



SEPT 22 - 24, 2026 | NAIROBI, KENYA

Designing APIs & MCP for AI Readiness



Nuwan Dias
VP and Deputy CTO

A group of five young adults (three women and two men) are sitting on a stone ledge next to a canal. They are all looking at their smartphones and smiling. The background shows a city street with buildings and a bicycle. A yellow line graphic is overlaid on the right side of the image, framing the text.

**APIs are built for
customer
experiences!**

A human centric retail experience

User actions

- Searches for “*Running shoes under \$150*”
- Filters by size and brand
- Views a product
- Adds it to cart
- Checks out

User experience

- Linear
- Intentional
- Rate-limited by human behavior
- Context held in the UI, not the API



APIs designed to support this experience

GET /products?query=running+shoes&max_price=150

GET /products/{id}

POST /cart/items

POST /checkout

“Today, most AI agents use the same APIs built for human centric experiences!”



An agentic retail experience

“Find me the best running shoes under \$150, in my size, available to deliver within 2 days, with the best reviews.”

Actions

- Query *hundreds* of products
- Repeatedly re-rank results
- Compare prices across vendors
- Re-check inventory and delivery windows
- Simulate multiple carts
- Retry aggressively when responses are unclear
- Chain calls across multiple APIs

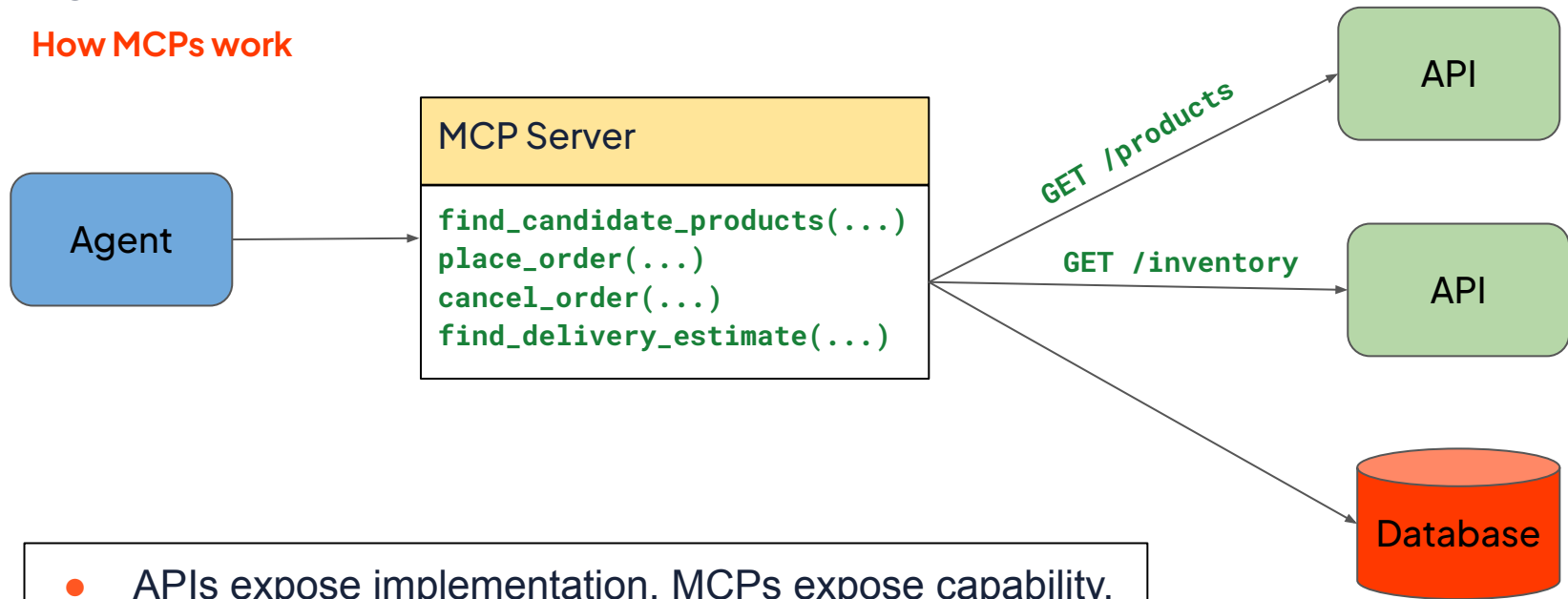
Experience

- Exploratory
- Non-linear
- Probabilistic
- Machine-driven
- Cost-blind unless constrained



