TFS DIGITAL

mFin Architecture Playbook 2.31

‘How to’ design ABC Product for ‘mFin’ Architecture

Release Date: 10.28.2020

TOYOTA FINANCIAL SERVICES

Table of Contents

Links to each section provided below

Team

Product Owners
Walter Campbell
Toshihiro Masukawa

Executive Sponsors
Vipin Gupta
Ravi Chirumamilla

Intended Audience	
Product Designers	Release Masters
Developers	Digital Factory Owners

Release Date	Copyright
10.28.2020	Toyota Financial Services

TOYOTA FINANCIAL SERVICES

Print Guidance: Please be friendly to planet Earth

Enterprise Architecture Design

Goal

To help transform TFS into a Multi-brand Digital business and unlock the power of TFS business by enabling faster and better design decisions.

Enterprise Architecture's key role is help TFS navigate the interconnectedness and complexity across strategy, business capabilities, process automation, applications, data, infrastructure, security and transformation initiatives.

Acceptance Criteria

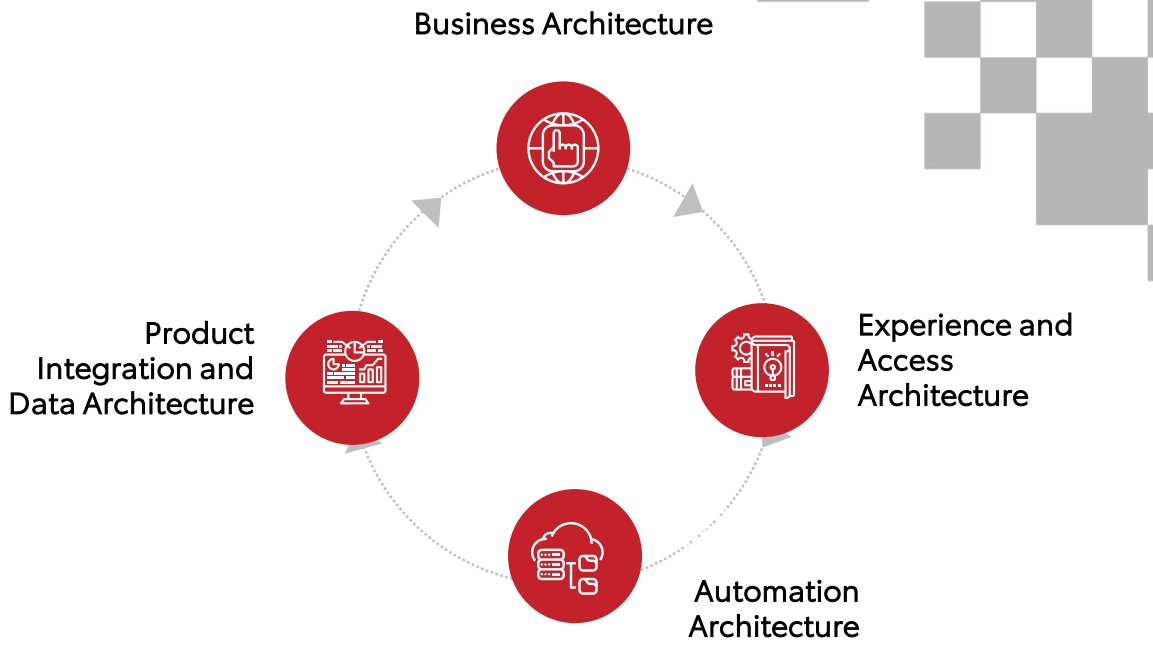
Focus on design direction, design enablement and design assurance.

With following priorities:

- Improve Agility
- Increase Efficiency
- Increase Consistency
- Design Quality, Code Quality, Functional Quality and Run Quality
- Enable Autonomy of Product Designers

How

- Consulting and engaging with Domains/Factories
- Creating best practices/guiding principles
- Continual focus on pragmatic designs that will push forward our incremental modernization
- Be the influencer in mFin MEP and identify options to bridge gaps
- Stay focused through time and cost pressures



Path Forward

2018

- Disconnected experience
- Slow Decision workflows
- Siloed system and data
- Shortage of digital expertise

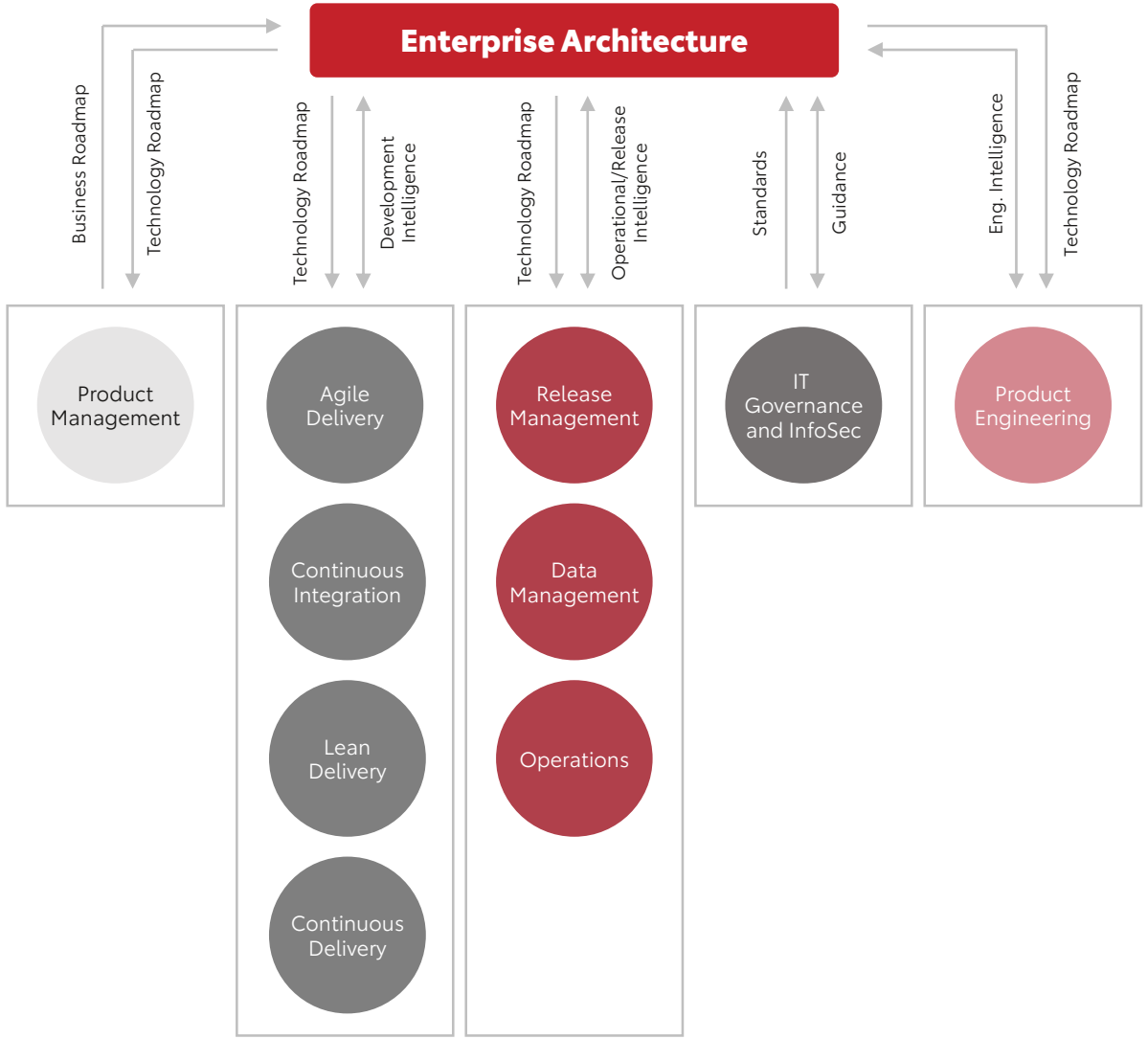
Crossing the "Digital Chasm"

2021+
Multi-Brand Digital Business

- **Simple and Self-Service** Empowering Experience
- **Speed and Agility** of Decisions, Changes and Actions
- **Smart and Automated** Zero-touch paper-less workflows
- **Secure and share** Global platform eco-systems

TOYOTA FINANCIAL SERVICES

Enterprise Architecture and Factory Engagement



The Enterprise Architecture Design provides factories with

- Unified Eco-System with Shared Standards and Common Platforms
- Standards on pragmatic designs that will push forward incremental modernization
- Overview, interconnectedness and impact of digital footprint of factory products/ services across enterprise.
- Standards, guidelines and design requirements for a Technical Product Roadmap
- MEP and Standards for Mfin Architecture
- Guardrails and guidelines on Most Essential Technology Enterprise Requirements (METER)
- Guiding principles and standards on InfoSec and Data Governance
- Enterprise Data architecture and Data Lake
  - Data catalogue requirements for compliance
  - Master Data management
  - Data normalization standards to harmonize data quality and consumption
  - Mechanism and data retrieval and restoration
  - TFS Cloud
- Enterprise Document Management Standards
- Best practices for Release Management and Deployments

TOYOTA FINANCIAL SERVICES

Architecting with Agility

Enterprise Architecture
(Domain Architects)

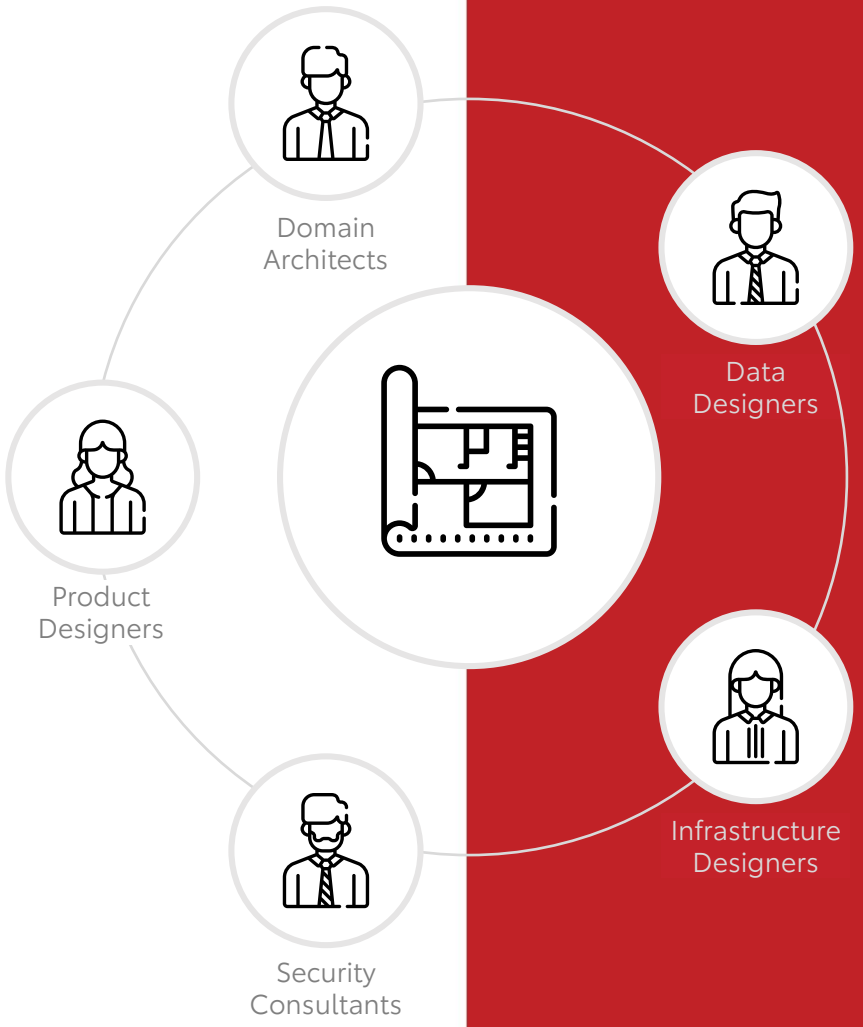
- Domain Technology Direction (Macro Decisions)
- Enterprise Technology Platforms & Roadmaps
- Integration Design Patterns
- Technology Standards

Application Architecture
(Product Designers)

- Product Designs
- Development Patterns & Practices
- Non-Functional Requirements

Security Architecture
(Security Consultants)

- Security Platform Direction
- Security Integration Designs
- Security Patterns & Practices



Data Architecture
(Data Designers)

- Enterprise Data Models
- Master Data Management
- Data Ingestion Patterns

Infrastructure Architecture
(Infrastructure Designers)

- Infrastructure Platform Designs
- New Platform POCs

TOYOTA FINANCIAL SERVICES

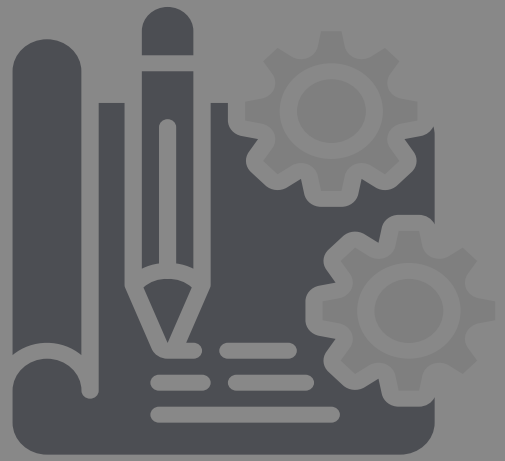
Connected, Autonomous, Shared, Electrification…

Toyota Financial Services

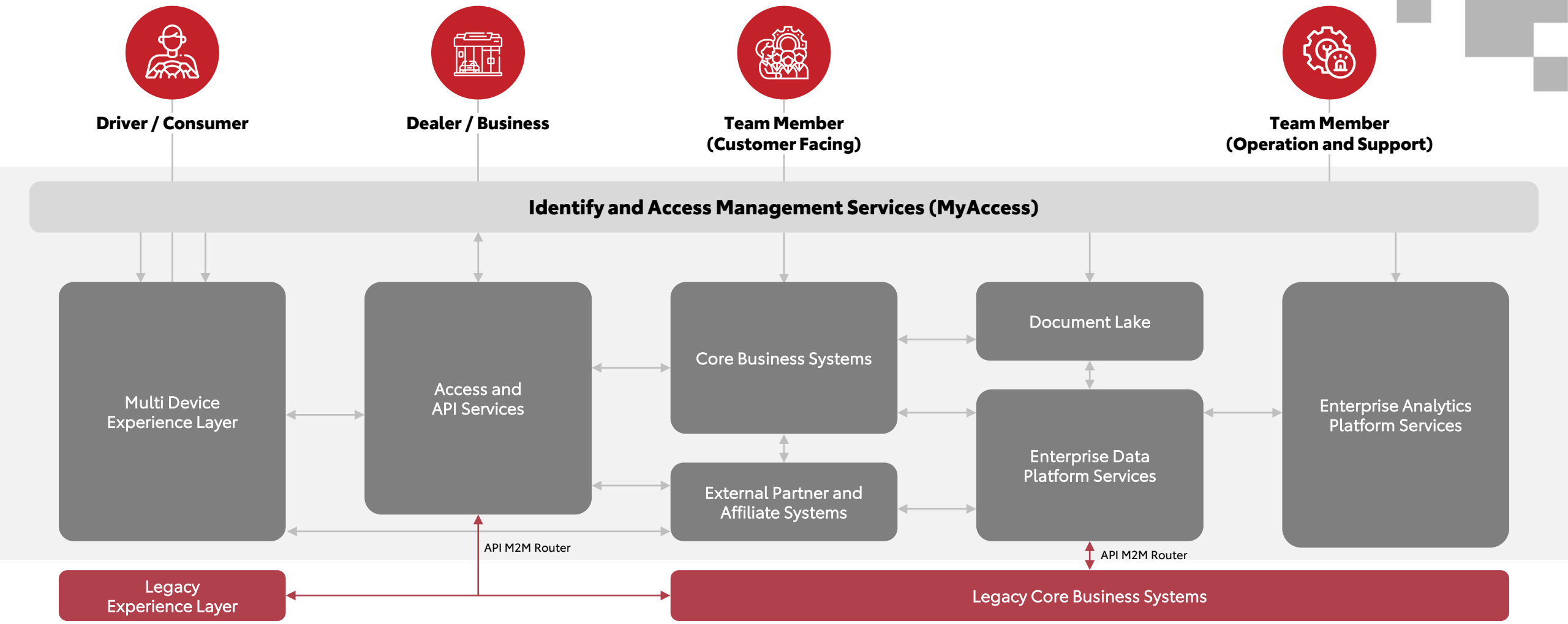
Captive Finance	+	Mobility Finance
Monetizing Ownership	+	Monetizing Usage
Business for One	+	Platform for Many

Lending, Insurance, Payments, Billing, Fleet…

Technology Architecture Direction

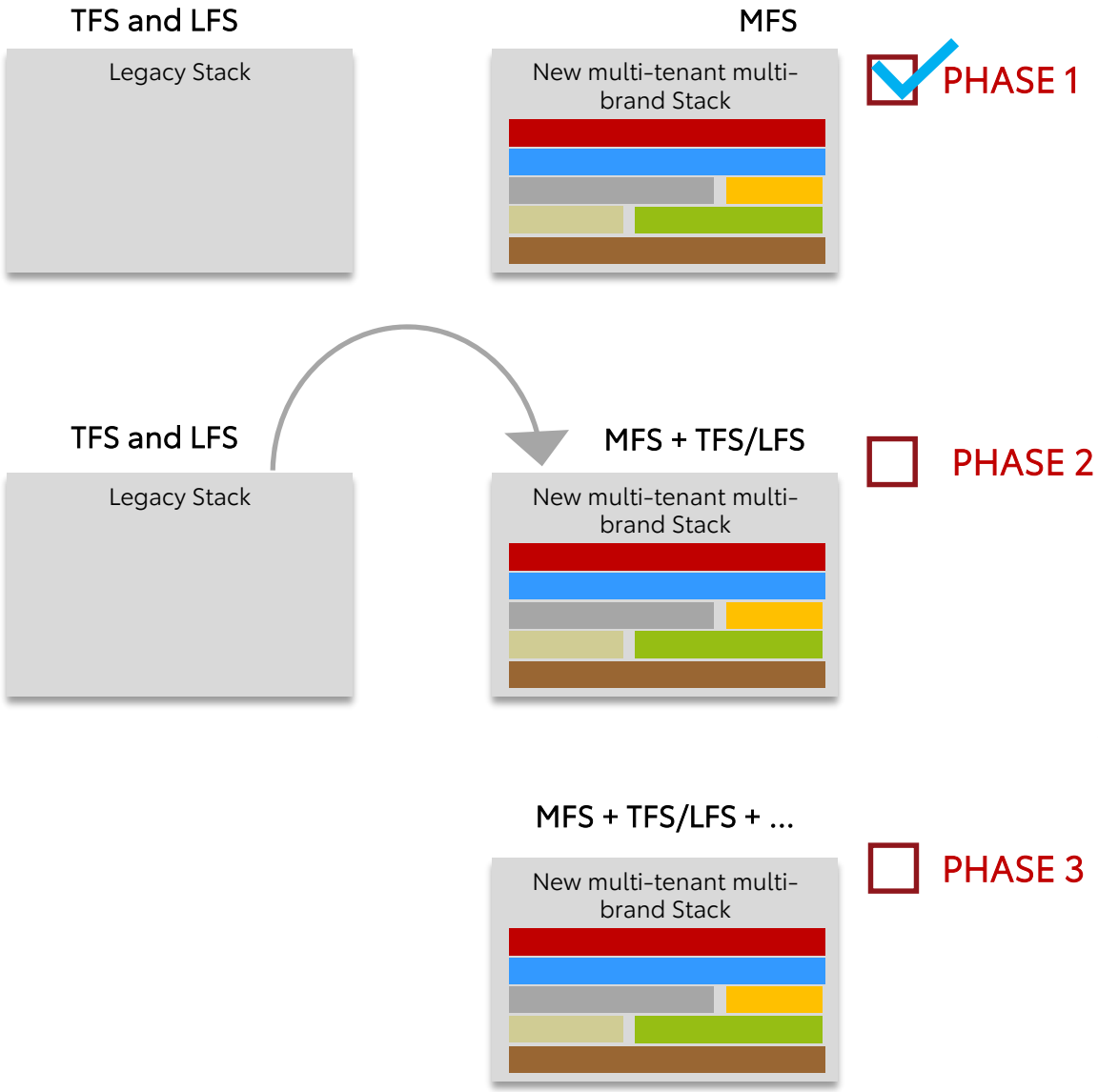


‘mFin’ Integrated Digital Enterprise Architecture



TOYOTA FINANCIAL SERVICES

Modernization Approach



April 2020

Introduce new mFin design based multi-tenant technology stack for the first tenant – MFS.

2020+

Introduce new tenant to new mFin Stack for TFS. Start moving TFS capabilities, systems and data to TFS tenant in mFin Stack

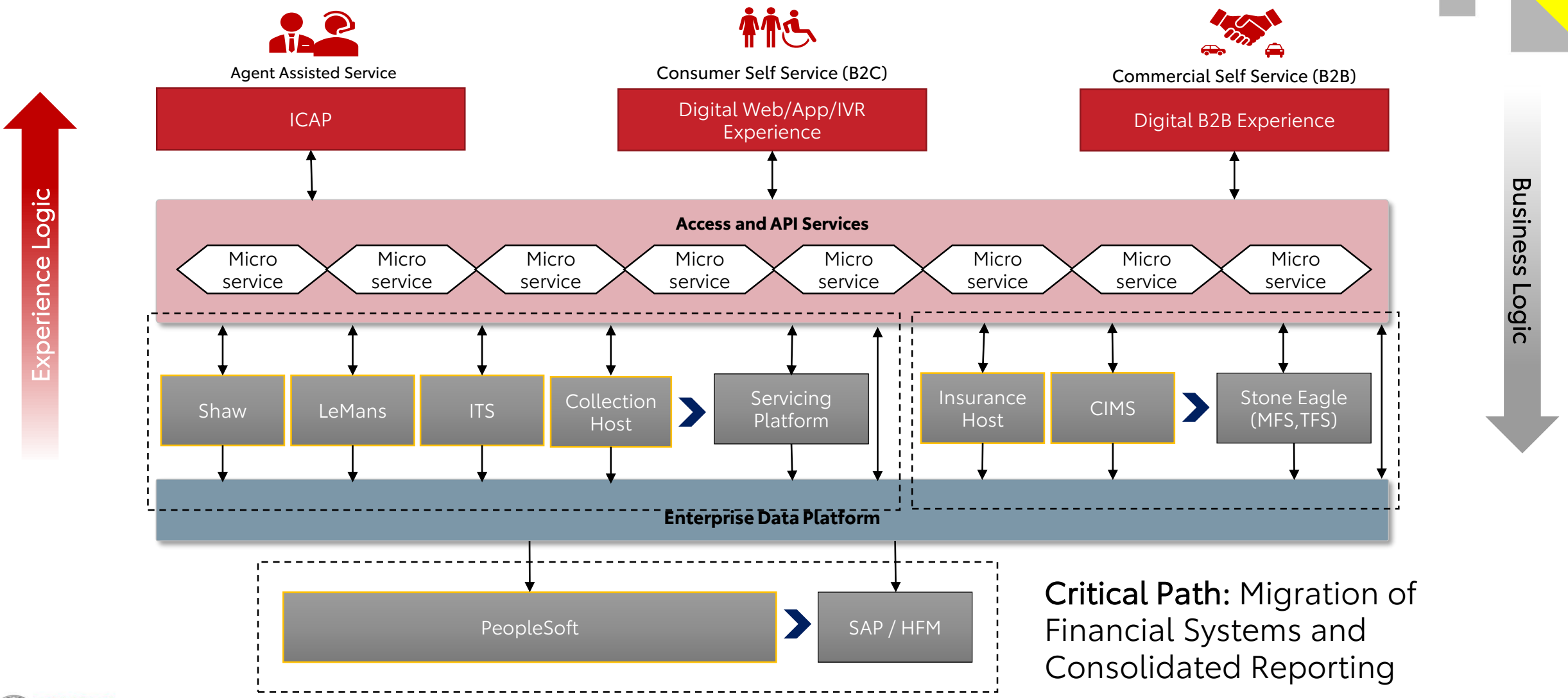
2023+

Complete transition of retained old stack systems to mFin multi-tenant approach. Decommission Legacy systems.

Customer, Dealer & Agent Experience

Experience Layer to provide a Single UI / UX during Transition Period

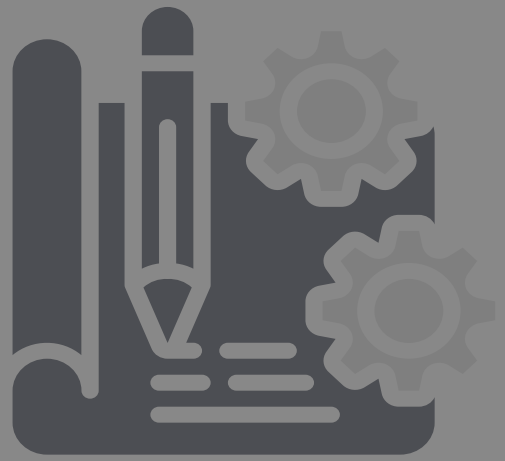
ILLUSTRATIVE



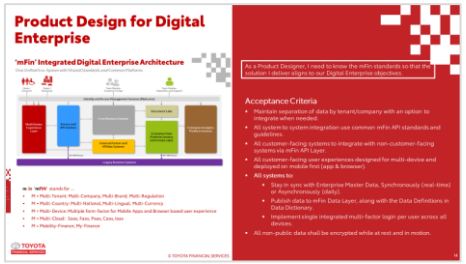
Critical Path: Migration of Financial Systems and Consolidated Reporting

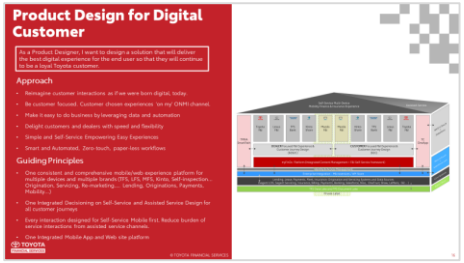
TOYOTA FINANCIAL SERVICES

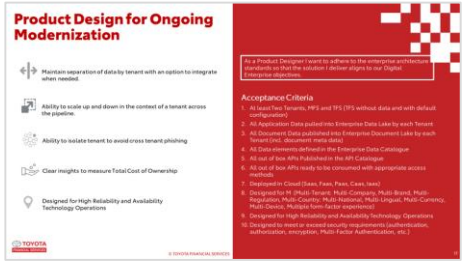
Product Roadmap =
Product Vision & Design

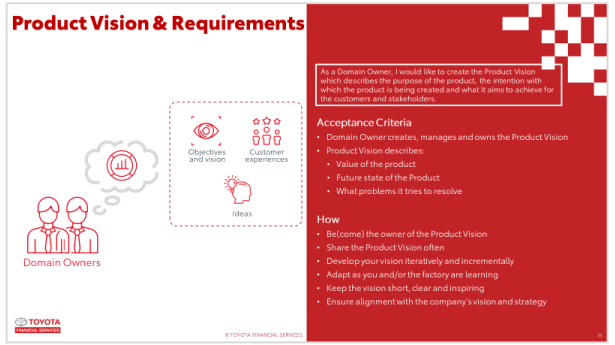


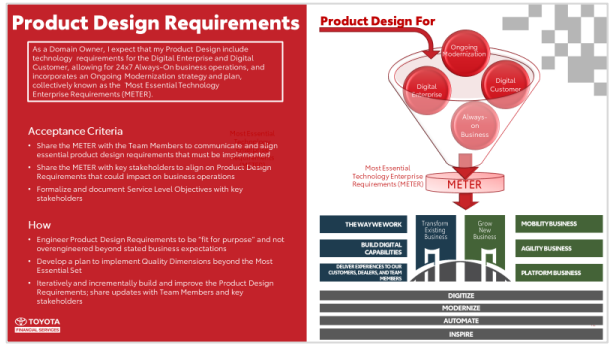
Product Vision & Design for the Product Roadmap

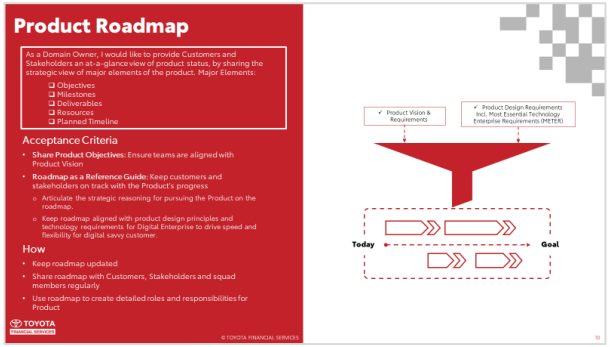












Product Roadmap

As a Domain Owner, I would like to provide Customers and Stakeholders an at-a-glance view of product status, by sharing the strategic view of major elements of the product, such as:

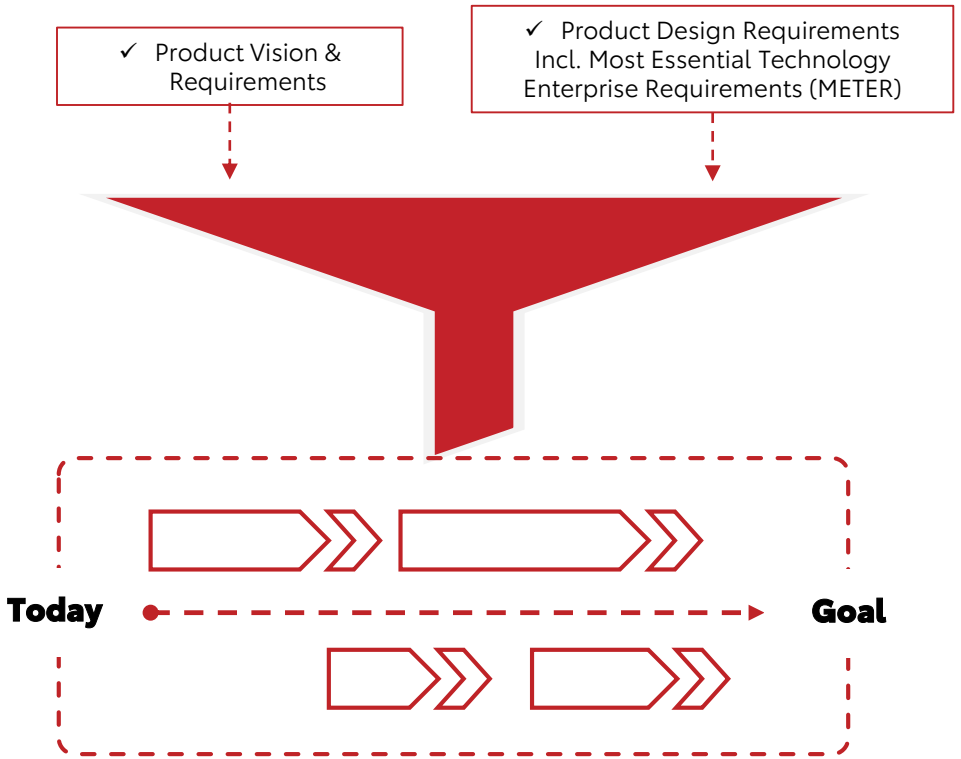
- Objectives
- Milestones
- Deliverables
- Resources
- Planned Timeline

Acceptance Criteria

- **Share Product Objectives:** Ensure teams are aligned with Product Vision
- **Roadmap as a Reference Guide:** Keep customers and stakeholders on track with the Product's progress
  - Articulate the strategic reasoning for pursuing the Product on the roadmap.
  - Keep roadmap aligned with product design principles and technology requirements for Digital Enterprise to drive speed and flexibility for digital savvy customer.

How

- Keep roadmap updated
- Share roadmap with Customers, Stakeholders and squad members regularly
- Use roadmap to create detailed roles and responsibilities for Product



TOYOTA FINANCIAL SERVICES

Product Vision & Requirements

As a Domain Owner, I would like to create the Product Vision which describes the purpose of the product, the intention with which the product is being created and what it aims to achieve for the customers and stakeholders.

Acceptance Criteria

- Domain Owner creates, manages and owns the Product Vision
- Product Vision describes:
  - Value of the product
  - Future state of the Product
  - What problems it tries to resolve

How

- Be(come) the owner of the Product Vision
- Share the Product Vision often
- Develop your vision iteratively and incrementally
- Adapt as you and/or the factory are learning
- Keep the vision short, clear and inspiring
- Ensure alignment with the company's vision and strategy

TOYOTA FINANCIAL SERVICES

Product Design Requirements

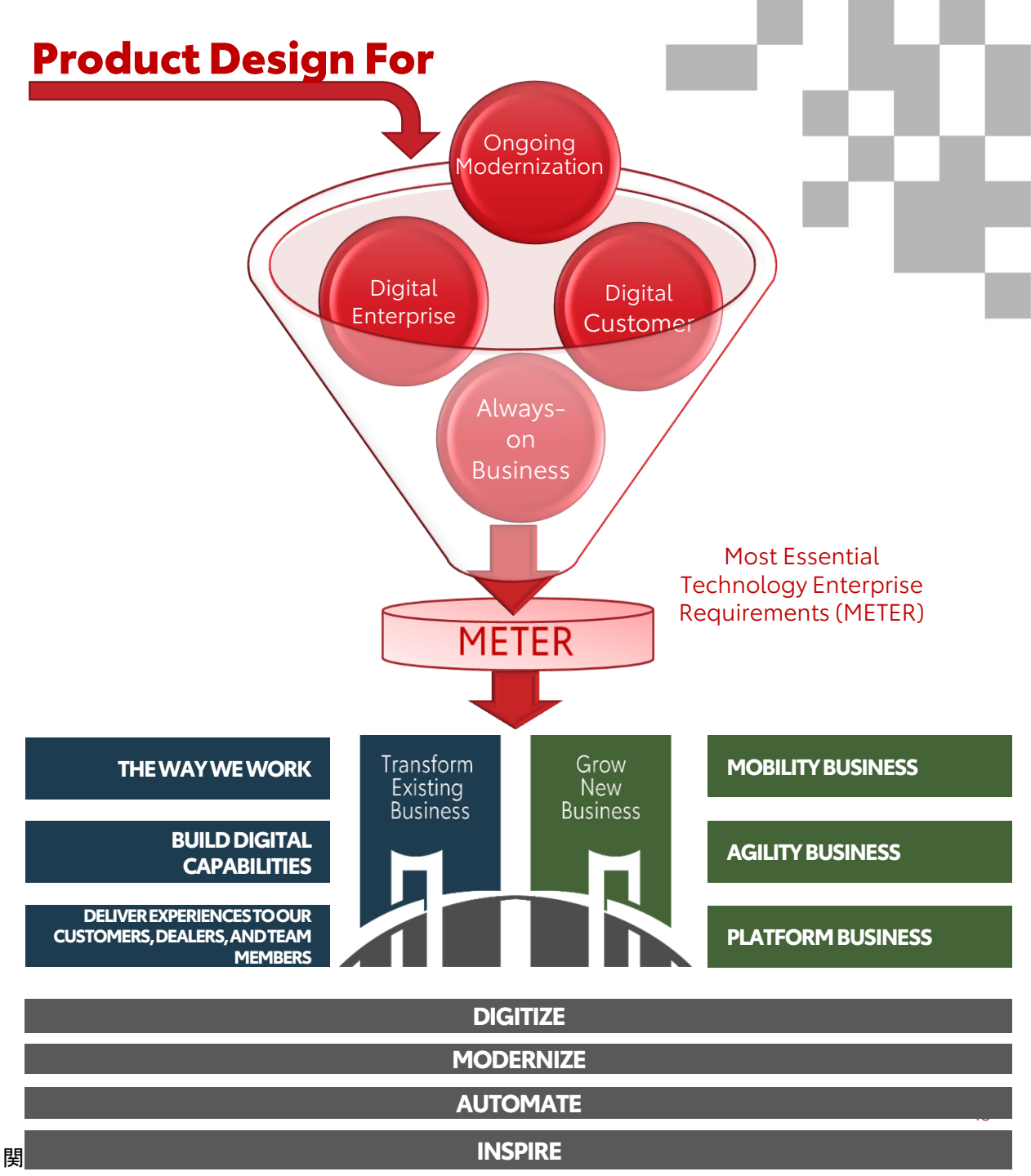
As a Domain Owner, I expect that my Product Design include technology requirements for the Digital Enterprise and Digital Customer, allowing for 24x7 Always-On business operations, and incorporates an Ongoing Modernization strategy and plan, collectively known as the Most Essential Technology Enterprise Requirements (METER).

Acceptance Criteria

- Share the METER with the Team Members to communicate and align essential product design requirements that must be implemented
- Share the METER with key stakeholders to align on Product Design Requirements that could impact on business operations
- Formalize and document Service Level Objectives with key stakeholders

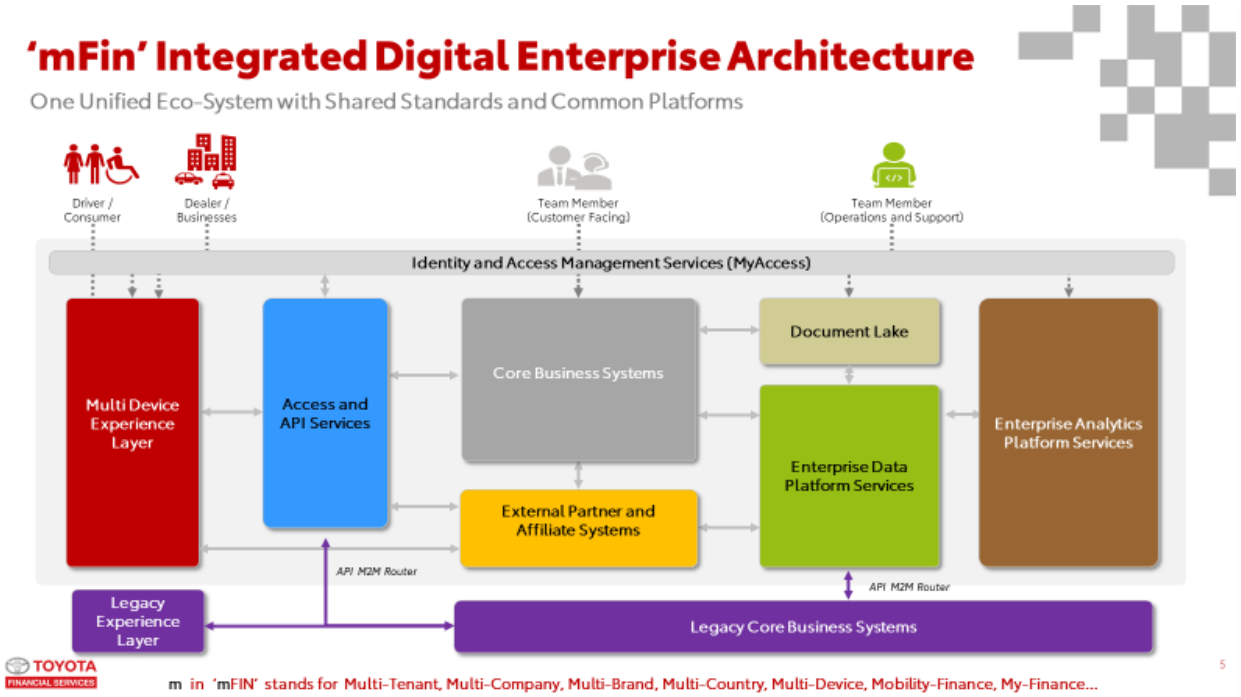
How

- Engineer Product Design Requirements to be “fit for purpose” and not overengineered beyond stated business expectations
- Develop a plan to implement Quality Dimensions beyond the Most Essential Set
- Iteratively and incrementally build and improve the Product Design Requirements; share updates with Team Members and key stakeholders



TOYOTA FINANCIAL SERVICES

Product Design for Digital Enterprise



m in 'mFIN' stands for ...

- M = Multi-Tenant: Multi-Company, Multi-Brand, Multi-Regulation
- M = Multi-Country: Multi-National, Multi-Lingual, Multi-Currency
- M = Multi-Device: Multiple form-factor for Mobile Apps and Browser based user experience
- M = Multi-Cloud: Saas, Faas, Paas, Caas, Iaas
- M = Mobility-Finance, My-Finance

As a Product Designer, I need to know the mFin standards so that the solution I deliver aligns to our Digital Enterprise objectives.

Acceptance Criteria

- All customer-facing user experiences designed for multi-device and deployed on mobile first (app & browser).
- All customer-facing systems to integrate with non-customer-facing systems via mFin API Layer.
- All system to system integration use common mFin API standards and guidelines.
- All out of box APIs Published in the API Catalogue; and ready to be consumed with appropriate access methods.
- All Application Data pulled into Enterprise Data Platform by each Tenant
  - All Data elements defined in the Enterprise Data Catalogue
  - Maintain separation of data by tenant/company with an option to integrate when needed.
- Systems to stay in sync with Enterprise Master Data, Synchronously (real-time) or Asynchronously (daily).
- Designed to meet or exceed security requirements (authentication, authorization, encryption, Multi-Factor Authentication, etc.)
  - All non-public data shall be encrypted while at rest and in motion.
  - single integrated multi-factor login per user across all devices.
- Deployed in Cloud (Saas, Faas, Paas, Caas, Iaas); Designed for High Reliability and Availability Technology Operations

TOYOTA FINANCIAL SERVICES

Product Design for Digital Customer

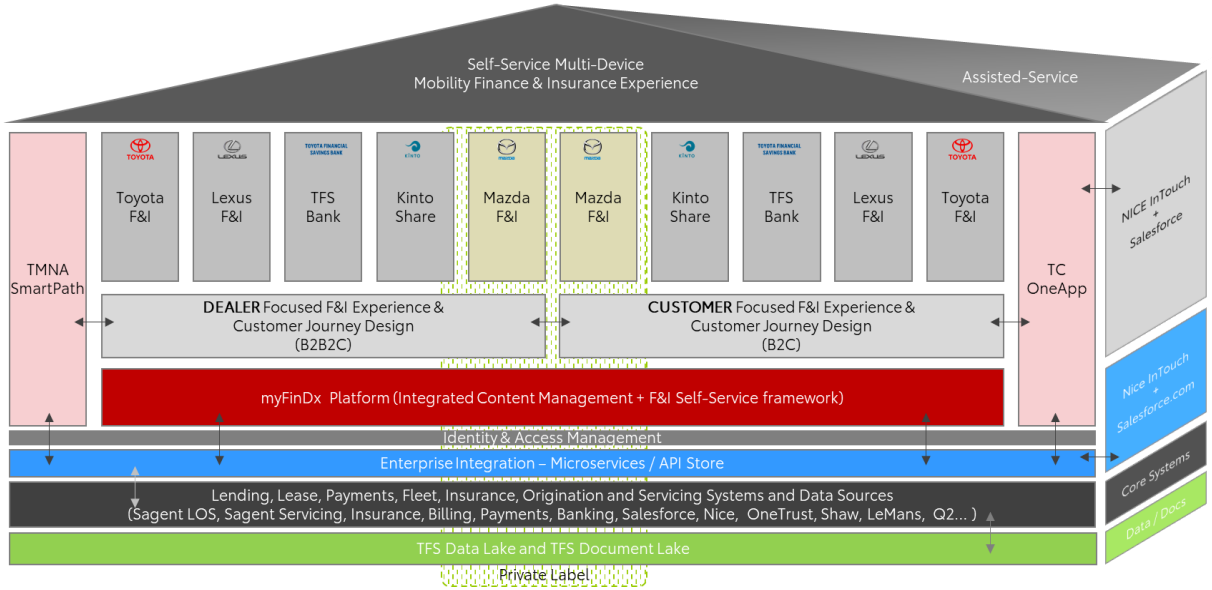
As a Product Designer, I want to design a solution that will deliver the best digital experience for the end user so that they will continue to be a loyal Toyota customer.

