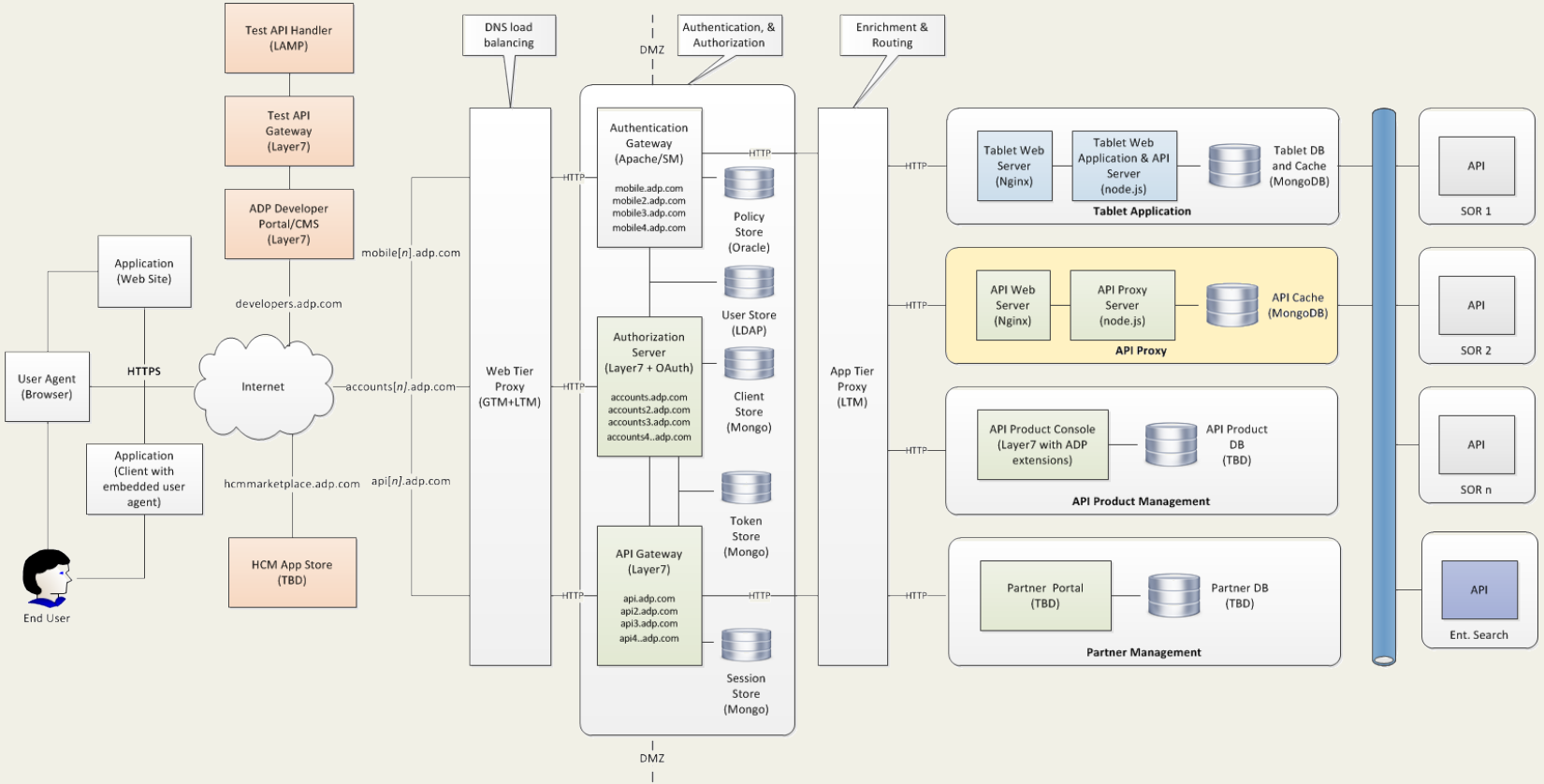
Web API Proxy

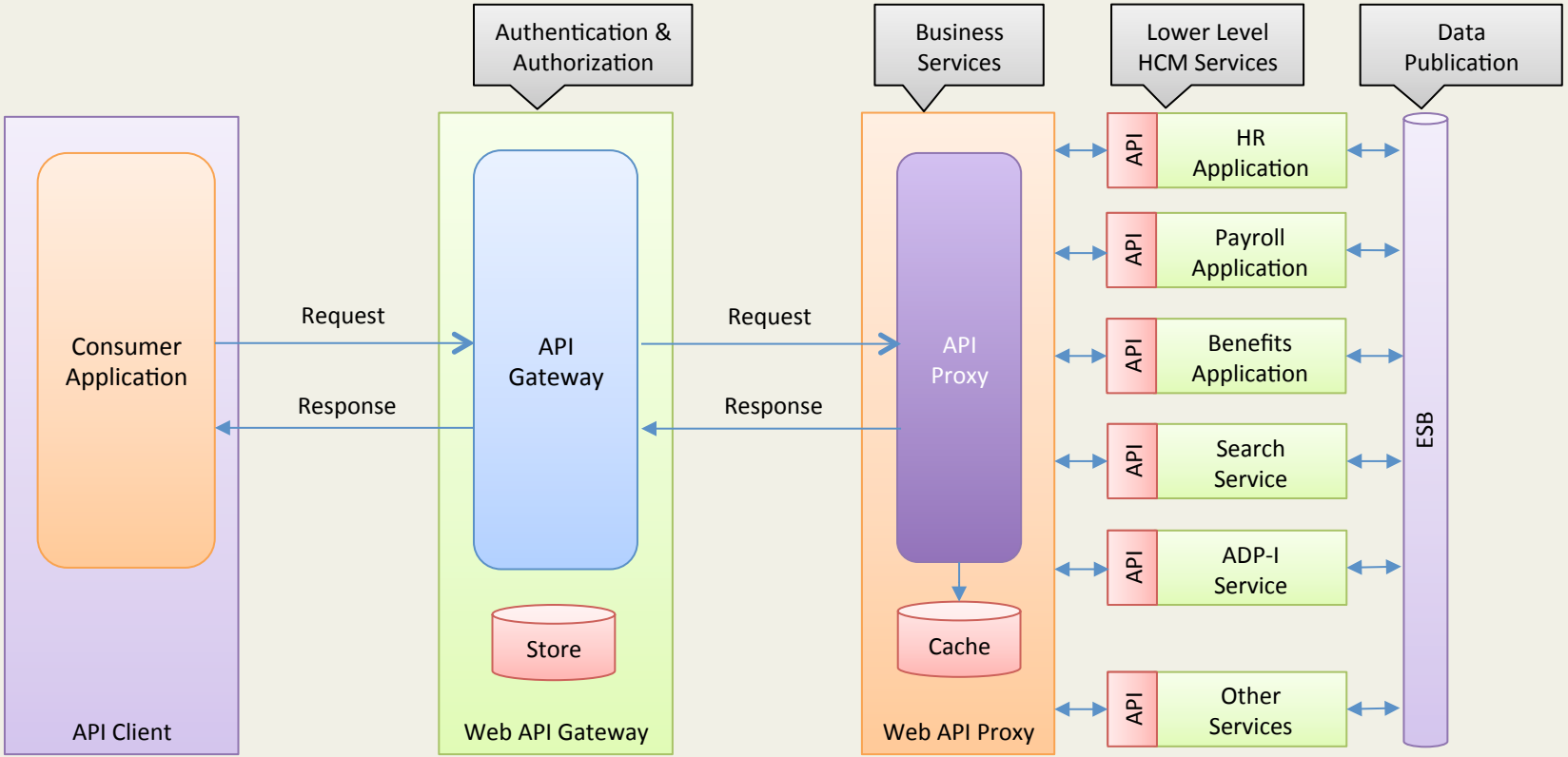
Overview

ADP Innovation Labs

Big Picture

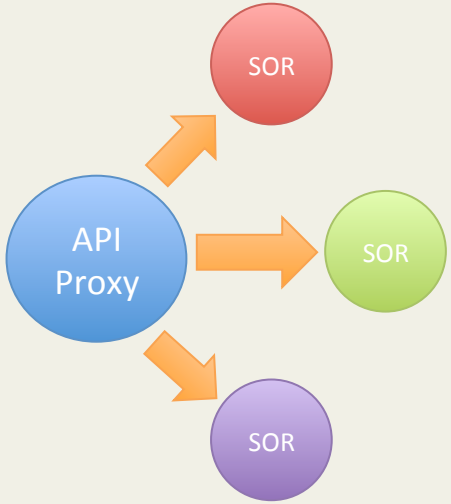

Test API Handler (LAMP)
Test API Gateway (Layer7)
ADP Developer Portal/CMS (Layer7)
Application (Web Site)
developers.adp.com
User Agent (Browser)
HTTPS
Internet
Application (Client with embedded user agent)
hcmmarketplace.adp.com
HCM App Store (TBD)
End User
mobile[n].adp.com
accounts[n].adp.com
api[n].adp.com
DNS load balancing
Web Tier Proxy (GTM+LTM)
DMZ
Authentication, & Authorization
Authentication Gateway (Apache/SM)
mobile.adp.com
mobile2.adp.com
mobile3.adp.com
mobile4.adp.com
Policy Store (Oracle)
User Store (LDAP)
Authorization Server (Layer7 + OAuth)
accounts.adp.com
accounts2.adp.com
accounts3.adp.com
accounts4..adp.com
Client Store (Mongo)
Token Store (Mongo)
API Gateway (Layer7)
api.adp.com
api2.adp.com
api3.adp.com
api4..adp.com
Session Store (Mongo)
DMZ
Enrichment & Routing
App Tier Proxy (LTM)
HTTP
Tablet Web Server (Nginx)
Tablet Web Application & API Server (node.js)
Tablet DB and Cache (MongoDB)
Tablet Application
API Web Server (Nginx)
API Proxy Server (node.js)
API Cache (MongoDB)
API Proxy
API Product Console (Layer7 with ADP extensions)
API Product DB (TBD)
API Product Management
Partner Portal (TBD)
Partner DB (TBD)
Partner Management
API
SOR 1
API
SOR 2
API
SOR n
API
Ent. Search

Logical Architecture



Web API Proxy

- API façade
- Profile-based authorization
- Routing determination
- Cache for performance and reliability
- Aggregation and orchestration
- Scalable & reliable infrastructure
- Logging and auditing
- Filters
  - Schema validation filter
  - Data privacy filter
  - Payload transformation filter
  - Data relevancy filter



API Façade+

- Decouple interface from implementation
- Promote internal independence and portability
- Centralized common functionality
  - Profile-based authorization
  - Caching
  - Routing determination
  - Schema validation
  - Payload Transformation
  - Data privacy filter
  - Others

Profile-Based Authorization

- For each user, a profile is computed based service authorization received (via ASA and/or AA API calls) from SORs
- Incoming URI is mapped to a feature
- User profile is checked to see if the feature is enabled/authorized for the user
- If authorization fails then HTTP 401 is returned