Agents use MCP servers. Can it help?

How MCPs work



- APIs expose implementation, MCPs expose capability.
- MCP is an integration layer, not a cure!

Design enhancements to make APIs AI-ready

Offer bulk fetch operations to reduce fan-out

Agents iterate over items looking for “best-possible” options. They end up calling endpoints N times for N items.

- **Scenario: Check candidate products to evaluate suitability**

```
for product in candidates:  
    GET /products/{product_id}:  
    check eligibility
```

←----- Fan out

- **Solution**

- Offer a bulk fetch operation

```
POST /products-batch-get  
{  
    "ids": [1, 4, 7, 13, 21]  
}
```



Use machine readable errors and semantics

- **Prevent using ambiguous error codes and messages**

- “**Product unavailable**” could mean many things, such as “discontinued, not available in stock, variant not available, etc”

```
if error contains "unavailable":  
    try variants endpoint  
    try inventory endpoint  
    try different fulfillment
```

- **Use machine readable error codes**

- Ex: **OUT_OF_STOCK, INVALID_VARIANT, PRICE_CHANGED**
- These provide direct context to agents to make appropriate decisions.
 - Budget, review ratings, delivery SLA, preferences, etc.

First-class Idempotency and safe retries

Not all retries by agents are safe, they can create duplicate write entries.

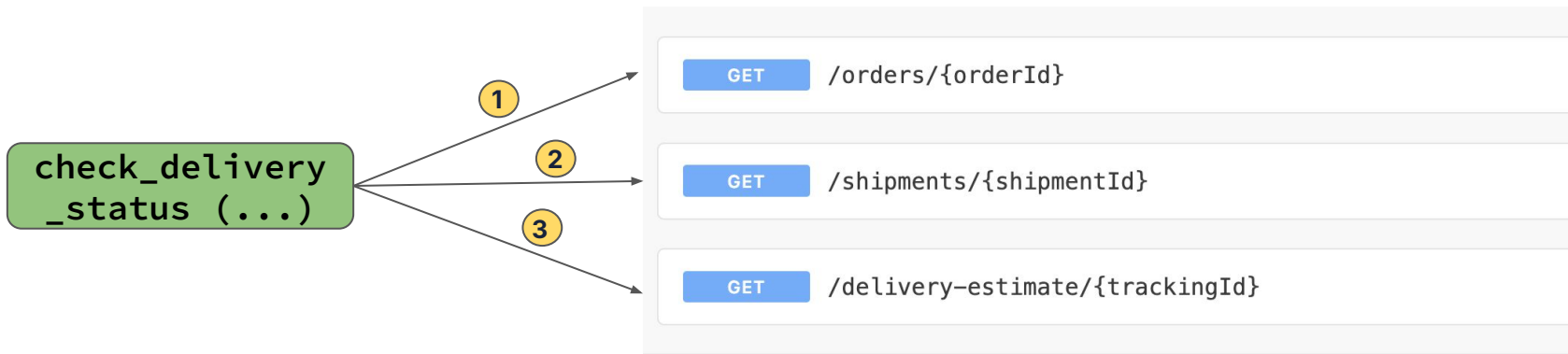
- **Scenario: Agent initiates a checkout**
 - Payment takes longer
 - Agent retries!
- **Use Idempotency-Keys to prevent duplicate writes**
 - Client obtains a unique idempotency-key for each unique request.
POST /checkout
Idempotency-Key: 8f2d-abc-9012
 - A given Idempotency key is only processed once by a Server.
 - For subsequent requests of the same Idempotency key, the initial response is sent back.