Approach

- Reimagine customer interactions as if we were born digital, today.
- Be customer focused. Customer chosen experiences 'on my' ONMI channel.
- Make it easy to do business by leveraging data and automation
- Delight customers and dealers with speed and flexibility
- Simple and Self-Service Empowering Easy Experiences
- Smart and Automated, Zero-touch, paper-less workflows

Guiding Principles

- One consistent and comprehensive mobile/web experience platform for multiple devices and multiple brands.
  - **Example:** (TFS, LFS, MFS, Kinto, Self-inspection, Origination, Servicing, Re-marketing, Lending, Originations, Payments, and Mobility)
- One Integrated Decisioning on Self-Service and Assisted Service Design for all customer journeys
- Every interaction designed for Self-Service Mobile first. Reduce burden of service interactions from assisted service channels.
- One Integrated Mobile App and Web site platform



TOYOTA FINANCIAL SERVICES

Product Design for Ongoing Modernization

- Maintain separation of data by tenant with an option to integrate when needed.
- Ability to scale up and down in the context of a tenant across the pipeline.
- Ability to isolate tenant to avoid cross tenant phishing
- Clear insights to measure Total Cost of Ownership
- Designed for High Reliability and Availability Technology Operations

As a Product Designer I want to adhere to the enterprise architecture standards so that the solution I deliver aligns to our Digital Enterprise objectives.

Acceptance Criteria

1. At least Two Tenants, MFS and TFS (TFS without data and with default configuration)
2. All Application Data pulled into Enterprise Data Lake by each Tenant
3. All Document Data published into Enterprise Document Lake by each Tenant (incl. document meta data)
4. All Data elements defined in the Enterprise Data Catalogue
5. All out of box APIs Published in the API Catalogue
6. All out of box APIs ready to be consumed with appropriate access methods
7. Deployed in Cloud (Saas, Faas, Paas, Caas, Iaas)
8. Designed for M (Multi-Tenant: Multi-Company, Multi-Brand, Multi-Regulation, Multi-Country: Multi-National, Multi-Lingual, Multi-Currency, Multi-Device, Multiple form-factor experience)
9. Designed for High Reliability and Availability Technology Operations
10. Designed to meet or exceed security requirements (authentication, authorization, encryption, Multi-Factor Authentication, etc.)

TOYOTA FINANCIAL SERVICES

Product Design for 'Always On' Business

As a Domain Owner, I want to design my product for the 24x7 'Always On' Digital Business, so that I can meet and exceed the needs and expectations of the digital savvy customer.

Acceptance Criteria

- **Functional Fitnesss** - Provide relevant and adequate functions that meet stated and implied needs of customers
- **Technical Performance** - Enable the specified level of performance, reliability and security that aligns with customer on-demand and privacy expectations
- **Functional Experience –** Deliver software product can be used by customers to experience specific value & to achieve specific goals with effectiveness, efficiency, & satisfaction
- **Functional Flexibility –** Deliver product that is flexible, can easily be modified and reused in response to customer needs to remain agile & responsive
- **Maintainability –** Deliver product that can be supported and sustained, to remain efficient and cost effective

How

- Implement instrumentation to proactively monitor and manage key application performance and security metrics
- Implement auto scaling and business continuity to meet unplanned events
- Implement and enable application and integration event logging that provide useful and actionable information for root cause analysis and incident remediation
- Stream system and application event logs into the Enterprise Data Platform to perform predictive and prescriptive analytics and AI

TOYOTA FINANCIAL SERVICES

Information Security Requirements

Access Control Security
(Integrate with TFS IAM tools, Least privilege access, MFA, etc.)

Application Security
(Secure coding, Encryption, API Security, etc.)

Data Security
(AES 256 Encryption, Key Management, Access control, etc.)

Network Security
(Private connections, TLS 1.2, etc.)

Security Monitoring
(Code Scan, Vulnerability Scan, SIEM integration, etc.)

3rd Party
(Contracts, Vendors meet TFS Security requirements, etc.)

How do I ensure that all applicable Information Security requirements are incorporated into routine Factory releases?

Acceptance Criteria

- Security requirements are to be implemented for all solutions (on premises and cloud implementations).
  - Refer to the **TFS Security guidebook** for tips on secure coding practices.
  - Consider security controls as part of your DevOps practice (aka DevSecOps). Additions, changes, or modifications to applications or data may require that **additional information security controls** be implemented, as required by regulatory entities.
  - Run a security scan and remediate vulnerabilities for every release before deployment.

How

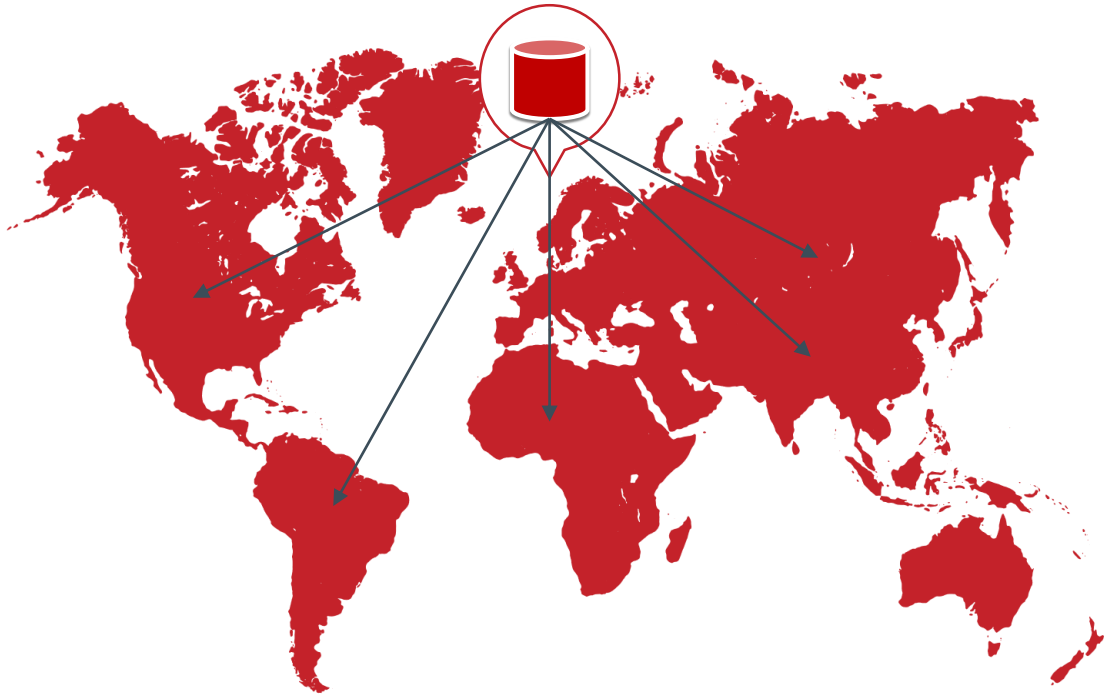
- Complete the **Information Security Questionnaire** for every Release at the end of the grooming sessions, once the scope for a release is finalized.
  - Work with your Domain Security champions(DSC) to create security stories and consult on secure design for your releases/sprints to implement the required security controls.
  - Please refer to the latest published **InfoSec Policies**

TOYOTA FINANCIAL SERVICES

How to Design for Multi-Tenancy

mFin Multi-Tenant Framework

Pattern 1
Centralized



- Developed and Operated by a global team.
- Deployed in **1 location.**
- Used/consumed by each region and county.

Use Case

- Globally uniqueness required
- Limited real-time requirements

Pattern 2
Federated

- Developed by a global team.
- Deployed in **multiple locations.**
- Operated by a global/local team.
- Used/consumed by each region and county.

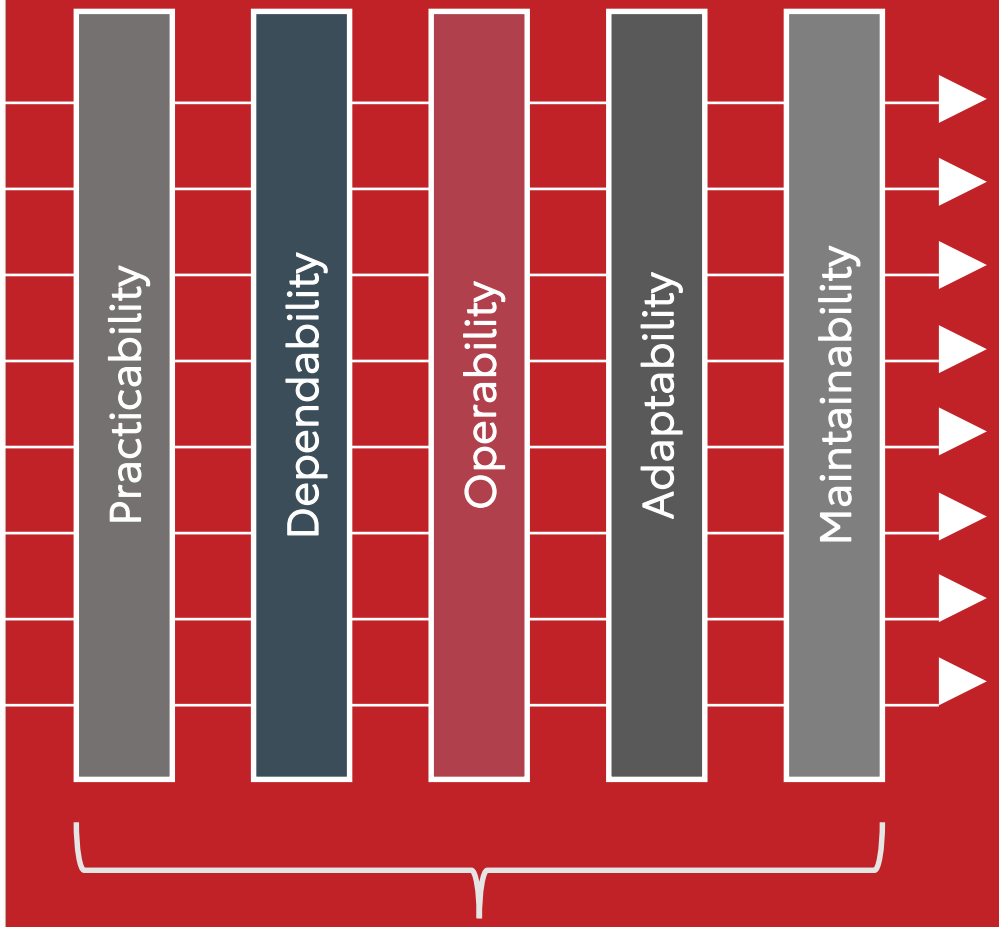
Use Case

- Real-time customer experience
- Data sovereignty, regulatory compliance

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework METER Alignment

- Tenant Isolation
- Data Partitioning
- Tenant Configuration
- Tenant Usage Analysis
- Tenant-Level Metering
- Tenant Monitoring
- Tenant Service Level Agreement
- Tenant Provisioning



TOYOTA FINANCIAL SERVICES

mFin – One Platform for Many Brands, Tenants

1

Pattern 1
Separated APP & DB instance model

- **Performance Tuning Effort:** Low
- **Operating cost:** High
- **Initial development cost:** Low

This pattern works for a limited number of tenants, but it does not scale well beyond 3-5 tenants. The overhead of managing multiple independent instances with different configurations and divergent tenant expectations becomes a constraint to expanding the tenant base.

2

Pattern 2
Single APP Instance, Individual DB Schema

- **Performance Tuning Effort:** Medium
- **Operating cost:** Medium
- **Initial development cost:** Medium

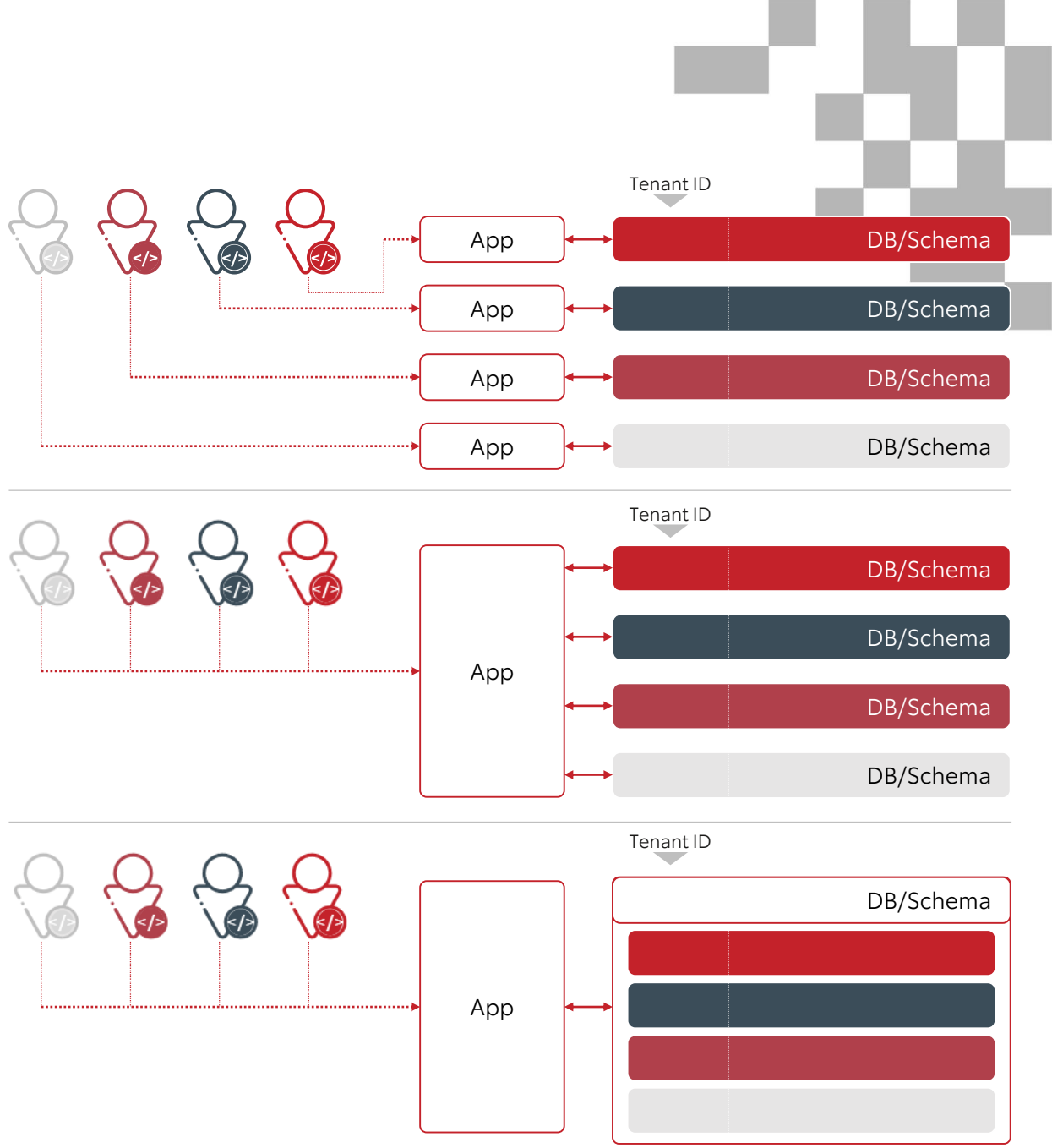
This pattern is common in Software as a Service (SaaS) offerings, given that tenant data and rules can easily be separated. Reuse of common data structures and business rules becomes a challenge to management and growth over the longer term, and this type of solution can easily support 10+ tenants if properly managed.

3

Pattern 3
Single APP Instance, Common DB Schema

- **Performance Tuning Effort:** High
- **Operating cost:** Low
- **Initial development cost:** High

This pattern is what most SaaS providers aspire to offer. The solution is complex to design and implement and offers automated onboarding of new tenants. Example: Amazon Marketplace.



TOYOTA FINANCIAL SERVICES

mFin Tenant ID Guideline
Tenant Isolation

- Tenant is defined as "a Legal Entity that consents to use our infrastructure but requires to maintain separation of data, documents and access due to contractual, legal or regulatory definition".*
- A Tenant ID is a 4-digit alpha numeric code that uniquely identifies a Tenant
- A Tenant ID is structured using a pattern; 'tnnn' where 't' is a default alphabet & 'n' is any numeric number
- Tenant ID **'t001'** represents **'Toyota/Lexus USA', 't002'** represents **'Mazda USA'**

*Tenant could be a company, financial institution, start up, software company, SFC, Dealer... any separate legal entity.

'Tenant ID' MUST be part of:

- Every Data Domain Entity (e.g. Contract, Account, etc. must all include Tenant ID as part of their Key/Identifier) for Uniqueness & Integrity purposes
- Every API Signature for Access Control, Routing, Orchestration, Auditing & Monitoring purposes
- Every Secure Identity, Entitlement, Crypto Key, Crypto Token, etc. for Security & Privacy purposes
- Every Tenant-Specific Customized Code, Module & Configuration for Functional Segregation & Correctness purposes
- Every Tenant-Specific Technology Platform & Infrastructure Component (e.g. Tenant-Specific Kubernetes Pods, etc.) for Physical & Logical Isolation purposes
- Every Runtime User Session, Log Entry, Monitoring Probe, etc. for Service Management purposes

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Tenant Isolation

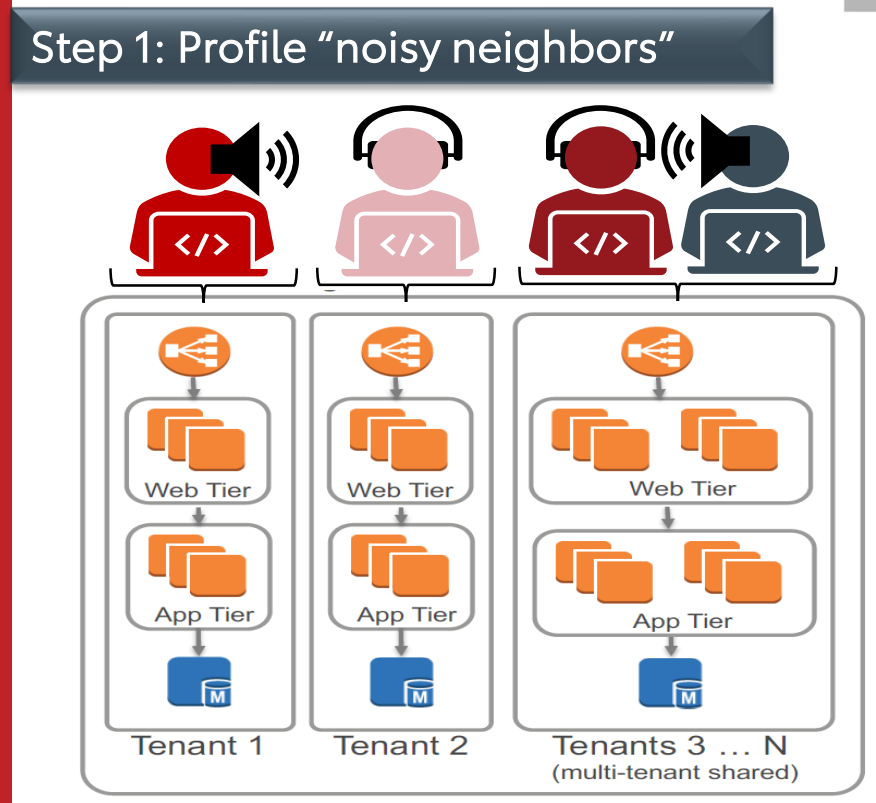
How do I meet tenant specific SLA's in a multi-tenant environment?

Considerations

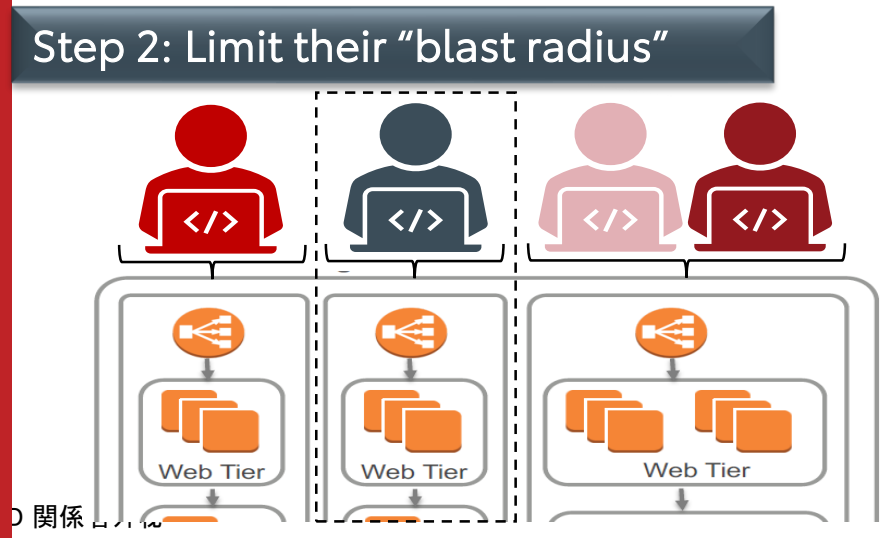
- "Noisy neighbors" are tenants who consume excessive compute and storage resources in a shared / pooled environment
- The "blast radius" of failures / processing delays caused by "noisy neighbors" could negatively impact on all tenant SLAs
- Basic isolation constructs include security roles, runtime policies, throttling, de-prioritization, and segregation of critical services
- Non-critical services can be shared / pooled between tenants, but critical services should have segregation options by design
- Carefully consider the data migration strategy beyond initial implementation in case a "noisy neighbor" has to be migrated

Solution

- Build tenant personas to understand system utilization patterns
- Collocate tenants with similar needs in a shared / pooled environment
- Build multi-tenant solution from the start to avoid rewriting the application for tenant #2
- Invest in tenant specific fault tolerance, isolation and recovery
- Consider bulkheads / circuit breakers to contain the "blast radius" of a "noisy neighbor"
- Automatically scale the stack up and down; build this into the architecture, don't implement manual scaling
- Server-less functions (FaaS) is a popular option in SaaS solutions to provide tenant isolation with tenant specific context injected at runtime



Managing "noisy neighbors":
- Start with throttling,
- then deprioritize services,
- next segregate critical services, and finally
- stand up a separate instance.



mFin Multi-Tenant Framework Data Partitioning

DOs

Tenant master table

Tenant ID	Tenant Name
1	Asahi
2	Sapporo
3	Kirin

Attribute master table

Tenant ID	Attribute ID	Attribute Name
1	1	Email
1	2	Phone
2	1	Birthday
2	2	Area
3	1	Phone
3	2	State

Customer common information table

Tenant ID	Customer ID	Customer Name
1	1	Walter Campbell
1	2	Srikanth Boyapali
2	1	David Eddy
2	2	Reza Ghamsari
3	1	Kento Izumi
3	2	Toshi Masukawa

Customer specific information table

Tenant ID	Customer ID	Attribute ID	Attribute Value
1	1	1	walter.campbell@toyota.com
1	1	2	469-782-4110
1	2	1	srikanth.boyapali@toyota.com
1	2	2	310-429-1820
2	1	1	5-Aug
2	1	2	Texas
2	2	1	9-May
2	2	2	California
3	1	1	310-426-4151
3	1	2	Headquarter drive 1-5, Plano
3	2	1	310-842-8569
3	2	2	Main street 11562, Frisco

Asahi

Sapporo

Kirin

DON'Ts

Asahi

Sapporo

Kirin

Customer ID	Customer Name	Email	Phone
1	Walter Campbell	walter.campbell@toyota.com	469-782-4110
2	Srikanth Boyapall	srikanth.boyapalli@toyota.com	310-429-1820

Customer ID	Customer Name	Birthday	State
1	David Eddy	5-Aug	Texas
2	Reza Ghamsari	9-May	California

Customer ID	Customer Name	Phone	Address
1	Kento Izumi	310-426-4151	Headquarter drive 1-5, Plano
2	Toshi Masukaw	310-842-8569	Main street 11562, Frisco

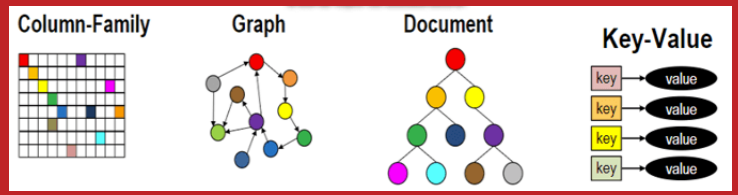
How do I code tenant specific data requirements in a database shared between multiple tenants?

Considerations

- Tenant specific data extensions in a relational database model (below left) results in a sparsely populated data structure.
- A flexible model that does not differentiate between tenant specific and common tenant data attributes is needed.

Solution

- The solution (top left) illustrates an alternative approach that provides the desired level of flexibility using relational database technology.
- Please note that the solution is not intended to describe a preferred or optimal solution for the stated requirement, but rather a different way to think about solutioning.
- Relational database technology is not the only option for storing and relating data, document databases, graph databases, etc. are commonly used alternatives that may offer better alternatives to storing tenant specific and common data.



mFin Multi-Tenant Framework
Tenant Configuration

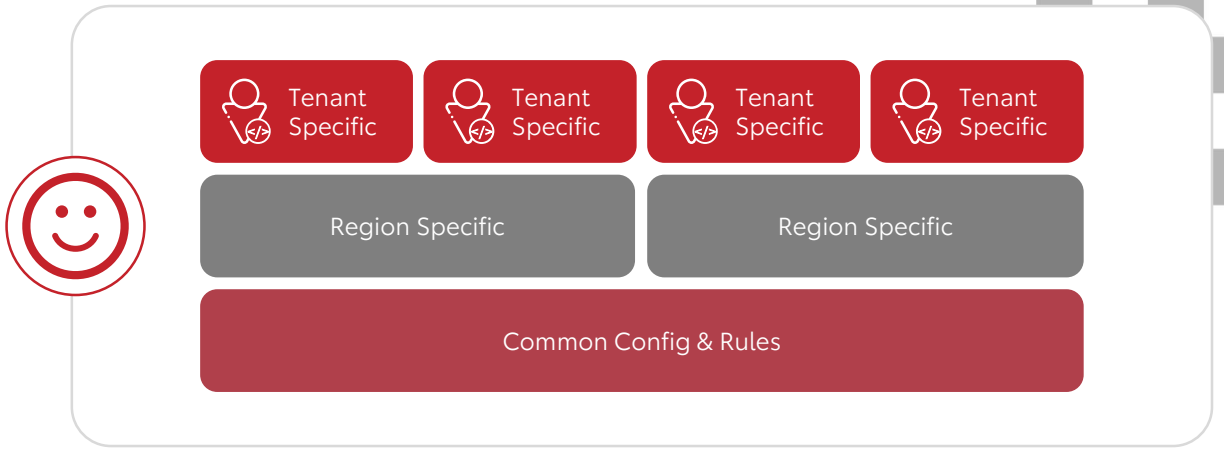
How do I code tenant specific overrides in a multi tenant environment?

Acceptance Criteria

- Minimize duplication and overrides of config /rules to maintain the smallest config/rule set as possible
- Config/rules that tenants typically differentiate on should be implemented as tenant overrides, while industry common config/rules should be in the common layer
- Testing is more complex given that a composite set of rules and overrides execute at runtime for a particular tenant
- Ideally developers should be oblivious to the fact that they're implementing config and rules in a multi tenant environment.
- The platform should take care of the multi tenant "plumbing", allowing developers to focus on business specific requirements.

How

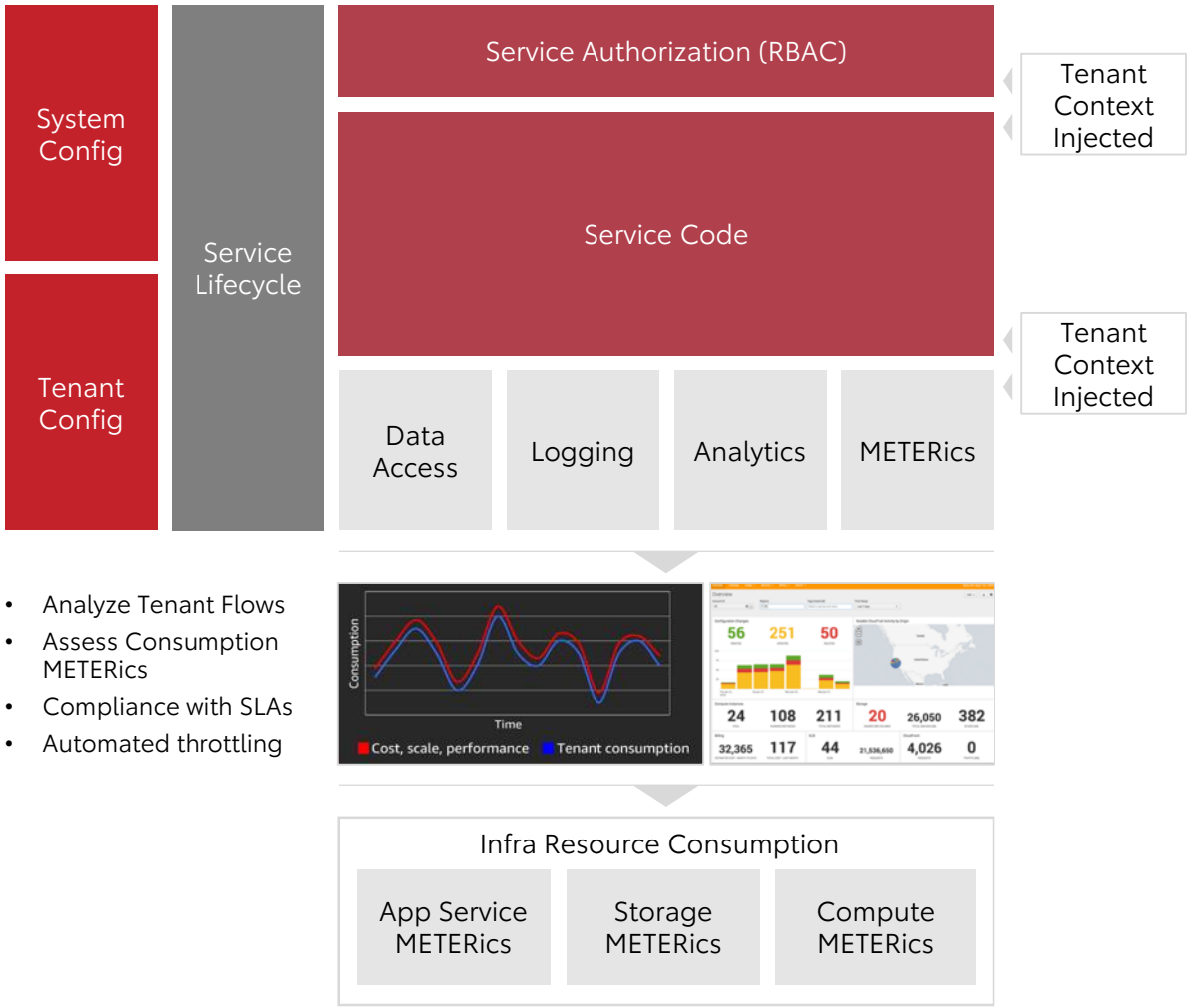
- Collocate tenants with similar needs in a shared/pooled environment to take advantage of the same rule base
- Structure config and rules to allow tenant specific overrides
- Config and rules specific to a region can be reused between tenants, with tenant specific overrides as needed
- Config and rules that are not region or tenant specific should be implemented in a common layer




```
public static void main(String[] args)
{                       int[] list ={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
                        int sum = sumListEnhanced(list);
                        System.out.println("Sum of elements in list: " + sum);
                        Switch(tenant)
                        {
                        case "Tenant 1" :
                                        System.out.println("Original List");
                                        printList(list);
                                        System.out.println("Calling addOne");
                                        addOne(list);
                                        System.out.println("List after call to addOne");
                                        printList(list);
                                        System.out.println("Calling addOneError");
                                        addOneError(list);
                                        System.out.println("List after call to addOneError. Note elements of list did not change.");
                                        printList(list);

                                        Common code repeated for each Tenant!

                        case "Tenant 2" :
                                        System.out.println("Original List");
                                        printList(list);
                                        System.out.println("Calling addTwo");
                                        addOne(list);
                                        System.out.println("List after call to addTwo");
                                        printList(list);
                                        System.out.println("Calling addTwoError");
                                        addOneError(list);
                                        System.out.println("List after call to addTwoError. Note elements of list did not change.");
                                        printList(list);

                        }
```

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Instrumentation and SLAs



- Analyze Tenant Flows
- Assess Consumption METERics
- Compliance with SLAs
- Automated throttling

How do I differentiate service levels between tenants subscribed to different tiers?

Acceptance Criteria

- Metering and monitoring of tenant specific instances are required to ensure SLAs are met, and to determine tenant specific profitability.
- Use concurrency (pooled, siloed) and experience throttling to differentiate tenant tiers.

How

- Design and enable tenant specific metering and monitoring for shared environments.
- Enable tenant throttling to limit the impact of “noisy neighbors”
- Use metering to discover operations by “noisy neighbors” that could be solved with less resource utilization overhead (for example: sync back service for data extracts)
- Automatically scale the stack up and down; build this into the architecture, don’t implement manual scaling

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Provisioning

How should I approach tenant provisioning? Is there value in automating provisioning given the limited number of potential tenants?

Considerations

- Multi-Tenancy Options 1 and 2 are prime candidates for auto provisioning.
- If Federated - Option 3 may also be a candidate if your global system implementation pattern
- Provisioning has different levels of automation, please refer to summary of levels on this slide.
- Note that **compliance can impact tenant onboarding** and automated provisioning, specifically data sovereignty.

Solution

- Automation of provisioning at the highest-level appropriate builds future tenant auto registration and onboarding capabilities.
- Provisioning automation also allows for optimal use of cloud computing resources for ongoing testing and ad-hoc POC's, with rapid provisioning and deprovisioning capabilities.
- Automation reduces errors and frees up expensive labor.

Level 1: Factory Provisioning

- Factory is required to validate every step of he provisioning process to enable a new tenant
- Provisioning scripts require manual intervention and customization per tenant

Level 2: Enterprise Team Provisioning

- Enterprise team is able to run provisioning scripts for new tenants with minimal manual intervention
- Enterprise team is able to validate provisioned services with test scripts
- Security policies are static and tenant specific
- Factory is engaged when provisioning or test scripts fail execution

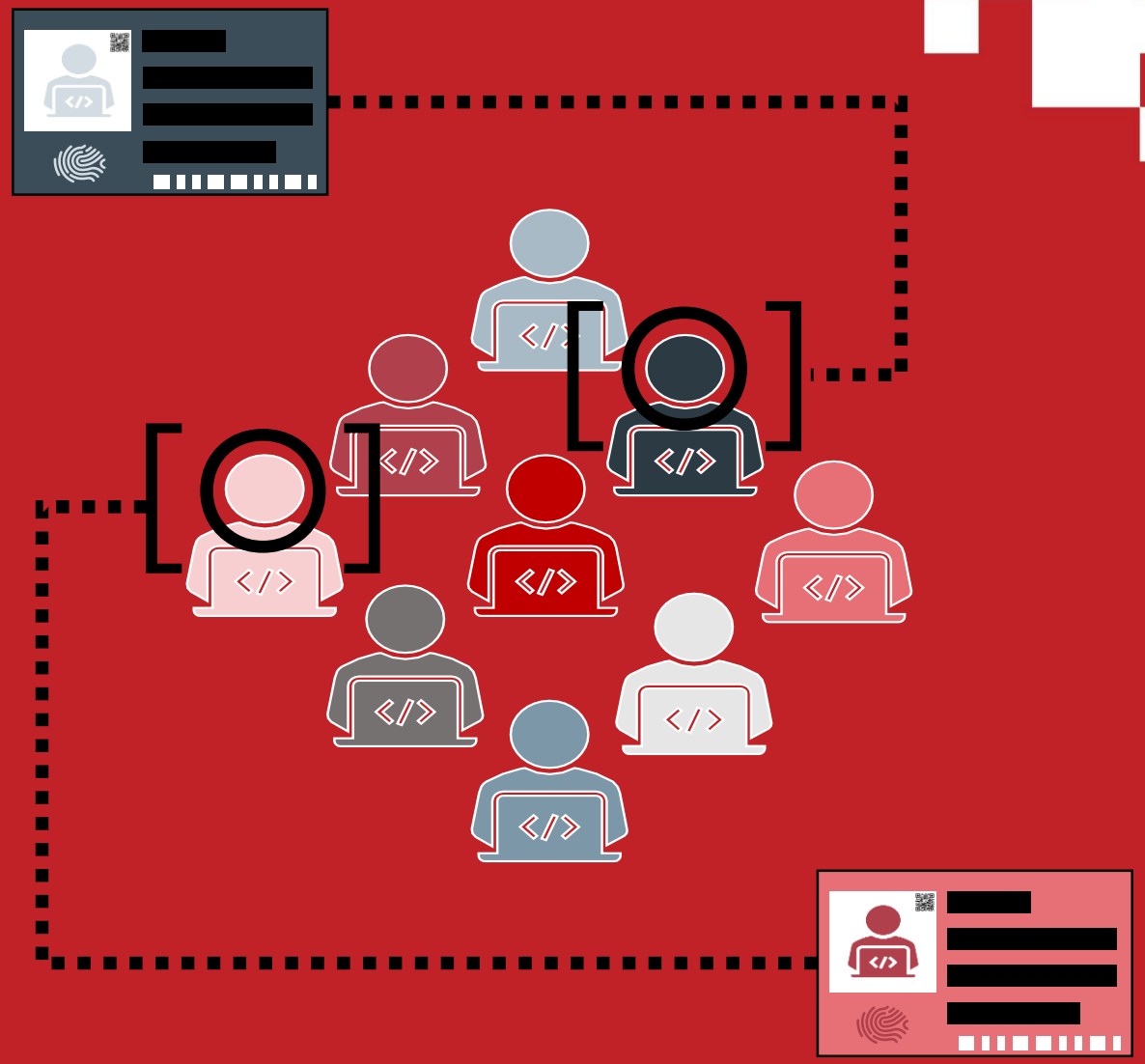
Level 3: Tenant Auto Provisioning

- Tenant can sign up for service and provisioning scripts run automatically
- Tenant may opt to sign up for optional services, or a higher service tier
- Security policies are dynamically generated and not static
- Tenant is able to discontinue optional services or revert to a lower service tier, deprovisioning scripts execute to remove services and/or capacity
- Enterprise team is engaged when auto provisioning or test scripts fail to execute; Factory team in cases of escalated issues

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework
Tenant Profiling

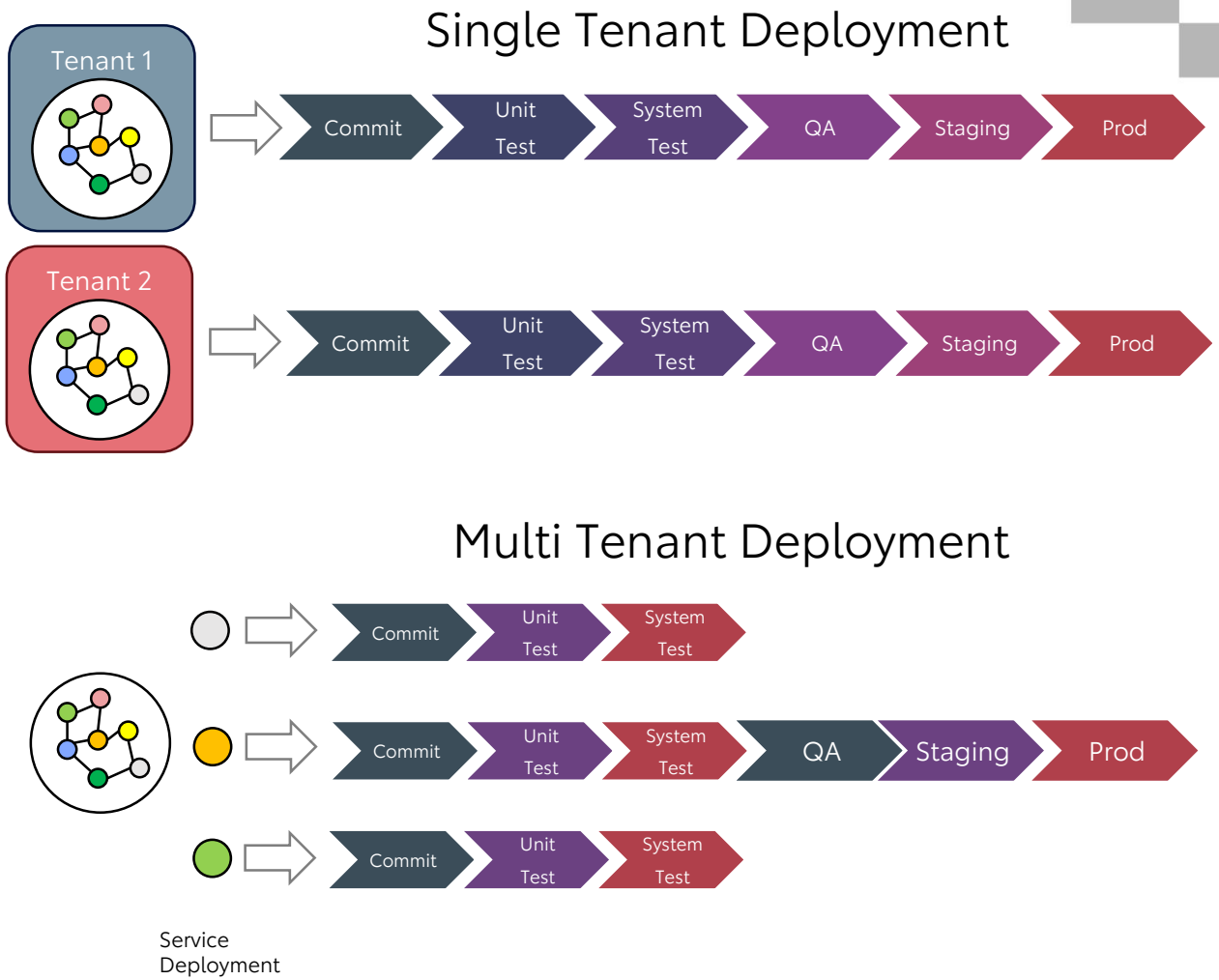
1. What level of isolation will be required by your tenants?
2. Are your tenants subject to specific compliance requirements that would imply a specific isolation level?
3. What are your tenant specific security requirements?
4. How much customization is expected for each tenant?
5. Will the solution offer different tiers and differentiated features per tier?
6. What is the expected load / usage profile of a typical tenant?
7. What SLA's will be offered to tenants and will these be differentiated by tier?



TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Build & Deploy Considerations

- Agility is our first priority; build abstraction at each layer to future-proof the solution
- Minimize tenant variation; build “most asked for features” in the core platform
- Centralize, version, and control tenant configuration and deployment
- Share and extend deployment automation wherever possible
- Prefer more granular deployments
- Design with isolation in mind
- Never accept downtime
- Invest in agile data migration and versioning strategies
- Expect to build and continuously evolve your own tooling
- Repeatability = Stability = Agility



TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Testing Considerations

Profiling and modeling tenant loads	• Build an application scaling profile and SLAs • Simulate tenant loads, scalability and application failures • Including “noisy neighbors” • Validate metering tiers and throttling • Validate auto scaling / self healing capabilities
Security and isolation testing	• Validate role-based access limits on cross-tenant access • Validate tenant isolation schemes • Assess data security constraints • Validate that the isolation model is working • Inject tenant context into another tenant, access should be denied
Validation storage scaling	• Test throughput and policies • Test bulk operations
Tenant Configuration	• Validate that shared, regional and tenant specific configurations result in a deterministic set of rules at runtime
Tenant Provisioning	• Test tenant provision scripts and scrips designed to ensure the provisioning scripts executed as intended
Tenant Data Migration	• Ensure tenant data can be migrated to a shared instance from a dedicated instance and vice versa
METER	• Validate compliance with Most Essential Technology Enterprise Requirements (METER)

TOYOTA FINANCIAL SERVICES

mFin Multi-Tenant Framework Frequently Asked Questions

1. What multi-tenancy option should I pick, 1, 2 or 3?

The option with the lowest ongoing operational cost (option 3) is preferred, but not always practical. Please consult with the Enterprise Architecture group for guidance.

2. Should external vendors use the IDS tenant ID? What if they already have a tenant ID? Should we have the vendor switch to our tenant ID?

