



Problem

How will run-time configuration state be coordinated? Not just across internal business services, but also across other disparate services that may or may not be otherwise connected to a common messaging platform.



Solution

Add distributed configuration and coordination capabilities (DCC) to the edge nodes and make them available through your platform, addressing this particular integration challenge. The implementation can be extended into partner and cloud environments and made available across those segmented networks. While other approaches can coexist, this service provides the universal way to solve widespread application state coordination where it is needed. This implementation is typically a secure, distributed, in-memory namespace that application affinity groups use to share values and state information. Its scope includes distributed run-time application configuration and coordination, which does not have a large amount of data involved, and as such fits alongside messaging and API gateways. Additional capabilities could be added to handle stateful connection tracking.



Constraints

1. Each application environment has already made and implemented its answer to distributed coordination—which could even be to not do it at all. In any case, the goal wasn't end-to-end coordination as a common solution—creating a gap.
2. The shift to API-centric direction in application development has somewhat relaxed decisions around what technology and approach is used behind the API (including programming languages which determine tools, etc.). This creates additional disparity in implementation.
3. Even if there were common and ready services for developers to use, there is no onboarding process, documentation or test environment to simulate what the integration platform looks like. Who owns cross-application integration as a service?



Steps

1. Optimize DCC for an implementation that has the broadest support and easiest path for consumer adoption.
2. Interconnect DCC with security services and access through the inspection zone.
3. Publish a service API for the DCC (add, change, del) and connectors.
4. Use the API to establish secure namespaces for components and take over their ongoing configuration management (configuration for: network, security, data mappings, message queues, etc.) starting with edge node services.
5. Implement event processing (and correlation if applicable) and logging (service usage, etc.).
6. Coordinate/integrate with policy enforcement for guardrails.



Forces

- The shift to API-centric direction in application development has relaxed the emphasis on what you use to build behind the APIs (including languages and tools used, etc.). This disparity in implementation can add complexity when distributed coordination across those APIs is needed.
- Real-time integrated business flows are increasingly becoming the norm (e.g., retail coupons, digital payments, with online credit and fraud checks – all in less than a second).
- Digital is also driving extreme geographically distributed transactions that may still need transactional integrity (e.g., logistics businesses needing to roll back a transaction across multiple partner functions, etc.).
- Application architects are beginning to have to think about widespread integration at scale.

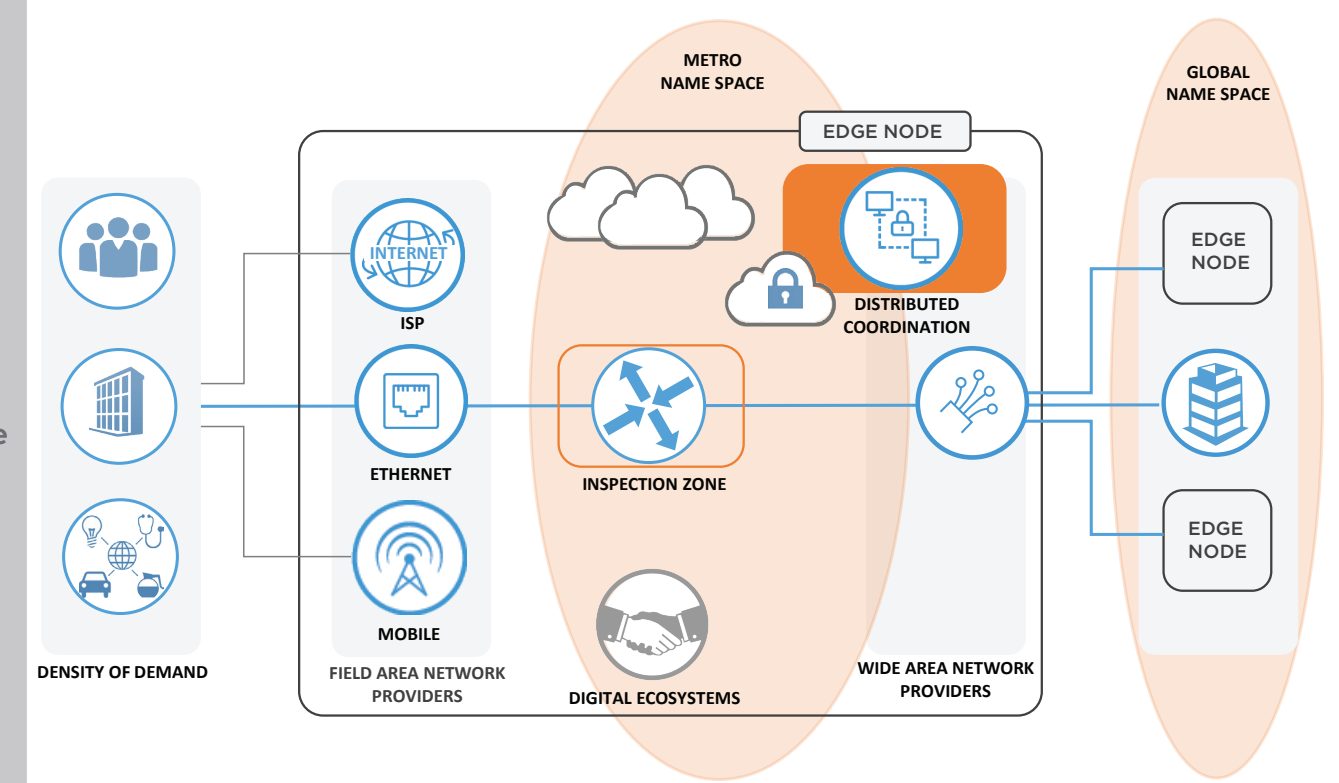


Results

- You have a secure, self-service, distributed configuration and coordination service with mixed local and global groups as needed.
- Implementation across the mesh of edge nodes either as local appliances or SaaS augmentation.
- Components can be parked when provisioned and then assigned their functional configuration. You may not want a faulty component to immediately go back into production. Configure a replacement.
- Administrative tools interact with the namespace for higher level orchestration.
- With the infrastructure-as-code, the namespace can be backed up to the distributed repository (Data Blueprint).



Reference View



* Data Blueprint — IOAKB.com