From APIs to MCP tools: Think agent-first

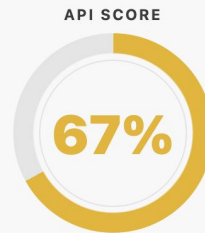
MCP Tool

API Endpoints



- Converting every API endpoint to an MCP tool is an anti-pattern.
- Decide what tools your agent needs first.

AI readiness checks for APIs



AI Readiness

A set of rules that ensures that REST APIs are well-documented and structured to be easily understood and utilized by AI systems.

PASSED CHECKS

Completed checks out of the total checks run

36/64

AFFECTED ENDPOINTS

Endpoints impacted by one or more findings

9

[View issues](#)

ERRORS

Issues that require immediate attention

6

[View issues](#)

WARNINGS

Potential risks and improvement opportunities

56

[View issues](#)

AI Analysis

AI-powered evaluation to uncover agent-readiness gaps and recommend prioritized fixes.

READY 15 findings available

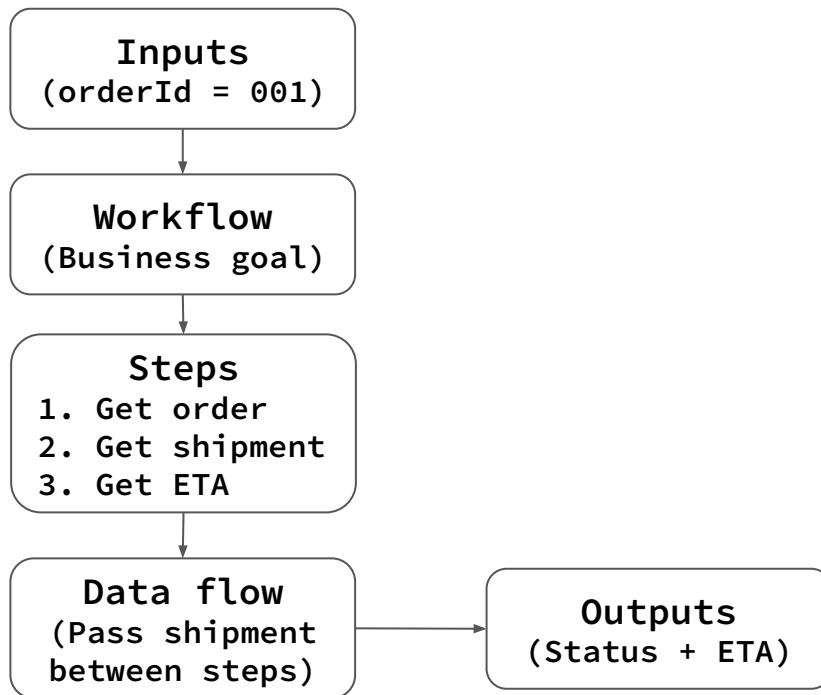
[View findings](#)





Arazzo

A mechanism that can define sequences of calls and their dependencies to be woven together and expressed in the context of delivering a particular outcome or set of outcomes when dealing with API descriptions.



In summary, AI-ready API designs result in APIs that are..