Tenant ID is intended to ensure that systems are able to segregate tenant data, primarily for internal integration purposes. External vendors are not required to adopt the IDS tenant ID scheme, but Factories responsible for managing external vendors are expected to append the IDS tenant ID to data sourced from these vendors that are intended for internal consumption.

3. When generating files, do I create a single file with all tenants or one file per tenant?

Please generate one file per tenant.

4. How is mFin alignment measured, captured and managed for Factories?

The mFin alignment report (soon to include METER) is a self assessment report that factories are expected to complete and maintain. EA assisted with an initial baseline assessment.

5. What are some of the more common design pitfalls when designing for multi-tenancy?

- Designing deployments that require excessive downtime or interdependencies
- Insufficient investment in deployment automation and testing
- Insufficient investment in management and monitoring
- Addressing analytics and metering too late in the deployment design
- Deprioritizing SLA policies
- Allowing “noisy neighbors” to negatively impact on all tenants
- Assuming authentication and authorization = tenant isolation (it’s not)
- Focusing on features before multi-tenancy

TOYOTA FINANCIAL SERVICES

Most Essential Technology Enterprise Requirements (METER)

mFin Multi-Tenant Framework Frequently Asked Questions (METER)

1. What are the Most Essential Technology Enterprise Requirements (METER)?

- METER describes not what the software will do, but how the software will do it, for example, performance requirements, external interface requirements, design constraints, and quality attributes.
- **Overall, METERs are "Dimensions of Quality"**

2. Where do I track METER for my software?

- Tracking METERs in JIRA at the Story or Epic level is one method of entering requirements into the backlog of the factory.
- METERs are to be consumed with each Factory effort and should be used to satisfy the 'Definition of Done' (DoD)

3. When do I address METER in my product lifecycle?

- METER should be adhered to throughout the product lifecycle
- METER quality is expected at the "Build", "Test" and "Run" lifecycle phases

4. Do I need to comply with all of METER for my software? Are METER prioritized so I can focus on the most important ones?

- Story or Epic Agnostic – DoD METER are not specific to each Factory activity
- Only METER that pertain to your specific business and technology requirements need to be addressed by the Factory
- The Most Essential Technology Enterprise Requirements is a subset of the overall Technology Requirements

TOYOTA FINANCIAL SERVICES

Most Essential Technology Enterprise Requirements

Practicability	Dependability	Operability	Adaptability	Maintainability
☐ Accurateness	☐ Availability / Reliability	☐ Compatibility	☐ Changeability	☐ Analyzability
☐ Completeness	☐ Compliancy	☐ Controllability	☐ Configurability	☐ Consistency
☐ Correctness	☐ Resiliency	☐ Interoperability	☐ Customizability	☐ Modularity
☐ Preciseness	☐ Responsiveness	☐ Learnability	☐ Extensibility	☐ Orthogonality
☐ Relevancy	☐ Recoverability	☐ Observability	☐ Portability	☐ Provability
☐ Suitability	☐ Safety	☐ Testability	☐ Reusability	☐ Supportability
☐ Usefulness	☐ Scalability	☐ Usability	☐ Togglability	☐ Sustainability

Functional Fitness - Degree to which a software product provides functions that meet stated and implied needs of customers when used under specified conditions & context

Technical Performance - Degree to which a software product reliably functions at a specified level of performance when used under specified conditions for a specified period of time

Functional Experience - Degree to which a software product can be used by specific users to experience specific value & to achieve specific goals with effectiveness, efficiency, & satisfaction in a specific context of use

Functional Flexibility - Degree to which a software product can be modified & reused by product team members to remain agile & responsive to business-driven changes

Maintainability - Degree to which a software product can be supported & sustained by product team members to remain responsive & cost effective

TOYOTA FINANCIAL SERVICES

Technology Enterprise Requirement

Practicability

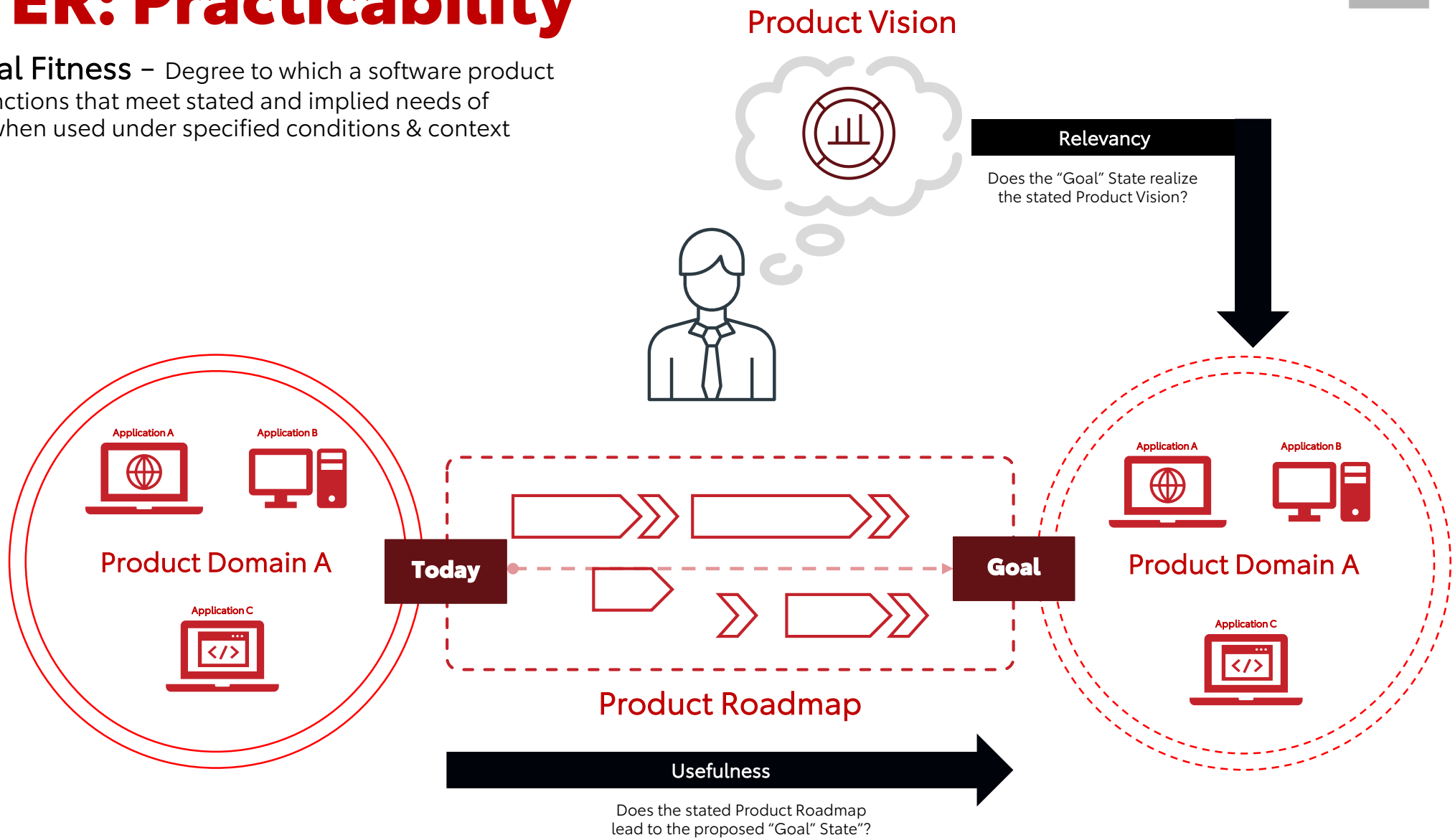
Accurateness	Degree to which a software product functions carefully & conforms closely to the true expectations as defined per the requirements & design specifications
Completeness	Degree to which a software product fully implements all features requested as per the requirements & design specifications
Correctness	Degree to which a software product answers business questions appropriately as defined per the requirements & design specifications
Preciseness	Degree to which a software product measures exactly, calculates rightly, performs specifically, & behaves strictly as defined per the requirements & design specifications
Relevancy	Degree to which a software product provides functions that are closely connected & related to the task at hand to accomplish specific objectives
Suitability	Degree to which a software product provides functions that are acceptable for specific users & appropriate for specific situations to accomplish specific objectives
Usefulness	Degree to which a software product provides functions that are practical & beneficial for specific users & for specific situations to accomplish specific objectives

Functional Fitness: Degree to which a software product provides functions that meet stated and implied needs of customers when used under specified conditions & context

See diagram on next page

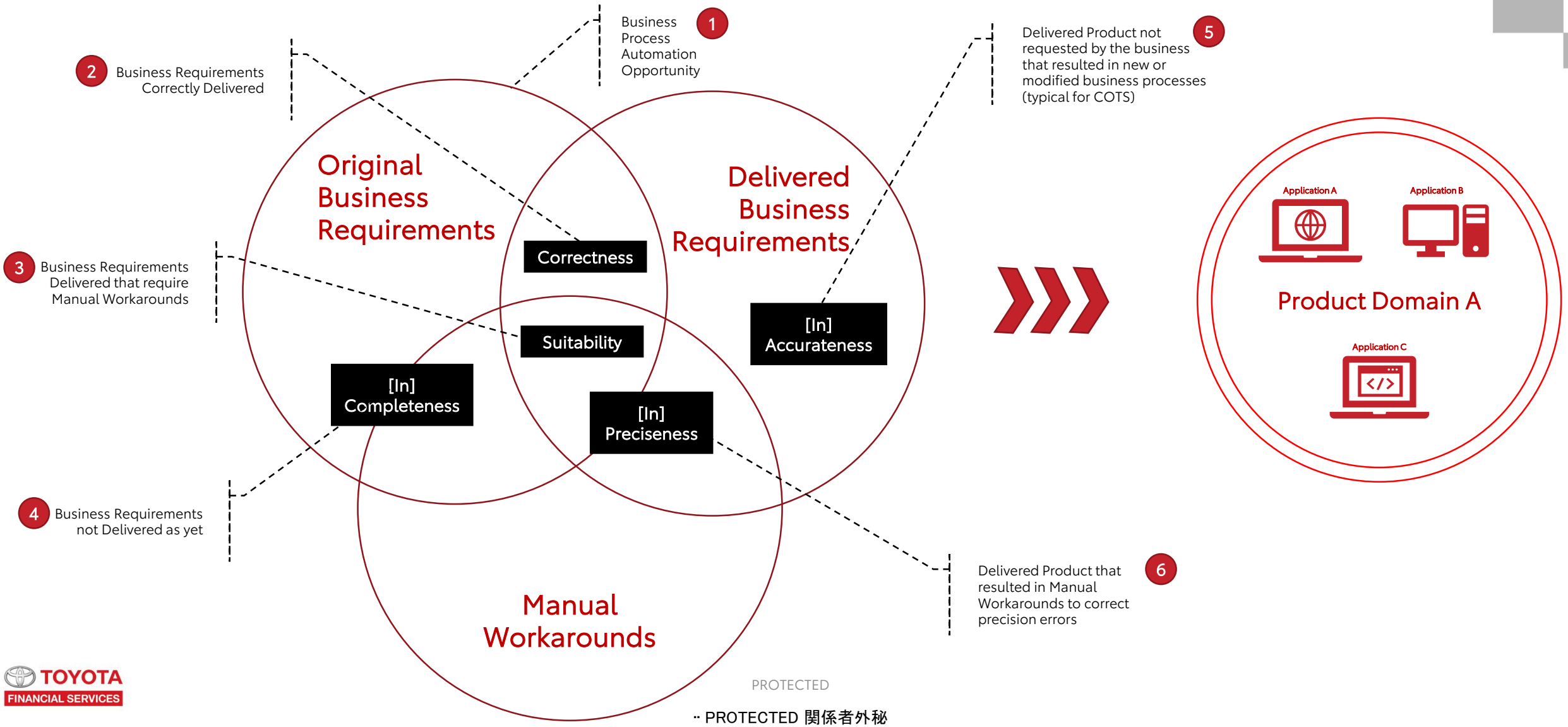
METER: Practicability

Functional Fitness – Degree to which a software product provides functions that meet stated and implied needs of customers when used under specified conditions & context



METER: Practicability … Continued

Functional Fitness – Degree to which a software product provides functions that meet stated and implied needs of customers when used under specified conditions & context



Technology Enterprise Requirement

Dependability

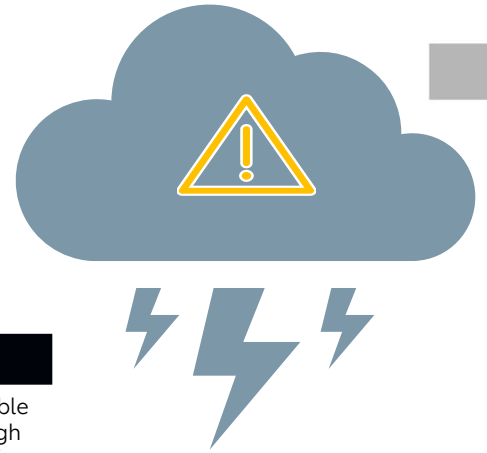
Availability	Degree to which a software product is in a specified operable and functioning condition to perform required functions at any given instant of time within a given time interval
Compliancy	Degree to which a software product reduces risk by adhering to legal & regulatory compliance policies & procedures and internal standards & rules
Reliability	Degree to which a software product remains operational over time without failure & maintains a specified level of performance under specified conditions for a specified period of time
Resiliency	Degree to which a software product recovers and remains functional from the customer perspective under stressful & chaotic conditions such as power outages, system crashes, natural disasters, etc.
Responsiveness	Degree to which a software product performs to user & system requests by responding within an acceptable time interval & specified latency without noticeable or unacceptable delay
Recoverability	Degree to which a software product is able to restore to the previously stable state after failure.
Scalability	Degree to which a software product allows increase of load & operates consistently & reliably under such increased load without impact on the responsiveness & performance of the system

Technical Performance: Degree to which a software product reliably functions at a specified level of performance when used under specified conditions for a specified period of time

See diagram on next page

METER: Dependability

Technical Performance – Degree to which a software product reliably functions at a specified level of performance when used under specified conditions for a specified period of time



Availability

Is the system available during defined business hours?

Resiliency

Is the system able to maintain high availability during power outages, crashes, natural disasters, etc.

Responsiveness

Is system response time adequate for 80% of the most frequently used transactions?

Scalability

Is the system able to maintain responsiveness under load?

Application A

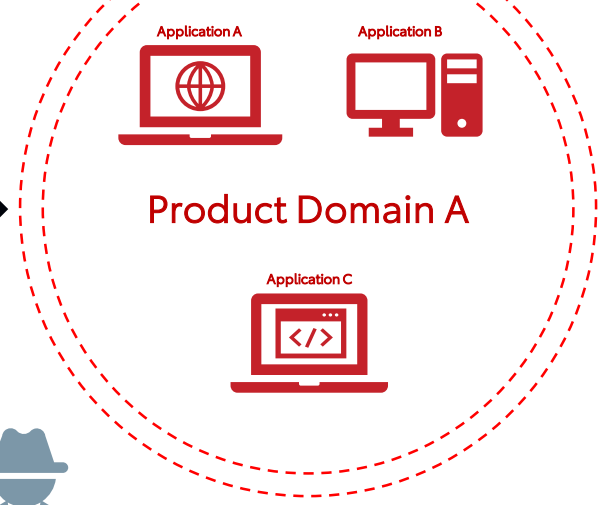
Application B

Application C

Product Domain A

Recoverability

Does the system have a tested TRP for failover?



Reliability

Are system performance and failure rates acceptable when measured over longer periods of operation under varying conditions?

Safety

Does the system implement relevant InfoSec policies and guidelines?

Compliancy

Does the system remain in compliance with relevant country specific laws & regulations; tenant policies & procedures?

Technology Enterprise Requirement

Operability

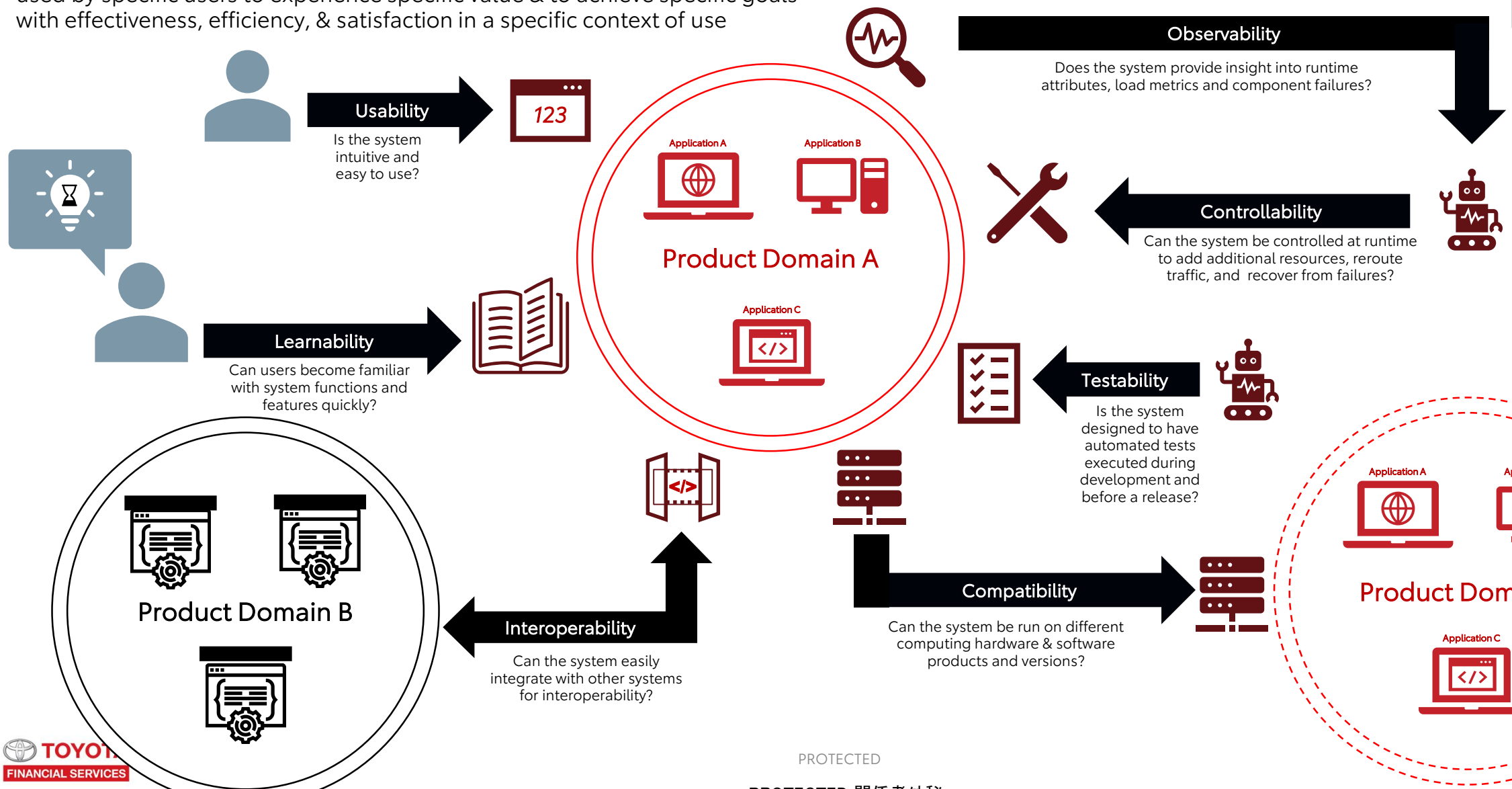
Compatibility	Degree to which a software product can be executed on different types of computing hardware & software environments without detrimental impact to its usability, functionality, & performance
Controllability	Degree to which a software product can be controlled to change its internal state through control inputs in order to manage the system safety & performance
Interoperability	Degree to which a software product communicates, exchanges, & interacts with other components or systems correctly & effectively without failure or loss of information
Learnability	Degree to which a software product allows users to quickly become familiar with and able to make good use of all their features and functions
Observability	Degree to which a software product can be observed to determine its internal state from its outputs in order to monitor the system safety & performance
Testability	Degree to which a software product can be verified & validated with ease against the requirements & design specifications
Usability	Degree to which a software product can be used by specific users to achieve specific goals with ease, effectiveness, efficiency, satisfaction, & experience in a specific context of use

Functional Experience: Degree to which a software product can be used by specific users to experience specific value & to achieve specific goals with effectiveness, efficiency, & satisfaction in a specific context of use

See diagram on next page

METER: Operability

Functional Experience - Degree to which a software product can be used by specific users to experience specific value & to achieve specific goals with effectiveness, efficiency, & satisfaction in a specific context of use



Technology Enterprise Requirement

Adaptability

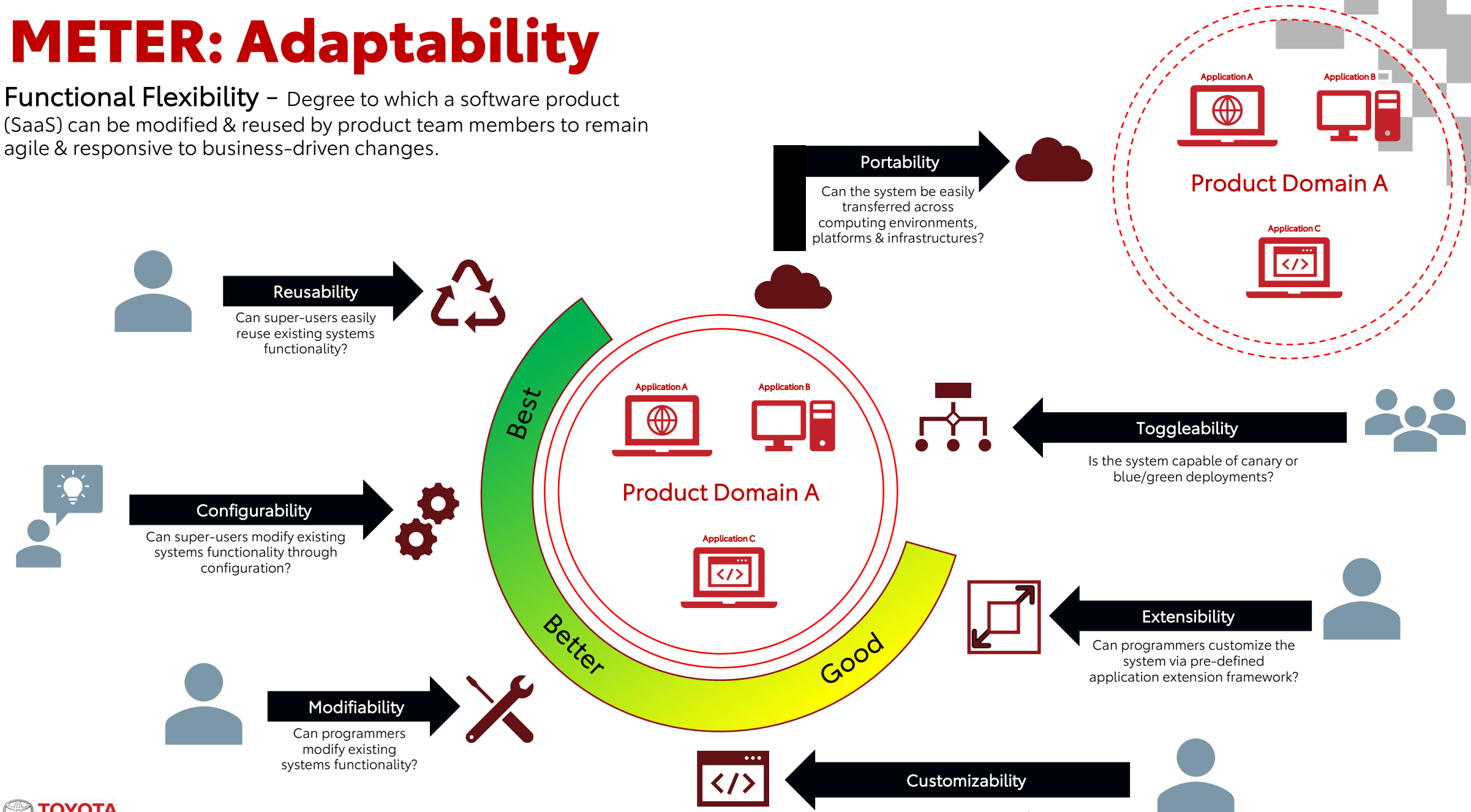
Changeability	Degree to which a software product & its features lends itself to be modified or altered with ease & speed with no or minimal & manageable impact to the overall system
Configurability	Degree to which a software product & its features can be dynamically personalized & continually shaped with ease by non-programmers without source code modification to meet an organization's industry-specific & organization-specific needs
Customizability	Degree to which a software product & its features can be dynamically personalized & continually shaped with ease by programmers with source code modification to meet an organization's industry-specific & organization-specific needs
Extensibility	Degree to which a software product & its features lends itself to be extended & added with new features with ease & speed with no or minimal & manageable impact to the overall system
Portability	Degree to which a software product can be transferred, installed, & used across different computing environments, platforms, & infrastructures with ease
Reusability	Degree to which a software product & its features lends itself to be used or repurposed with ease during design-time or development-time or run-time through referencing or composition or integration, etc.
Togglability	Degree to which a software product & its features can be dynamically enabled or disabled or made invisible or visible to a subset of users (e.g. canary releases) during runtime

Functional Flexibility: Degree to which a software product (SaaS) can be modified & reused by product team members to remain agile & responsive to business-driven changes.

See diagram on next page

METER: Adaptability

Functional Flexibility – Degree to which a software product (SaaS) can be modified & reused by product team members to remain agile & responsive to business-driven changes.



Technology Enterprise Requirement

Maintainability

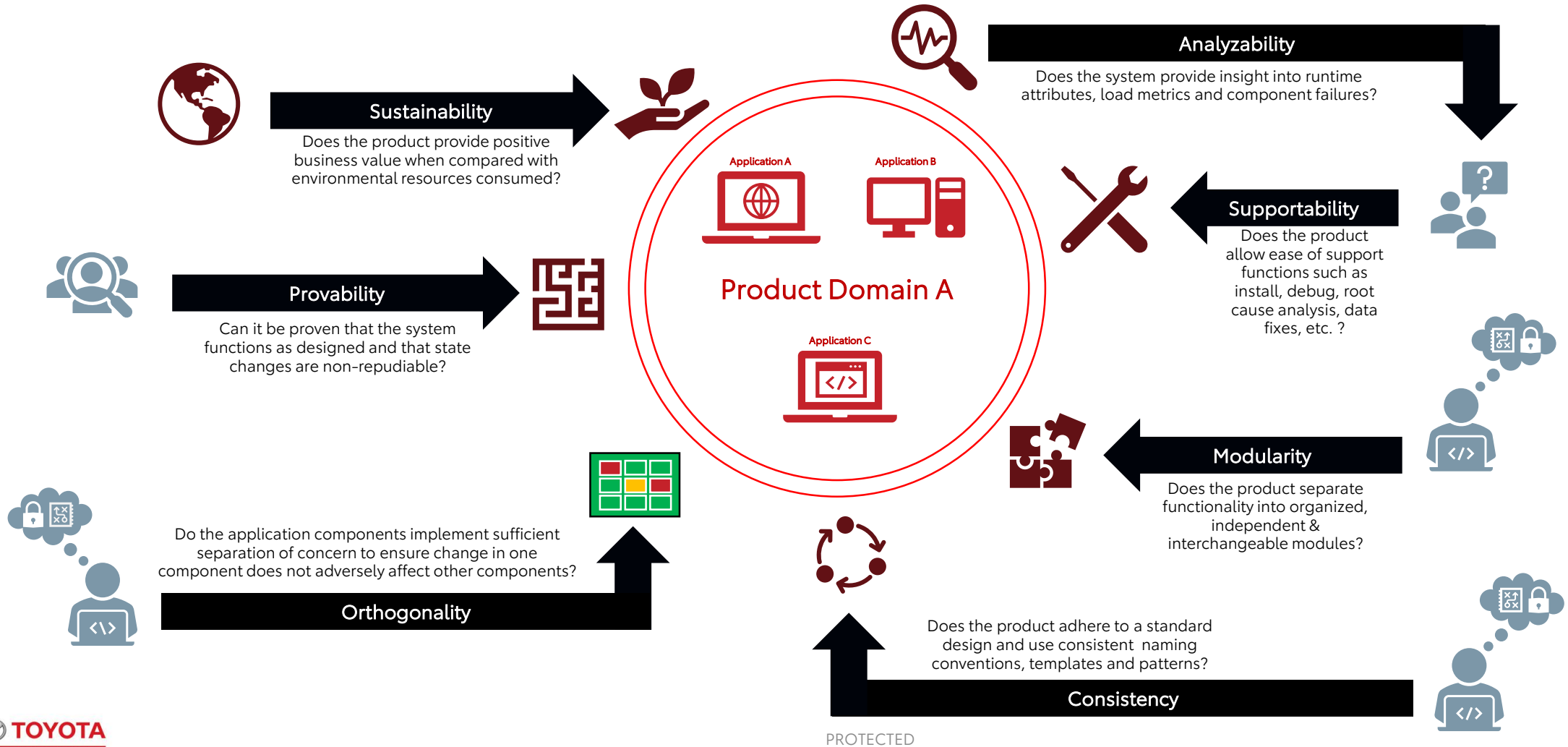
Analyzability	Degree to which a software product & its features can be analyzed with ease to diagnose for deficiencies & to identify causes of failures in the software through the utilization of chronological versions of traceable software assets
Consistency	Degree to which a software product & its features have been formatted, designed & constructed uniformly conforming to specific standards and designed & implemented to behave at runtime with repeatability & reproducibility characteristics
Modularity	Degree to which a software product & its features separates the functionality into independent & interchangeable modules, such that each contains everything necessary to provide the desired functionality
Orthogonality	Degree to which a software product & its features guarantees that modifying the functional or technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system
Provability	Degree to which a software product proves that the design & the implementation of system will perform as intended & provides assurance that the actions & the state changes performed by the system cannot be denied (i.e. non-repudiation)
Supportability	Degree to which a software product enables personnel to install, configure, and monitor software products, to reduce risks & resolve incidents by identifying faults & errors and by debugging, analyzing, & identify root causes, to solve problems & restore services, and to perform routine maintenance such as applying patches & upgrades, etc. with ease
Sustainability	Degree to which a software product delivers & captures business value relative to amount of assets & resources used from the social, economic, educational, health, & environmental perspective

Technical Operations: Degree to which a software product can be supported & sustained by product team members to remain responsive & cost effective

See diagram on next page

METER: Maintainability

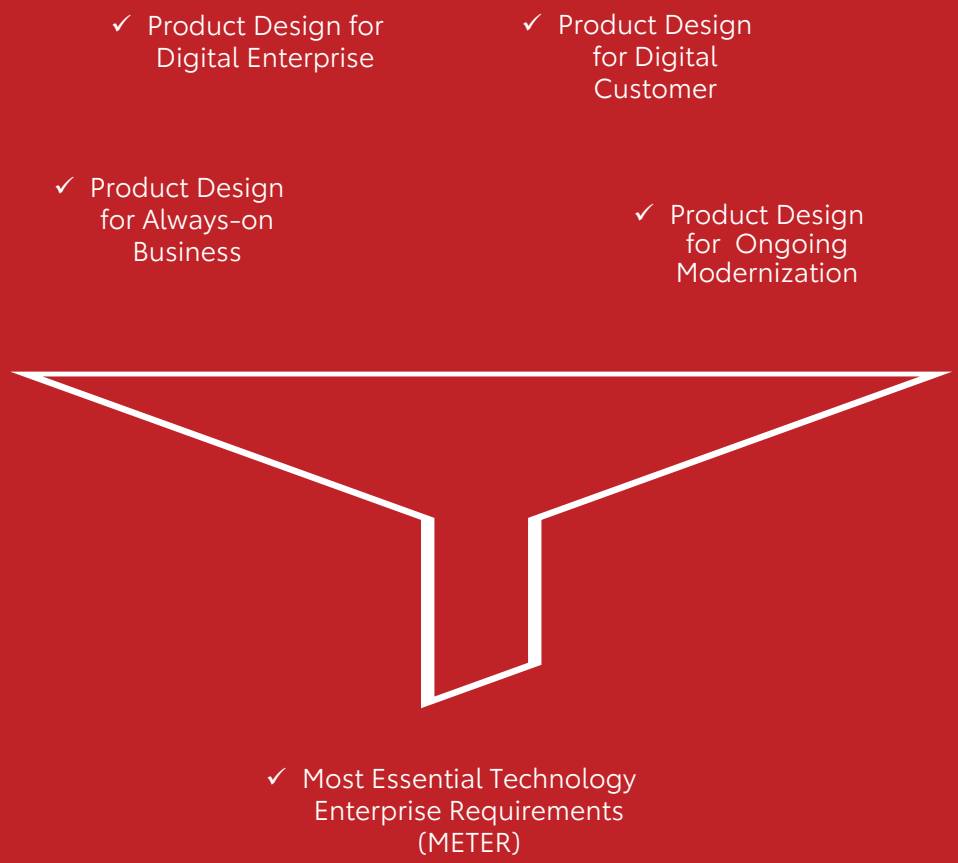
Technical Operations – Degree to which a software product can be supported & sustained by product team members to remain responsive & cost effective



PROTECTED

Product Design to METER Cross Reference

Product Design	Practicability	Dependability	Operability	Adaptability	Maintainability
		METER			
Digital Enterprise		✓	✓	✓	
Digital Customer	✓				
‘Always On’ Business		✓	✓		✓
Ongoing Modernization		✓	✓	✓	✓



Information Security and Identity Governance

Information Security Requirements

Access Control Security
(Integrate with TFS IAM tools, Least privilege access, MFA, etc.)

Application Security
(Secure coding, Encryption, API Security, etc.)

Data Security
(AES 256 Encryption, Key Management, Access control, etc.)

Network Security
(Private connections, TLS 1.2, etc.)

Security Monitoring
(Code Scan, Vulnerability Scan, SIEM integration, etc.)

3rd Party
(Contracts, Vendors meet TFS Security requirements, etc.)

How do I ensure that all applicable Information Security requirements are incorporated into routine Factory releases?

Acceptance Criteria

- Security requirements are to be implemented for all solutions (on premises and cloud implementations).
  - Refer to the **TFS Security guidebook** for tips on secure coding practices.
  - Consider security controls as part of your DevOps practice (aka DevSecOps). Additions, changes, or modifications to applications or data may require that **additional information security controls** be implemented, as required by regulatory entities.
  - Run a security scan and remediate vulnerabilities for every release before deployment.

How

- Complete the **Information Security Questionnaire** for every Release at the end of the grooming sessions, once the scope for a release is finalized.
  - Work with your Domain Security champions(DSC) to create security stories and consult on secure design for your releases/sprints to implement the required security controls.
  - Please refer to the latest published **InfoSec Policies**

TOYOTA FINANCIAL SERVICES

mFin InfoSec Tech Standards By User and Usage Type

Consumer/Dealer Identities (Untrusted/ Ungoverned)	Identity Manager & Multi Factor Authentication	TFS External Access Gateway incl. User Store, Access Manager, Risk Based Authentication
	One Time Passcode	Twilio for OTP via e-mail, voice call & SMS
Team Member (Trusted/Governed)	Single Sign On & Multi Factor Authentication	Okta for SAML & Oauth (Idaptive in limited use while existing API services are transitioned)
	Identity Lifecyle, Governance & Provisioning	MyAccess aka. MyAccess incl. Auto User Provisioning
Identity & Access Management Supported Protocols	TFS External Access Gateway	SAML 2.0, OAuth 2.0, Open-ID Connect, REST, JSON, User-Managed Access (UMA) 2.0., SCIM, FIDO Web Authentication etc.
	Okta	SAML 2.0, OAuth 2.0, FIDO U2F, Open-ID Connect, WS-Fed etc.
	MyAccess	REST/Webservice, JDBC, SCIM, Oracle, SQL and platform specific API's

TOYOTA FINANCIAL SERVICES

Managing Customer and Dealer Access in mFin Applications

Account Creation

Account Registration

Authentication/Login

Password Reset

Account Creation

Authentication/Login

For externally facing applications (Customer or Dealer) we need to integrate with security for authentication (incl. OTP and MFA), and additionally integrate with internal applications and/or external partners.

Considerations

- Confirm that the application specific business requirements for user provisioning, access authorization and decommissioning are understood and documented.
- Refer to the TFS Security guidebook for general security considerations.

Solution

- Integrate the application with TFS External Access Gateway using SAML or OAuth.
- **Integration details for Customer and Dealer Access**:
  - Engage with the Identity and Access Management team via TFS_External_IAM@toyota.com
- **Required firewall access requests:**
  - Engage with the Network Engineering team
- **For more details on how Customer and Dealer Security Integration at the API layer is implemented:**
  - Refer to *"How to Integrate using APIs"* section of this playbook.

Customer Account Creation

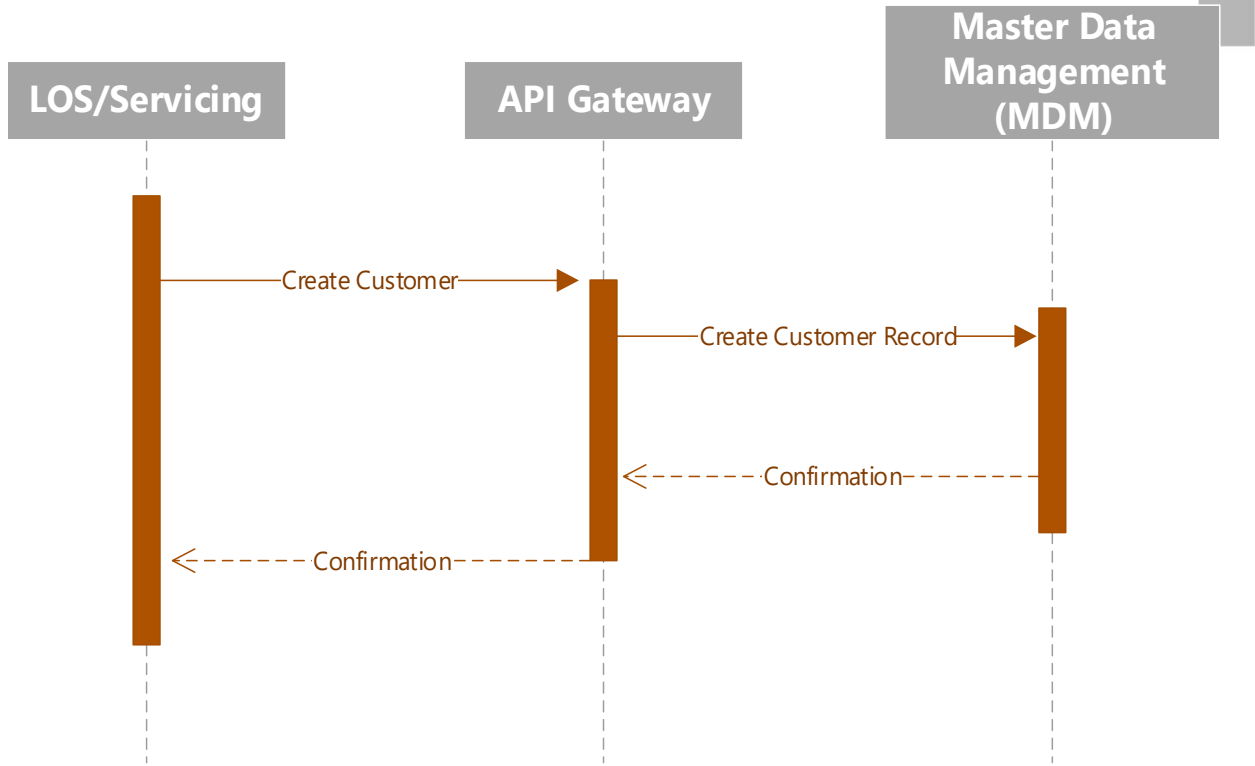
How are customer accounts created prior to them being able to access digital services through web and mobile?

Considerations

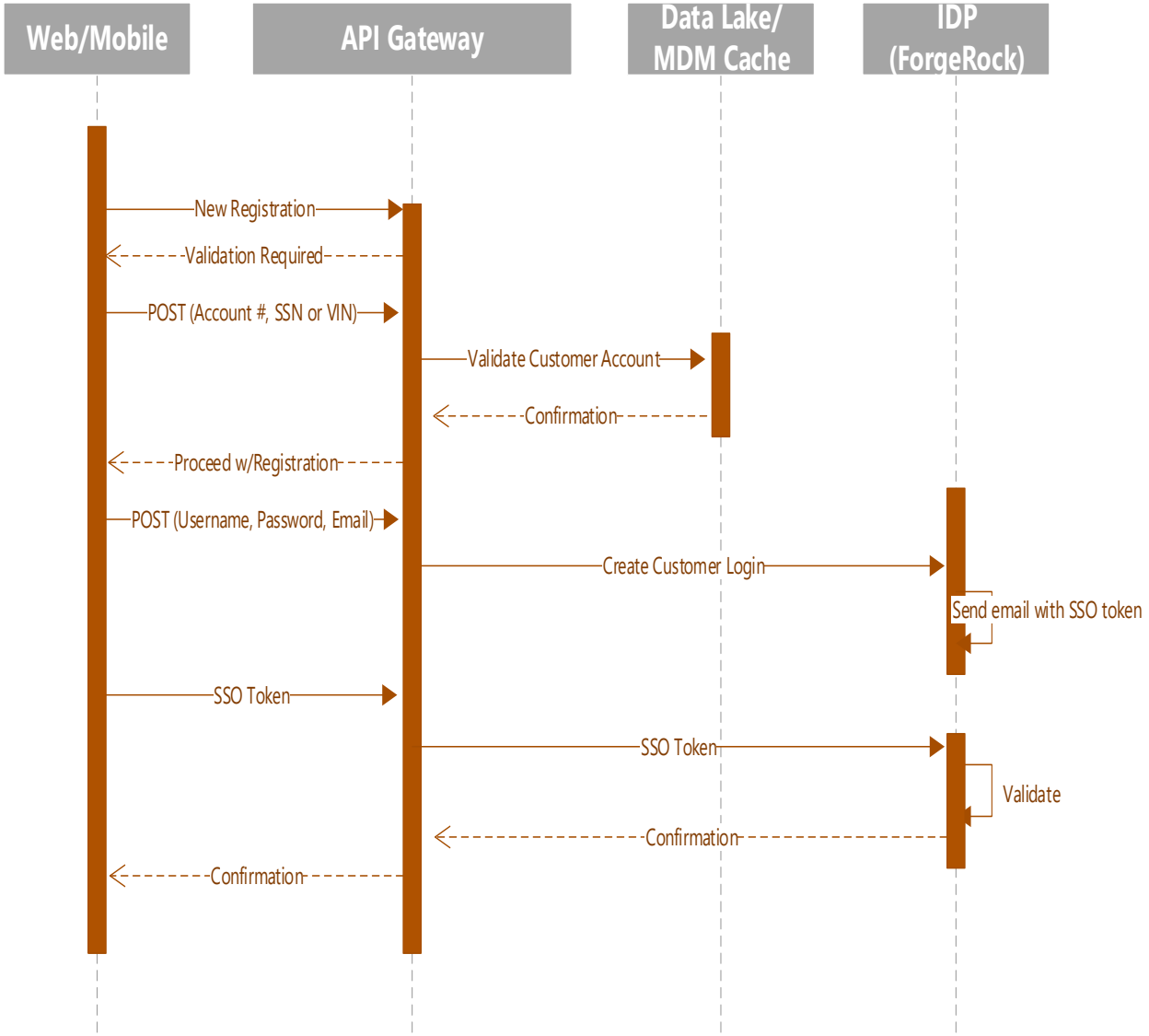
- Customer personal information and account details must reside in Master Data Management before online access can be established.

Solution

- LOS and Servicing are the primary routes to establish an initial customer record.
- When potential customers apply for a loan and their request is approved, the originations and services systems will notify MDM through API Layer, with both customer and account information.
- MDM will store the provided customer and account information required for the registration process.



Customer Account Registration



How can customers have a consistent experience across digital systems to register and set up their online login for use with web and mobile systems?

Considerations

- The API Gateway should remain the primary avenue for system integration and provides a common interface point.
- TMCC is the identity provider for customer records and this function will not be outsourced to a vendor or fragmented across vendors.

