catalogs	bronze_prod | silver_prod | gold_prod	Keep	Top-level behavior, ownership, and access boundaries are obvious
Source schemas in Bronze	bronze_prod.salesforce.opportunity	Keep	Bronze should still reflect where the data came from
Business schemas in Gold	gold_prod.finance.mrr_monthly	Keep	Gold should think like the business, not the source
Vague suffix soup	gold_final_v2 | cleaned_ready	Kill	Versioning language hides purpose and creates confusion
Business schema in Bronze	bronze_prod.finance.invoice	Kill	Bronze should not pretend source data is already business-aligned

Source schema in Gold	gold_prod.salesforce.account_raw	Kill	Gold is not where raw source tables go to wear a tie

Naming decision check

Before you create a new object, ask three questions: 1) What layer contract does it satisfy? 2) Is it still source-oriented or already business-oriented? 3) Who is the intended consumer? If you cannot answer those cleanly, the name is probably hiding a design problem.

2. Layer Ownership Matrix

Who should own what, who should influence it, and what "done" looks like.

A clean namespace does not save you if nobody owns the layer. Ownership is where medallion architecture either becomes an operating model or decays into organized confusion.

Primary ownership by layer

Layer	Organized by / purpose	Primary owner	Key partners	Done when
Bronze	Source or vendor / controlled raw intake	Data Engineering or Platform Engineering	Security, source system owners	Data lands reliably, is replayable, and is traceable
Silver	Business-aligned reusable structures	Data Engineering	Data stewards, analytics engineering, domain owners	Entities are conformed, reusable, and quality rules are enforced
Gold	Trusted business-facing data products	Analytics Engineering, BI, or domain data product owner	Finance, RevOps, leadership stakeholders	Metrics are certified, definitions are stable, and adoption is intentional
Platinum	System-facing and AI-ready serving outputs	Data Engineering plus consuming system team	Application teams, ML or AI teams, platform teams	Downstream systems can rely on shape, privacy, freshness, and contract stability
Semantic	Governed business consumption layer	Analytics Engineering or BI	Business owners, reporting consumers	Business users can answer questions without wandering into raw layers

Role expectations by persona

Persona	What they should own	What they should influence	What they should not be doing
Data Engineers	Bronze pipelines, Silver conformance, serving contracts	Quality rules, lineage, access patterns	Building surprise gold metrics without business definition
Analytics Engineers / BI	Gold outputs, semantic models, trusted consumption paths	Business metric logic, certification, documentation	Treating Bronze as an analyst playground
Data Stewards / Domain Owners	Definitions, stewardship, approval of trusted outputs	Silver conformance rules, Gold business meaning	Owning ingestion mechanics they do not operate
Platform / Security	Access controls, masking, environment guardrails	PII policy, policy enforcement, monitoring	Acting as the sole owner of business definitions
Business Users	Validation, adoption, decision-making	Feedback on trust and usability	Creating ad hoc production tables because it felt faster

Ownership pressure test
If a table breaks at 2 a.m., who fixes it? If a KPI definition changes, who approves it? If a sensitive field leaks, who failed the guardrail? If those answers are fuzzy, ownership is not real yet.

3. Access Model Checklist

Least privilege is not bureaucracy. It is how you stop mistakes from scaling.

Broad access feels efficient right up until everyone builds a different definition of revenue or someone drags sensitive raw fields into a dashboard. Keep consumers close to trusted layers.

Recommended access pattern

Layer	Default access	Purpose of access	Guardrail
Bronze	Data Engineering and Platform Admins only	Operate ingestion, replay, troubleshoot raw landing issues	No casual analyst or business-user access
Silver	Data Engineers, Analytics Engineers, select analysts, approved data science use	Build from standardized reusable structures	Masking, hashing, and data minimization should already be enforced
Gold	Analysts, BI teams, approved business consumers	Consume trusted business-facing data products	Certification before broad exposure
Platinum	Only teams that need system-facing or AI-ready contracts	Serve applications, reverse ETL, APIs, models, or agents	Access by use case, not curiosity
Semantic	Broad business use through approved tools	Make governed answers easy to consume	Keep the business away from raw blast radius

Access checklist

- Bronze access is restricted to the people responsible for ingestion and platform operations.
- Silver access is broader than Bronze, but still intentionally gated and logged.
- Sensitive fields are masked, hashed, or removed before broad Silver and Gold exposure.
- Gold outputs are certified before they become the default business path.
- Semantic is the preferred layer for broad non-technical consumption.
- Platinum access is granted only when teams need stable serving contracts for systems or AI workflows.
- Sandbox spaces are isolated from production and have cleanup expectations.

Common access anti-patterns

- Analysts querying Bronze as a normal operating model
- Business users pulling raw source snapshots into reporting tools
- Everyone reading everything because permissions were "temporary"
- AI or application teams hitting unstable analyst tables instead of governed serving contracts

> **Guardrail to remember**
> The farther raw data is from the average consumer, the fewer accidental problems you create. Fewer security surprises. Fewer conflicting numbers. Fewer meetings where five smart people all explain revenue differently.

4. Governance Checklist

Governance is not the deck. Governance is the default behavior of the platform.