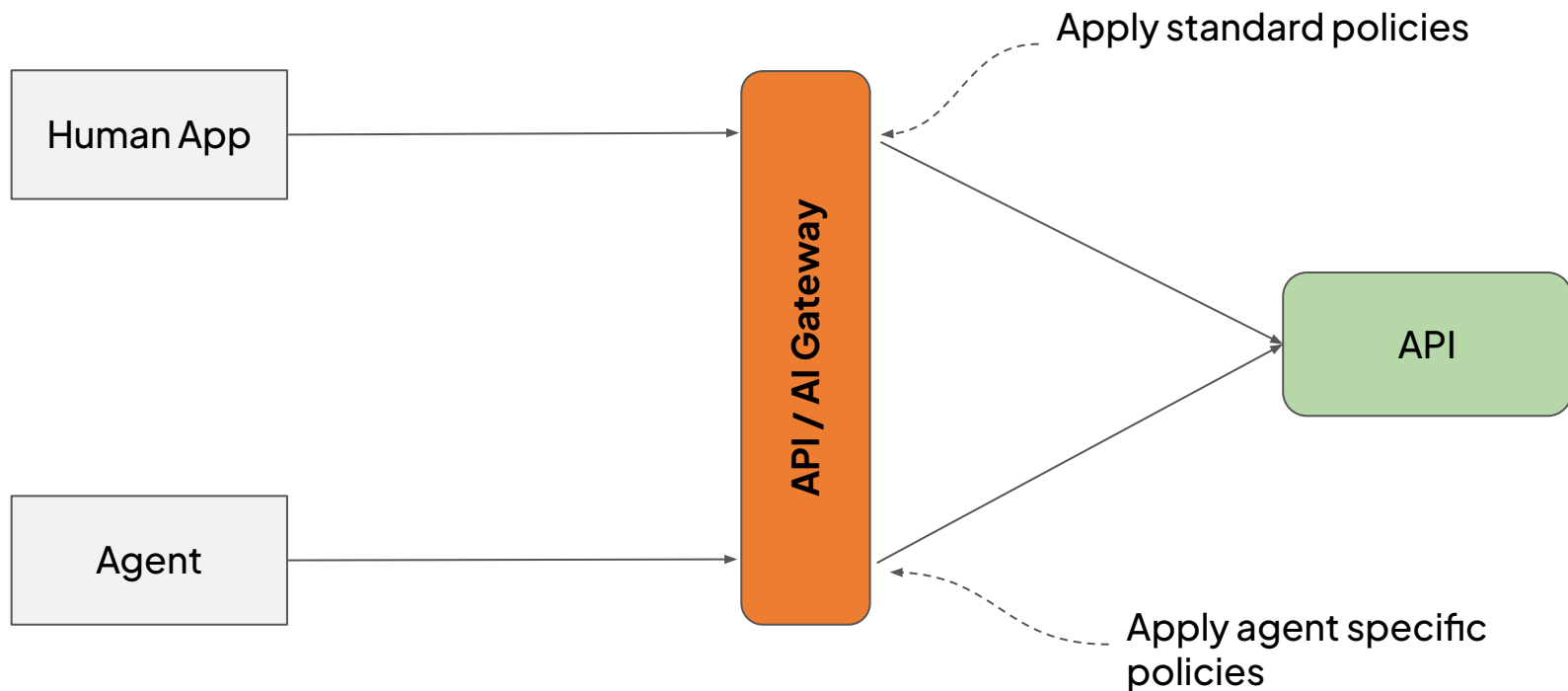
- **Semantic**
 - Clear meaning with machine readable error codes
- **Bounded**
 - Provides limits, pagination, field selection
- **Safe**
 - Idempotency, dry run, explicit side effects
- **Efficient**
 - Bulk operations
- **Auditable**
 - Reason codes, correlation IDs



Runtime enforcements for making APIs AI-ready

Identify the caller to apply relevant policies

Not all consumers are to be treated the same!



Safe-write protections for autonomous clients

- **Enforce idempotency** on *all* writes
 - Require `Idempotency-Key`, reject if missing
- **Two-phase commit / confirmation gates**
 - Block irreversible operations unless a `confirmed=true` + user consent token is present
- **Dry-run requirement**
 - Require `checkout-dry-run` within N minutes before allowing `confirm-checkout`
- **Write scopes**
 - Allow agents to “create cart” but not “place order” unless elevated scope