Solution

- The experience layer captures the necessary information from the customer during their registration and can leverage the API Gateway to optimize interactions with the Identity Provider (IDP) and Master Data Management systems for verification.
- After the customer's personal information is verified against their account their login account can be registered with the IDP to facilitate future logins across web and mobile applications.

Customer Adaptive Authentication with MFA

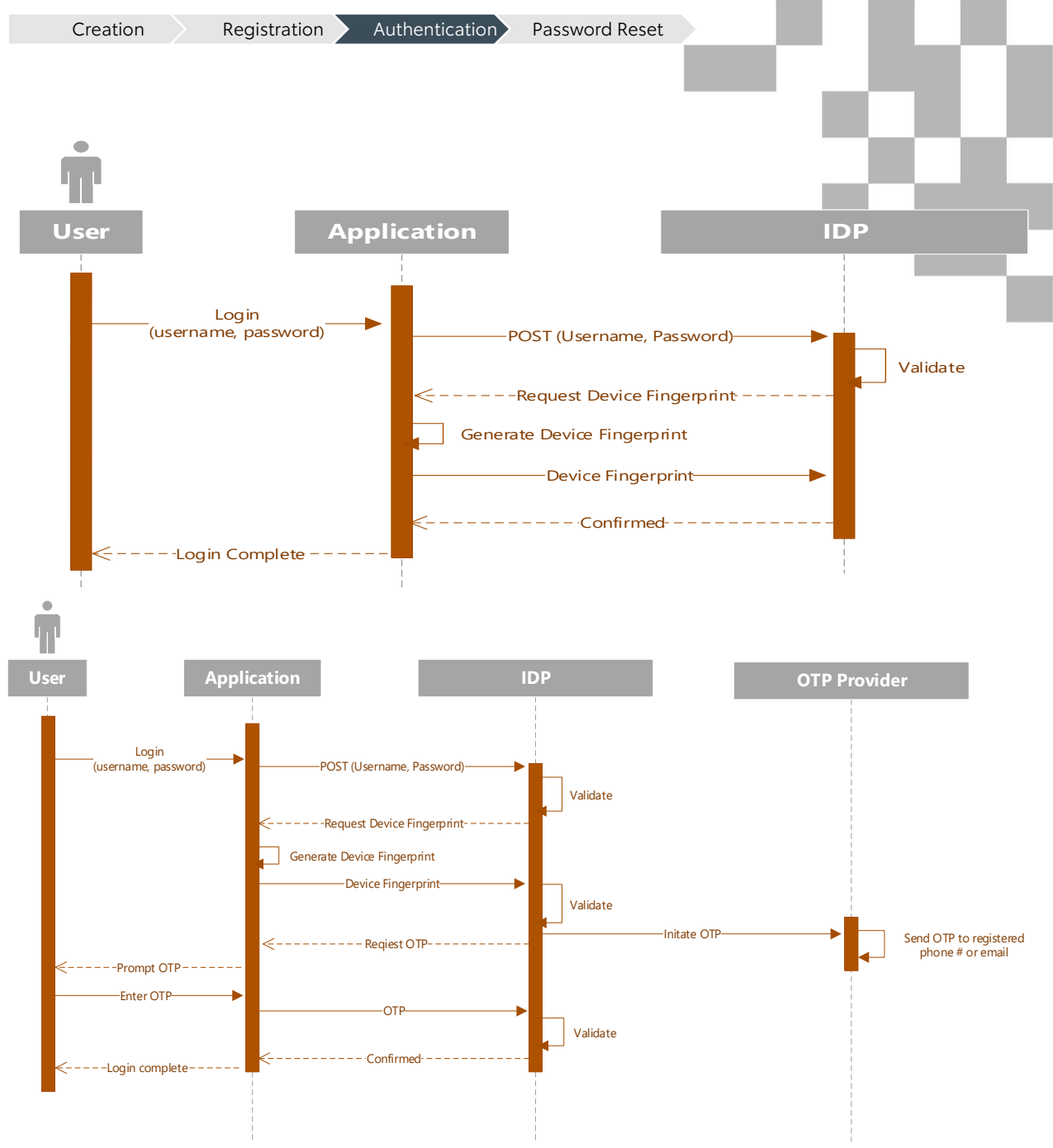
How can customers be allowed to access their account information safely and securely, while also retaining a convenient digital experience?

Considerations

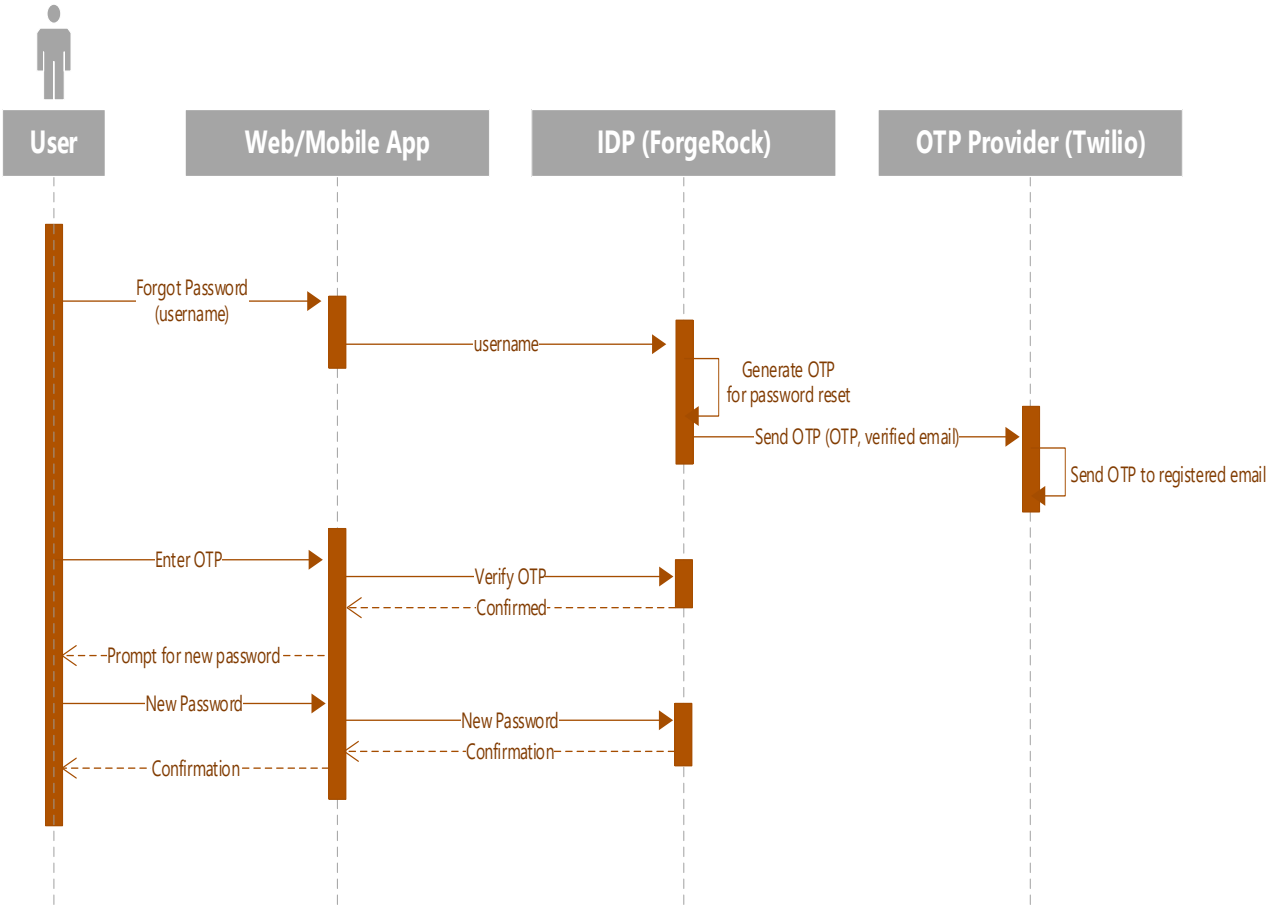
- Protecting the information and systems for our customers and our brand is a key part of providing a seamless digital experience for our customers and should be balanced with fast and easy processes.

Solution

- Adaptive authentication provides increased security for customer's logins, while also enabling a streamlined experience.
- When using an unrecognized device to access their account, customers are prompted for an additional layer of security by using a one-time passcode as a second factor.
- This can also be triggered when certain risk factors are detected that may put the customer's data at risk.
- Contact TFS_External_IAM@toyota.com for additional information on enabling this.



Customer Password Reset



How can customers securely reset their password in the case that they forget their credentials?

Considerations

- Password recovery features are necessary but can also be exploited to take over individuals' accounts to steal information.

Solution

- Enhancing the password recovery process with a second factor of authentication provides additional protection to our customers through leveraging something that they have in addition to the information that they, or perhaps an attacker, will know.
- The IAM team can assist with enabling this for customer login portals.

Dealer Account Creation

How to create dealer user accounts?

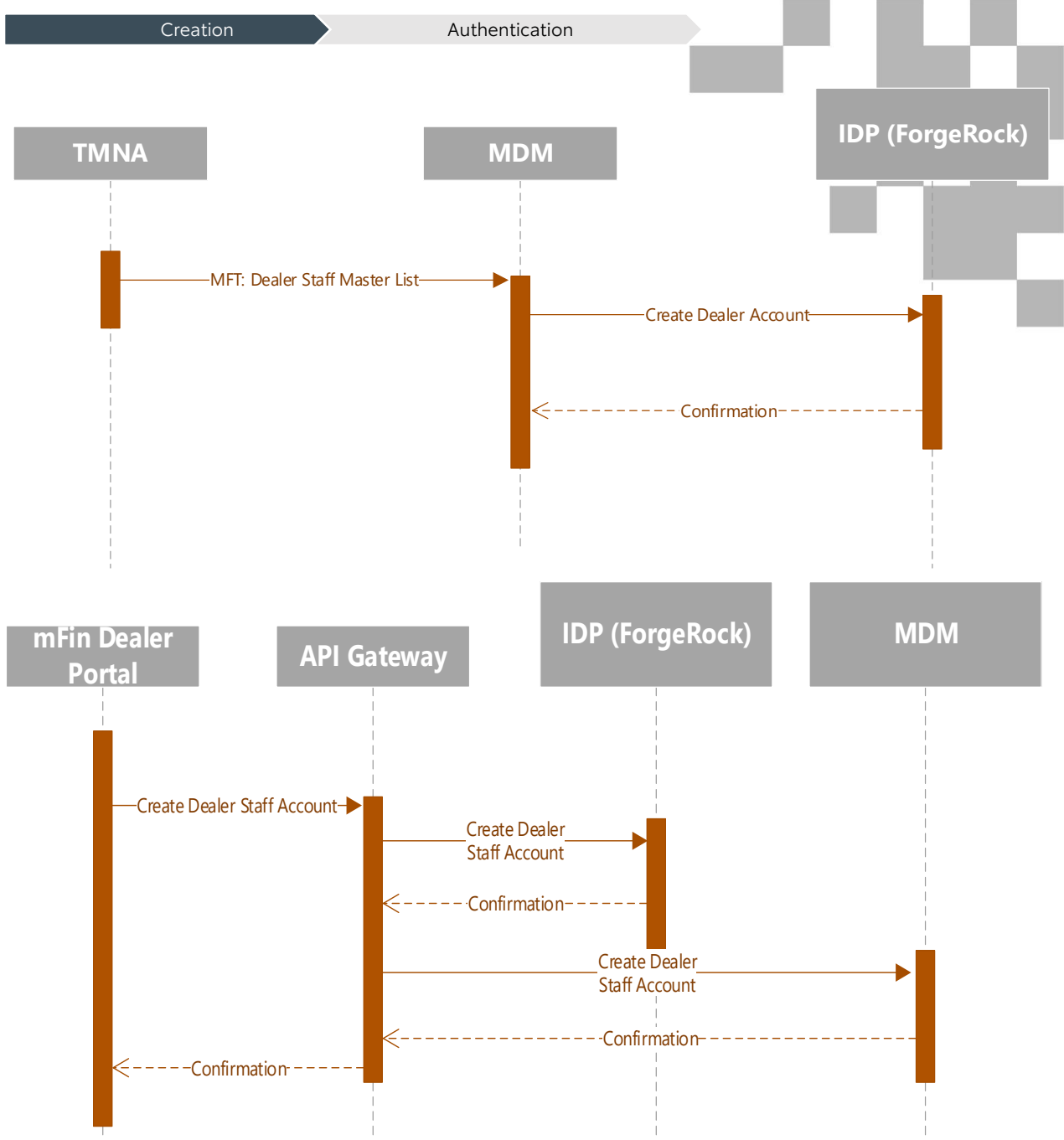
Dealer users are divided in two groups, the dealer admins and dealer staff. The first group's information is sent by an mFin Tenant to TFS, and the second group that represents dealer local hires for their dealerships. The information of both groups will be managed through Dealer Portal.

Considerations

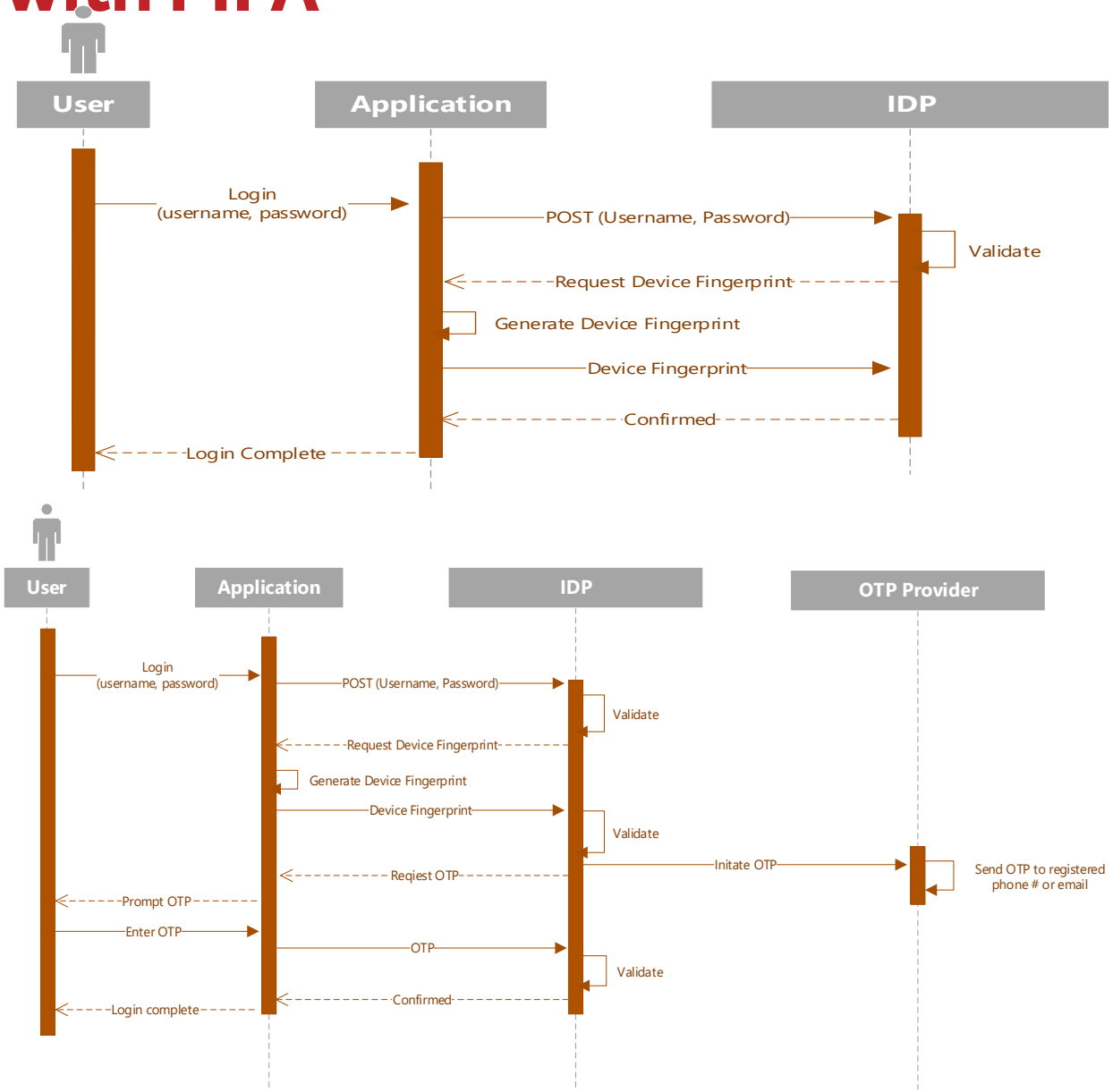
- Dealers have two types of users that we must support, Dealer Admins who server as principal points of contact within the dealerships and Dealer Staff members for each dealership.
- TMCC is the identity provider for dealer records and this function will not be outsourced to a vendor or fragmented across vendors.
- Dealer Staff members may work at more than one dealership at a time or be hired on a temporary basis, and we therefore need a solution to quickly provision and deprovision these types of accounts.

Solution

- A federated user access admin works well to meet this requirement.
- Toyota and Lexus dealerships have both Dealer Admins and Dealer Staff accounts populated through a data feed from TMNA.
- Going forward, dealerships for mFin tenants will have their admin accounts provisioned, after which they can directly create and manage accounts for the dealer staff members.



Dealer Adaptive Authentication with MFA



How can dealer staff members be allowed to access systems to support our mutual customer safely and securely, while also retaining a convenient digital experience?

Considerations

- While dealers are our valued partners in serving our customers, they access our systems from outside of our managed environment, just as customers do.

Solution

- Adaptive authentication provides increased security for dealers' logins, while also enabling a streamlined experience.
- When using an unrecognized device to access systems, they are prompted for an additional layer of security by using a one-time passcode as a second factor.
- This can also be triggered when certain risk factors are detected that may put a customer's data at risk.
- Contact TFS_External_IAM@toyota.com for additional information on enabling this.

Managing Team Member Access in mFin Applications

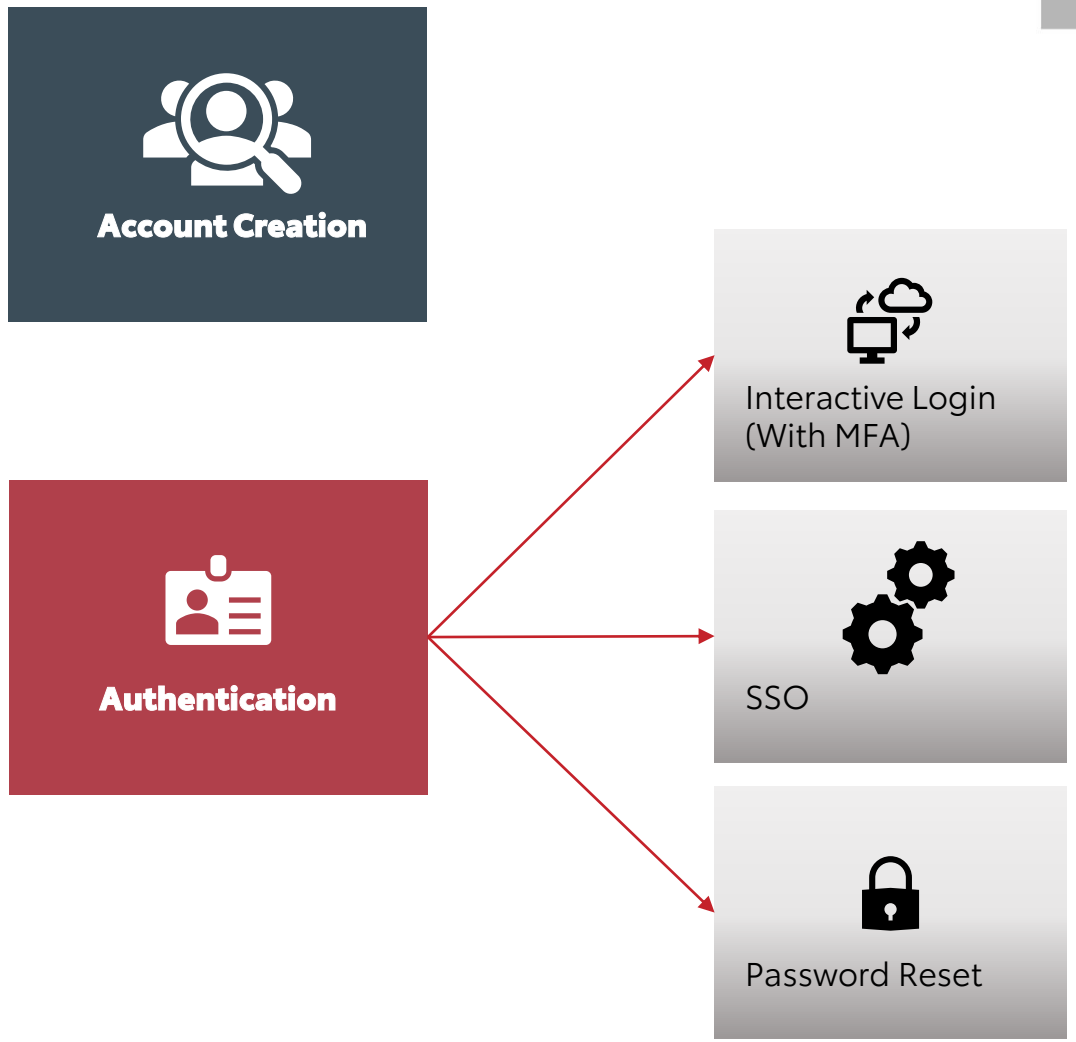
How do we leverage the existing security infrastructure while also ensuring that we protect customer, dealer and associate information?

Considerations

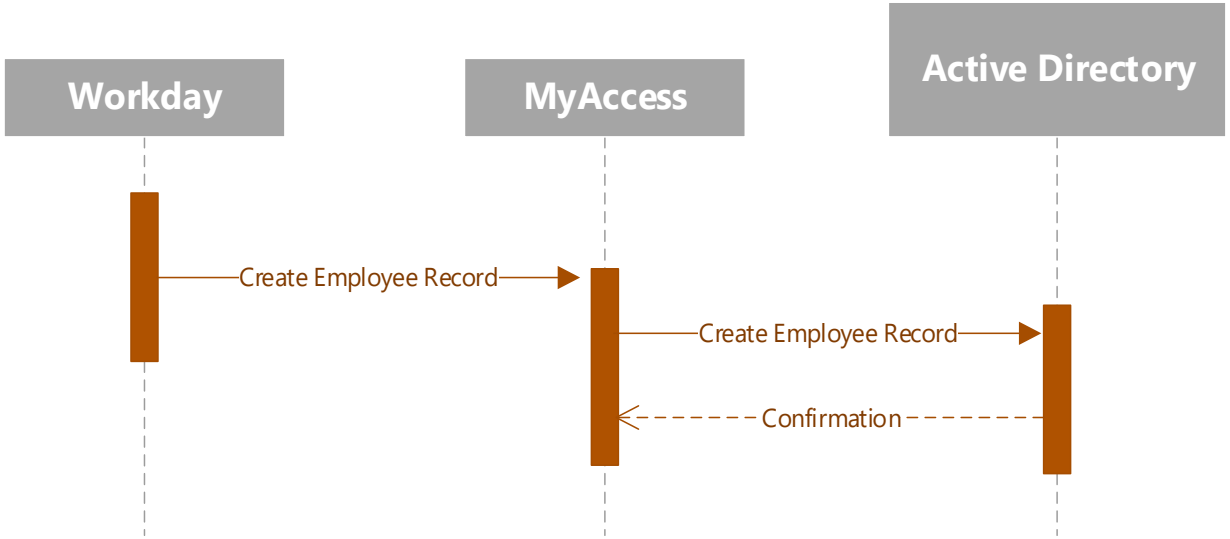
- For implementing new applications (COTS or Custom) or upgrading existing applications we need to integrate with security for authentication (incl. SSO and MFA), authorization and possibly integrate with other applications and/or external partners.
- Confirm that the application specific business requirements for user provisioning, access authorization and decommissioning are understood and documented.
- Refer to the TFS Security guidebook for additional security considerations.

Solution

- The IAM Team provides several secure options for associates to gain access to internal systems.
- Enhanced security is provided by Mult-Factor Authentication and Single Sign-On as well as risk-based triggers for adapting authentication processes seamlessly.



Team Member Account Creation



How are Team Members added to TFS application systems, and how access is provided to applications within mFin architecture platform?

Considerations

- In addition to additional account creation, access must be managed over time throughout the lifetime of the user in our systems.

Solution

- MyAccess is the centralized way for new Team Member accounts to be added to our systems, as well as how access rights are managed over time.
- Once new Team Members join and are added to our systems, MyAccess will continue to be used for requesting additional access and reviewing Team Members' access annually to ensure that the right access, no more and no less, is maintained.

Team Member Adaptive Authentication with MFA

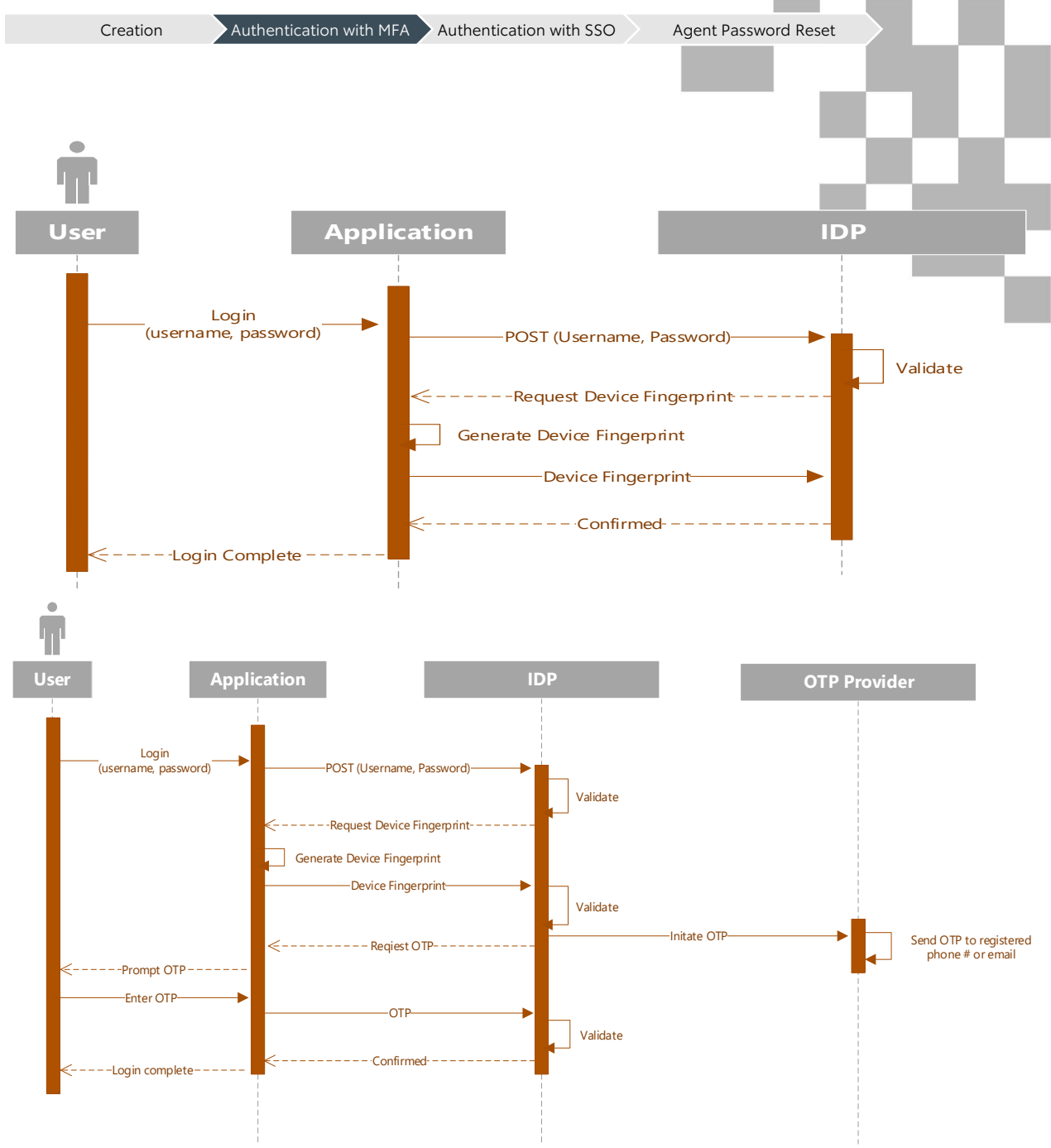
How to increase our applications security level beyond simple username / password concept?

Considerations

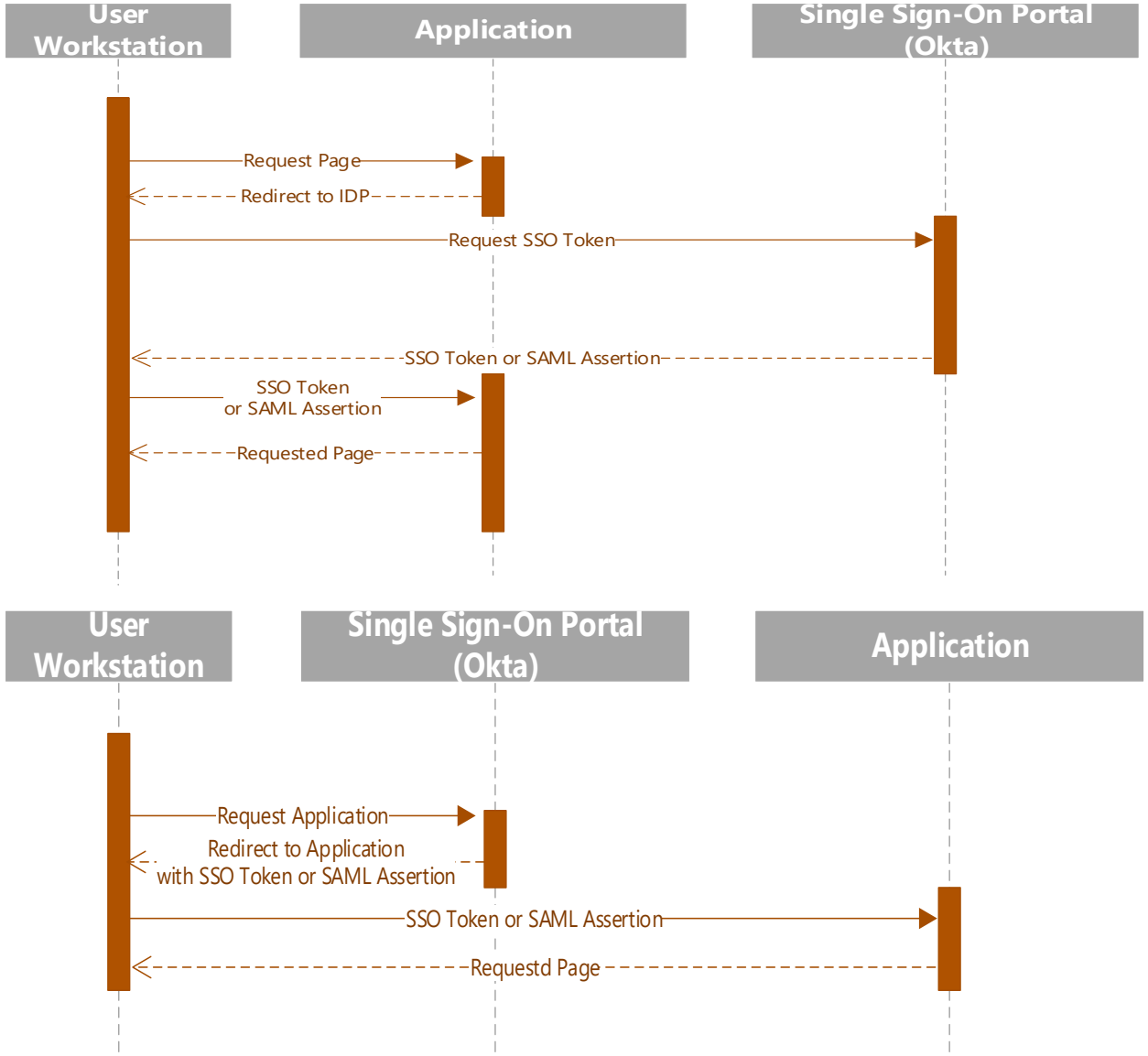
- Multi-Factor authentication should be considered for the extra layer of security.
- Multi-Factor authentication is a requirement and not a desired functionality anymore.
- There are multiple approaches to MFA, including sending a random code to user's smartphone, requesting answers to previously answered security questions, fingerprint or facial recognition, etc.

Solution

- Adaptive authentication provides increased security for Team Members' logins, while also retaining ease of use.
- The use of a second factor of authentication is triggered in certain risk scenarios, such as access from outside the corporate network or from remote locations, to ensure the highest level of assurance for critical systems.
- Contact Mytfsaccess@toyota.com for additional information on enabling this.



Team Member Passwordless Authentication with SSO



How can we move beyond usernames and passwords for ensuring secure access?

Considerations

- Single Sign-On is a common industry practice for providing security authentication without hassle of entering passwords manually.

Solution

- Okta provides a secure portal for Team Members' access.
- Prior to users leveraging Okta for passwordless authentication there must be trust established between Okta and the application.
- Contact Mytfsaccess@toyota.com for assistance with configuring your application to use Okta.

CSC Agent Password Reset

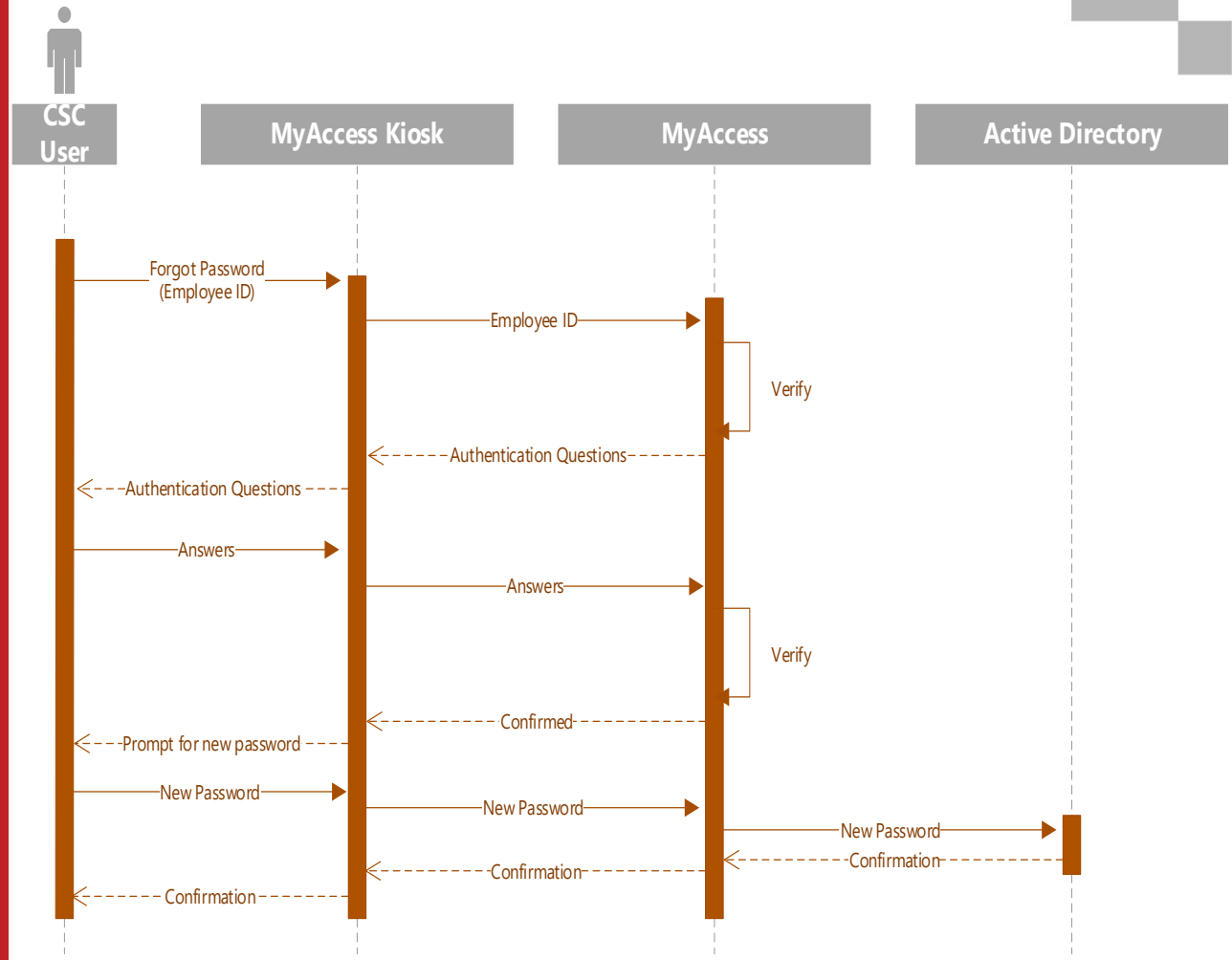
How can CSC Agents reset their password without having to engage support?

Considerations

- Password resets should be simple and straightforward, yet also secure to protect company systems and customer data.

Solution

- The IAM team provides a self-service kiosk at each CSC which allows agents to reset their password based on known information.



APIs

API

What is an API?

The Acronym API stands for Application Programming Interface

In speech, and in the mFin architecture playbook, “API” means three (3) different things*

1. **The interface** - how the client gains access to a function or service and what are the “protocols” and interoperability mechanisms.

   Speech Example: **“This is a REST API not a SOAP API”**

   → **API Interoperability Standards**

2. **The implementation –** this is the part of the API that actually does computation, performs task, e.g. the number crunching. It’s the running code base that affords the service or function.

   Speech Example: : **“The API had to be rewritten as a microservice.”**

   → **API Contributor and Implementation Standards**

3. **The instance –** the instance refers to both the interface and implementation running on some computer or computer(s).

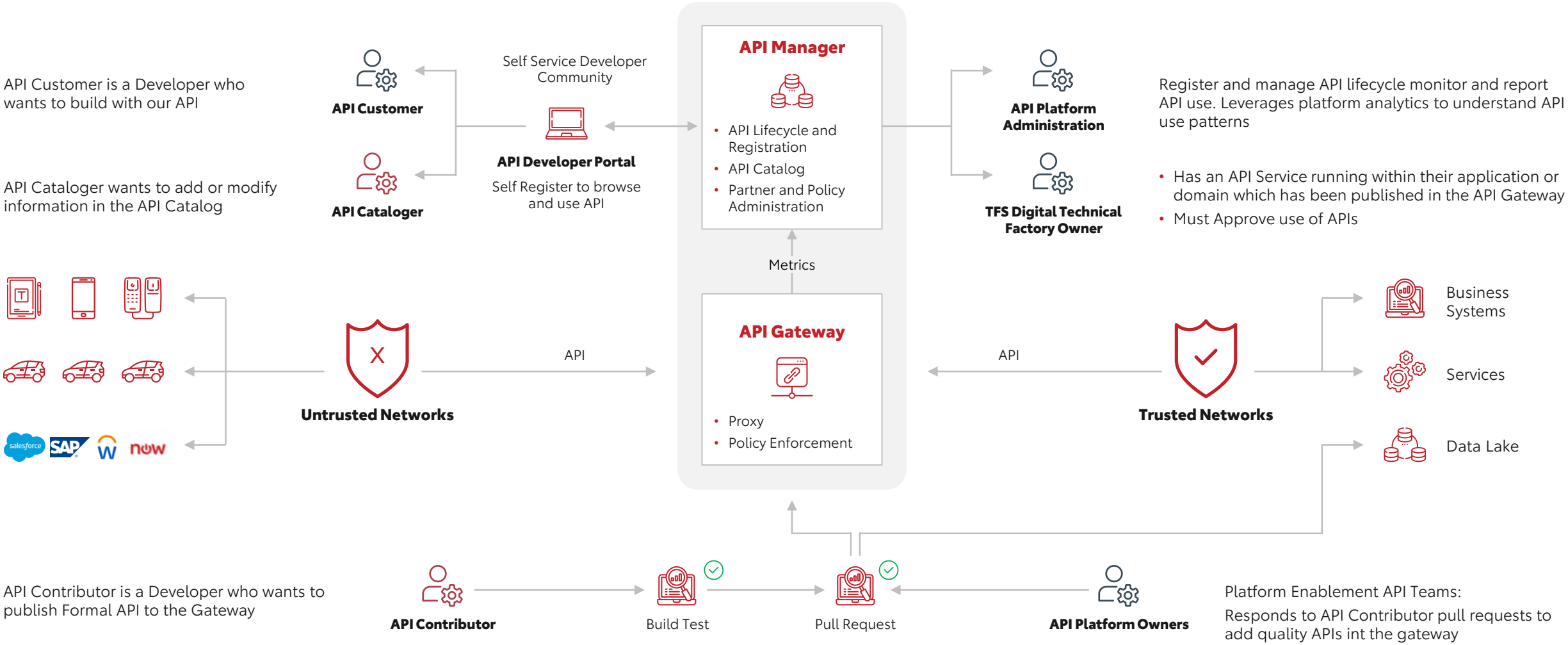
   Some standard API instances would be: development; test; stage; production; disaster recovery

   Speech Example: **“The API in stage is down!”**

   → **API Instance Standards**

TOYOTA FINANCIAL SERVICES

API Services Overview



API Quick Start Guide for Developers

API Question	Response
1 How to Access to API Catalog?	All developers are provided View access to the API Catalog located here API Developer Exchange.
2 How do I contact the API Factory?	For additional details or support, please email the API Services Gateway Factory
3 What is the Security Process to Consume an API?	*Most* API access is provisioned in OKTA System. API consumers will need to register with service accounts in OKTA and use OAuth protocol. The process to request OAuth access is documented here OAuth Server Access
4 What is the API CICD process?	Developers can deploy API's from Github to Mulesoft CloudHub environment using the API Standard CICD process
5 How to develop an API?	Here you can find a complete overview of how to develop API proxies using Anypoint Designer, Exchange, Github, Jenkins and Artifactory etc., New API Development
6 What are the gateway API Proxy Patterns?	API security involves controlling access to your APIs, guarding against malicious message content, accessing and masking sensitive encrypted data at runtime, protecting your backend services against direct access, and other important safeguards. API Gateway Security Patterns
7 What are the API Gateway Design Patterns?	API Gateway design patterns are used to represent some of the best practices adapted for service tier and delivery tier APIs. API Gateway Design Patterns
8 Where to I find information for Developer Onboarding?	Document provides the configuration steps that are required for developer access Developer Onboarding Document
9 Where do I find Developer standards and checklists?	Please refer to Developer standards and Checklist

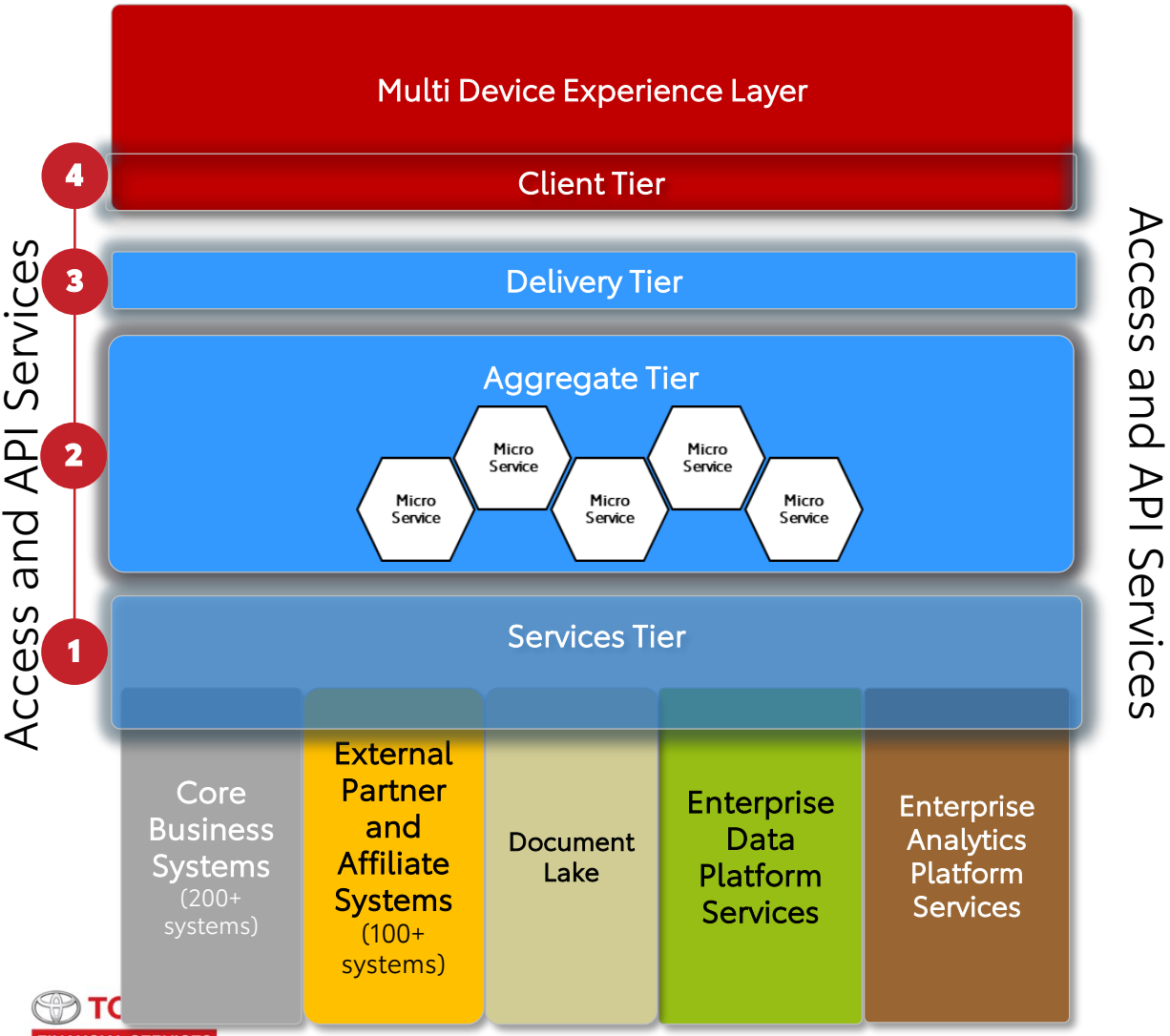
TOYOTA FINANCIAL SERVICES

API Development

- API Tiers
- API Naming Guidelines
- Building Multi-Tenant APIs
- API Security and Token Handling

API Tiers

How API(s) are organized to prevent point to point patterns and to optimize reuse with best performance.