If governance only exists in a policy PDF, it is not embedded. It is just a PowerPoint deck with ambitions. Good architecture makes the right behavior the easy behavior.

Minimum governance controls by topic

Control area	What good looks like	Where it starts	Who usually owns it
Naming standards	Consistent catalog, schema, and table patterns with clear rules	From day one	Architecture lead / Data Engineering
Ownership metadata	Every trusted object has an explicit owner or owning team	Silver and above	Data Engineering + Analytics Engineering
Documentation	Critical tables and fields are described in plain language	Silver and above	Owning team
Lineage	Teams can trace upstream and downstream dependencies	Bronze onward	Platform + owning team
Data quality	Checks are defined by layer, not bolted on randomly	Bronze onward	Owning team + stewards
PII controls	Masking, hashing, minimization, and policy enforcement are deliberate	Silver onward for broad exposure	Security + platform + owning team
Certification	Gold and Semantic outputs are marked as trusted on purpose	Gold and Semantic	Analytics Engineering / BI / domain owner
Change control	Breaking changes are reviewed, communicated, and versioned intentionally	Gold and Platinum especially	Owning team

Governance checklist by layer

Layer	Must-have controls	Nice-to-have controls
Bronze	Source metadata, freshness checks, lineage, restricted access	Schema drift alerts, landing scorecards
Silver	Owner, documentation, conformance checks, masking or hashing, key health	Steward sign-off on core entities
Gold	Certification, KPI validation, SLA monitoring, business definitions, master data discipline	Adoption scoring, usage monitoring
Platinum	Serving contract validation, privacy review, downstream compatibility, latency expectations	Automated contract tests and rollback playbooks

Semantic	Governed access, business glossary alignment, clear naming, report traceability	Consumer usage analytics

Governance starter list

If you are early, start with five things: naming rules, explicit owners, layer-based quality checks, restricted Bronze access, and certification for trusted Gold outputs. Those five get you out of wishful thinking territory fast.

5. When to Use Platinum

Use Platinum when the platform must serve systems, models, or agents - not just dashboards.

Not every team needs a Platinum layer. But when you need one, pretending Gold covers everything usually creates brittle integrations, messy reverse ETL, and AI workflows that consume the wrong data.

Platinum makes sense when ...

Use case	Why Gold is not enough by itself	Example output
Reverse ETL / activation	The downstream system needs a stable, narrow, system-facing payload	platinum_prod.sales.crm_activation_feed
API or application serving	Applications need a contract they can rely on, not analyst-shaped tables	platinum_prod.customer.customer_profile_api
Operational sync	ERP, CRM, support, or operations tools need curated machine-consumable data	platinum_prod.operations.erp_sync_payload
Agent context / AI workflows	Agents need governed context tables, chunk-ready content, or embedding-ready text	platinum_prod.ai.support_agent_context
Model or feature serving	Low-latency or model-consumption paths need stable, governed inputs	platinum_prod.ai.feature_ready_customer_state

Do not use Platinum when ...

- It is only for human reporting and a Gold or Semantic output already solves the need.
- The team has not stabilized Gold definitions yet and is trying to skip the trust step.
- Someone wants a Platinum layer because it sounds advanced, not because a real serving contract exists.
- There is no identified downstream system, model, or agent that actually depends on it.

> **Simple decision rule**
> If this output is mainly for humans, keep it in Gold or Semantic. If systems, models, or agents need a stable serving contract, Platinum probably makes sense.

Platinum readiness checklist

- You know the downstream consumer and the exact contract it needs.
- Privacy review has happened before the data reaches applications or AI workflows.
- Latency and freshness expectations are explicit.
- The output is documented, owned, and monitored.
- You can explain why this cannot simply remain a Gold table.

6. Implementation Starter Blueprint

A practical sequence for teams cleaning up an existing mess or building from scratch.

You do not need to redesign the universe in one sprint. You do need to make a few intentional decisions early so the platform does not drift into architecture soup.

First seven decisions to make

Step	Decision	Why it matters
1	Lock the catalog pattern: <layer>_<environment>	It sets the top-level contract for behavior, ownership, and access
2	List Bronze schemas by source or vendor	It prevents business logic from leaking into raw intake
3	Define Silver business domains	It gives reusable conformed structures a clean home
4	Name Gold products the business will actually trust	It separates trusted outputs from random downstream tables
5	Decide whether Platinum is real or just fashionable	It avoids building a serving layer with no consumer
6	Assign owners for each layer and key outputs	It turns architecture into an operating model
7	Enforce the access and governance minimums	It stops cleanup debt from compounding

Pressure-test questions for your team

- Can we explain our naming rules in under two minutes?
- Do our Bronze schemas reflect sources, not business domains?
- Can business users stay in Gold or Semantic for most questions?
- Do we know which Gold outputs are certified and who owns them?
- If we say we need Platinum, can we name the system, model, or agent that depends on it?

> **Need help pressure-testing this architecture?**
> Gambill Data works with teams that need practical help cleaning up naming, ownership, access, serving contracts, and modern data-platform design. Reach out at chris.gambill@gambilldataengineering.com if you want a second set of eyes on your setup.