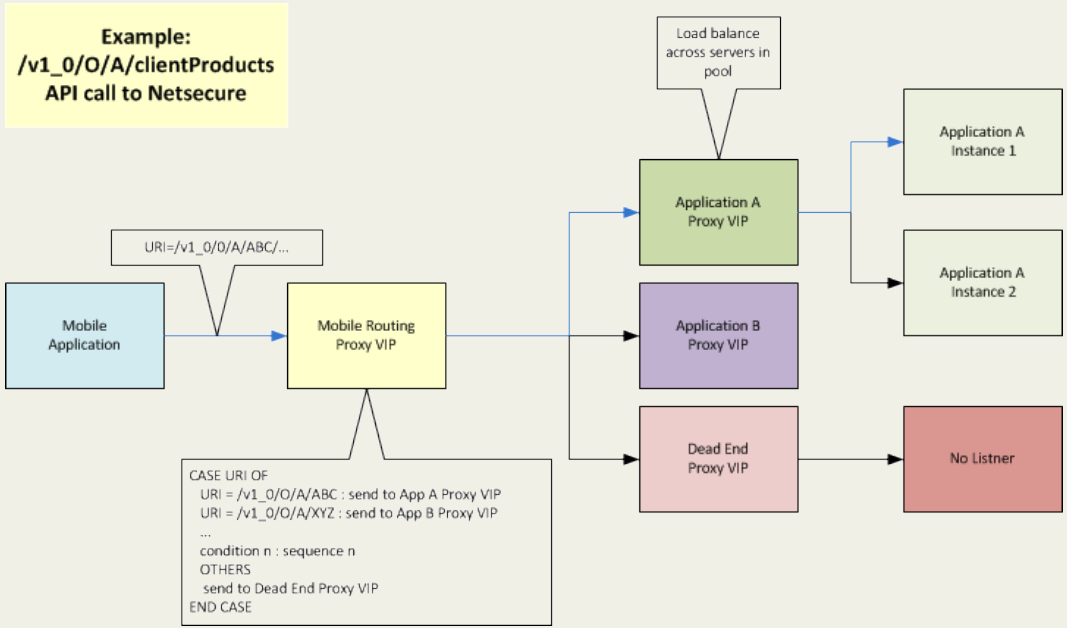
Routing

- Routing to internal service providers
- Routing based on service type (URI)
- Routing based on service provider (SOR)
- Routing based on product subscription (service authorization)
- Routing based on service instance (SORURI, ResourceGroup and SORContext)
- Routing patterns:
    1. URI-Based
    2. URI+SOR-Based
    3. URI+SOR+ResourceGroup-Based
    4. URI+SOR+SORURI-Based
    5. URI+SOR+SORURI-Based with SORContext

Routing Parameters

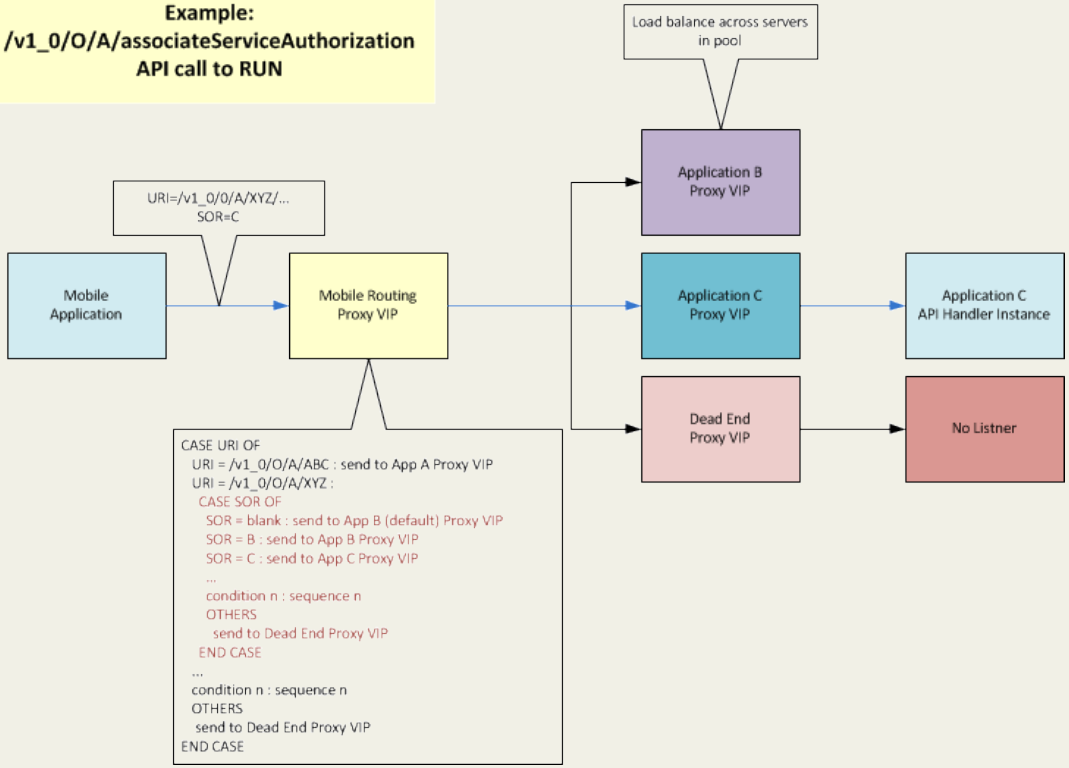
Parameter	Source	Mapped	Example(s)
URI	API Specifications	N/A	/v1_0/O/A/payStatements /v1_0/O/corporateContacts
SOR Header	AssociateServiceAuthorization JSON *applicationIdentifier.id*	No	IPay Run
SORURI Header	AssociateServiceAuthorization JSON *applicationInstanceUri.href*	No	http://10.15.80.103 http://wspproxy.adphc.com
SORContext Header	AssociateServiceAuthorization JSON *instanceIdentifier.id*	No	z92s
Portal ResourceGroup	LDAP	N/A	R8:PORTALNASPI, PORTALNAS01, PORTALNAS02, PORTALMASPI, PORTALMAS01, PORTALMAS02
			Vantage:PORTALHRII1
			WFN:PORTALWFN01, PORTALWFN02, PORTALWFN03