1. The **services tier** connects core business systems, partners and other services into the API ecosystem. It therefore provides the other tiers with the data and functionality they require.

2. The **aggregation tier** serves as a hub for integrating internal and external services using real-time, bidirectional communication. Services in this tier are built as **microservices** following domain driven design (DDD) principles including aggregate design pattern

3. The **delivery tier** is responsible for optimizing delivery of events and data between the platform components. **It receives events from the client** tier to ensure that content is optimized for each device.

4. The **client tier** generates events that leverage the delivery, aggregation, and services layers to generate event-specific responses including responses to invocation of business services.

Our mFin API Tiers were inspired by "The Four-Tier Engagement Platform" by Forrester Research."

Services Tier

DO WE ALLOW SELF SERVICE API CREATION?

- YES. Services Tier APIs are written by YOUR factories. **Your teams are FREE to innovate, create and discover all the 'Service' APIs you can possibly think of!**

WHAT IS THE STANDARD TECHNOLGY PRODUCT

- **Your Factory Standards apply.** We recommend microservice concepts for custom services ☺
- **Legacy applications** may not be ready to expose their services as APIs. We have two standard tools to help solve this problem:
  - Mulesoft Anypoint Platform has hundreds of out of the box connectors
  - IBM zOS connect is used for exposing Mainframe or AS400 services

API STANDARDS

This guide contains the basic quality and interoperability standards that you need to make your APIs work in mFin. Please read this guide!

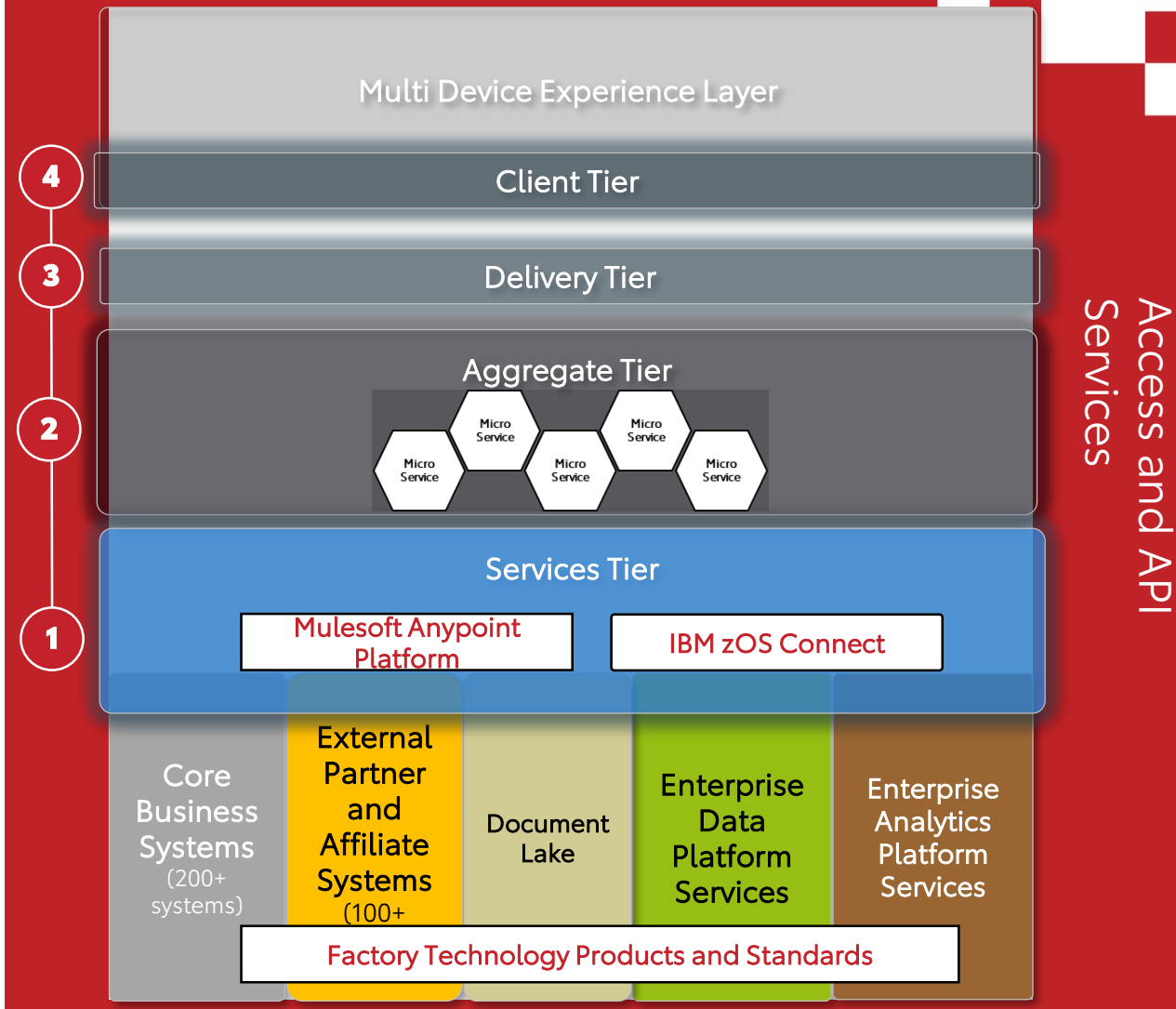
HOW DO OTHERS USE SERVICE TIER API(s)?

To be part of the standard ecosystem all teams will use the **Enterprise Services Gateway** to access your APIs*. **mFin Standards will be enforced this also includes rules about the Aggregate Tier API(s)**

*This does not apply to API(s) which are internal to your systems (private or protected)

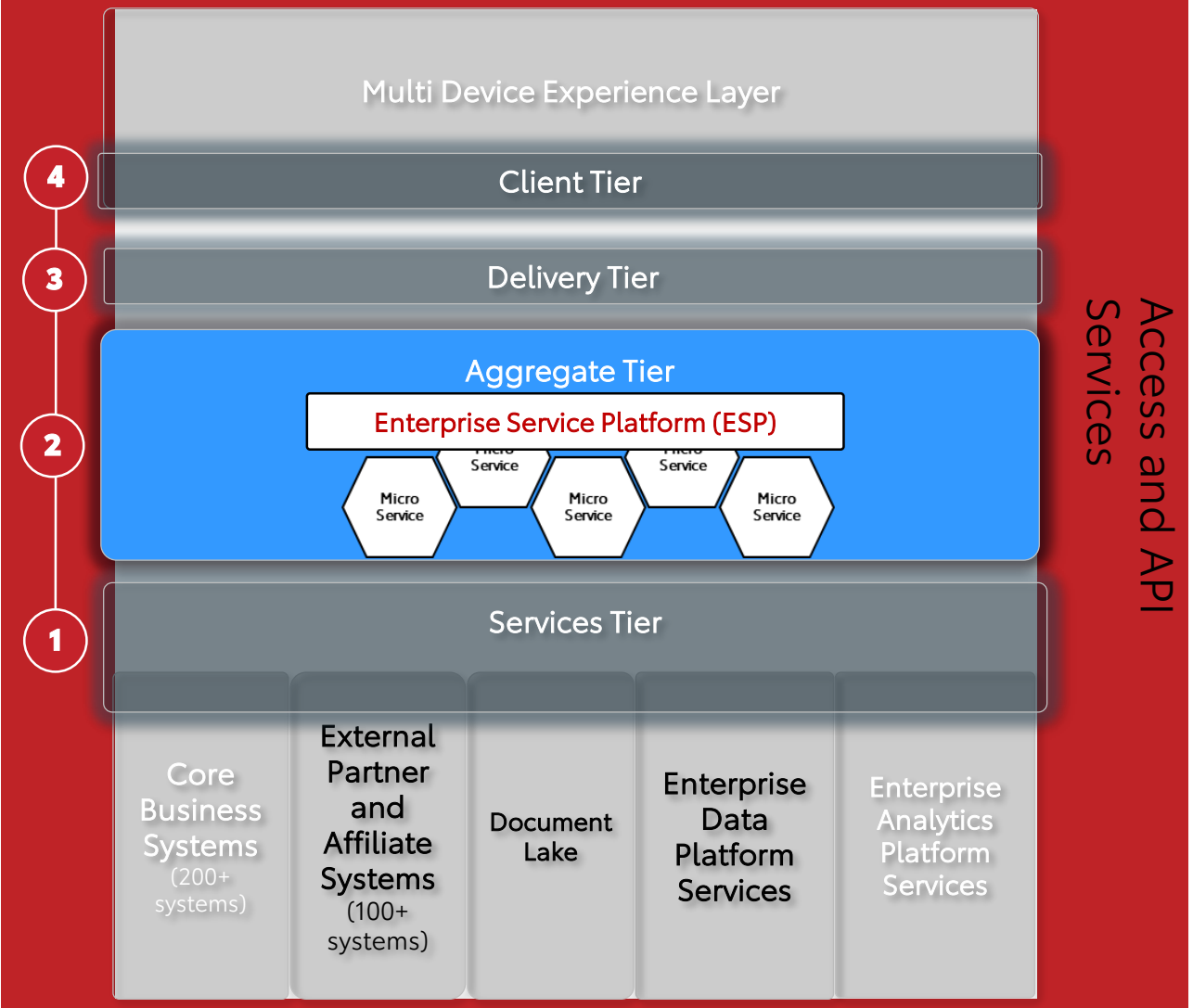
This is Your Tier!

Your Factory writes and provides the API(s)



Aggregate Tier

This tier is not self-service for API creation



DO WE ALLOW SELF SERVICE API CREATION?

NO. This TIER is NOT SELF SERVICE for API CREATION. THERE ARE NO PLANS FOR SELF SERVICE Aggregate Tier API(s)

WHAT IS THE STANDARD TECHNOLGY PRODUCT?

- The Enterprise Services Platform (ESP) is the standard product for Aggregate Tier APIs
    - APIs for Aggregate are ONLY deployed within the ESP

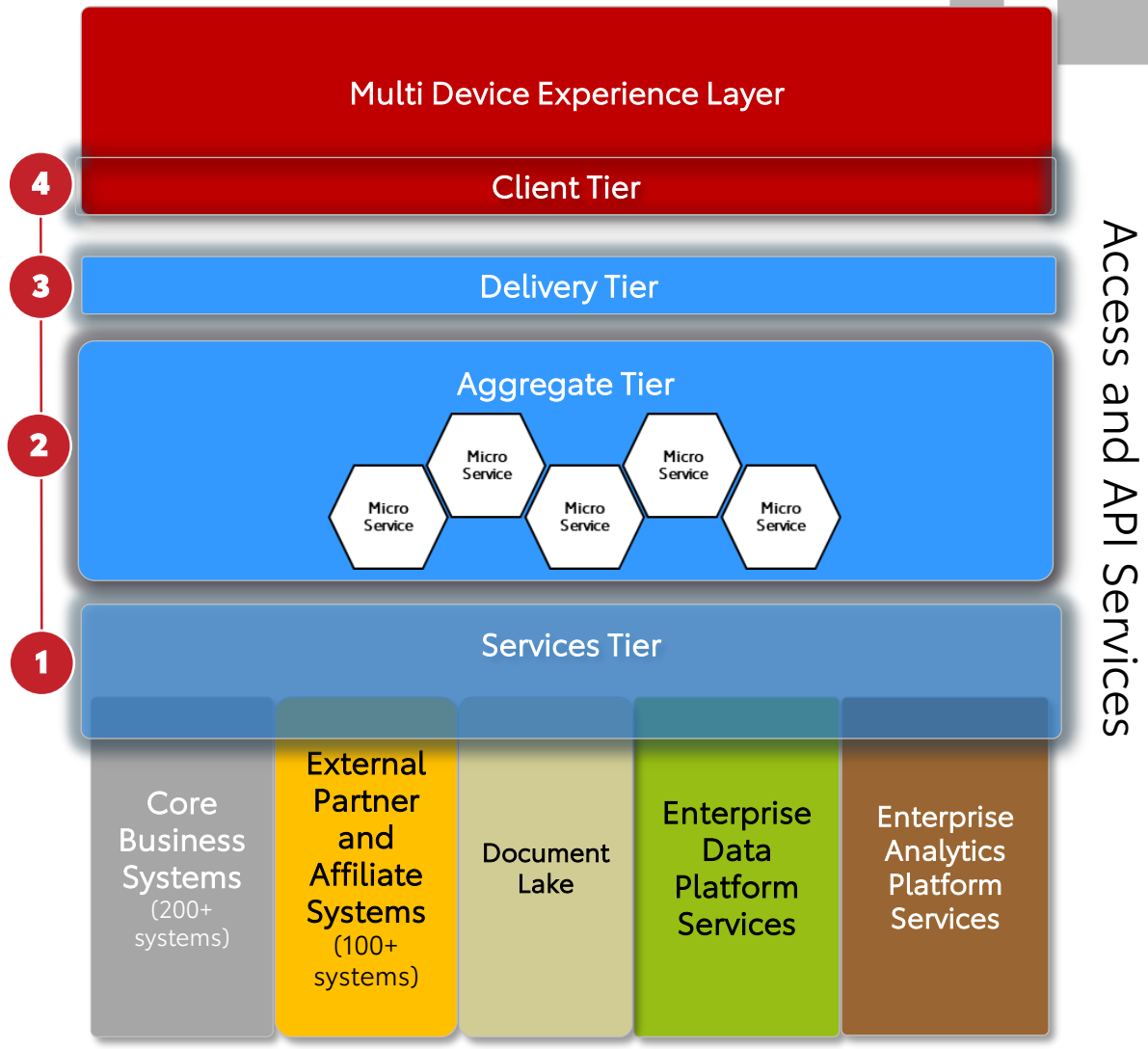
The Aggregate Tier has very specific Enterprise Integration Functions

1. Aggregate Tier provides microservices to allow **real time access to Enterprise Data Platform information and MDM systems**
2. Aggregate Tier is used to **ENSURE consistency for business processes which span multiple systems of record or master data systems**. This applies to internal and external services which must be kept synchronized for a business process.
    - Human or USER workflow capability is NOT within the Aggregate Tier. Human workflow is within Core Business Systems or within the applications provided in other Tiers
3. Aggregate Tier contains the enterprise event streaming systems used for streaming business events and for supporting **observer and saga patterns between enterprise systems and external partners and affiliates**.
    - Infrastructure events for monitoring or alerting are NOT part of the Aggregate Tier services however such infrastructure events may originate in this Tier.

API Implementations by Tier

1. **Service Tier** APIs ARE Written by **Factory Teams.**
   - Your Factories are FREE to innovate and create all the 'Service" APIs you can possibly think of. Go crazy. Be innovative. Solve business problems!
2. **Aggregation Tier** APIs ARE Written by **Enterprise API Domain Factory Teams.**
   - Aggregation Tier are NOT PLANED for "Self Service". We have no plans of enabling aggregation tier APIs as self service.
3. **Delivery Tier** APIs ARE Currently Written by the the Platform Enablement API team.
   - Delivery Tier APIs will be available as SELF SERVICE in FUTURE.
4. **Client Tier** use of APIs is Written by Factory Teams.
   - Use of APIs is 100% self service
   - Your factory developers have access to the Enterprise Services Catalog to search for available APIs
   - More Self-service tools will be added each month to streamline security and attestation

Which factories write the actual API implementations for mFin services?



Delivery Tier

DO WE ALLOW SELF SERVICE API CREATION?

- YES. COMING SOON! Your factories developers will be able to use Mulesoft Anypoint Platform to create your own Delivery Tier APIs and policy enforcement points.

- **Delivery Tier** APIs **ARE Currently created by the** Platform Enablement API team. This includes **Back Ends For Front Ends (BFF).**

WHAT IS THE STANDARD TECHNOLGY PRODUCT

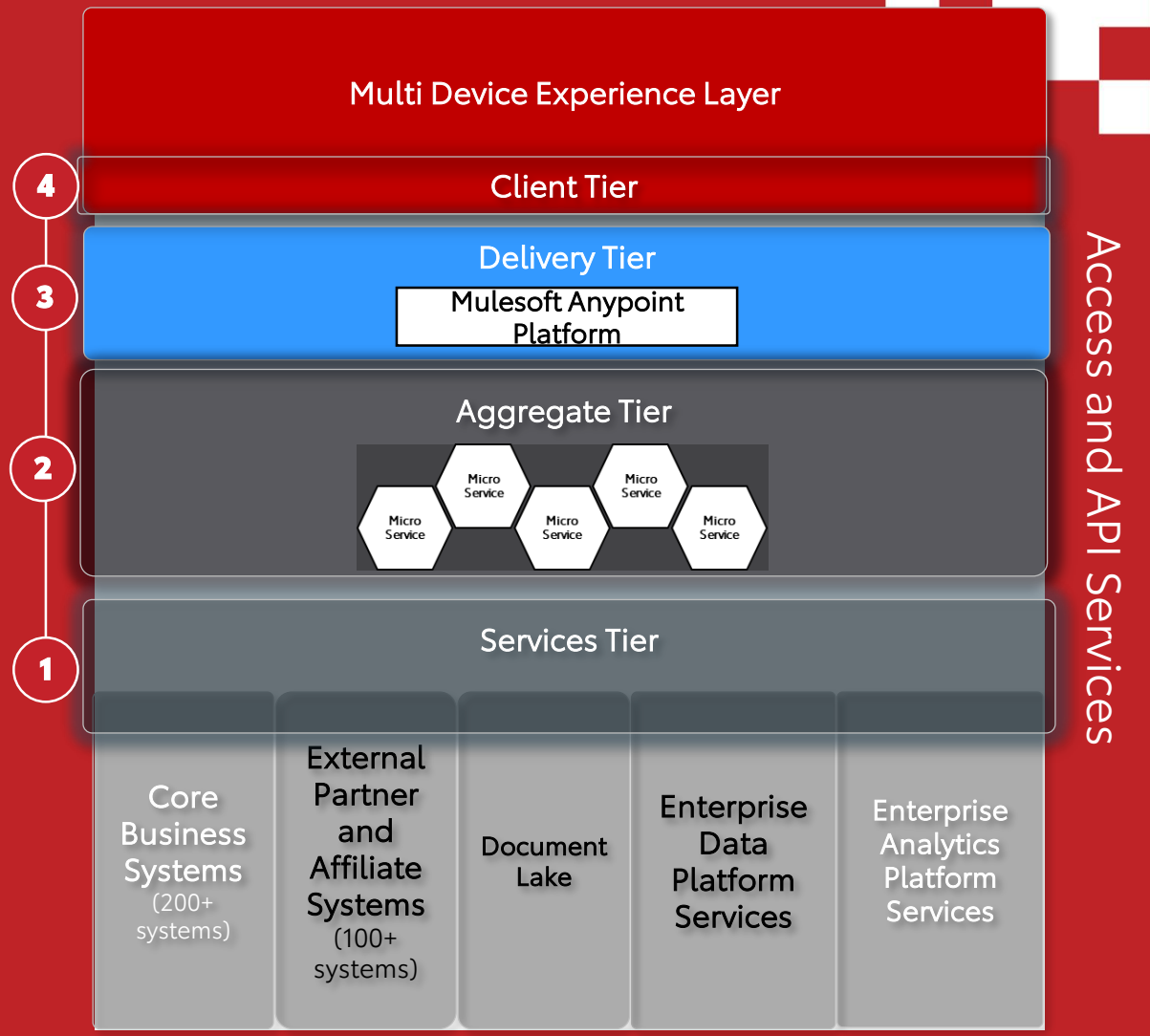
- Mulesoft Anypoint Platform is the standard technology product to be used for implementations of Delivery Tier APIs

Client Tier

SELF SERVICE API USE?

- YES. Client Tier use of APIs is 100% SELF Service!
- Your factory developers have access to the Enterprise Services Catalog to search for available APIs
- More Self service tools will be added each month to streamline security and attestation

Client Tier is 100% self service
Delivery Tier is currently provided by Platform enablement API team



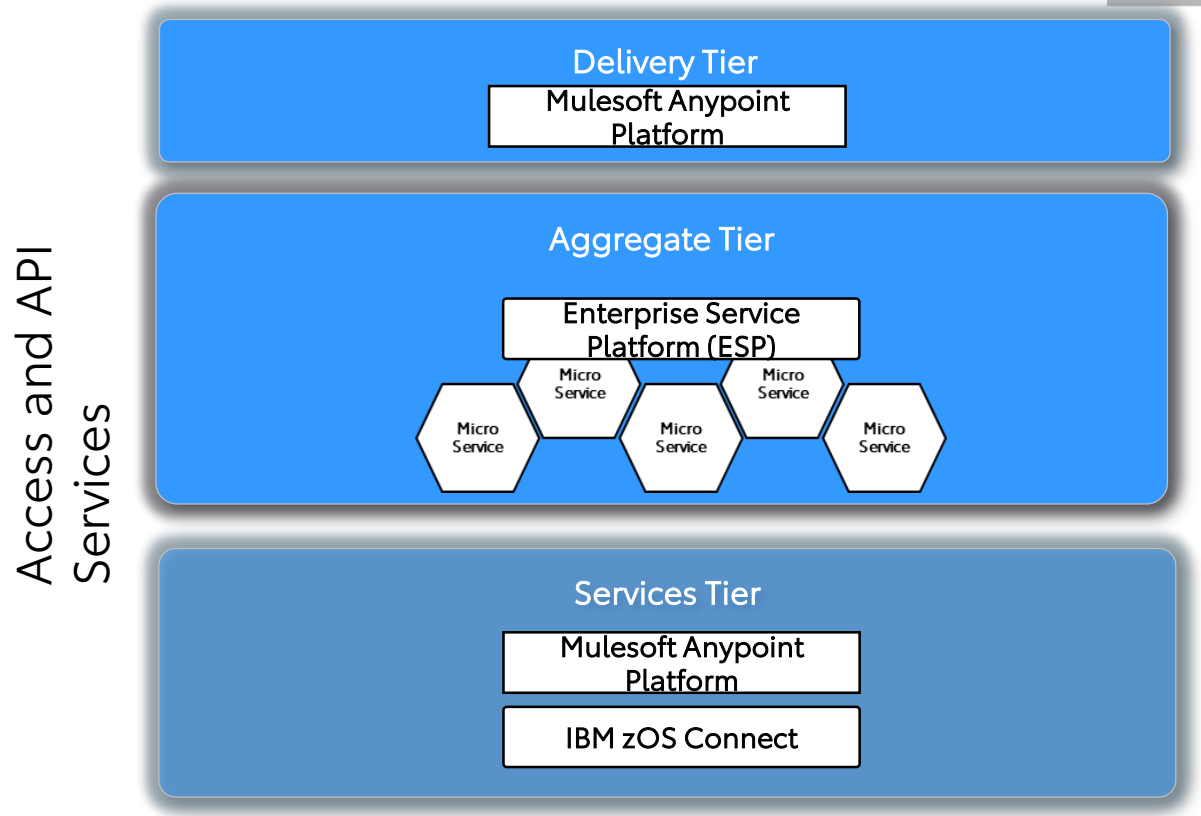
API Tool Standards

API solution Style

Gateway	Delivery Tier: **Mulesoft Anypoint Platform, Mulesoft Anypoint Exchange Plugin**
Gateway Plugin	
Gateway 3rd Party Plugin	
Vendor Application	Services Tier: **IBM zOS Connect** (Mainframe/AS400) OR **Mulesoft Anypoint Platform, Mulesoft Anypoint Exchange Plugin**
Microservice	Data Services and Aggregate Tier: **Enterprise Service Platform (ESP)**: AWS, Kubernetes, Java, Spring, Kafka, MongoDB, Redis, Elastic Stack Heroku: Special purpose for Salesforce integration and rapid proof of concept

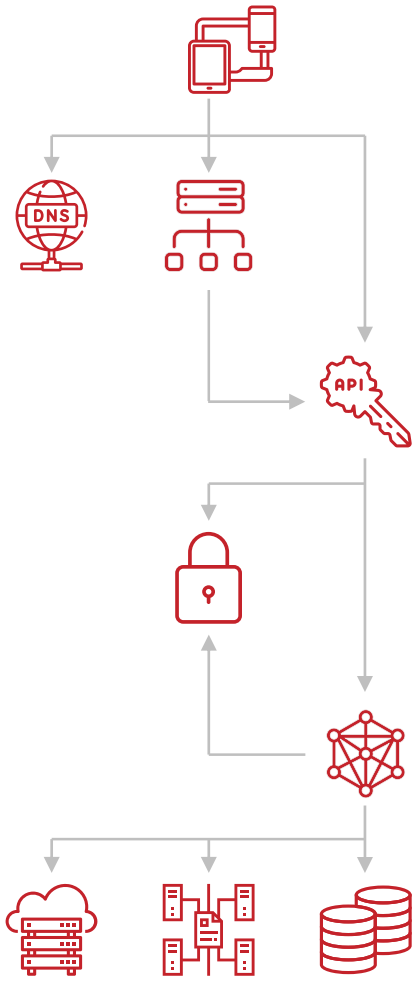
and their Solution Styles

Tool Standards



If you are building APIs without these tools, then your factory is not following the API standards!

API Interoperability Standards



API Consumers
- Mobile Apps
- Single-Page Apps (web)
- Third-party services

Edge Services
- Apply service-naming conventions
- Map FQDNs to TenantIDs

API Gateway Services
- Apply service-naming conventions
- Map FQDNs to TenantIDs
- Assert validity of credentials

Identity & Access Management
- Construct Bearer Tokens
- Support OAuth2 flows
- Provide Token Exchange API
- Provide Introspection APIs

Microservices
- Utilize common libraries for:
  - Token Exchange (e.g. in lieu of Service Accounts
  - Token Relay
  - Resource Redirection

Resource Dependencies
- Tenant-isolation (topics, databases, etc.)
- Tenant-aware or –dedicated (e.g. webservices)

mFin applications are a mix of services.

What are the standards which should be followed to allow them to work together seamlessly?

Acceptance Criteria

- Naming should be standardized.
- Tenant isolation should NEVER be capable of being defeated
- Tenant ID should be available EVEN if the native service was not created with Tenant ID
- The system should not require SERVICE Accounts which defeat role-based access controls
- Downstream services should have strong isolations between tenants for all resources including databases, other APIs and streaming events and topics

How

- API Naming, security and interoperability standards

Consequences

- Supports an identity-based audit mechanism that can trace through layers of the enterprise systems
- Support for Attribute-Based Access Control (ABAC) based on end-user identity at the API.
- All APIs have access to Tenant ID
- Identity of a requestor is securely passed to an API.
- API can verify that requestor is authorized to perform an operation.
- APIs cannot be fooled by fake interoperability tokens
- APIs can introspect on a request to obtain additional metadata on the originating user
- Microservices provide appropriate isolation of resources (databases, other APIs and event topics)

TOYOTA FINANCIAL SERVICES

mFin API Naming Guidelines

For branding and proper digital experience we need a consistent address and naming structure for our APIs. How do we prevent API(s) from looking like a hodgepodge collection of other companies' services?

Considerations

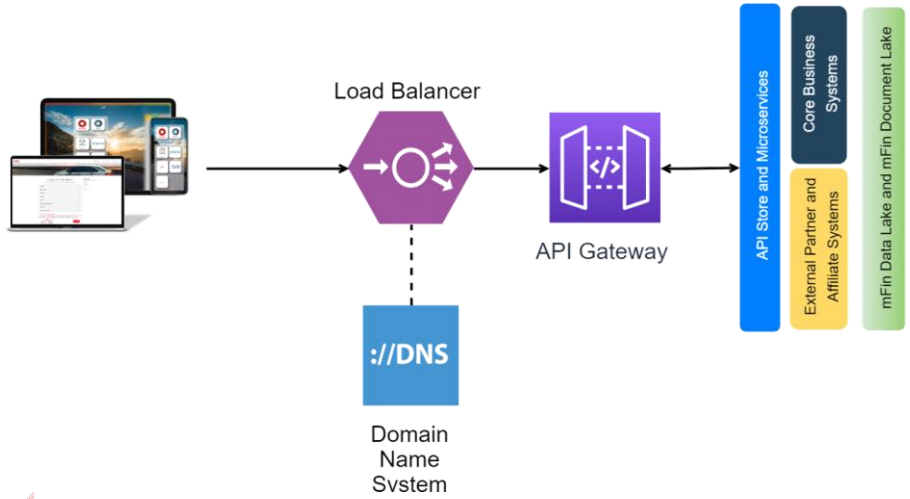
Core business systems and applications will be located in different clouds with different web addresses for their APIs and API gateways

Solution

Utilize API Gateway Pattern and Domain Name System (DNS) to abstract the maze of underlying services

- Naming scheme *rooted* at the concepts of *Tenant, Service and Resource*.
  - Basic form: *service.tenant.tld/resource*
  - Simple: **`auth.toyotafintech.io/user`**
  - Extended: **`rates.leasing.toyotafintech.io/tier/1/terms`**
- Follow REST resource naming conventions & modeling
  - Context path must be made up of ***NOUNS*** *and* ***free of VERBS***.
  - Build on HTTP Verbs for requested operation (eg `GET`, `POST`, `DELETE`, etc)
  - Example: `Due Date Change`
    - Bad: **`POST /performDueDataChange`**
    - Good: **`POST /dueDateChange`**
      - Creates a new event (model) to represent a Due Date Change "request"

… or why All API(s) via the gateway?



`service.subdom.dom.tenant.tld/ctx/subctx/resource`

more specific — more specific

Service Location

- *service*-based taxonomy
- interpreted by network devices
- elasticity based on infrastructure organization

Resource Addressing

- *business model*-based taxonomy
- interpreted by the app
- elasticity based on model/entity complexity

Service	Implementation	Description
origination.toyotafintech.io	*.sagent.com	Originations
insurance.toyotafintech.io	*.stoneeagle.com	Insurance products, servicing, & management
customer.toyotafintech.io	*.salesforce.com	Customer management
employee.toyotafintech.io	*.workday.com	Workforce management

mFin API Tenant Service

mFin applications are a mix of services. Not all resource models contain tenant id.

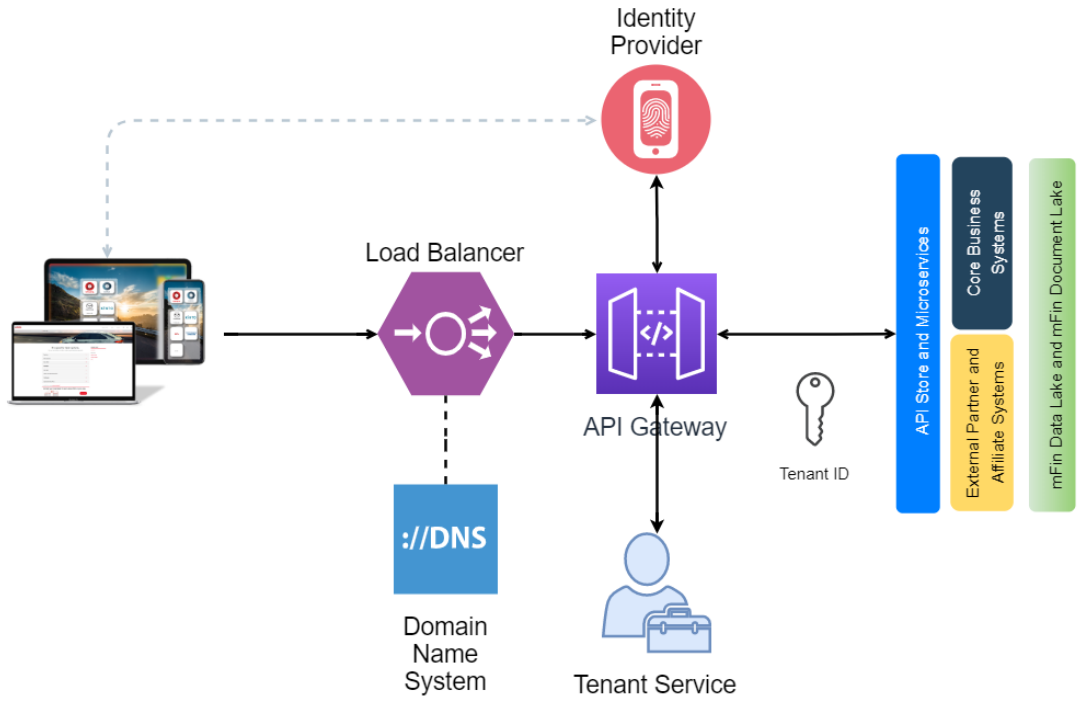
Consideration

- APIs need to work in process chains and all API in the chain will need to have tenant id available.
- It should be technically impossible for a user to be aware of another Tenant and its users. The resource model should be as clean as possible to leave no trace of tenants for hacking, abuse or suggestion of a conflict of interest.
- The existence of other Tenants should therefore not leak into the API model.

Solution

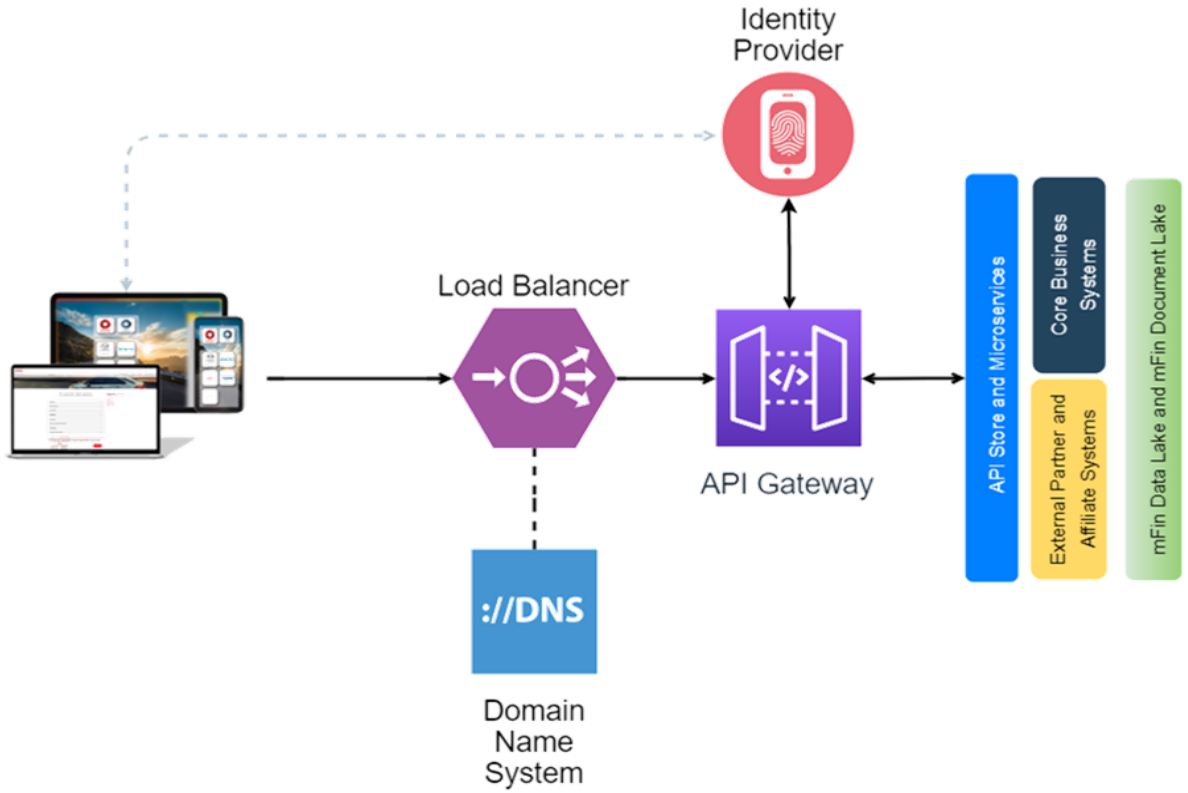
- API Tenant service will inject appropriate tenant ID into API requests for downstream systems.

… another reason to use the API Gateway for mFin



mFin API Security

How to secure APIs used by People Accounts and Services Accounts



API(s) are the gateway to sensitive information. How do we provide high security including both authentication and authorization; single sign on; and allow for proper attestation of security rights for API(s)?

Considerations

Core business systems and applications will be located in different clouds with different web addresses and have different security mechanisms for their APIs and API gateways

Solution

API Gateway Pattern and Identity Provider (IDP)

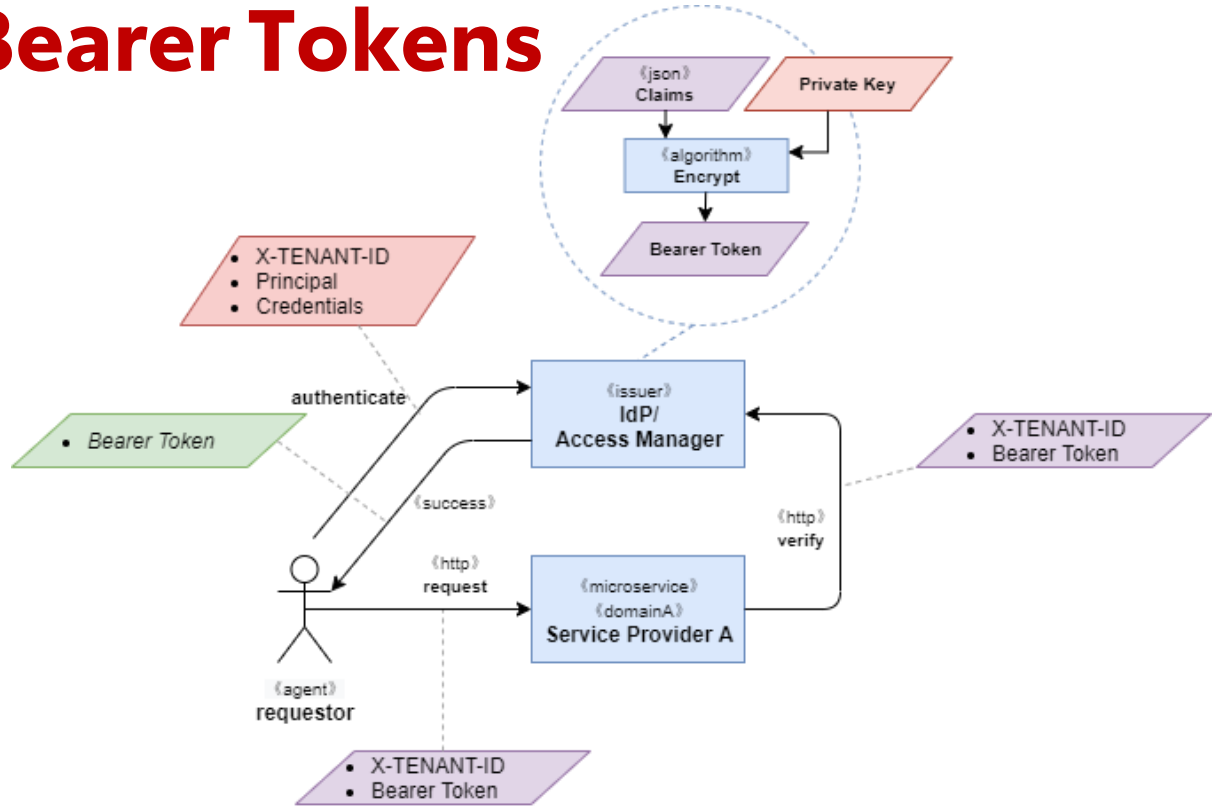
Two Variations:

1) System to System (Service Accounts)
IDP: Centrify
Pattern: API's are Secured using OAuth 2.0 and Mutual TLS.

2) Customers and Dealers (People Accounts)
IDP: ForgeRock
Pattern: API's are Secured using OAuth 2.0 and SSO.

Coming Soon: API Provisioning via SailPoint

Bearer Tokens



JSON Web Token (JWT)

Header `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX`
`VCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiw`
Payload `ibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4`
`iOnRydWV9.TJVA95OrM7E2cBab30RMHr`
Signature `HDcEfxjoYZgeFONFh7HgQ`

Payload example

Token Issuer
Subject
Audience
Expiration
JWT UUID
TenantID

```
{
  "iss": "https://auth.toyotafintech.io/",
  "sub" : "https://idp.toyotafintech.io/user/2779763313420",
  "aud" : [ "ot", "cep", "dfs", "aem" ] ,
  "exp" : "1579762695777",
  "jti" : "B3hn00CAd86KNdpruzgbUg1VsR8NxAxb",
  "https://toyotafintech.io/tenantId" : "0001",
  ...
}
```

Service Providers often need to verify that a requestor is authorized to perform an operation.

How do you securely communicate the *identity* & *attributes* of a requestor to Service Providers?

Considerations

- Tamper-resistant; authenticity of "proofs" should be easily verifiable.
- The mechanism must allow for *custom* payloads within the token.
- The mechanism must be extensible (additional attributes in the future)

Solution

- Use an IdP/Access Manager to issue tokens for requestors who successfully authenticate.
- Provide an API for *validating* issued tokens.

Implementation

- JSON Web Tokens (**RFC 7519**) is an open standard for representing claims (attributes) between two (2) parties.

Consequences

- Identity of a requestor is securely passed to a service provider.
- Service Provider can verify that requestor is authorized to perform an operation.

Bearer Token Relay

How will applications and services be able to identify users and authorize use of secured resources from behind the gateway?

Considerations

- The power of API style development is to promote innovation and reuse of service and to compose them in new and inventive ways.
- “Beyond” single sign on. Downstream services must be capable of fine grain authorization.