Deep observability: trace the *task*, not just the request

Better telemetry is critical for managing agent traffic

- **Propagate Task-ID/Session-ID across calls**
 - X-Task-Id, `transparent`
- **Per-task metrics**
 - Calls per task
 - Cost units per task
 - Retries per task
 - Top endpoints per tool
- **Audit logs** for tool-invoked writes
 - Who, what, why



Policy enforcement that is capability aware

- API Gateways usually enforce policies by resource path
 - **POST /orders -> authorization policy**
- AI-ready API Gateways should,
 - Enforce policies by capability
 - By tool name: **place_order -> authorization policy**
 - Enforce limits by capability
 - Limit **place_order** to 1 call per minute per consumer



Caching and deduplicating data

- Agents retry a lot
 - Re-evaluate, re-rand, re-check
- An AI-ready API Gateway can cache responses to prevent back-end APIs being exhausted.
 - Short TTL caching for expensive read operations
 - Stale-while-revalidate for list/search operations
 - Request coalescing (deduplication, with one backend call for many waiters)

Gateways are critical to making API runtimes AI-ready

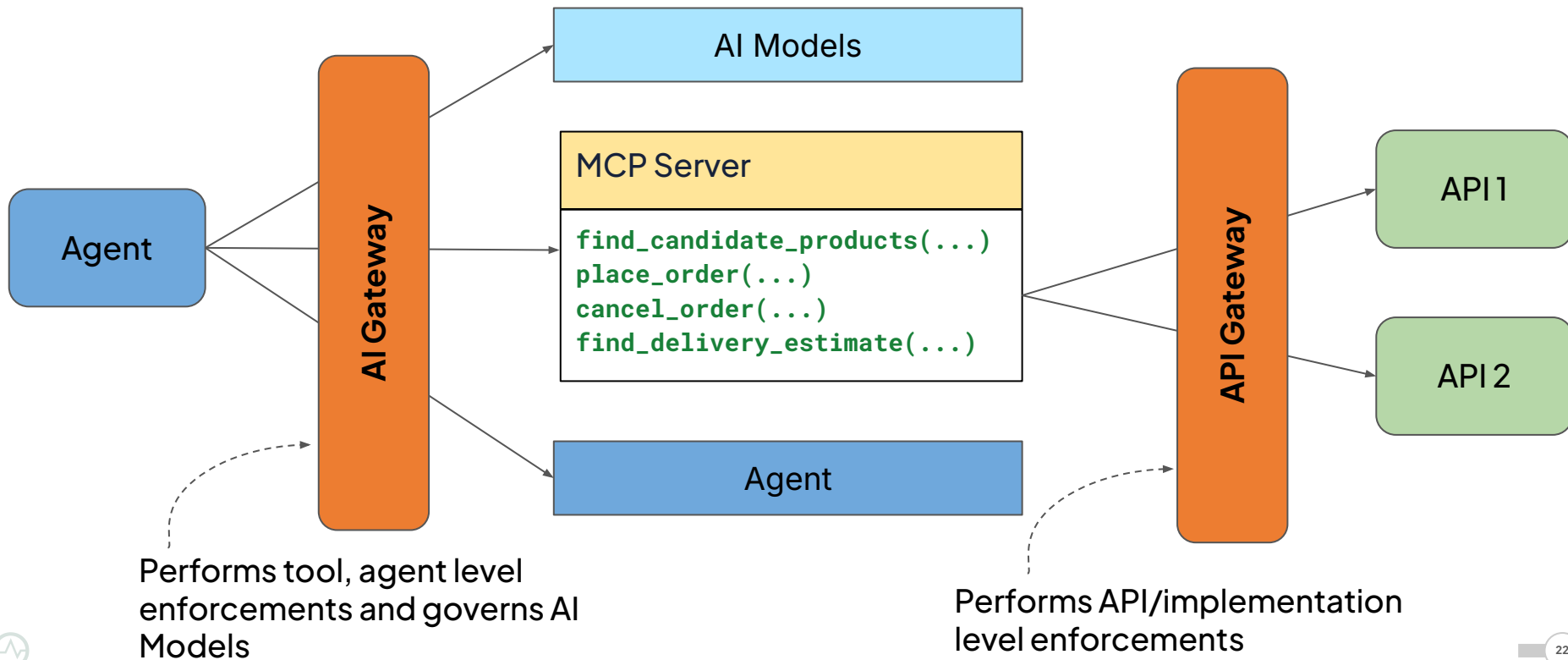
Modern AI-ready API Gateways provide

- **Identity:** agent/tool/user context
- **Budget controls:** rate + concurrency + cost units
- **Safety:** idempotency + confirm + dry-run
- **Efficiency:** pagination + field selection + dedupe/caching
- **Semantics:** standardized errors + validation
- **Resilience:** anomaly detection + circuit breakers
- **Observability:** task-level tracing + audit



Gateways are critical to making API runtimes AI-ready

Reference architecture of Gateways in action in the enterprise



Discoverability enforcements for making APIs AI-ready

If Developer Portals are for humans, what is it for Agents?

Today's API discovery assumes

- A human reads docs
- A human requests credentials
- A human interprets examples
- A human understands business constraints

Agents need

- Machine-readable capability descriptions
- Explicit constraints
- Structured auth flows
- Clear side-effect semantics
- Discoverable affordances

Beyond OpenAPI: Make APIs self describing for AI

OpenAPI describes structure, agents need semantics. Enhance your APIs by adding

- Explicit operation intent descriptions
- Clear side-effect classification
- Retry semantics
- Idempotency requirement flags