URI-Based

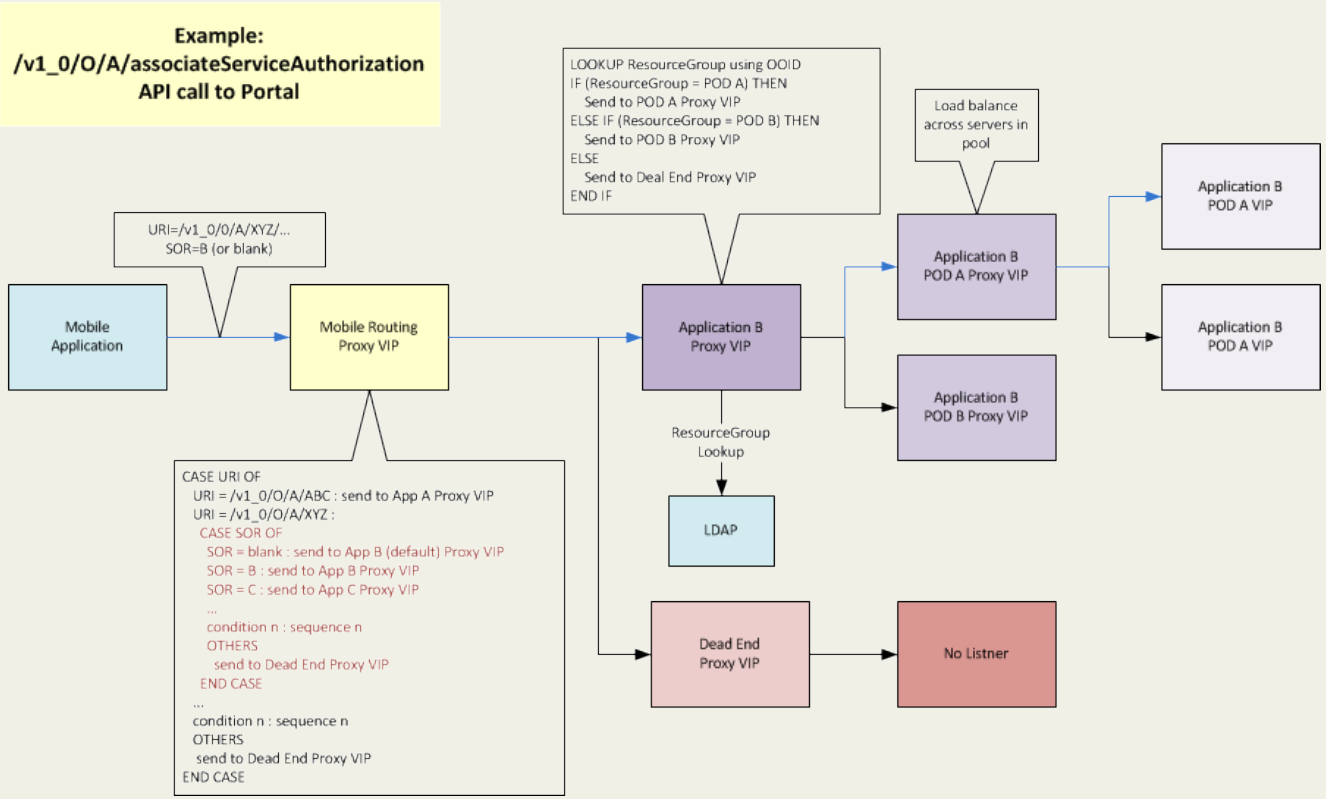

Example:
/v1_0/O/A/clientProducts
API call to Netsecure
Load balance across servers in pool
Application A Instance 1
Application A Proxy VIP
Application A Instance 2
URI=/v1_0/0/A/ABC/...
Mobile Application
Mobile Routing Proxy VIP
Application B Proxy VIP
Dead End Proxy VIP
No Listner
CASE URI OF
URI = /v1_0/O/A/ABC : send to App A Proxy VIP
URI = /v1_0/O/A/XYZ : send to App B Proxy VIP
...
condition n : sequence n
OTHERS
send to Dead End Proxy VIP
END CASE