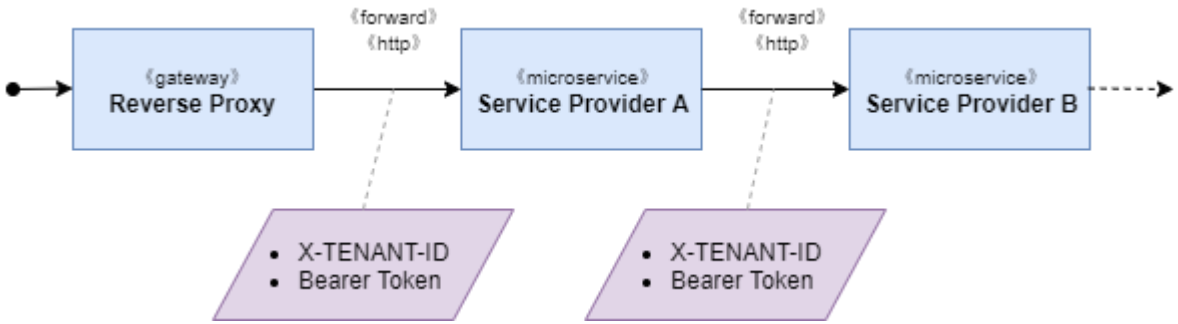
Solution

- Service providers reuse the access token to authenticate with upstream services they depend on, thus propagating the caller security context further in the infrastructure.
- Each upstream service is OAuth2 capable and can process JWT tokens

Consequences

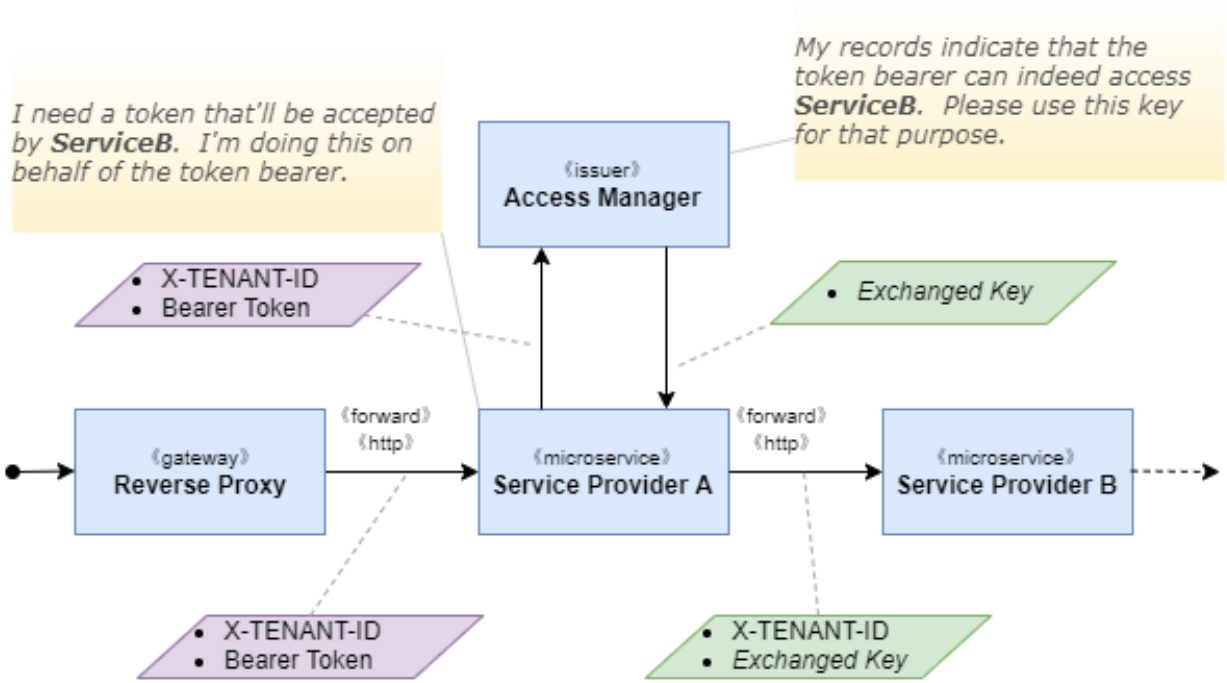
- Support for Attribute-Based Access Control (ABAC) based on end-user identity at the API.
- Avoidance of generic service accounts, when connecting upstream of an API - The same access token received by API (or ResourceProvider) is utilized when calling a dependency.
- Supports an identity-based audit mechanism that can trace through layers of the infrastructure

… Propagate the end-user security context beyond the API Gateway



Token Exchange

… how to get rid of the "service accounts" in APIs



How do you authorize an authenticated user to invoke another service in a *different* security domain?

Considerations

- User has a secret which is exchanged

Solution

- The Access Manager should provide an API for allowing bearer tokens (from one domain) to be exchanged for valid keys to another domain.
- The Access Manager can determine if the requesting client (holder of the bearer token) is *allowed* to invoke the operation being requested.
- If the Access Manager approves of the *exchange* the requesting Service Provider can now perform the request *on behalf of* the client in the origin domain.

User/Token Introspection

Provide a means for an API to obtain additional metadata on a User or Token during access decisioning

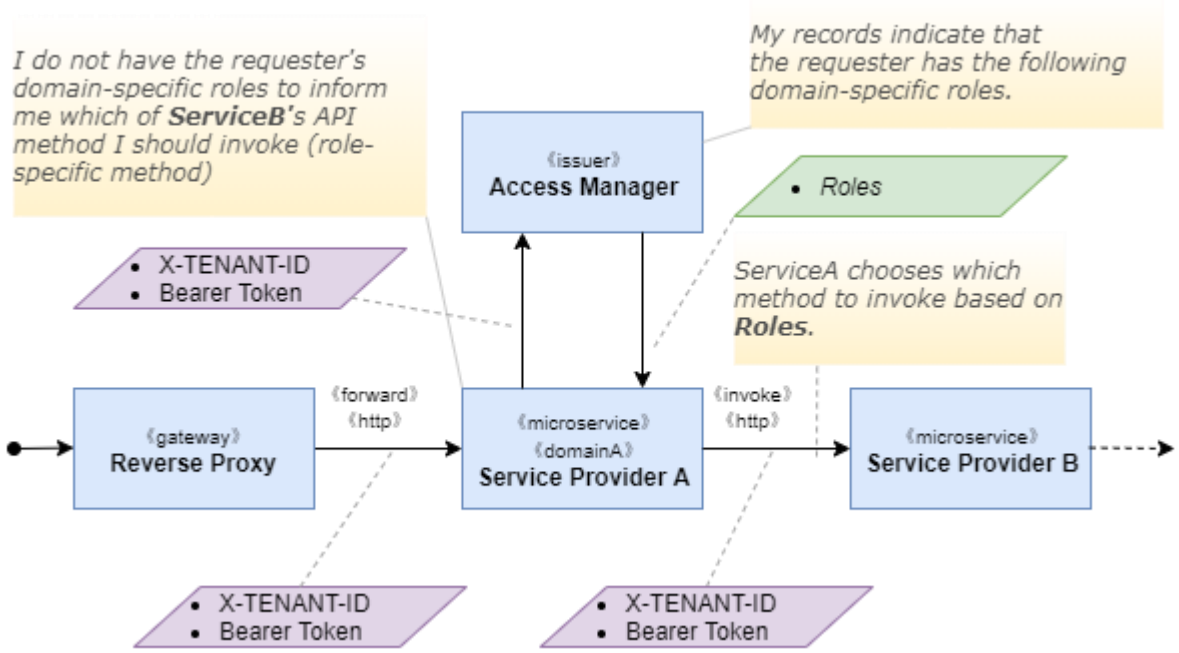
How does an API obtain required information not carried in the Access Token payload?

Considerations

Required info was not part of original requested scope/s during authentication (ie., this is a scope expansion)

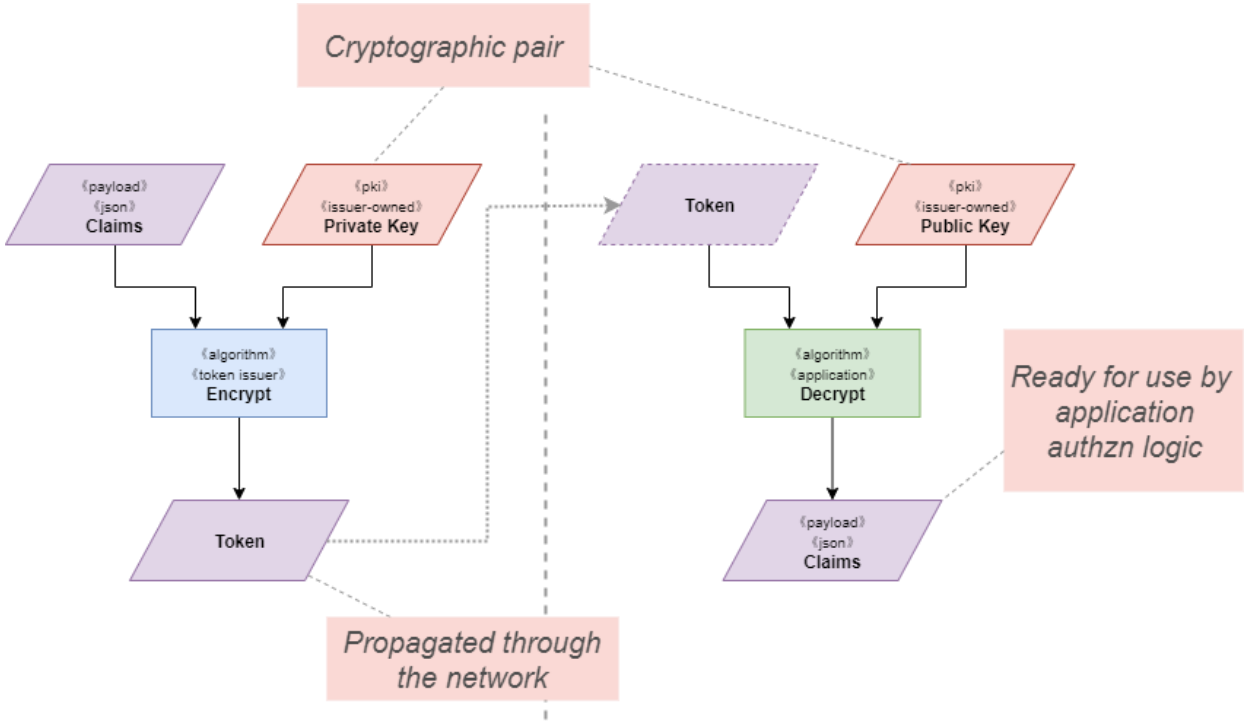
Solution

OAuth2.0 specification allows for "User Introspection" and "Token Introspection" endpoints that API can invoke to obtain additional information



Independent/On-the-Spot Token Validation

How to validate tokens without redirecting to the identity provider



Token relay patterns create a risk that exchanged tokens may not be valid. How can an API validate JWT tokens with minimal overhead to all the requests passing through API gateway?

Considerations

- Must be fast
- Must be 100% accurate

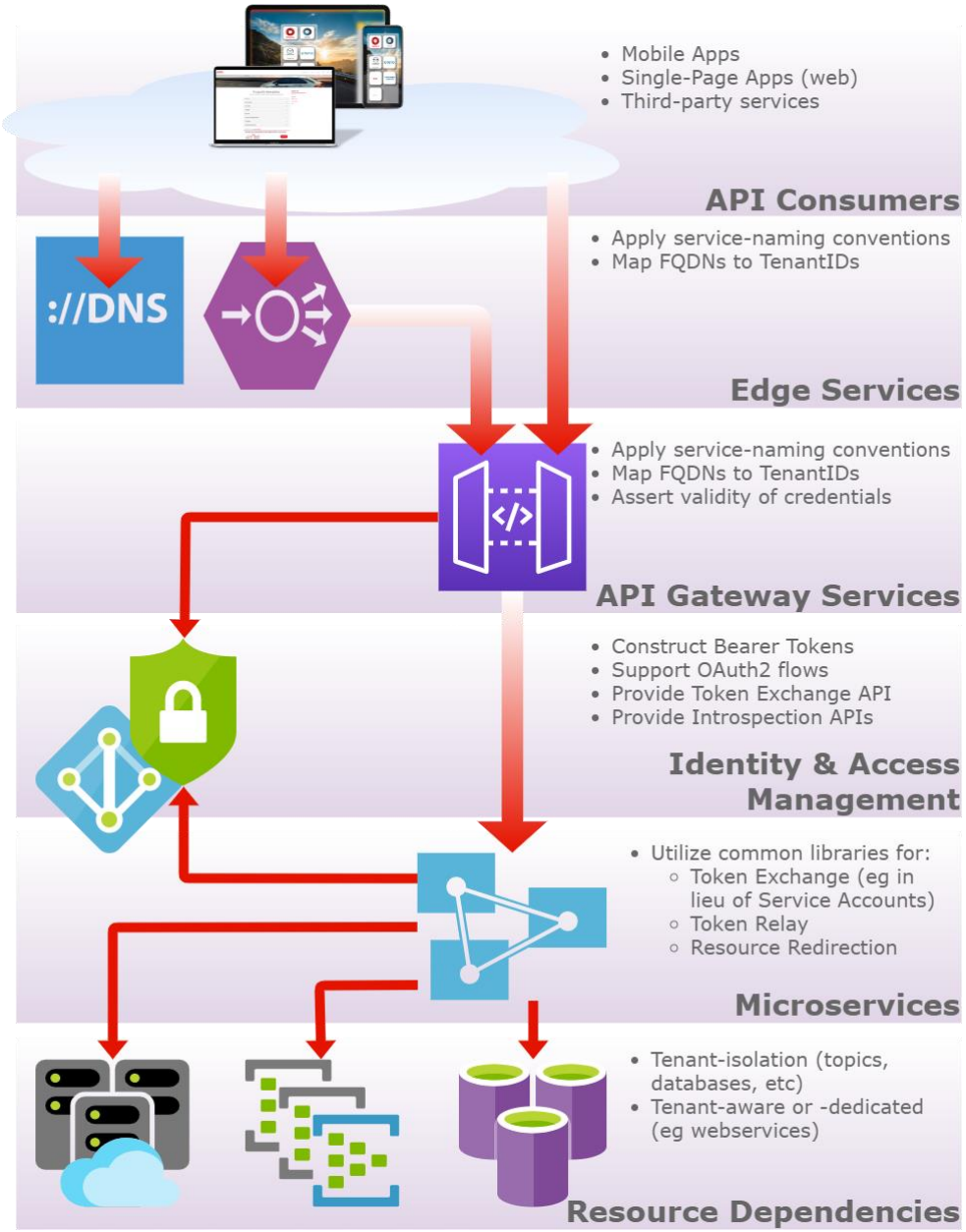
Solution

- Use *asymmetric algorithmic functions* to *encrypt* or *sign* tokens.
- Use the (disseminated) public keys of the issuer (ie encrypter/signer) to assert a key's integrity.
- Security is further enhanced by including an expiration (TTL) within a token's payload to restrict validity within a certain time period
  - ➢ Tampering with any part of the token is easily discovered
  - ➢ Tokens are guaranteed to go stale due to TTL
    - This forces revalidation of credentials from time to time.

Putting it all together

1. API Consumers
   - REST is the primary interaction style with services
     - Use HATEOAS (https://restfulapi.net/hateoas/) for navigation and model relationships.
   - JSON as the notation of choice to express business data
   - HTTP as primary protocol; non-functional data elements should be able to be satisfied by this layer.
2. Edge Services
   - Map FQDNs to TenantIDs
   - Apply the service-naming conventions
3. API Gateway Services
   - Map FQDNs to TenantIDs
   - Apply the service-naming conventions
   - Provide Tenant Management Services
4. Identity Provider and Access Management
   - Support Bearer Tokens containing *standard* claims
   - Support Token Exchange
   - Support Introspection Services
   - Support for OAuth2 flows
5. Microservices
   - Build mFin (aspect-oriented) libraries for
     - Database Connection Factories
     - Kafka Connection Factories
     - REST Client Connection Factories
     - Token support (relay, exchange, introspection)
6. Resource Dependencies
   - Isolated/dedicated per Tenant.

...let's go build this!



API Management

- API Catalog
- API Checklists & Requirements
- API Core Products

API Catalog Factory Responsibility

How can we scale API development to ensure it is not a bottleneck? How can we reduce coupling between teams and still promote reuse?

Considerations

- API(s) should be reused! API should carry as little "coupling" as possible. API(s) should be produced at speed with high quality

Solution

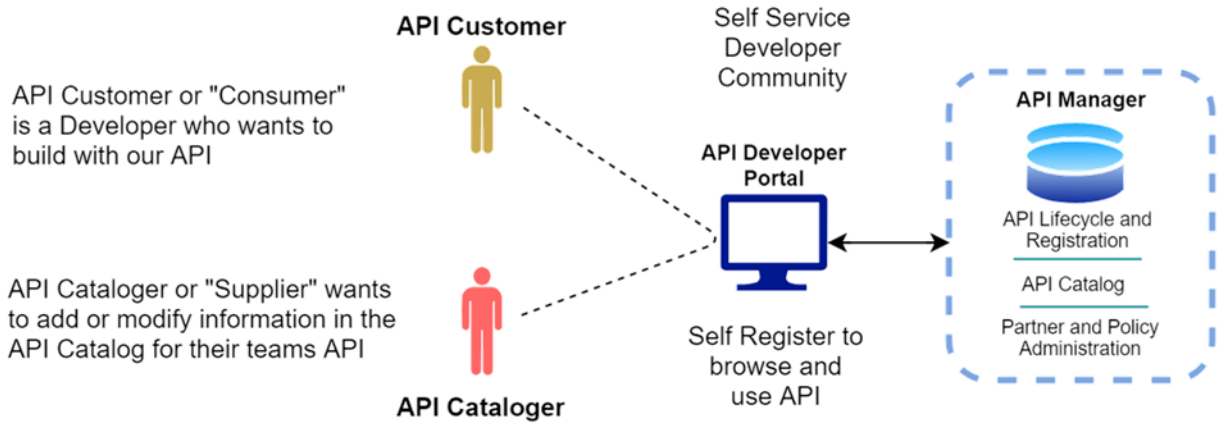
- API Catalog and Self Service

- Productize and Publish
- Encourage collaboration
- Drive self-reliance
- Improved Speed and Quality
- Improve efficiency and accelerate Innovation

Your Factory is 100% responsible for your APIs

- Your Factory is 100% responsible to publish and maintain all APIs provided by your systems and business partners in the API Catalog. The API developer portal should contain all information for others to use your APIs.

- API customers (the rest of the enterprise) should use the **API Catalog and Portal** to browse obtain API specifications and to provide feedback about APIs via crowdsourcing tools in the Exchange and Github.



Self service roles for all factories

Re-sequence

API Customer or "Consumer" is a Developer who wants to build with our API

API Customer

API Cataloger or "Supplier" wants to add or modify information in the API Catalog for their teams API

API Cataloger

Self Service Developer Community

API Developer Portal

Self Register to browse and use API

API Manager

- API Lifecycle and Registration
- API Catalog
- Partner and Policy Administration

API Manager

- API Lifecycle and Registration
- API Catalog
- Partner and Policy Administration

API Platform Administration

Register and manage API lifecycle monitor and report API use. Leverages platform analytics to understand API use patterns

TFS Digital Technical Factory Owner

- Has an API Service running within their application or domain which has been published in the API Gateway
- Must Approve use of APIs

API: Consumer, Catalog and Factory Roles

Acceptance Criteria

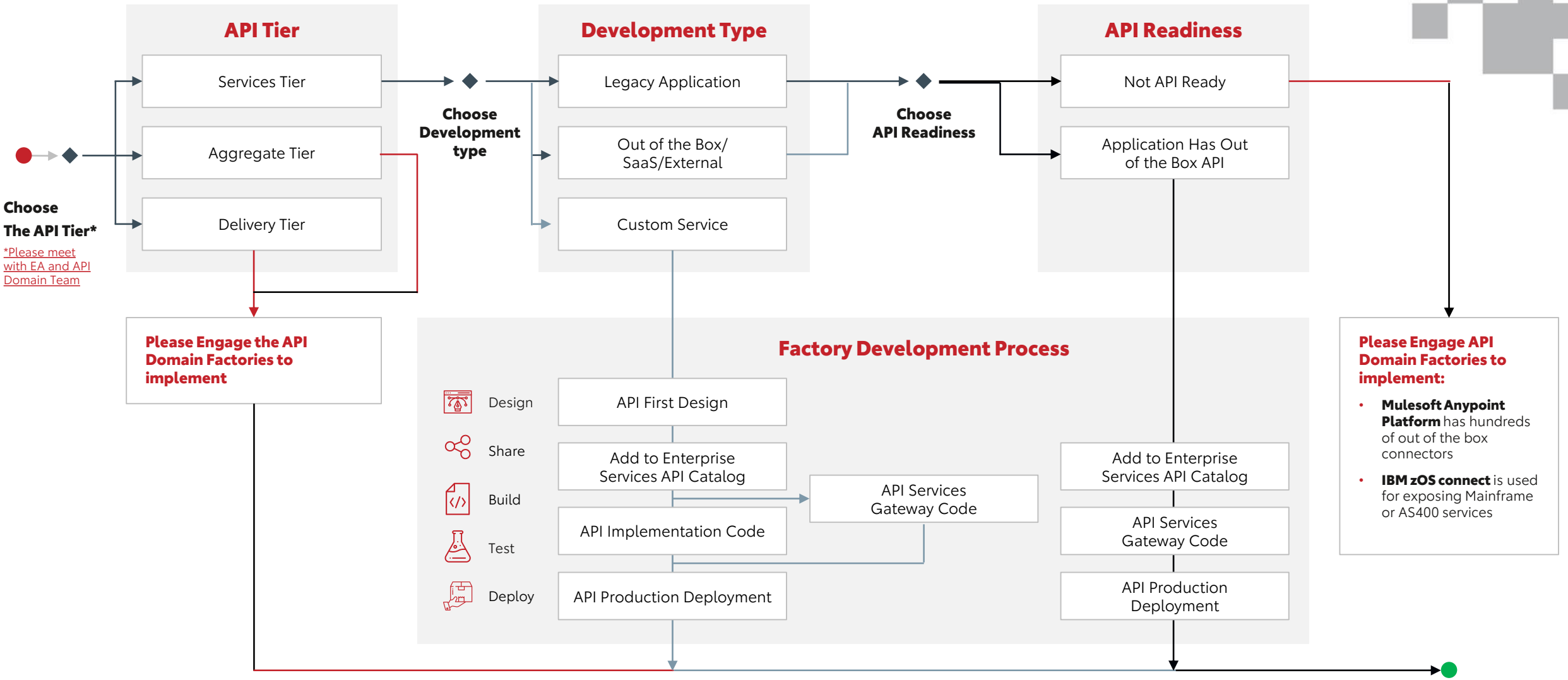
- Scaled APIs development for quick turnaround for high quality APIs
- Reduce coupling within APIs across teams

How

- API Catalog and Self Service
  - Productize and Publish
  - Encourage collaboration
  - Drive self-reliance
  - Improved Speed and Quality
  - Improve efficiency and accelerate Innovation
- API(s) should be reused
- Factory Developers responsible to develop, publish and maintain APIs for their system in the API catalog
- The API developer portal should contain all information for others to use the APIs.
- API customers (the rest of the enterprise) use the API Catalog and Portal to browse obtain API specifications and provide feedback about APIs via crowdsourcing tools in the Exchange and Github

TOYOTA FINANCIAL SERVICES

API Contribution Process



API Checklists and Requirements*

 **Design** |  **Share** |  **Build** |  **Test** |  **Deploy**

API First Requirements

Domain Modeling
Microservice API development requires Bounded Context Domain Modeling. Please see the section later in this in this guide (API Microservice Domain Modeling).

Source Control
Repository is created for your work with a "readme" (using Markdown). All code, specs, designs etc. must be in source control.

Create API Specification
API specification is created using Open API Specification (Swagger) or RESTful API Modeling Language and follows all interoperability requirements of this guide.

API Design Specification
Write down: SLA and Performance requirements; Max concurrent transactions; RTO and RPO; Usage limits; Security Roles and policy.

Create/Update API Catalog
Your APIs MUST be entered into the TFS Enterprise Service Catalog (API Catalog).

Deploy Mock Services
Your client and customers are able to start using your API via Mock Services.

API Development Requirements

Interoperability and Security
Please see the section later in this in this guide for interoperability requirements. (Naming, Tenant, Security, Token Relay, etc.).

Complete Code Implementation
Note you have TWO code bases for a custom service Tier API:
- API Services Gateway Code (Mulesoft Code)
- API Implementation Code

Unit Tests
80% Minimum of all your code must be covered by unit tests with 100% of written tests passing.

Automated Build
100% Automated Build and Deployment for:
- API Services Gateway Code (Mulesoft Code)
- API Implementation Code (Microservice Code) Including all databases

Automated Integration Tests
An automated test must be created to test all aspects of your API including security using (Postman (legacy), Ready API (new standard).

Deploy To Test / Stage Environment
Deployment is 100% automated including gateway policy, implementation code, and data bases.

API Production Requirements

System Testing
Functional testing and UAT testing is performed as part of the larger systems in which your API is used.

Performance Testing
Less than 500 millisecond response time under design load (the number of concurrent transactions specified in the API design).

Logging
Developer logging is required. All PII data MUST be obfuscated in Logs.

Monitoring
Your API must have a monitoring harness and be visible in the enterprise API Gateway analytics.

Correlation
Your API logging includes correlation IDs in logs. Please see interoperability requirements for details.

Deployment to Production
Deployment is 100% automated including service gateway code, Implementation code, and data bases/tech components.

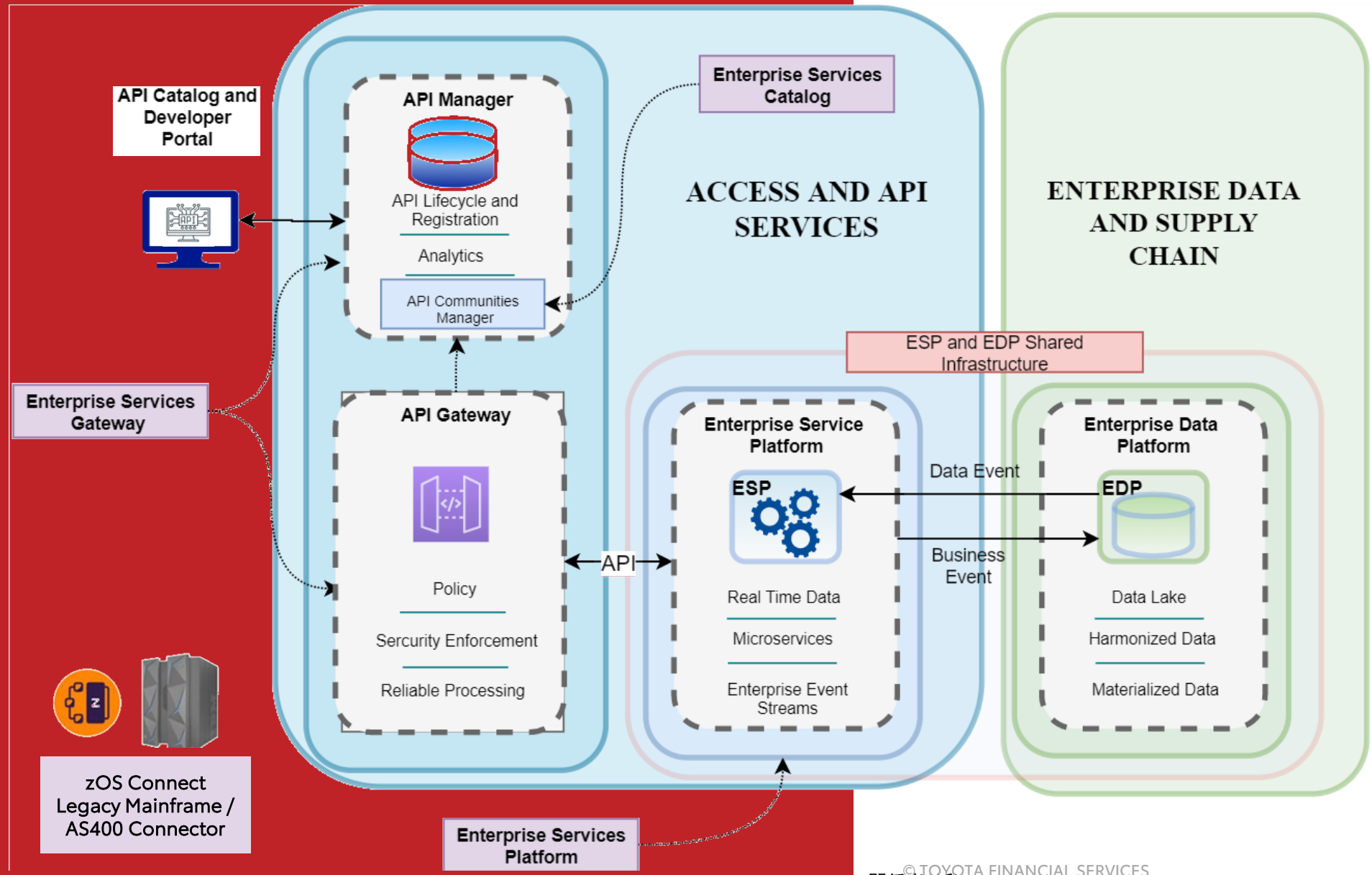
Update API Services Catalog
Update TFS Enterprise Service Catalog (API Catalog) for all production deployments.

* This is a subset of METER and requirements SPECIFIC to APIs. Please be prepared to prove your API(s) meet these requirements.

TOYOTA FINANCIAL SERVICES

Enterprise API Core Products

What are the API Products maintained by the Enterprise API Factories?



1. Enterprise Services Platform (ESP)
2. Enterprise Services Gateway
3. Enterprise Services Catalog

zOS Connect is the preferred tool for legacy mainframe or AS400 service connections. zOS will be retired with the mainframe ☺.

API Best Practices

- API First Design
- API Development Patterns by Tier
- API Microservice Domain Modeling
- API Data Service Patterns
- API FAQs

API

What is an API?

The Acronym API stands for Application Programming Interface

In speech, and in the mFin architecture playbook, “API” means three (3) different things*

1. **The interface** - how the client gains access to a function or service and what are the “protocols” and interoperability mechanisms.

   Speech Example: **“This is a REST API not a SOAP API”**

   → **API Interoperability Standards**

2. **The implementation –** this is the part of the API that actually does computation, performs task, e.g. the number crunching. It’s the running code base that affords the service or function.

   Speech Example: : **“The API had to be rewritten as a microservice.”**

   → **API Contributor and Implementation Standards**

3. **The instance –** the instance refers to both the interface and implementation running on some computer or computer(s).

   Some standard API instances would be: development; test; stage; production; disaster recovery

   Speech Example: **“The API in stage is down!”**

   → **API Instance Standards**

TOYOTA FINANCIAL SERVICES

Build Services “API First”

Regardless of the type of application your factory is developing your ultimate goal is to have that application be a participant in the mFin ecosystem of services. All factories are asked to build APIs as their primary interfaces!

Considerations

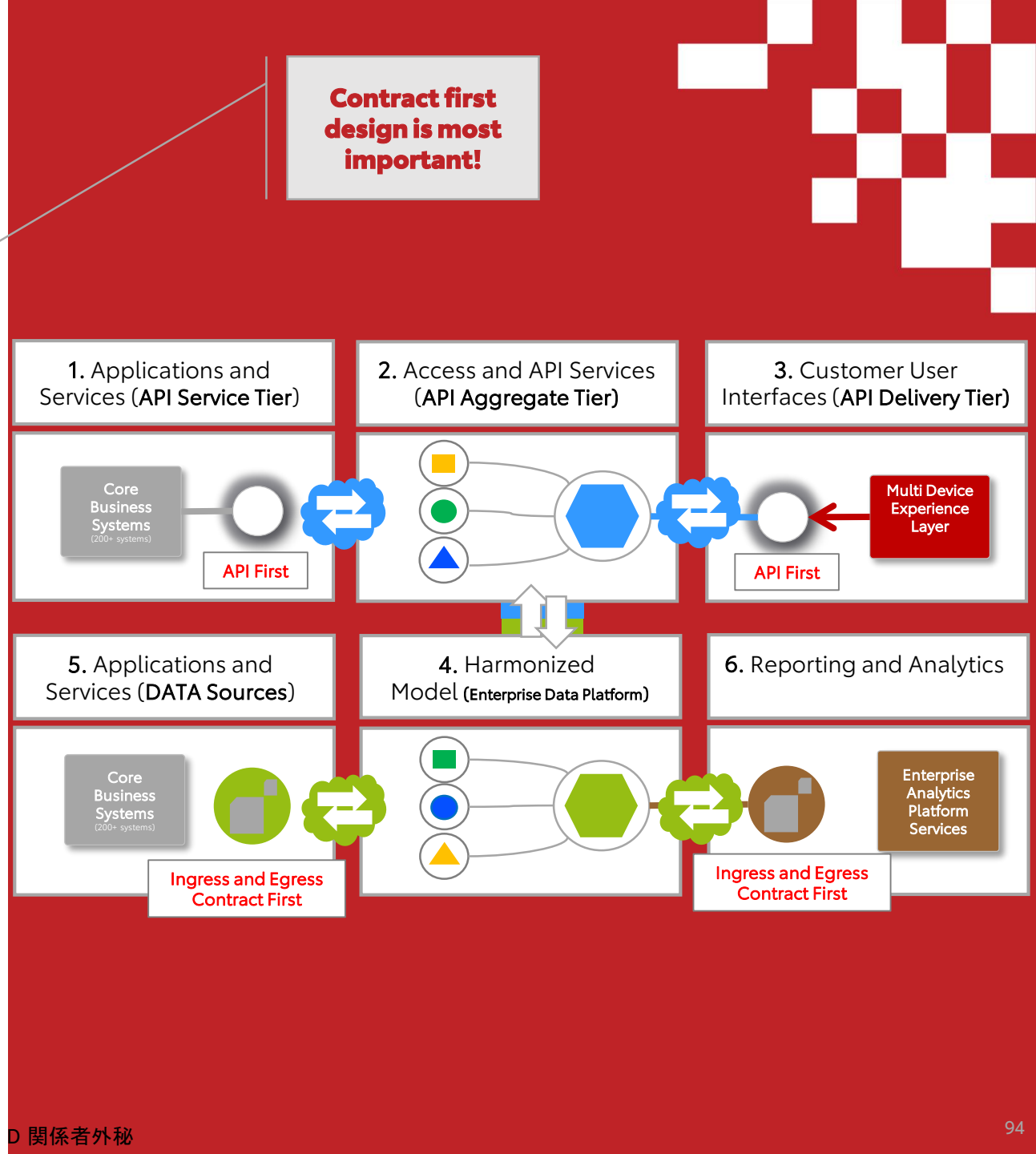
You have fully embraced the mFin guides. You are building cloud-native applications with microservices, and after code gets checked in to your repository, builds happen and test are automatically run, and you have release candidates running in minutes.

Now another mFin team uses your service. Every time you change you break them! Now take this and multiply by a hundred or a thousand services!

Solution

Build Services “API First” To avoid integration nightmare(s). Treat all your interfaces INCLUDING batch data as contracts or **“API First”**

- Start with the API as public contract. Your interfaces are first-all class artifacts.
- “API first” gives teams the ability to work against each other’s public contracts without interfering with internal development processes.



API Development Patterns

How can APIs be developed to ensure optimal performance when "one size does not fit all"?

Considerations

Mobile and web experience applications have very different service level needs than internal systems. API(s) from different systems have different styles, protocols and service levels. Some systems are not "API" enabled at all.

Solution

API Tiers and API Development Patterns

Different API Tiers require different styles of API and correspondingly different standard development patterns of API(s)

API "Cookbook"

mFin Architecture	API Tier	API Development Patterns
Multi Device Experience Layer	Client	NONE – This tier resides on device or client
Access and API Services	Delivery via API Gateway	- Translator - Cache - Policy (Proxy) - Data Composition
	Aggregate via ESP and API Gateway	- Process Aggregate - Business Event Stream
Data Supply Chain	Aggregate via ESP and API Gateway	- Data Services - Data Event Streams
Core Business Systems	Services via API Gateway	- System - System Connector - Translator - Reliable Messaging - Policy (Proxy)
External Partner and Affiliate Systems		

API Development Patterns & Solution Styles

API Development Pattern	Description	Example	API Solution Style
These are the standard set of solutions for API(s). These are the things you can ask the API team(s) to build.			
System Connector	When systems / partners do not have native API support connectors are needed to enable API access.	zOS Connect for Legacy Mainframe	- Vendor - Gateway Plugin - Gateway 3rd Party Plugin
Translator	Protocol and data type translations may be needed. These should be limited to EXCEPTION CASES to prevent overuse and coupling with integration systems.	CCPA Adobe system cannot read native REST API and requires data in XML format.	- Gateway Plugin - Gateway 3rd Party Plugin
Proxy	The proxy is the policy enforcement point on the API Gateway	Web SSO API Proxy	- Gateway
Process Aggregate	These are important API(s) and should be employed whenever "Transactional" integrity is required between two systems or two API.	Due Date Change	- Microservice
Data Service	These are important API(s) and should be employed whenever data services are needed. **Data Services are the ONLY way to access real time information from the Enterprise Data Platform.** Direct access to the Enterprise Data Platform is considered an anti pattern in mFin.	CCPA Data Fulfillment Service	- Microservice
Cache	In memory cache are used with various caching strategies (cache ahead, cache on change) to provide fast data service to improve customer experience	Fast Customer Lookup for IVR and Web	- Microservice
Stream	ESP platform supports two kinds of streams: Business Event Streams and Data Streams. Business Events are published when key business events occur (e.g. New Contract). Data Streams are based on event sourcing patterns.	Nice event streaming	- Microservice
Reliable Messaging	Reliable messaging delivery is offered to enable observer patterns, guaranteed delivery, replay and durability of business events between systems. (Enables Enterprise Observer patterns)	Dealer MDM notifications	- Microservice
Data Composition	Read only services which combine the data from more than one API (Backend For Front Ends BFF)	Dealer web user lookup	- Gateway

API Microservice Domain Modeling

Enterprise Domain Architect

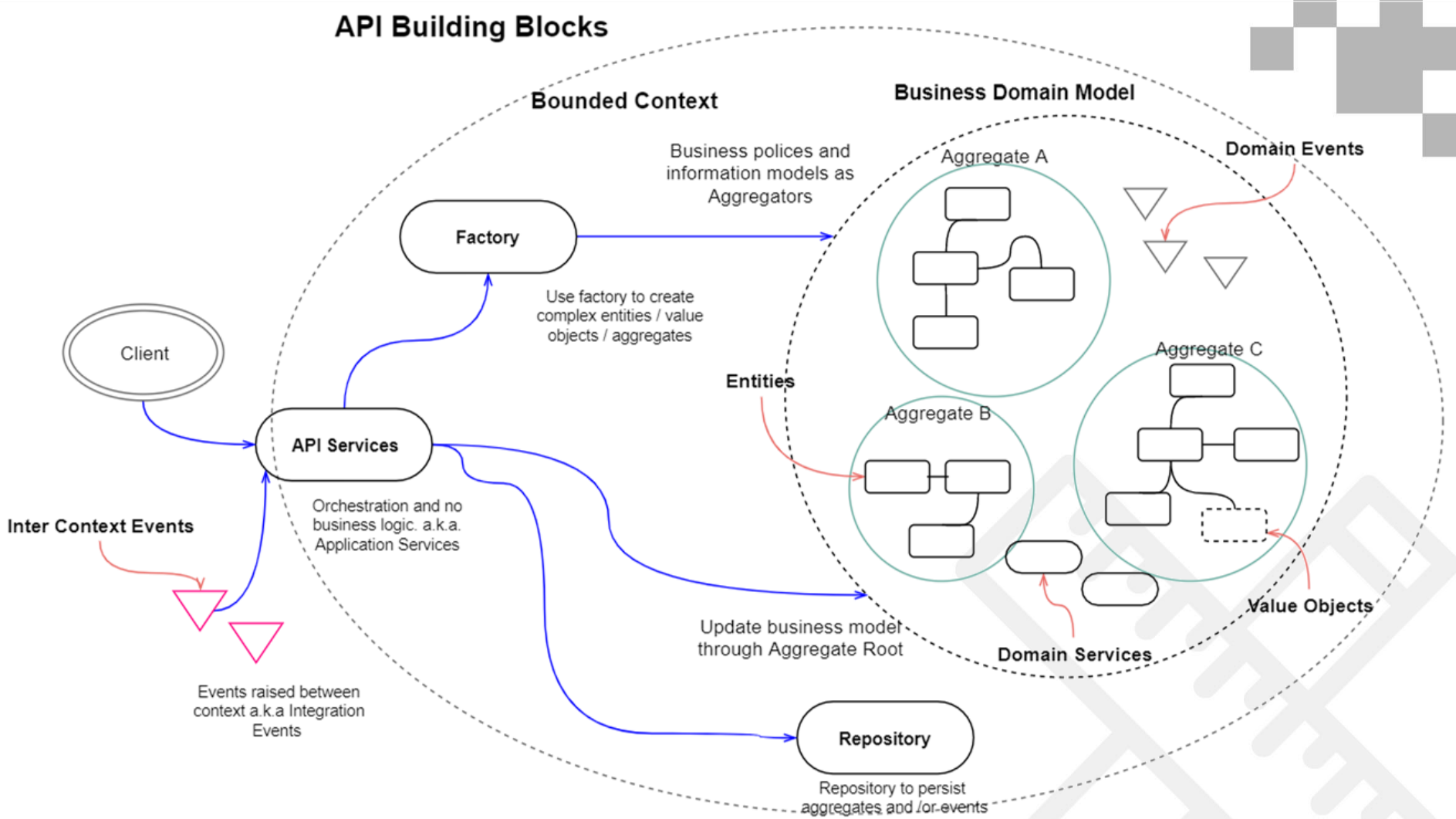
Product Designer

EA Domain Architect Works with Designers on bounded context for applications, integration microservices and data supply chain.

Example:

Bounded Context :Finance Contract
Entity: Borrower, Vehicle, etc.

Aggregate: Due Date Change, Late Fee Waiver, etc.



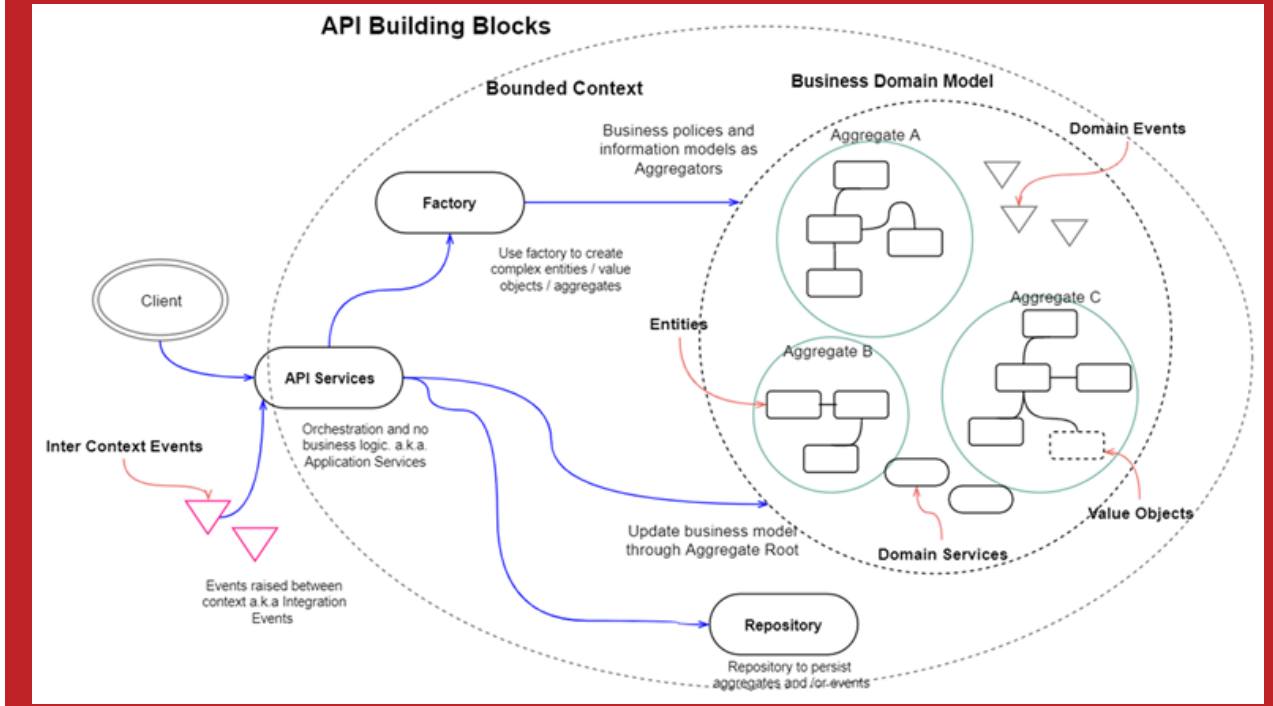
Credit: Domain-Driven Design, by Eric Evans, which provides a broad framework for making design decisions and a vocabulary for discussing domain design

API Microservice Domain Driven Design (DDD) Terms

Term	Meaning
Bounded Context	Bounded Context - defines tangible boundaries of the API systems based on the business area. The bounded context is the biggest possible service where the rules, terms and information would be consistent. For our purposes it's the boundary of the service internals but it also includes the API of Service (the communication with client and other services). The bounded context is a cluster of "codesigned services" that are designed to work with each other
API Service	API Service (a.k.a. "application service") comprises the client or customer facing interfaces. It orchestrates how the outside world interacts with the service internals ("Make a Payment" – "Withdraw Money" etc.)
Aggregates	Aggregates - a cluster of objects within a business domain used to further breakdown the bounded context. Every aggregate has a specific root and a boarder and within that border all the business rules (a.k.a. Domain Invariants) should be satisfied. (Examples "Due Date Change" for finance or "Insurance Claim" for insurance domain)
Domain Services	Domain Services – these are used to orchestrate interactions which cannot be attributed to a specific entity or value object in the domain (example user identification or authentication)
Inter Context events	Inter Context events – these are events which can trigger services effects across multiple bounded contexts
Entity	Entity - is a mutable object. It can change attributes without changing its identity. All entities have a unique ID. Entities are modeled as resources using object role modeling (ORM)
Value objects	Value objects - are like entities but their attributes should never change (they are immutable). They are known to the system only by their characteristics and do not have unique identifiers. A persons address could be modeled as a stream of immutable "address" value objects (using the event sourcing pattern)

All API microservices are organized in this fashion within the microservice codebase.