- Cost hints
- Rate classification
- Sensitivity classification

```
x-ai-metadata:  
  side_effect: irreversible  
  retryable: false  
  requires_confirmation: true  
  sensitivity: financial  
  estimated_cost_unit: 5
```

Provide a machine readable capability registry

Instead of scraping API docs, you can expose something like

GET /.well-known/ai-capabilities

```
{
  "capabilities": [
    {
      "name": "recommend_products",
      "description": "Find best products within constraints",
      "safe_for_agents": true,
      "side_effect": "none",
      "max_results": 50
    },
    {
      "name": "place_order",
      "safe_for_agents": false,
      "requires_confirmation": true
    }
  ]
}
```

Provide explicit Auth and Delegation Patterns

Humans are told to “get an API key”. Agents need

- OAuth On-Behalf-Of flows
- Scoped access tokens
- Short-lived delegated credentials
- Consent tokens for high-risk operations

```
x-auth:  
  flow: oauth2  
  delegation: required  
  consent_required: true
```

Create an “Agent Portal” layer

The governance/control-plane layer needed for tools

An Agent portal enforces:

- Curated tool definitions
- AI-safe operation tags
- Example invocation plans
- Risk classifications
- Runtime guardrail policies

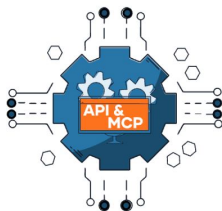
Now discovery is truly capability-driven, not endpoint-driven.



Agent Ready API Portal

APIs, MCP Servers, & Workflows - Ready for AI Agents and Developers

Discover, connect, and orchestrate APIs, MCP tools, and workflows to build applications, AI agents and automations. Browse our catalogs, subscribe and get going in minutes.

[Browse our APIs →](#)[Browse our MCP Servers →](#)[+ Discover with AI](#)

Explore & Subscribe to our APIs & MCP Servers

- Sign up with us



Developing Applications

- Generate application keys

Utilize SDKs to generate application code



Conclusion

AI-readiness isn't about adding a new spec. It's not about adding an MCP server. And it's not about flipping a gateway flag.

- Enforce proper design: so APIs encode intent, safety, and semantics.
- Govern the runtime: so probabilistic behavior is governed and observable.
- Enable discovery: so capability publication is intentional and controlled.

With agents, your APIs are not just interfaces. They are decision surfaces.





SEPT 22 - 24, 2026 | NAIROBI, KENYA

Thank You!