URI+SOR-Based

Example:
/v1_0/O/A/associateServiceAuthorization
API call to RUN

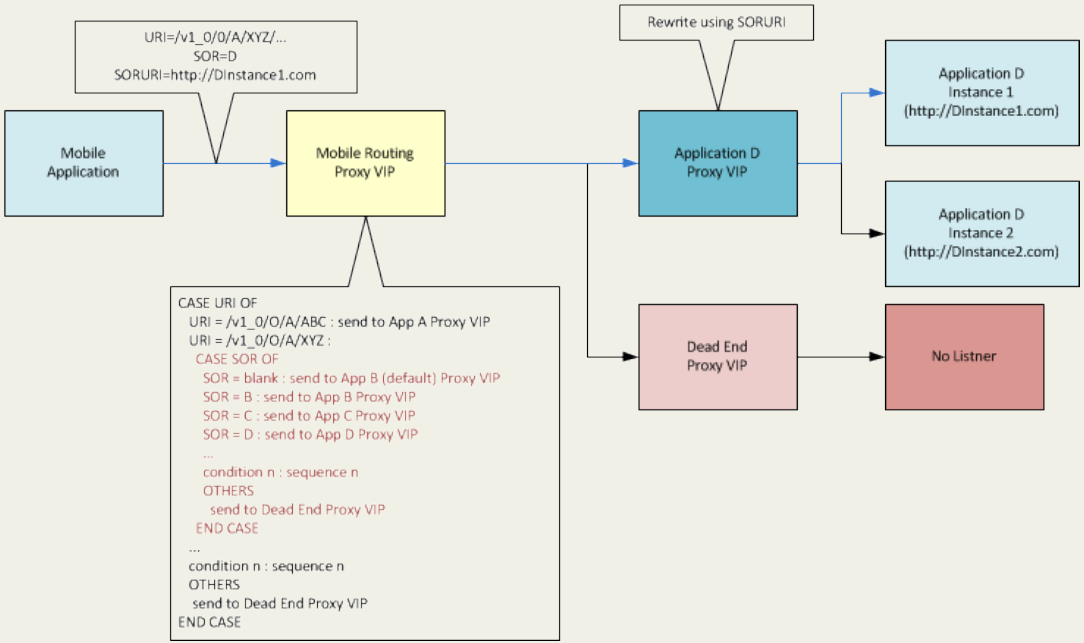


URI+SOR+ResourceGroup-Based



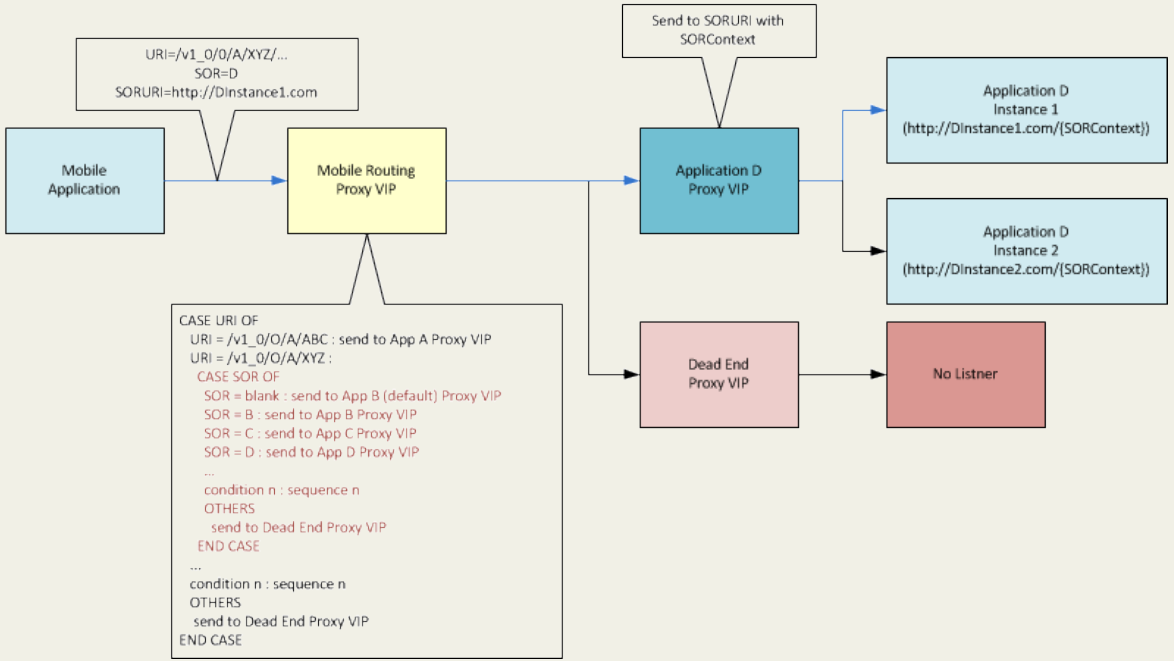
URI+SOR+SORURI-Based

Example:
/v1_0/O/corporateContacts
API call to RUN (future)



URI+SOR+SORURI-Based with SORContext

Example:
/v1_0/O/corporateContacts
API call to Enterprise

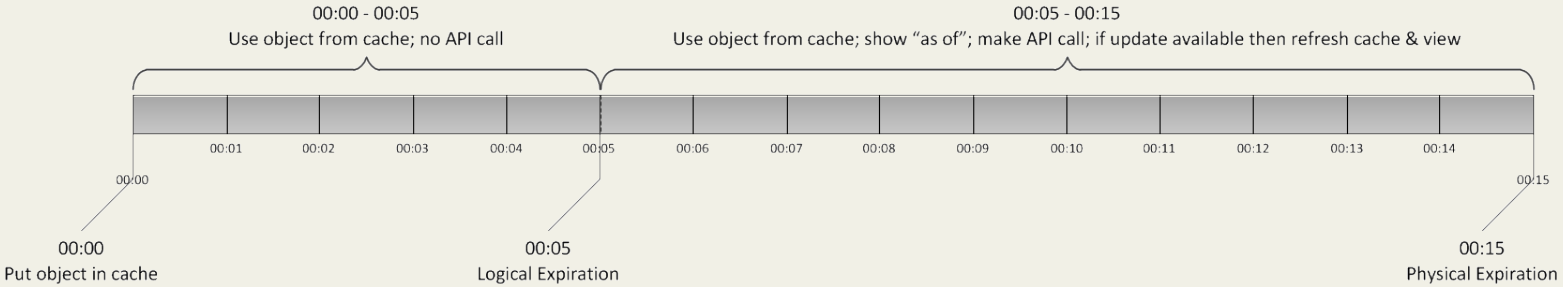


Caching

- Proxy caches API responses (JSON objects) based on configurable policy
- MongoDB NoSQL cache database – 3 shards, four-node replica sets
- Nodes distributed between two data centers
- Complete DC failure tolerated
- Cache configuration based on object type
- Each cache object has logical and physical expiration
- Enforce SLA on backend SORs when object available in cache
- Serve from cache, if available, when backend SOR down

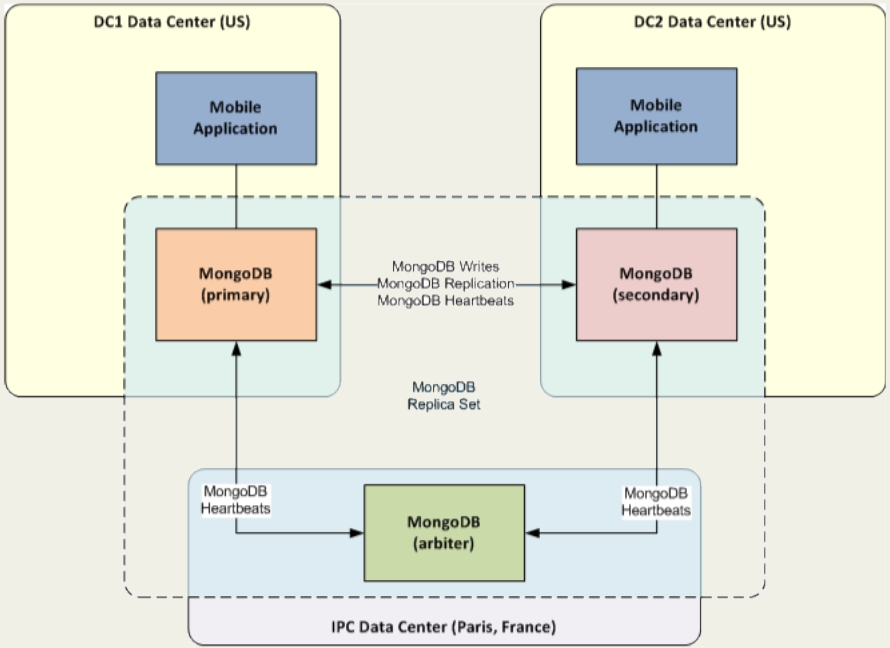
Cache Policy

- Logical and physical expiration policy per object
- Client displays cache data while attempting to retrieve updates from server in background; once update available, view is dynamically updated