Factories	Factories – a pattern to separate the use of the object from the construction of the object
Repositories	Repositories – a pattern to manage aggregate persistence and retrieval. Polyglot database and database per microservice patterns are promoted.
Event sourcing	Event sourcing – a pattern for persisting the state of an object and finding the current state by traversing through the history of those saved states



API Data Service Patterns

Eventual consistency between systems should be assured. But how?

Design Goal

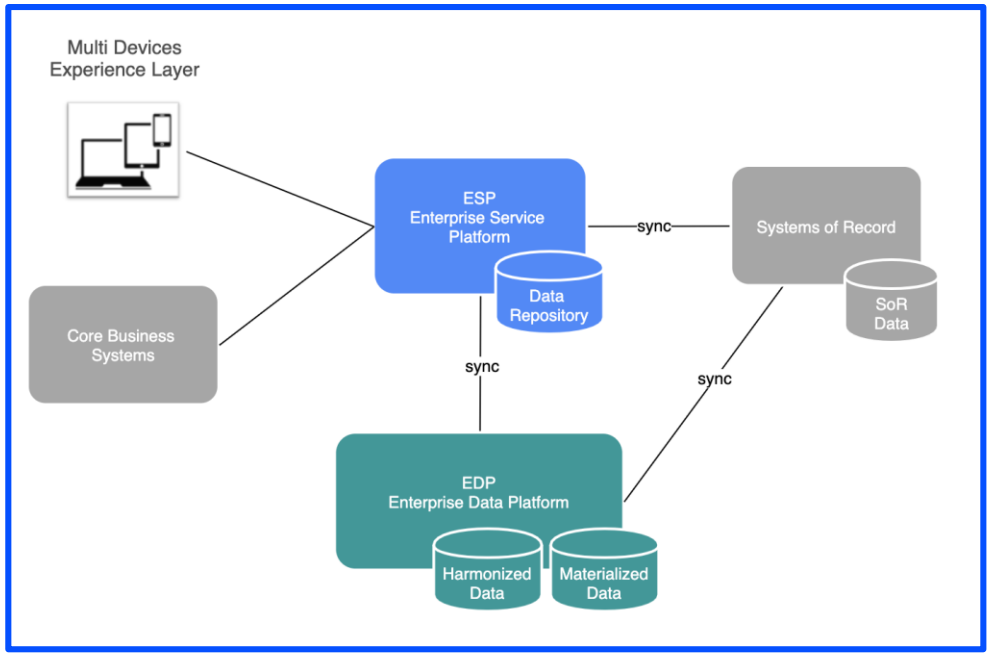
A key goal of this design effort is to define architecture patterns to synchronize domain data between systems:

- Systems of Records (SoR),
- Enterprise Data Platform (EDP), and
- Enterprise Service Platform (ESP), as shown in the figure.

Desired Consequences

- Highly reliable
- Real time access to change data
- Supports high volume and throughput
- Maintains loose coupling between solution components
- Rapidly adaptable and changeable by the business. We will be able to evolve and adapt to fast changing business needs.

ESP Data Service - **Figure 1**



What are the key patterns needed to support Data Services?

Key patterns used

#1 **Event Notification Pattern**: Services send event messages to notify other systems of a domain change.

- Messages do not carry the data that has changed but use a URI to tell where to get the data.

#2 **Event-Carried State Transfer Pattern**: Services send event state messages to notify other systems of a domain change.

- Messages carry data that has changed but consumers do not need to contact the source system in order to process the change.

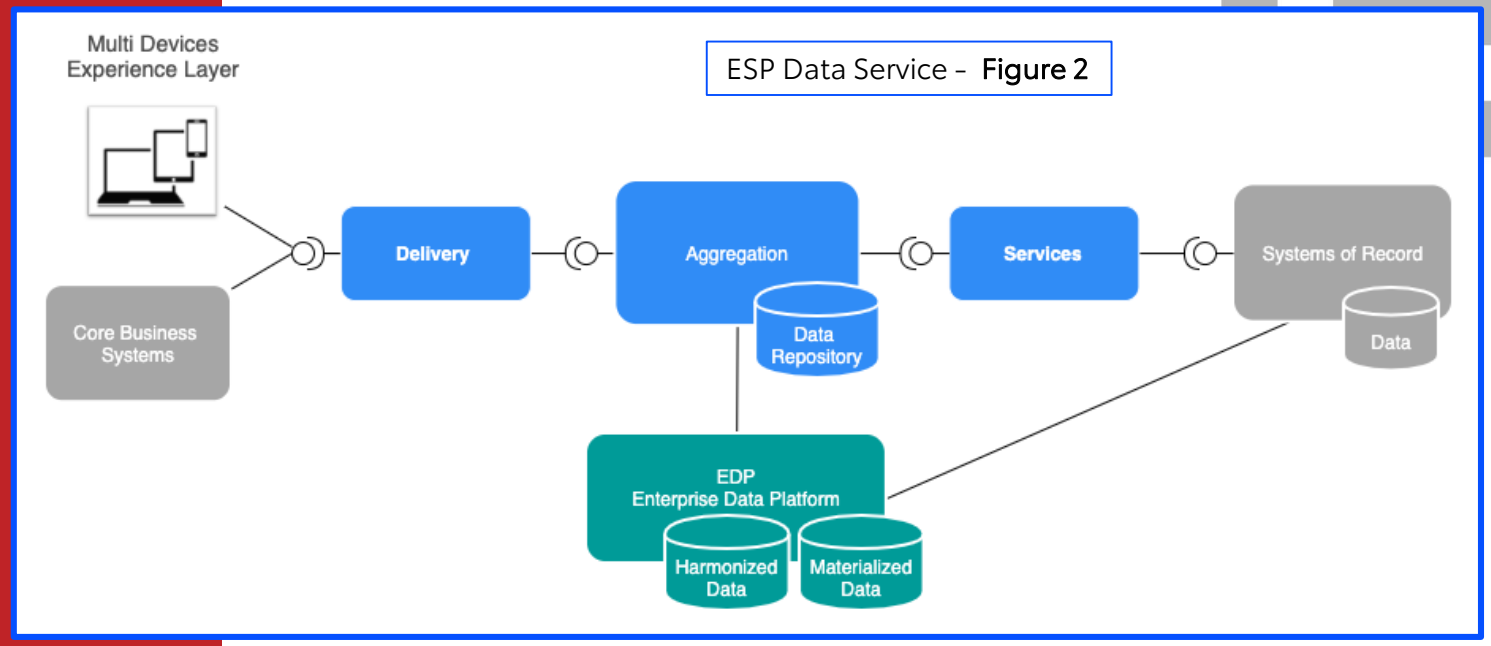
#3 **Event-Sourcing Pattern**: this pattern persists the state of a business entity as a sequence of state-changing events.

#4 Use **Four-Tier architectural** pattern which segregates the following tiers: Services tier, Aggregation tier, Delivery tier, Client tier

TOYOTA FINANCIAL SERVICES

ESP Data Services – Tier View

Domain – Four Tier Architecture View



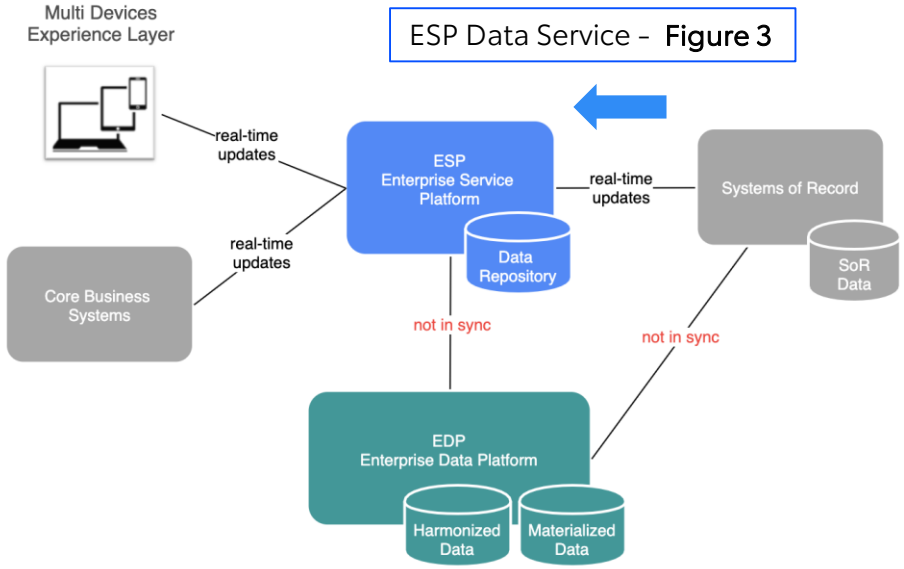
Data service design considerations:

- ALL data is persisted in the EDP Harmonized Data store. Data quality is enforced through the use of <harmonization schema> .
- EDP Harmonized data cannot be consumed through direct access – EDP data is consumed via Data Services exposed as APIs through ESP.
- EDP Materialized Data is consumer specific.
- ESP services aggregate data from multiple data sources including EDP Materialized Data.
- ESP services data is cached into "local" Data Repositories for performance reasons as well as to reduce processing load on data sources.
- Data quality rules are enforced in real time during aggregation of data from all sources. Note that Materialized Data is already harmonized.
- Realtime aggregation rules are enforced and errors are handled in real time.
- All configurations are based on JSON/Avro schemas.

To develop the design, we will analyze the events that can result in domain data inconsistences. Based on that analysis we will design the ESP and EDP components to address the challenges.

Business Events Cause Inconsistency

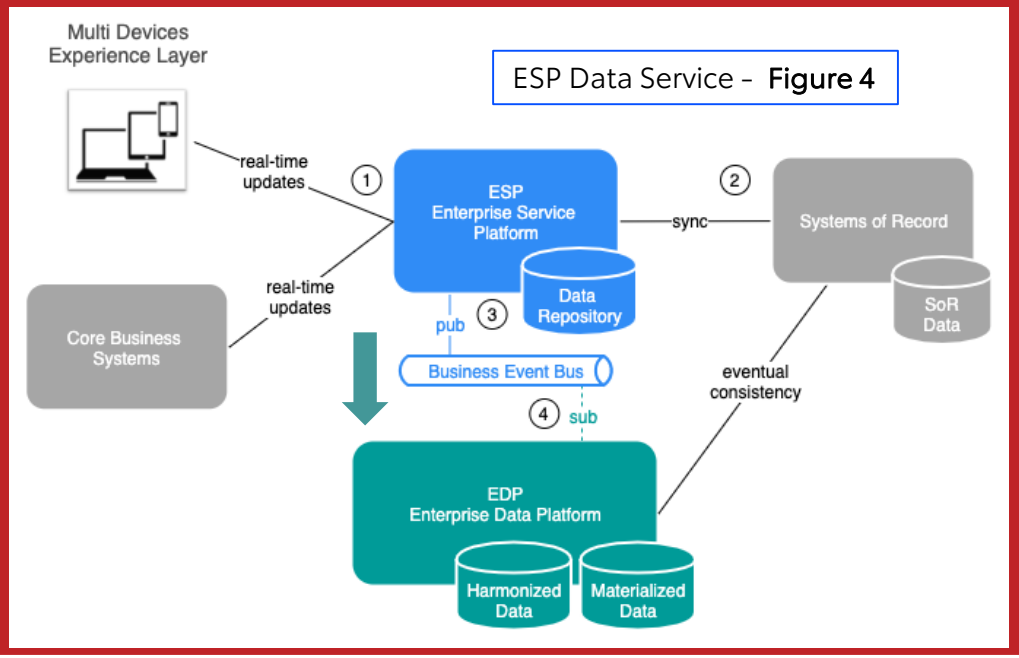
Business Events create inconsistency: Starting from a consistent state shown in ESP Data Service - Figure 2, real-time Business Events that change the System of Records states might leave EDP data in inconsistent state as shown below in ESP Data Service - Figure 3



Typically, amount of data changed during Business Events is not large, so we selected **Event-Carried State Transfer Pattern** vs Event Notification to synchronize data between ESP and EDP.

How to ensure consistency when business changes things via API?

Solution: To ensure data integrity, our design calls for implementation of Event-Carried State Transfer Pattern between ESP and EDP platforms as depicted in ESP Data Services - Figure 4.

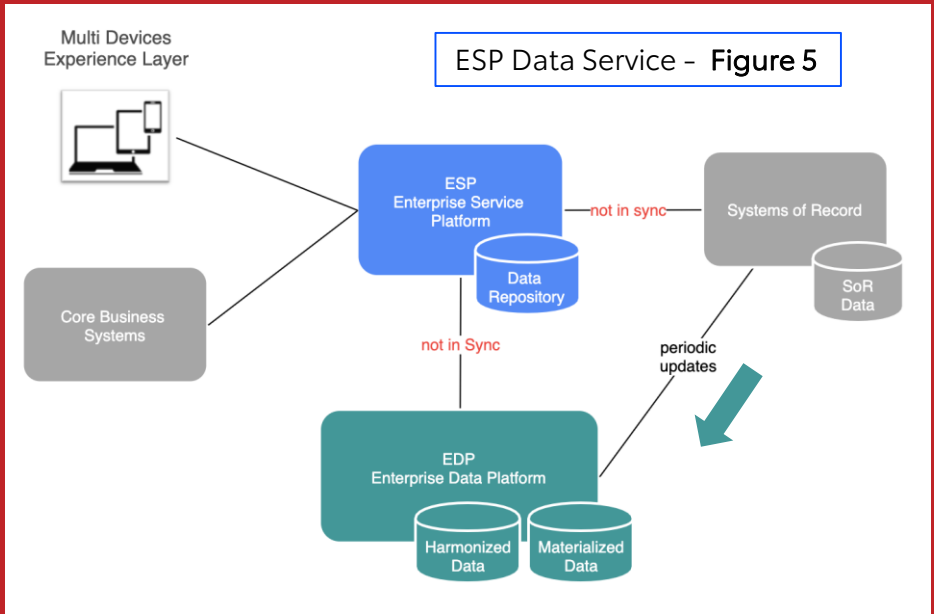


Solution Narrative:

1. Customers, Dealers and CBS make changes to SoR in real-time.
2. ESP aggregate services update SoR using Service Tier APIs – at this point rules are enforced, and errors are handled.
3. On success Data Repository (local cache) is updated and data sources (EDP and others) are notified in real-time by publishing Business Event.
4. The Business Event is consumed asynchronously by subscribers including EDP.

Data Events Cause Inconsistency

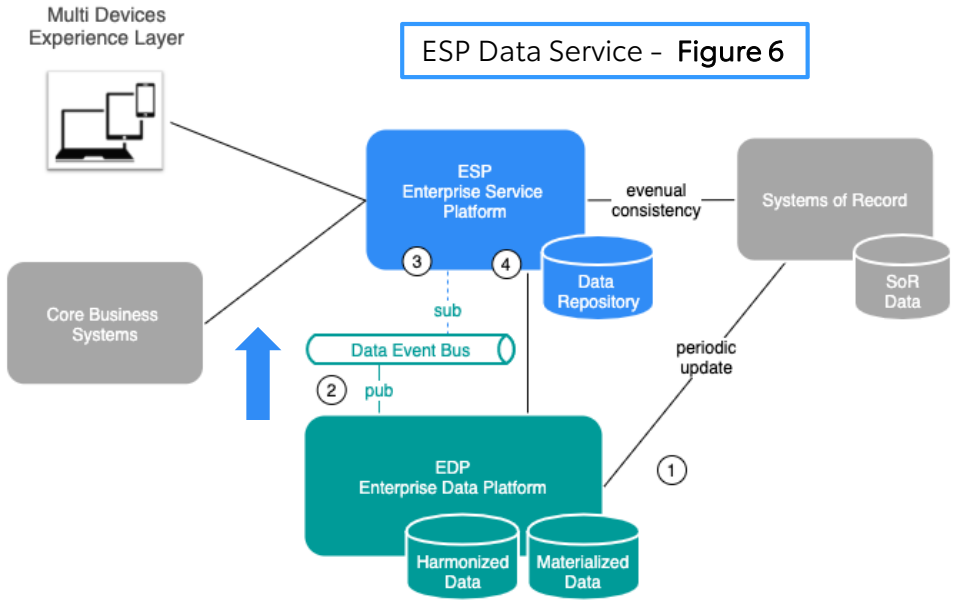
Data Events create inconsistency: System of Records update EDP periodically (at least daily) leaving ESP data in inconsistent state as shown in Figure 5.



ALL mFIN systems must send Enterprise Data Platform all their data daily

Data flows in batch to Enterprise Data Platform. How do we keep systems in synch?

Solution: To ensure data integrity, our design calls for implementation of Event Notification Pattern between ESP and EDP platforms as depicted in Figure 6

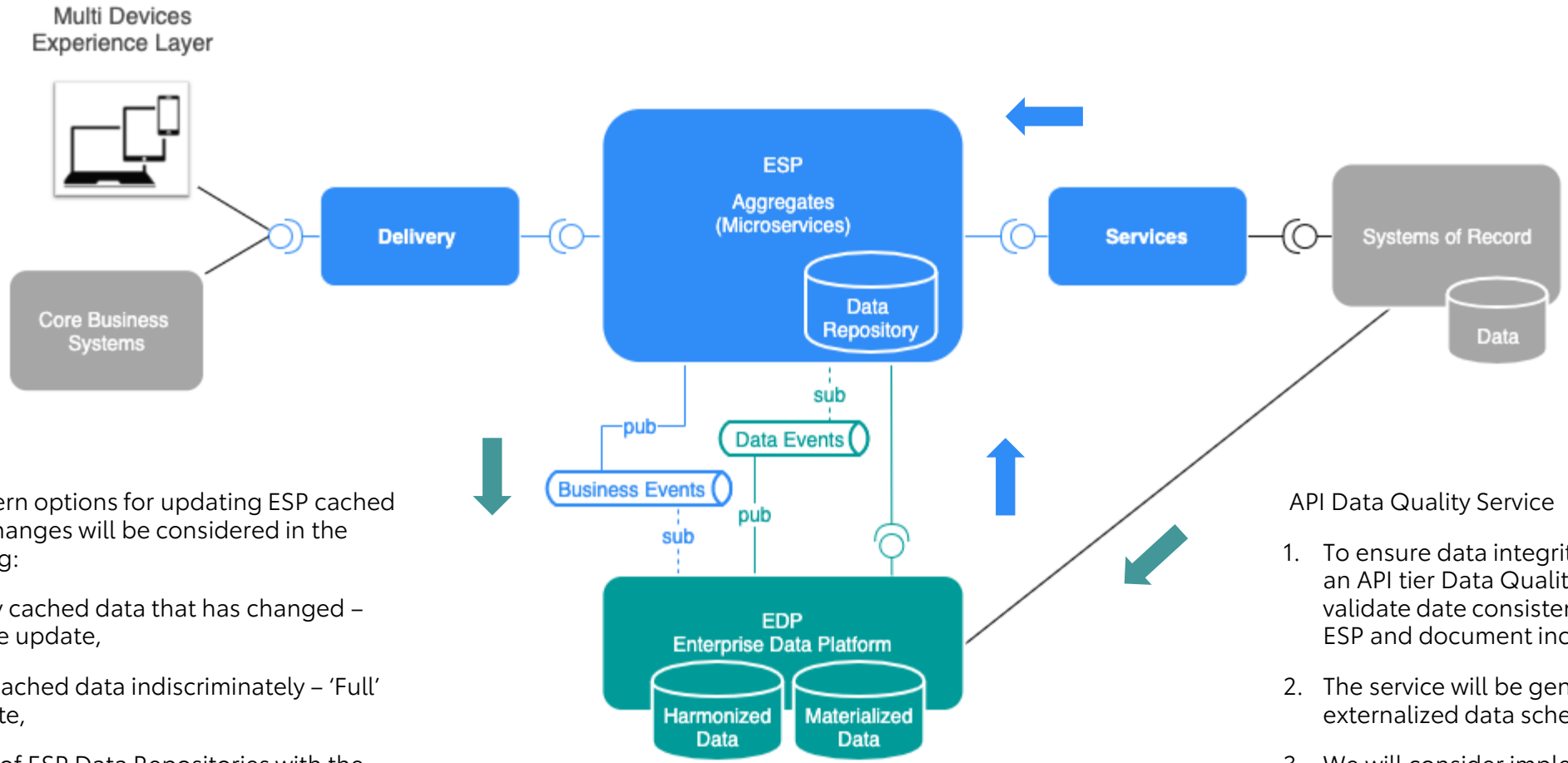


Solution Narrative:

1. Data in EDP is periodically updated with changes from SoR. Note that all data from SoR needs to be available in EDP
2. Upon successful Harmonization and Materialization EDP notifies ESP and others about a change in its data in real-time by publishing a Data Event.
3. The Data Event is consumed by ESP and next action is determined based on the call back information.
4. ESP calls-back the publisher to gets the data and updates the Data Repository (local cache) with changes

API Event-Driven Four-Tier Data Services Architecture

Ensuring eventual consistency



ESP Data Service - **Figure 7**

Additional Pattern options for updating ESP cached data with the changes will be considered in the design including:

1. Update only cached data that has changed – 'Delta' cache update,
2. Update all cached data indiscriminately – 'Full' cache update,
3. Full update of ESP Data Repositories with the current EDP materialized data – 'Full' business layer update.

API Data Quality Service

1. To ensure data integrity we will advocate for an API tier Data Quality Service which will validate date consistency across SoR, EDP, ESP and document inconsistencies.
2. The service will be generic, and it will utilize externalized data schemas of the systems.
3. We will consider implementing the service utilizing AWS Serverless Framework.

TOYOTA FINANCIAL SERVICES

API: Frequently Asked Questions

	API Question	Response
1	Who is responsible for putting API(s) into the catalog? Who keeps the catalog updated when APIs for systems and partners change?	Who is responsible for putting API(s) into the catalog? Who keeps the catalog updated when APIs for systems and partners change?
2	Do I need to publish all my system's API(s) into the catalog?	No. Publish all **API which are intended for use by external systems or business partners** into the API catalog.
3	Why do we publish APIs into the catalog?	Other teams need to know what APIs are available to spur innovation.
4	Our APIs are all tool related such as Jenkins and Jira. Do they need to be in the catalog?	Yes. Major tool API should be cataloged so developers can innovate. However, please use your own judgement. For example: Do not try and catalog all Amazon AWS APIs.
5	Do all APIs get proxied in the TFS Digital API gateway?	No. TFS Digital API gateway is the policy enforcement point for **APIs that are shared between systems or with external entities.** API(s) which are internal to your systems (private or protected) DO NOT need to be protected by the TFS Digital API Gateway.
6	Our APIs are all tool related such as Jenkins and Jira. Do they need to be proxied in the TFS Digital API gateway?	Currently we require mFin systems (not tools) to be protected by the TFS Digital API Gateway. However, coming soon, we will have a tool solution/standard for tools. The plan is to leverage AWS API Gateway as the API Gateway standard for tools.
7	If a published API doesn't do exactly what I need, what should I do next?	You should contact the API provider and see if it can be enhanced easily. Always feel free to come to the Enterprise API Domain teams with your API ideas.

TOYOTA FINANCIAL SERVICES

Enterprise Data and MDM

1 Clarifying Terms

Data Harmonization in an EDP Platform

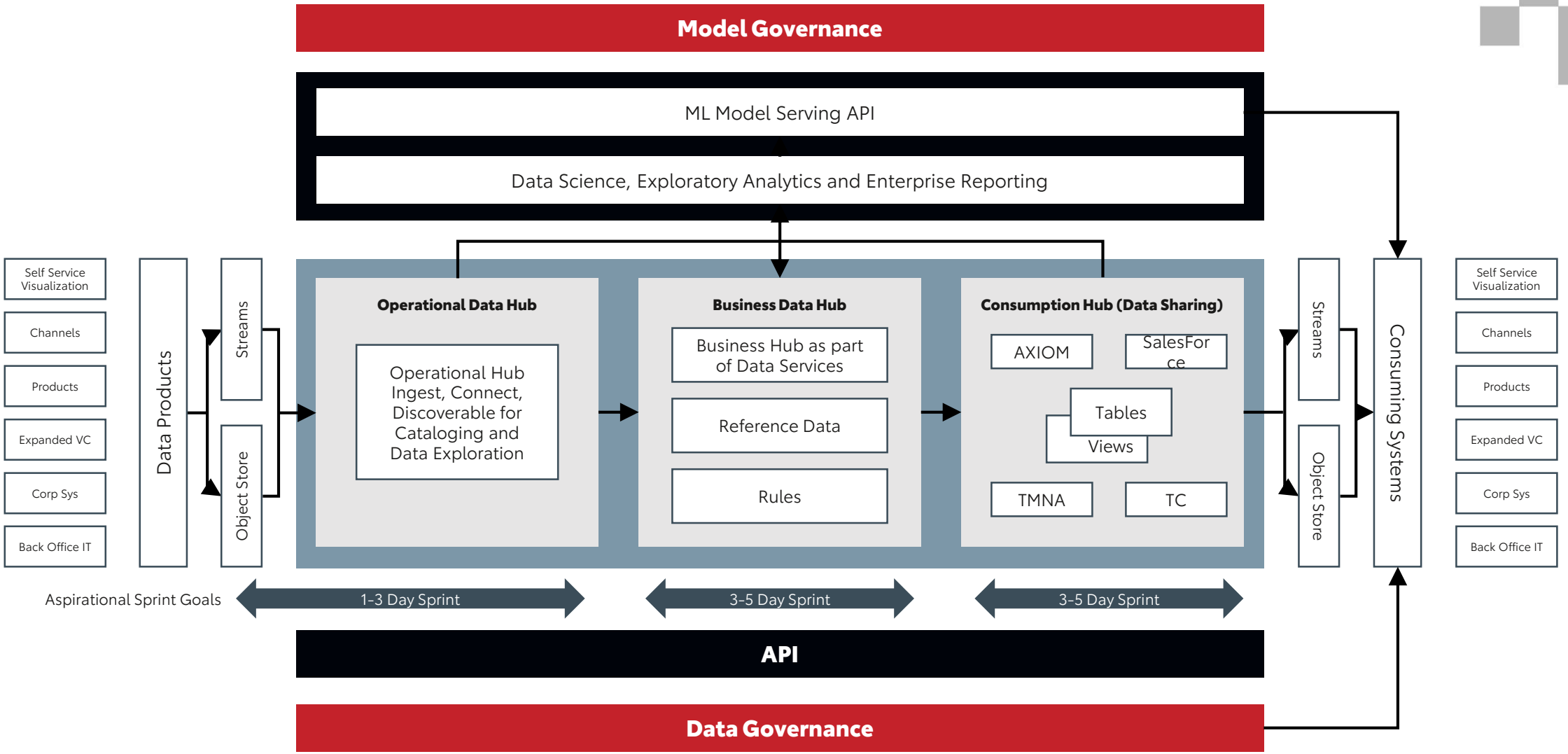
- Data is in different formats with different field names and schemas in the various silos, but when it is ultimately served up, the records must look at least similar to be processed as a group. Harmonization falls into three categories:

1. **Naming Differences:** The simplest to handle, naming differences, occur when different silos have an identical value with identical meaning, but use a different name. E.g. “PERSON.firstName” vs. “IDENTITY.GIVEN_NAME” in two different source tables—but holding the same sort of data.

2. **Structural Differences:** Somewhat more complex than Naming Differences, structural differences occur when the number and combination of fields differ across silos. E.g. “boxes_available” in one system might be total boxes in the warehouse plus a count_per_box field to derive total items, but in another system, may directly represent total_items (regardless of boxing). Further, one may represent available inventory not already allocated to unfilled orders, where the other may represent all physical inventory, regardless of open orders.

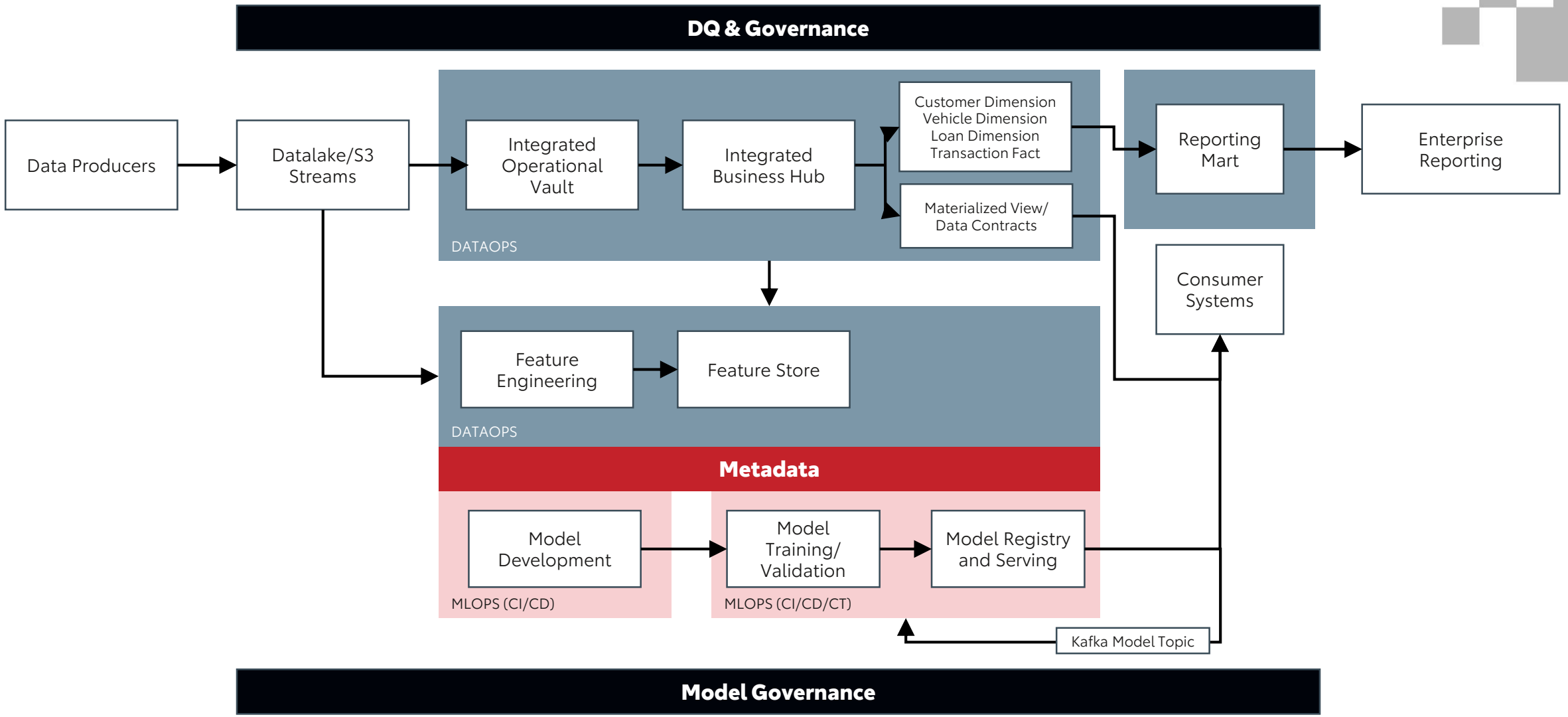
3. **Semantic Differences:** These are the worst. One system may have three patient statuses, and another may have five. These statuses will often overlap and be hard to map to one another ({Scheduled, Needs_Followup, Inactive} vs. {Intake, Scheduled, Telemedicine-only, Discharged}).

Goal EDP 2.0 : Conceptual View

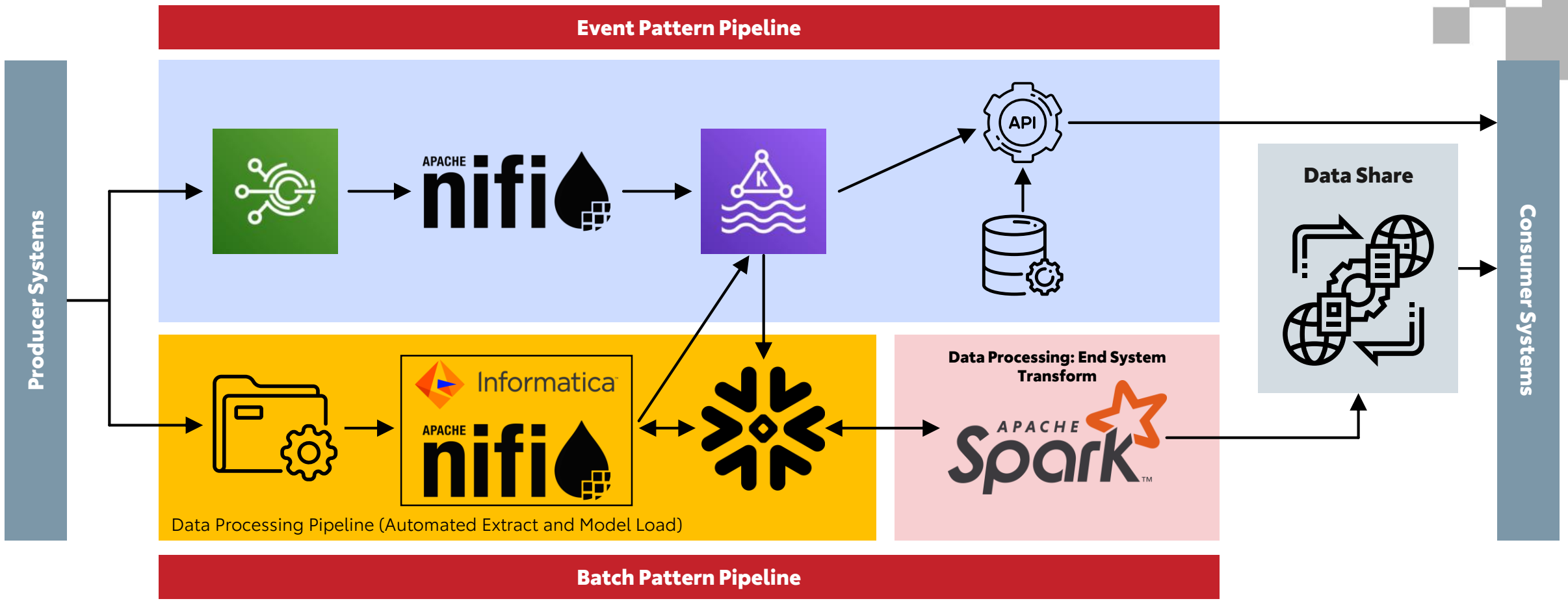


Level Down

Micro Batching and Streaming Framework



Reference Data Pipeline



Data Vault

Data Vault Organization of Objects:

HUB - Consists of unique list of Business Keys and Multi Tenancy

SATELLITE - Consists of descriptive data of parent Hub or Link that can change over time

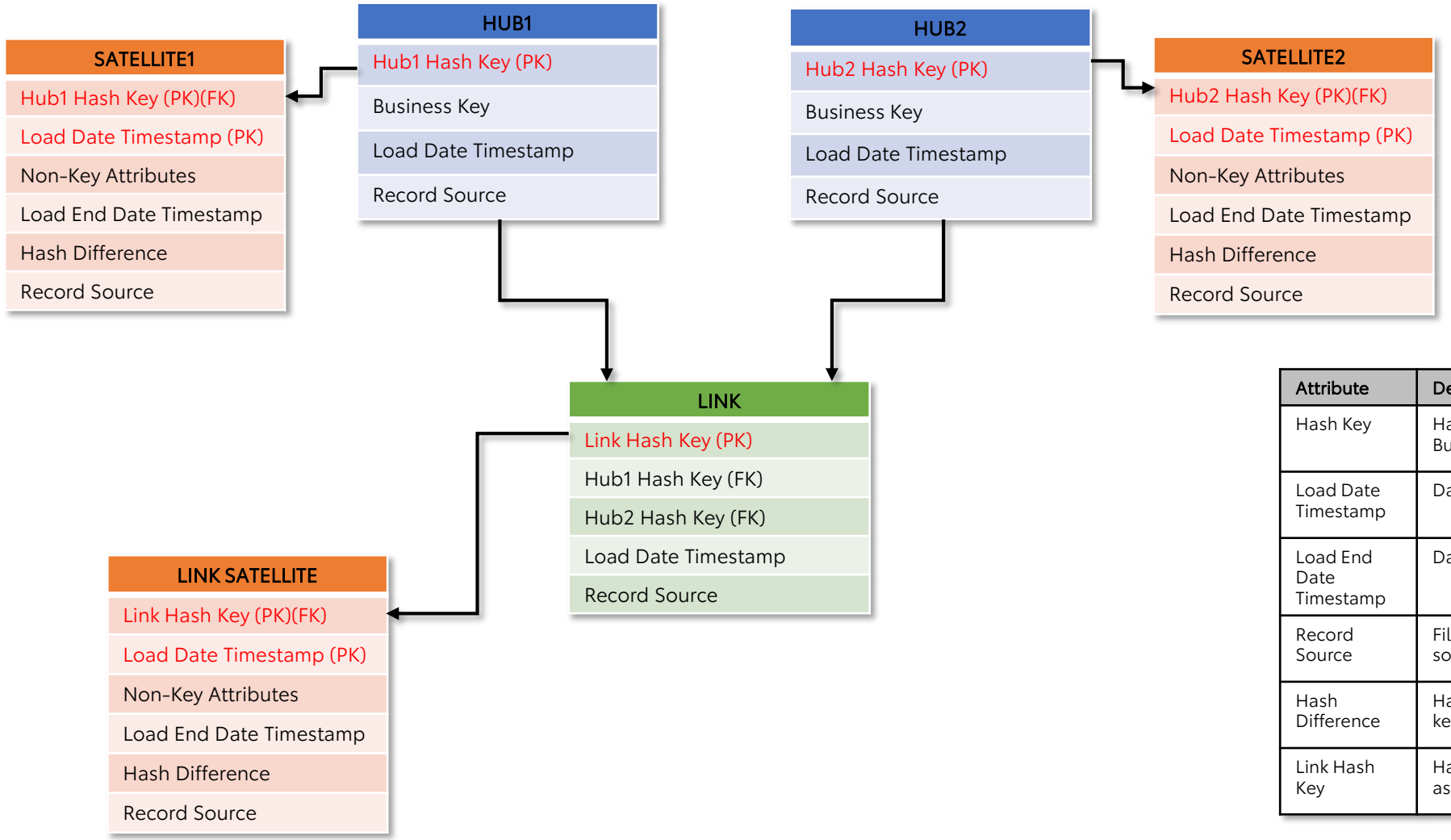
LINK – Represents relationships/associations, hierarchies, transactions and events

HUB
Hub Hash Key (PK)
Business Key
Load Date Timestamp
Record Source

SATELLITE
Hub Hash Key (PK)(FK)
Load Date Timestamp (PK)
Non-Key Attribute1
Non-Key Attribute2
Load End Date Timestamp
Hash Difference
Record Source

LINK
Link Hash Key (PK)
Hub Hash Key (FK)
Load Date Timestamp
Record Source

Data Vault – Overview of Data Model

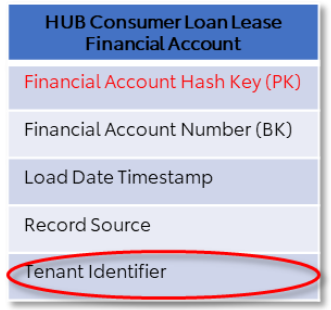


Attribute	Description
Hash Key	Hash representation of Business Keys
Load Date Timestamp	Date and Time of data insert
Load End Date Timestamp	Date and Time of expiry data
Record Source	File name or Table name of the source system
Hash Difference	Hash representation of all non-key attributes and meta data
Link Hash Key	Hash representation of associative Hub Business Keys

Data Vault – Multi Tenancy and Product Line of Business

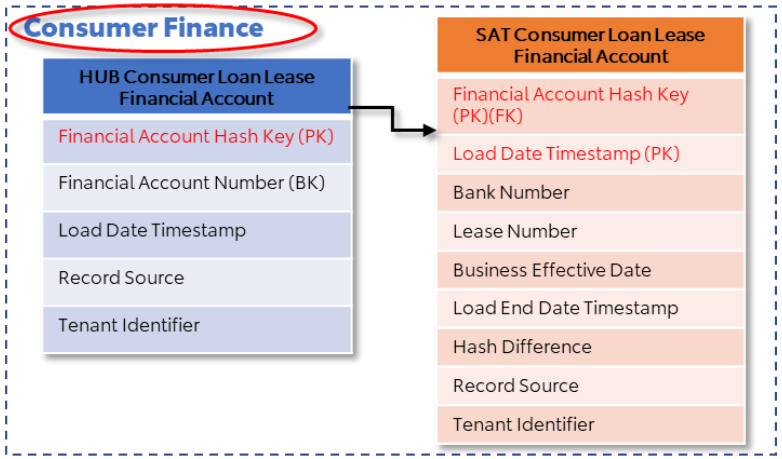
Multi Tenancy:

- Data segregation at Hub level with Tenant Identifier
- Different Vaults
- Different S3 buckets



Product Line of Business:

- Data segregation by Domain
  Example: Insurance, Commercial Finance, Consumer Finance, etc.
- Introducing Hubs, Satellites and Links by Product Line of Business



Harmonize & Materialize Data for Consumption

All data pulled into the Enterprise Data Platform will be harmonized and materialized for consumption by applications/factories.