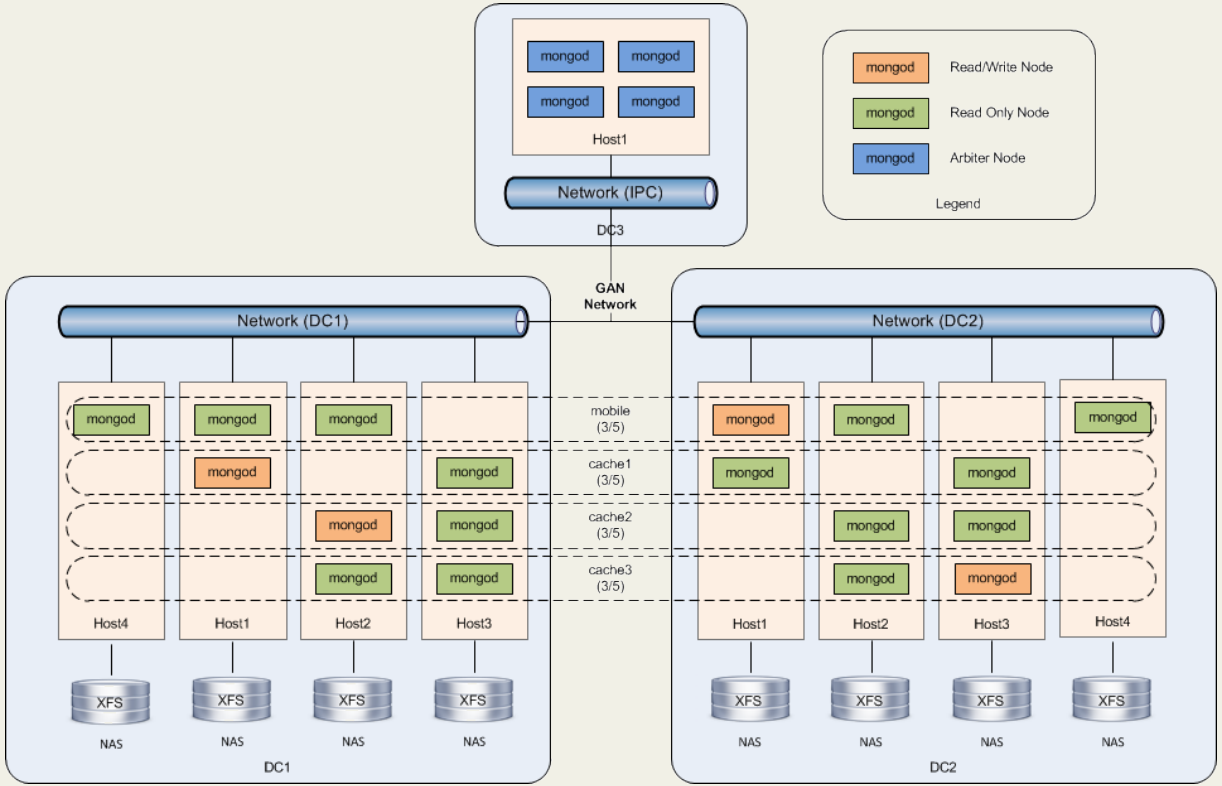


MongoDB

- NoSQL (non-relational) document-oriented (BSON) data store
- Schemaless, no joins or complex transactions
- Replica sets and asynchronous replication for durability
- Local read
- Horizontally scalable architecture
- Automatic partitioning and management
- Data center awareness
- Replication from nearby members
- Protection against split-brain (arbiter)

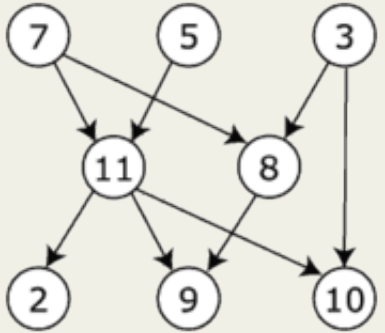


MongoDB Deployment



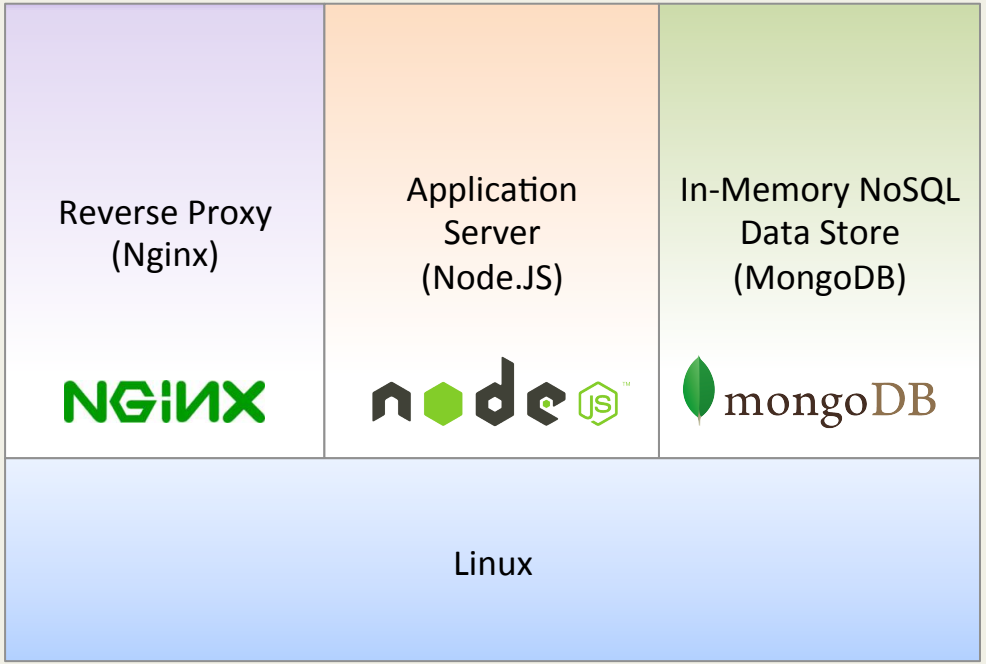
Aggregation and Orchestration

- Bridge the gap between client need and API capabilities
- Create new APIs and representations using existing APIs and representations
- Framework and libraries for orchestration
  - API handler/controller/container
  - PigeonKeeper – component for orchestrating the execution of processes
  - Support for job described by directed acyclic graph of tasks
  - Node.js implementation



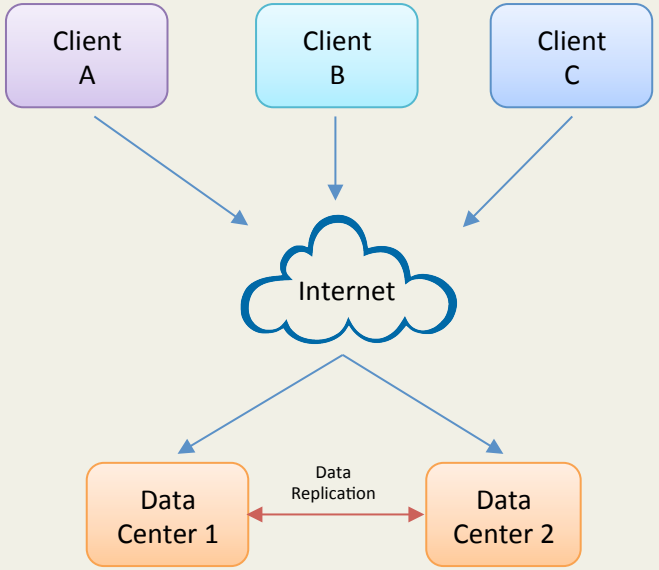
Server Stack

- Scalable stack
- Scalable, high performance web server (Nginx)
- Non-blocking server architecture (Node.js)
- NoSQL data store (MongoDB)
- Lighter and simpler stack



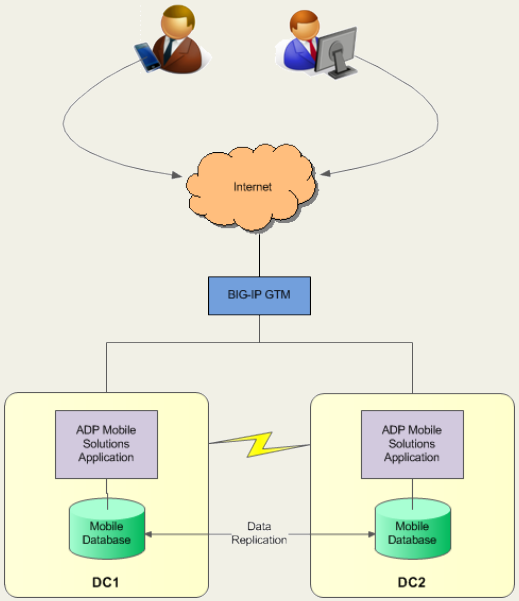
Infrastructure

- True active-active data center deployment
- Service location transparency for end user
- Request distributed based on best performing and available data center
- Complete autonomy at web and application tiers
- Node failures gracefully handled
- Data center failure gracefully handled
- Protection against split-brain condition
- Improved performance and reliability
- Ease of maintenance (rolling upgrades)
- Target 100% uptime

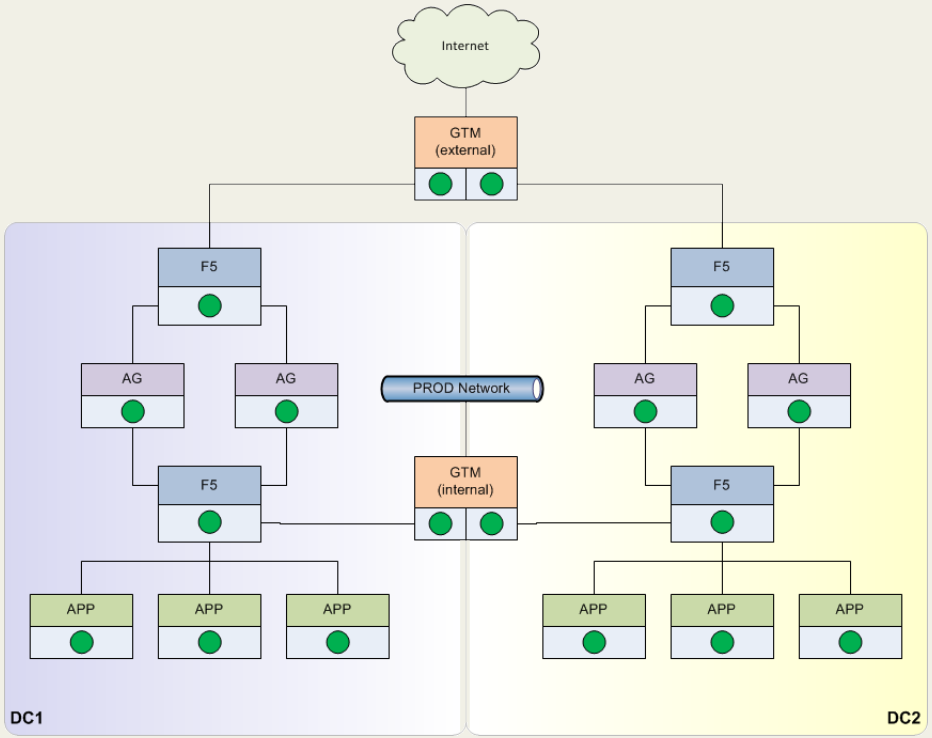


Location Transparency

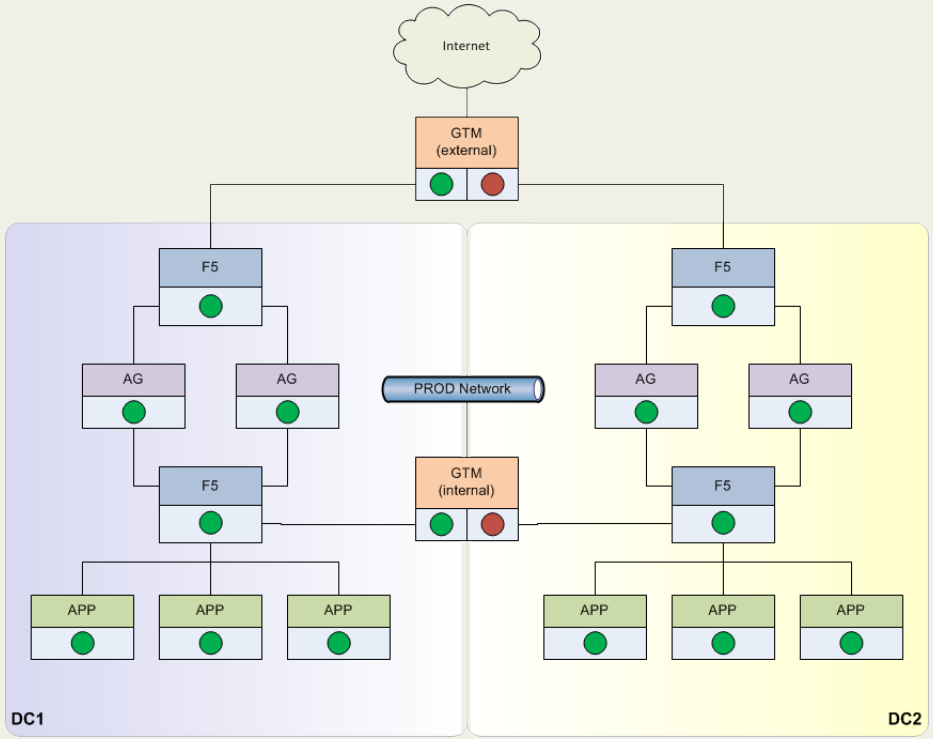
- BIG-IP Global Traffic Manager (GTM)
- Global URL(s)
- Directs user to best performing (QOS) data center
- Automatically routes traffic to available data center when one data center is overloaded or unreachable
- Can be configured to do geo-location-based routing
- Enable traffic management



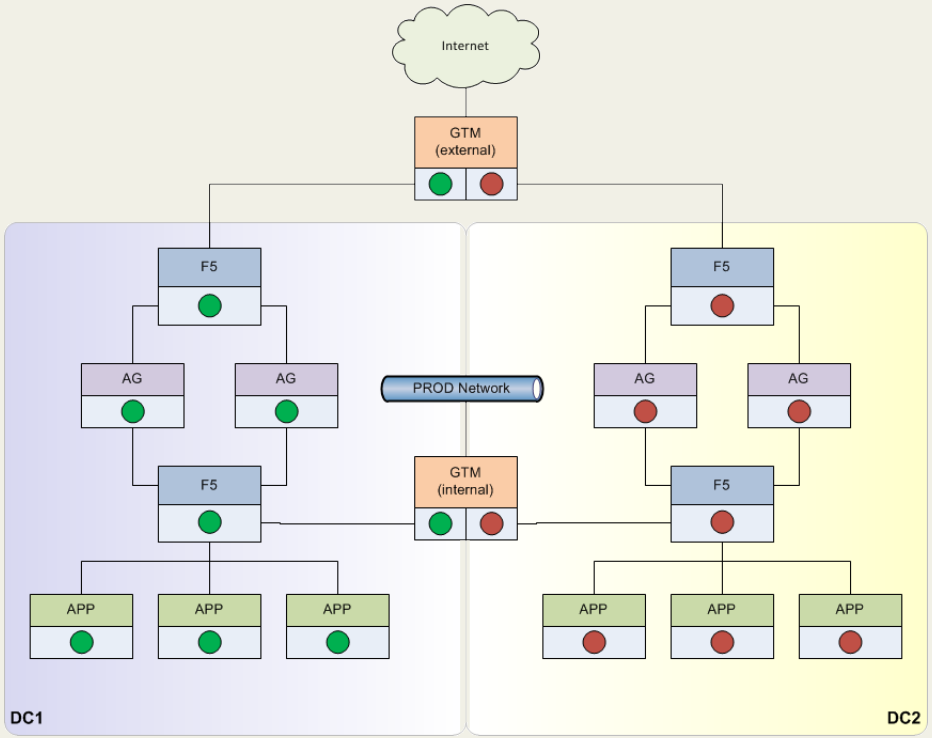
GTM


Internet
GTM
(external)
F5
AG
AG
F5
PROD Network
GTM
(internal)
F5
AG
AG
F5
APP
APP
APP
APP
APP
APP
DC1
DC2

GTM Traffic Management



GTM Monitor


Internet
GTM
(external)
F5
AG
AG
F5
PROD Network
GTM
(internal)
APP
APP
APP
DC1
F5
AG
AG
F5
APP
APP
APP
DC2

Filters

Schema Validation

- Inbound: request schema validation
- Outbound: response schema validation
- Configurable validation policy
  - Enforcement by environment
  - Enforcement by URI
  - Enforcement by API risk level

Data Privacy

- Enforce data privacy requirements
- Leverage context API (for data masking)
- Support for “reveal” action with proper authorization

Payload Transformation

- JSON to XML transformation
- Flattened JSON
- Backward compatibility