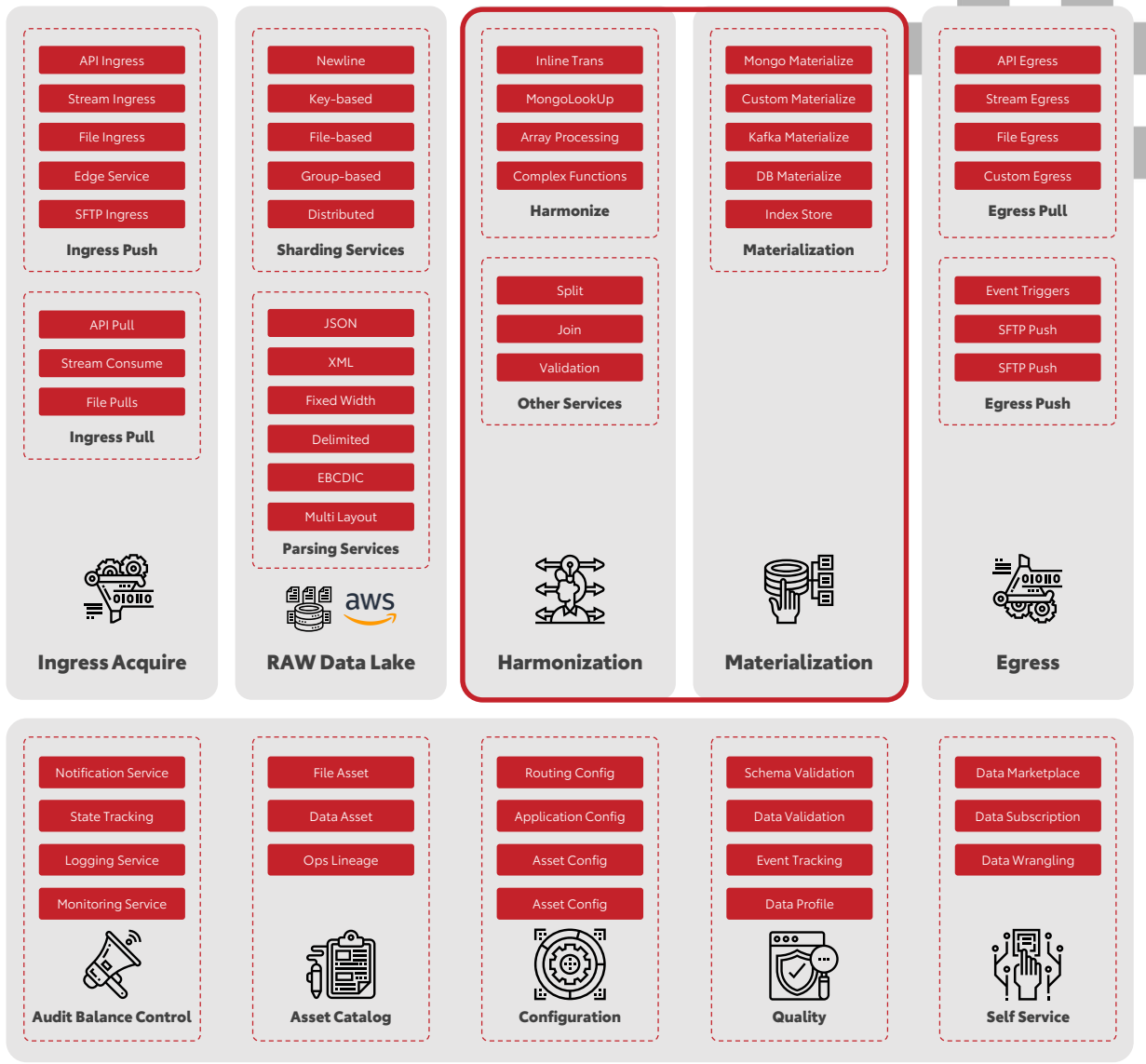
Factories need to provide the event and time-based specifics (transactional and operational) to achieve the expected data materialization for their consumption and use.

Acceptance Criteria

- Harmonized output stores fine-grained well-linked dataset in a standardized format with high cohesion for enterprise use.
- Structural data transformation meets and fulfills business needs.
- Ensure content level derivations, calculations, aggregation/summarization and splits are in alignment with business requirements.
- As functions are incrementally build and the systems may evolve, it utilizes a flexible data model to process new or changed data entities.
- Factory must demonstrate and adhere to the METER operational quality dimensions.

How

- Implement structural level (i.e.., structural data transformation) & content level (i.e., derivation, calculation, aggregation, summarization logic) semantic data integration & data quality functions from the shared enterprise perspective.
- Maintain versioning by utilizing flexible data model as data entities may arrive at different times in different sequences from varying data sources.
- Data enrichment and transformation; process the formatted source data with harmonization logic using a set of microservices and stores the outputs into the harmonized & materialized data stores.
- Ensure data quality is maintained and meets the end user expectations.



TOYOTA FINANCIAL SERVICES

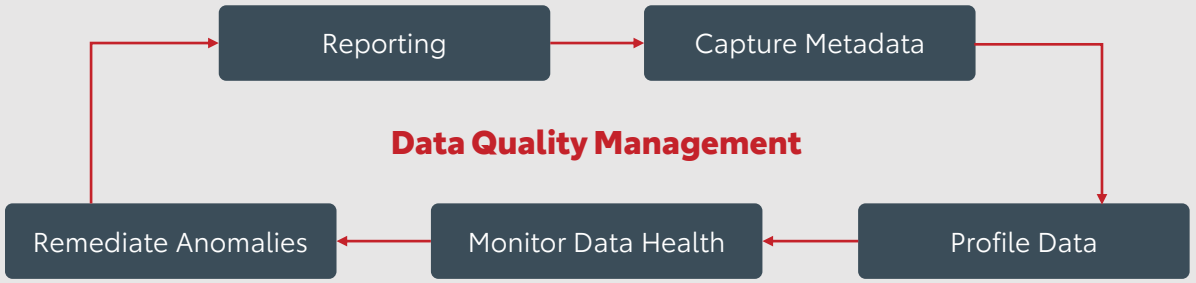
Data Quality Management

Data Quality Mgmt. Process Overview

Inputs:	Process/ Activities:	Outputs/ Deliverables:
• EDC Intake Form • Define Data Quality Expectations • Business Requirements • Business Rules • Data Requirements • Business Metadata • Technical Metadata • Data Sources & Data Stores • Data Lineage	• User Stories Created, Refined • Perform Initial Data Quality Assessment • Identify & Prioritize Improvements • Develop & Deploy Data Quality Operations for Repeatability • Review with TPO/BPO for Definition of Done • Correct, Measure, Monitor & Report on Data Quality Levels & Findings	• Data Quality Framework • Analyses from Data Profiling Based on Epic Definition • Recommendations based on Root Cause Analysis of Issues • DQM Procedures for Future Use • Data Quality Reports • Data Quality SLA • DQ Policies & Guidelines

✓ Metadata Mgmt. ✓ Data Quality Mgmt. ✓ Data LifeCycle Mgmt.

Data Quality Improvement Lifecycle



Applications provide real-time feedback which needs to be included in a day-to-day operations, in order to be effective. They also require access to quality data from any source, whether on-premise, in the cloud, or in any other new data platform.

Lack of front-office controls (e.g., poor quality of data entry at system of origin with no/limited validation) will impede in achieving the enterprise strategic goal of quality management.

Acceptance Criteria

- Ensure the data/information quality is maintained & matches to the user consumption
- All Factories shall plan, implement, and control activities that apply data quality management techniques to data, in order to assure it is fit for consumption and meets the needs of data consumers
- Ensure no entity instance will be recorded/identified more than once based upon how that instance is identified for uniqueness
- Ensure and validate the data is valid and it confirms to the syntax (format, type, range) of its definition
- Ensure the data is accurate, consistent and consumable

How

- **Factories must plan to identify the critical data, existing rules & patterns associated with their domains**
- **Factories, in collaboration with Data Quality Factory, will :**
  - conduct initial data quality assessment to identify, prioritize and perform root cause analysis of issues (baseline),
  - identify & prioritize actions based on business impact, develop preventive & corrective measures to improve data quality,
  - develop & deploy data quality operations to correct, measure, monitor & report data quality levels and findings

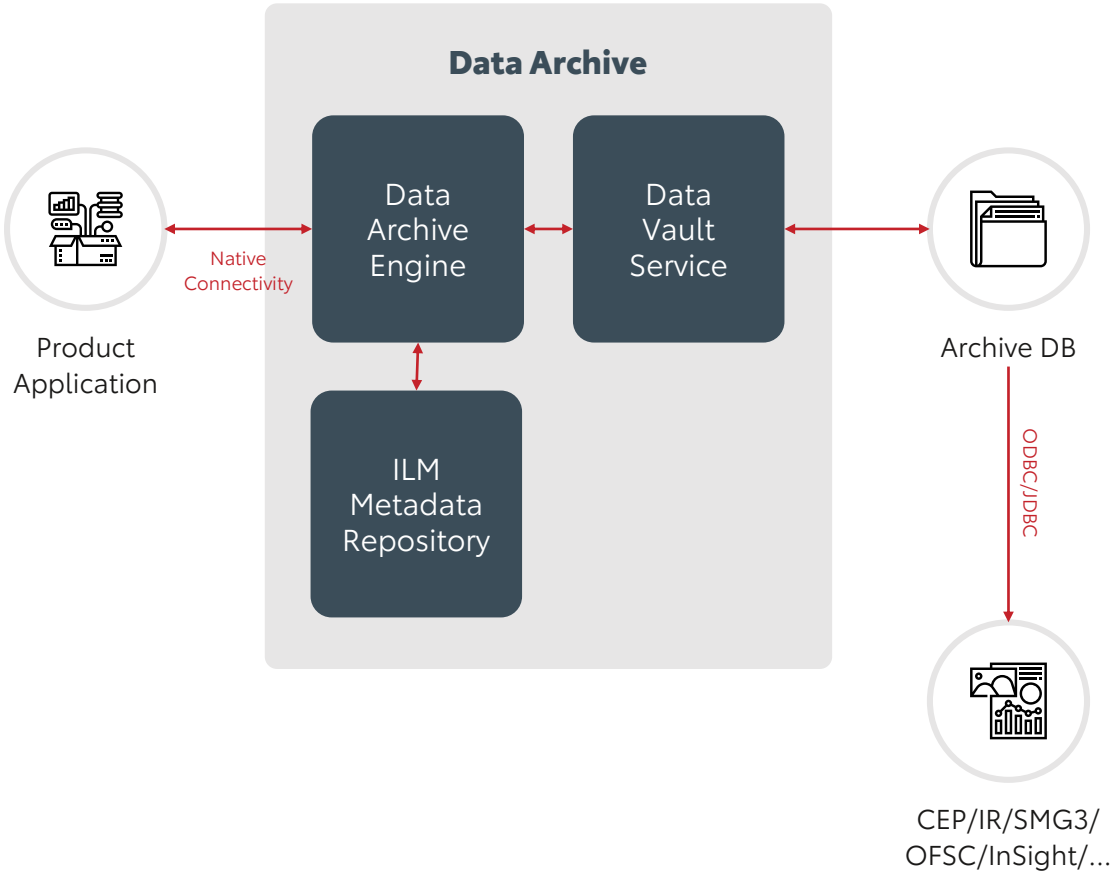
TOYOTA FINANCIAL SERVICES

Data Archival & Restoration

Data archiving is integral part of Data Supply Chain. Due to the expected growth in data over time we must have the archival and retrieval capability. The archived data must be performed in two key time spans (13 month and 13 years).

The archived data must be separated by the functional usage and will be stored according to the enterprise data retention guidelines. Special provision must be in place for email data archival and retrieval.

Data Archiving



Acceptance Criteria

- All factories must ensure that they comply with the data archival/retention and retrieval policies
- All factories must ensure their data archiving/purging in a rolling fashion as per data retention policy
- Ensure for archiving and restore activities fulfills the business requirements
- Enterprise data archival/storage and restoration requires a well-defined process that is practical, reliable and trustworthy at all phases of data supply chain
- Where, how, and length of time must be defined and maintained
- Data ownership, access, storage, backup and backup schedule must be defined
- Store inactive data in a separate solution for long term retention
- Overall storage costs reduced, application performance is improved, and resources are not constrained

How

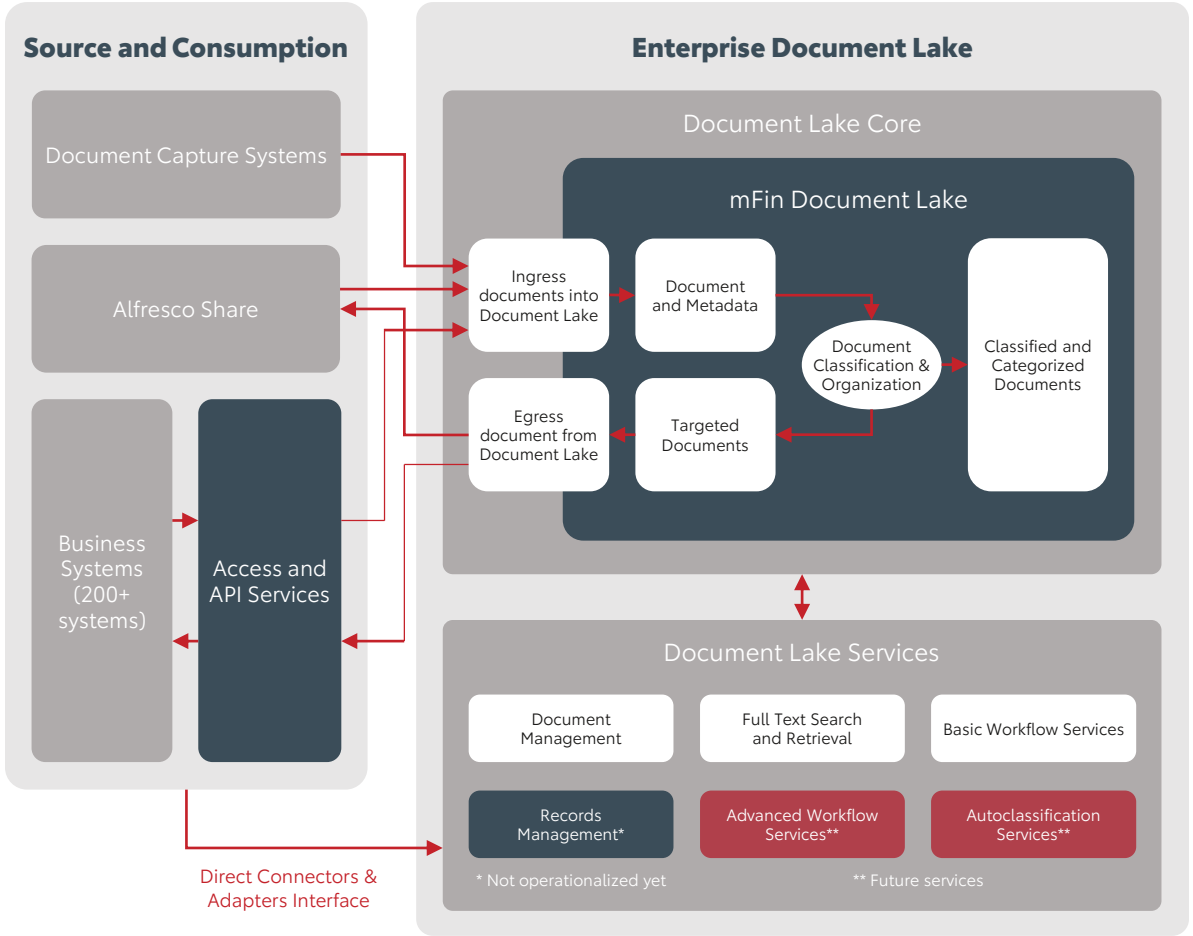
- **All factories must:**
  - Participate in the EDC intake process.
  - Engage with EDQ factory if they are out of compliance as per data storage and date retention policies to archive and purge data
  - Engage EDQ factory to build a process flow for automated archival, retrieval/restoration activities, if needed by the business

TOYOTA FINANCIAL SERVICES

Enterprise Document Lake

‘mFin’ Enterprise Document Lake

Enabling digitizing and automating content and processes



Point-to-point document feeds between applications are not permitted and will be considered an exception.

The Document Lake drives standardization, optimization and automation of documents, enabling ‘mFin’ tenants to utilize one system of record for all document processing requirements.

- The Document Lake platform is the enterprise solution for all document and record management requirements.
- Applications use the Document Lake ‘Egress’ function to request and consume documents.
- Legal, Privacy, Compliance & Security requirements for documents will be enforced by the Document Lake.
- Document processing, transportation and consumption processes will be clearly documented and maintained.
- The Document Lake ingests paper documents, Standalone documents (supporting multiple formats), and Electronic Documents associated with business applications.
- Documents collected in the Document Lake will be archived and will be available for retrieval and consumption as needed.
- Once ingested, documents are classified, organized, subjected to quality control, stored, versioned, processed, and made available for consumption via the ‘Egress’ function.
- Document content and metadata will be modeled for enterprise use and comply with the ‘mFin’ architecture guidelines and standards.

TOYOTA FINANCIAL SERVICES

Document Lake – Multi-tenant Pattern

The Document Lake complies with the multi-tenancy requirement (mFin Pattern 3) in the same storage space by utilizing a unique id per tenant (Tenant ID) and the 'Content Model Definitions'.

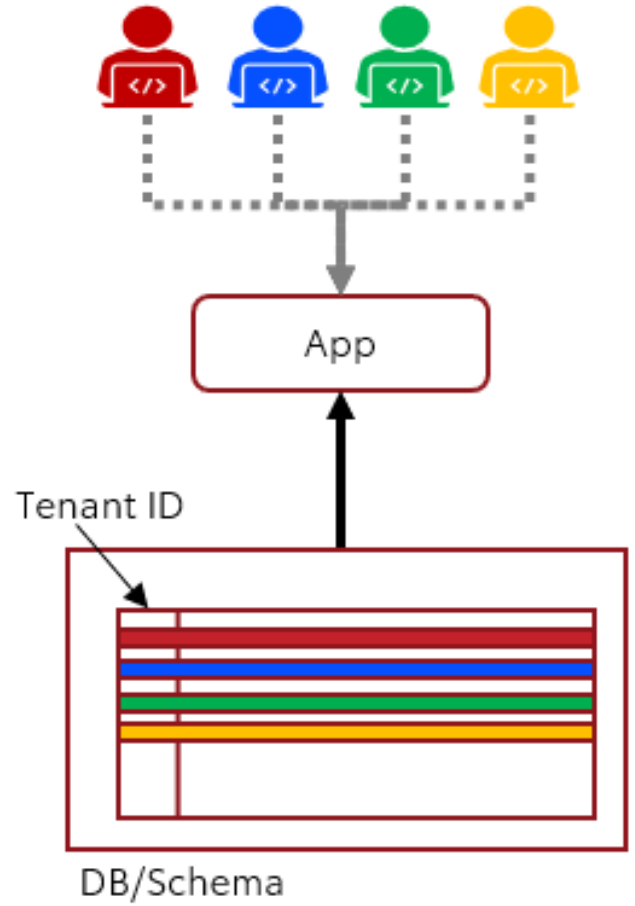
A 'Content Model Definition' is a collection of document containers with inheritance enabled.

Each 'Content Model Definition' will belong to one specific tenant.

Each definition will be assigned a unique identifier and a corresponding Tenant ID. Any document container instantiated within the definition will automatically inherit the corresponding Tenant identifier. This allows the logical separation of the Documents and the corresponding metadata.

All definitions will be configured in a single Document Lake repository (database)

Pattern 3 – Single APP Instance, Common DB Schema

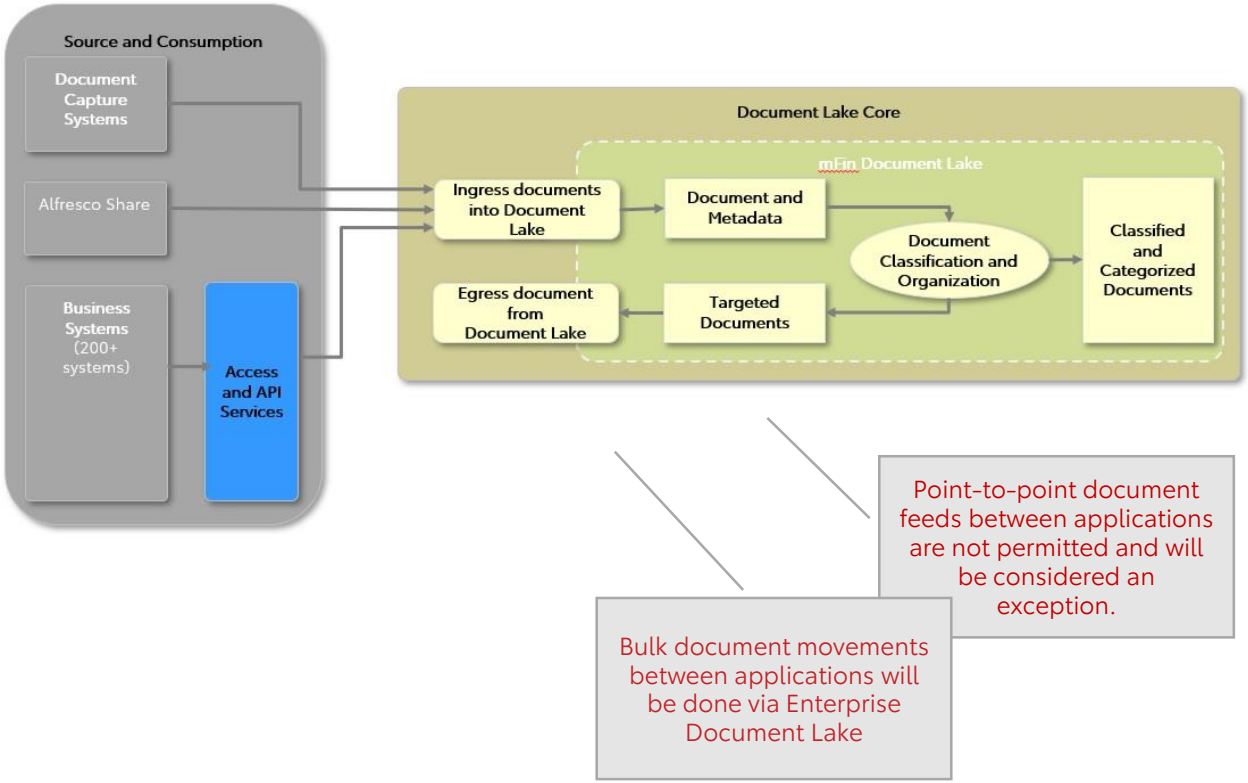


TOYOTA FINANCIAL SERVICES

Ingress documents into Document Lake

Requirements:

1. Complete Intake Process
2. Define Acceptance Criteria
3. Engage SMEs
4. Provide Metadata, Sample Documents
5. Document Pattern & Connectivity



How do I leverage the Enterprise Data Platform to ingest documents that are currently stored in multiple locations and in different formats?

Acceptance Criteria

- Ingestion is processed without any errors.
- Documents are classified and categorized correctly.

How

- The Document Lake supports ingestion of documents from multiple sources and in multiple formats. Accepted types include:
  - Paper documents: Paper documents will be digitized and ingressed into Enterprise Document Lake.
  - Electronic documents, affiliated or not affiliated with business systems: This type of documents will be ingested into the Document Lake through 'Push' or 'Pull methods, depending on the business processes around these documents.
- Ingested documents are next indexed, classified, and organized in the Document Lake Platform.
- All relevant meta data needs to be available during of the Document Lake ingestion process
- The API layer is the preferred method to ingest documents into the Document Lake.
- Alternative ingestion methods, applicable in certain scenarios, are as follows:
  - Alfresco Share application to ingest documents.
  - Document capture solutions to ingest scanned, faxed, and emailed documents.
- Bulk import patterns will be used for large document volumes
- Get started by completing Document Lake intake process!

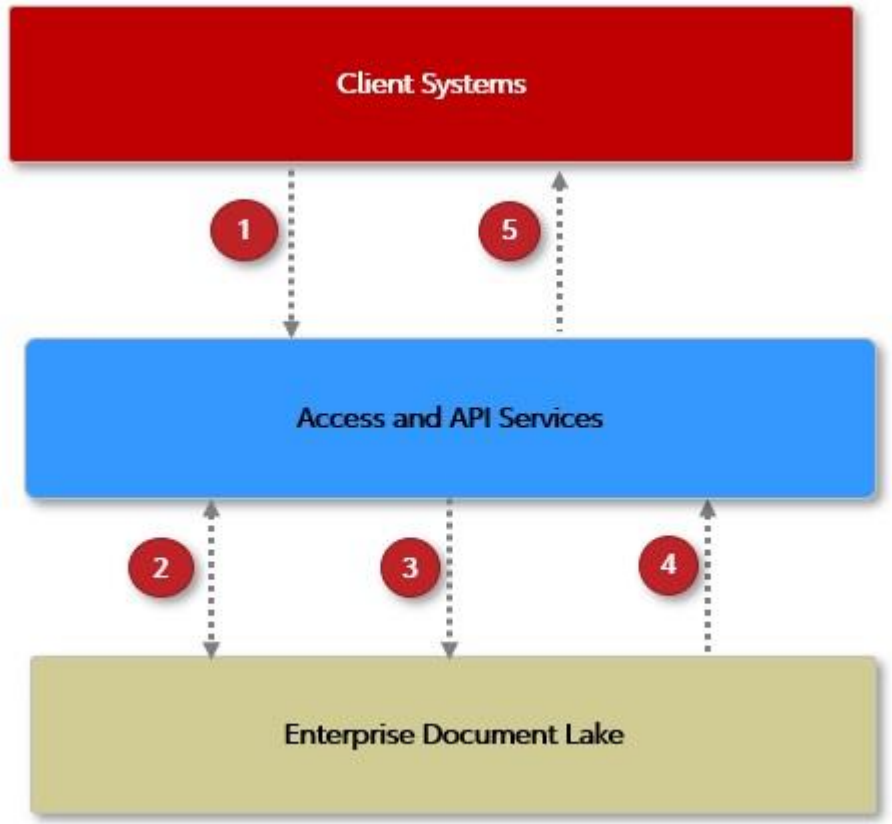
TOYOTA FINANCIAL SERVICES

Ingress Pattern 1 – Add Document

Ingress Pattern 1 reflects adding individual documents and associated metadata from client systems via published REST APIs, but without the capability to generate and pass security tokens.

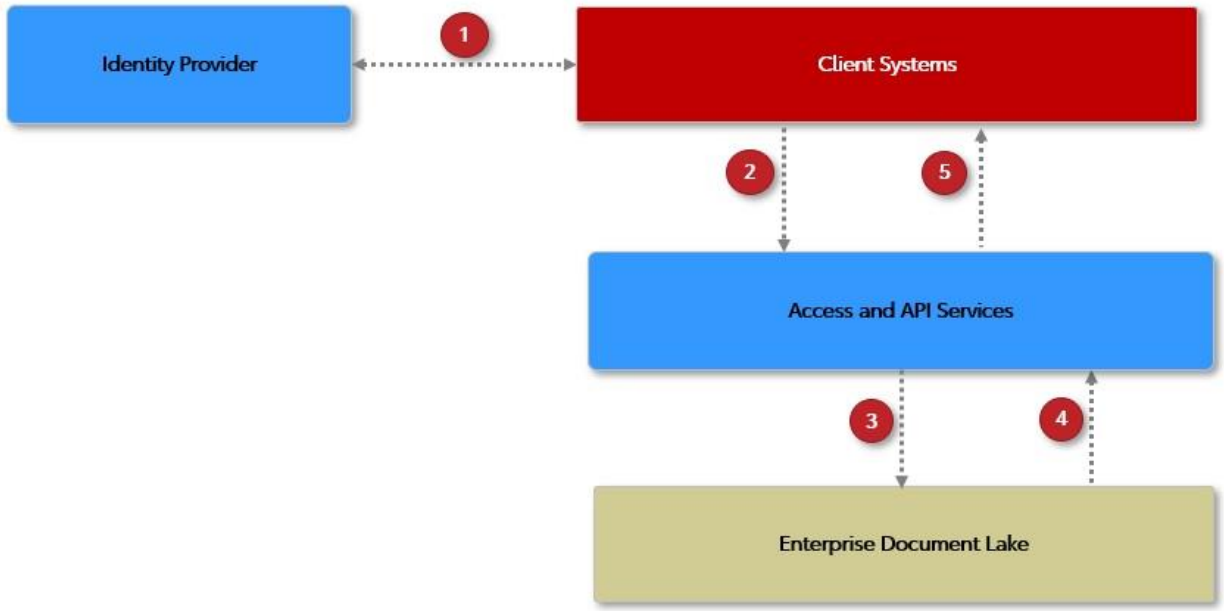
How

1. Client system will call the published REST API to add document.
2. The API Gateway will perform basic authentication utilizing a service account to establish connection to Document Lake.
3. The API Gateway will make a 'PUT' call to Document Lake to add document.
4. Document Lake will return a unique document identifier (or an error code).
5. API Gateway will pass the unique document identifier (or an error code) to the client system.



TOYOTA FINANCIAL SERVICES

Ingress Pattern 2 – Add Document



Ingress Pattern 2 reflects adding individual documents and associated metadata from client systems via published REST APIs, and with the capability to generate and pass security tokens.

How

1. Client system will authenticate with the identity provider.
2. Client system will call the published REST API to add document along with the security token.
3. The API Gateway will make a 'PUT' call to Document Lake to add document along with the security token.
4. Document Lake will authorize the user in the token and return a unique document identifier (or an error code).
5. API Gateway will pass the unique document identifier (or an error code) to the client system.

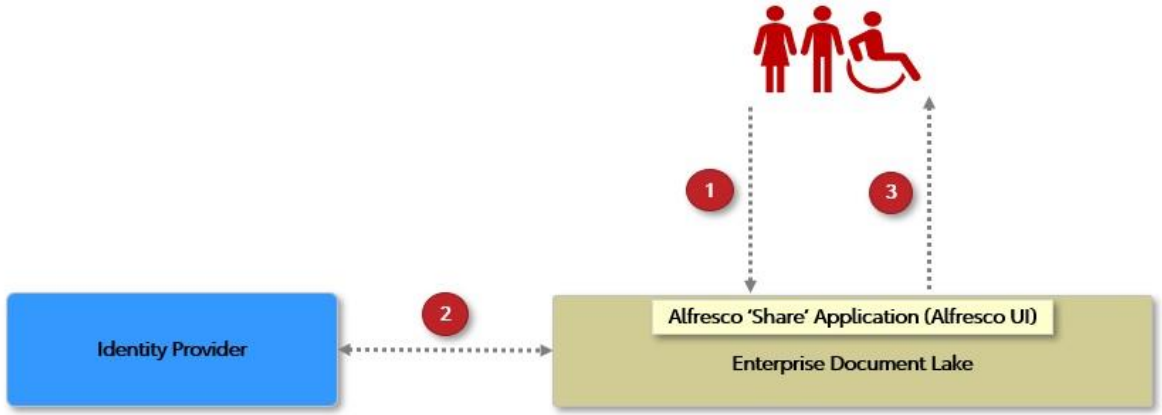
TOYOTA FINANCIAL SERVICES

Ingress Pattern 3 – Add Document

Ingress Pattern 3 reflects adding individual documents and associated metadata directly via the SSO enabled Alfresco Share application.

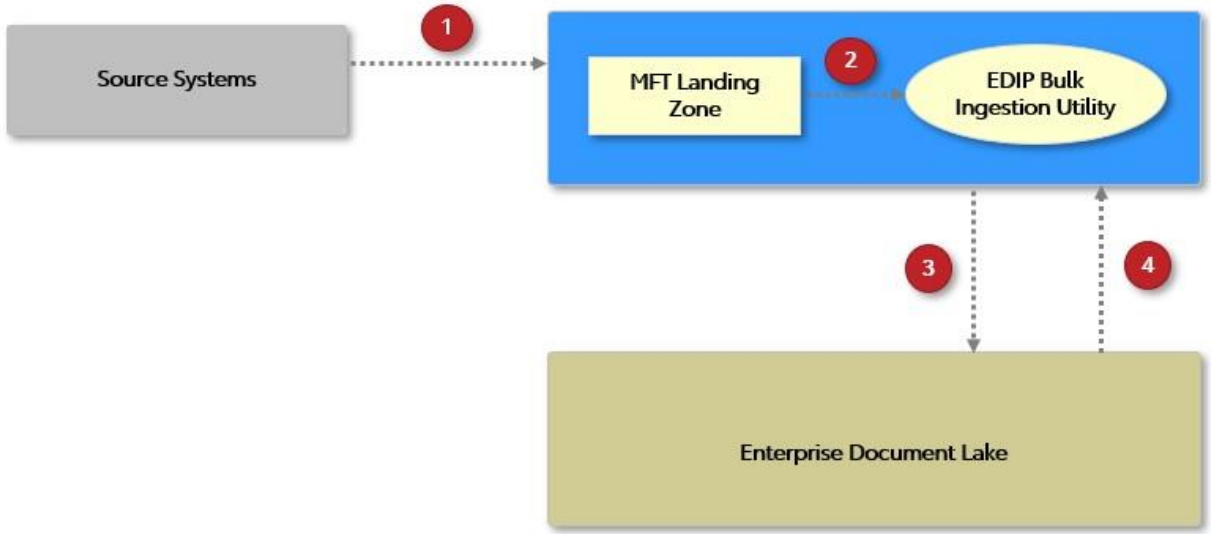
How

1. User will open the Alfresco Share Application.
2. Document Lake will redirect the user authentication to TFS Identity Provider for authentication. Authenticated user will be authorized by Document Lake.
3. User will add document into Document Lake.



TOYOTA FINANCIAL SERVICES

Ingress Pattern 4 – Bulk Document Ingestion



Ingress Pattern 4 reflects bulk loading documents and associated metadata from source systems.

How

1. Source system will utilize the MFT interface to push document batch and associated control file into a landing zone.
2. An event will be triggered to launch EDIP Bulk Ingestion Utility.
3. The EDIP Bulk ingestion utility will:
    I. Perform preprocessing
    II. Read the control file
    III. Extract document and associate the corresponding metadata
    IV. Add document and metadata into Document Lake.
4. Document Lake will return a unique document identifier (or an error code) for each committed document. The EDIP Bulk ingestion utility will reconcile and generate exceptions notice, if needed.

TOYOTA FINANCIAL SERVICES

Egress documents from Document Lake

How do I egress documents that are stored in the Enterprise Data Platform?

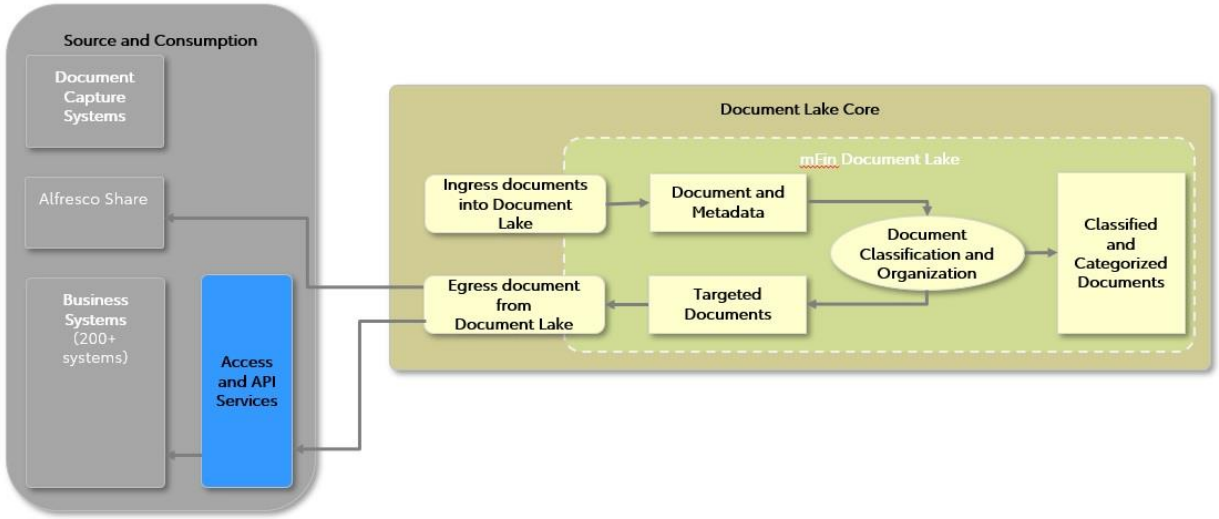
1. Developer File Catalog
2. Interface Ready
3. Audit and traceability of transactions

Acceptance Criteria

- Appropriate document metadata is provided for search and retrieval.
- Documents consumed by factories from Document Lake is secure, timely and seamless.

How

- Factories shall participate in the Document Lake egress process.
- Factories will primarily use API Layer to consume documents from Document Lake.
- In certain scenarios, Factories will use the Alfresco Share application to retrieve documents.
- Factories can retrieve documents by providing the appropriate metadata as search criteria.

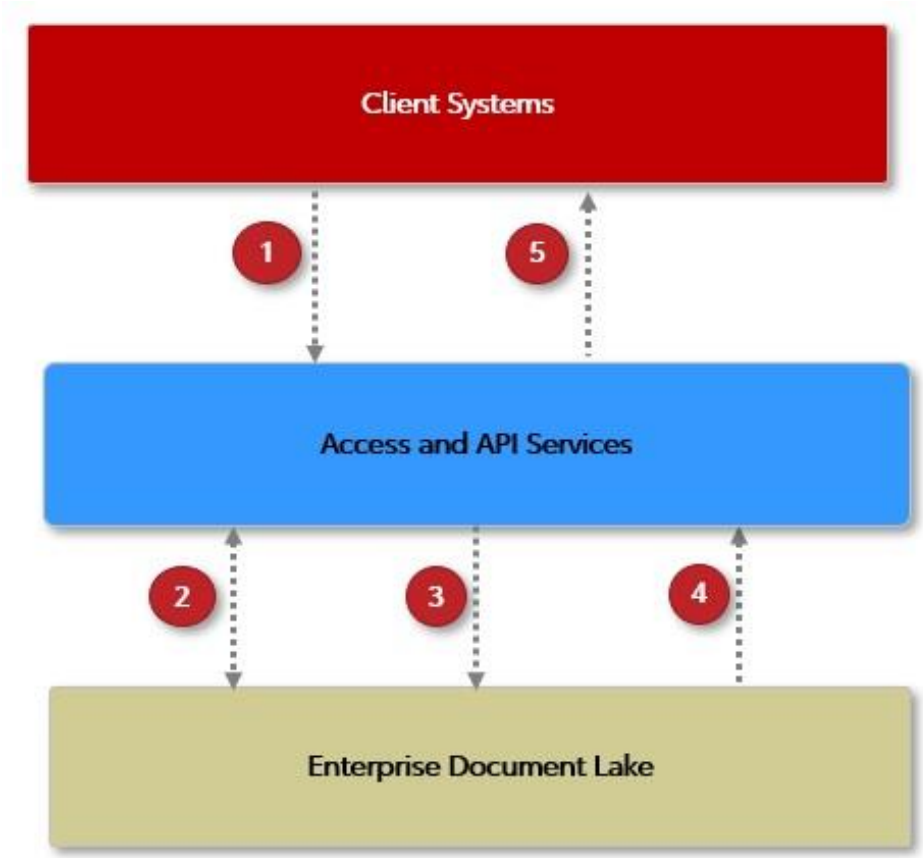


Point-to-point document feeds between applications are not permitted and will be considered an exception.

Bulk document movements between applications will be done via Enterprise Document Lake

TOYOTA FINANCIAL SERVICES

Egress Pattern 1 – Retrieve Document



Egress Pattern 1 reflects retrieving individual documents and associated metadata from client systems via published REST APIs, but without the capability to generate and pass security tokens.

How

1. Client system will call the published REST APIs to retrieve document.
2. The API Gateway will perform basic authentication utilizing a service account to establish connection to Document Lake.
3. The API Gateway will make a 'GET' call to Document Lake to retrieve document.
4. Document Lake will return content.
5. API Gateway will return content to the client system.

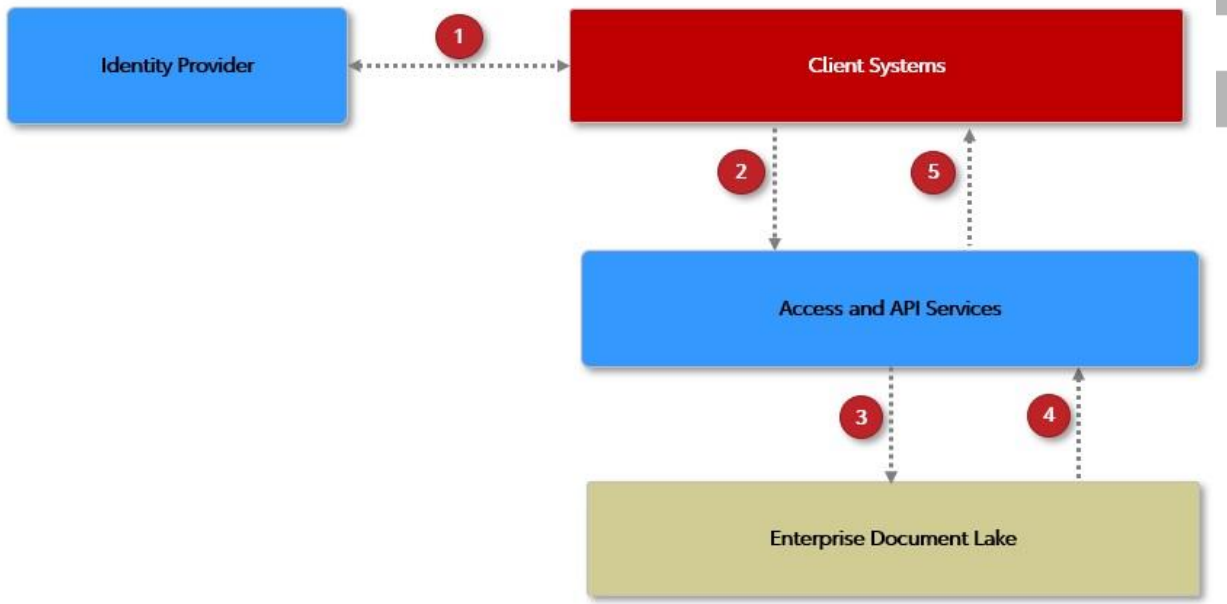
TOYOTA FINANCIAL SERVICES

Egress Pattern 2 – Retrieve Document

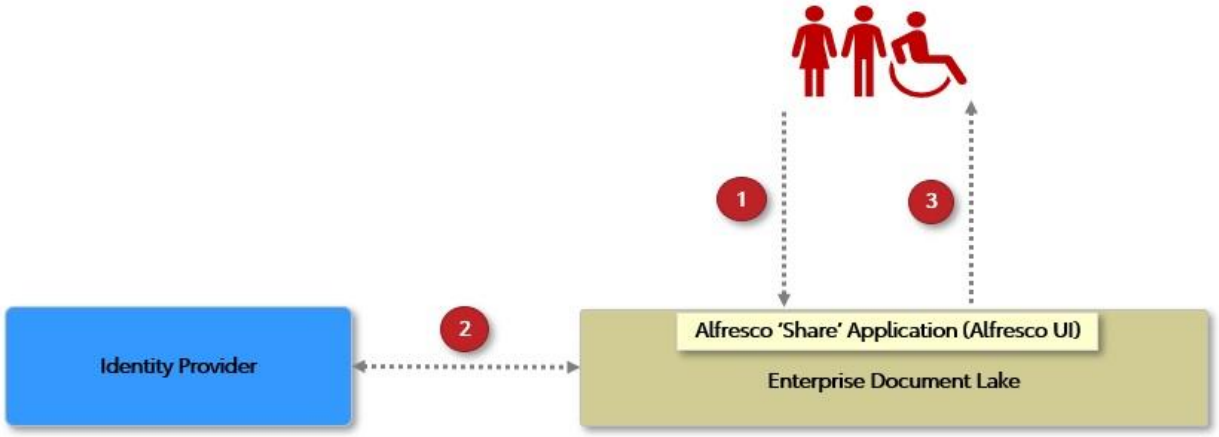
Egress Pattern 2 reflects retrieving individual documents and associated metadata from client systems via published REST APIs, and with the capability to generate and pass security tokens.

How

1. Client system will authenticate with the identity provider.
2. Client system will call the published REST API to retrieve document along with the security token.
3. The API Gateway will make 'GET' calls to Document Lake to retrieve document along with the security token.
4. Document Lake will authorize the user in the token and return the content.
5. API Gateway will pass the content to the client system.



Egress Pattern 3 – Retrieve Document



Egress Pattern 3 reflects retrieving individual documents directly via the SSO enabled Alfresco Share application.

How

1. User will open the Alfresco Share Application.
2. Document Lake will redirect the user authentication to TFS Identity Provider for authentication. Authenticated user will be authorized by Document Lake.
3. User will retrieve document by searching document metadata into Document Lake.

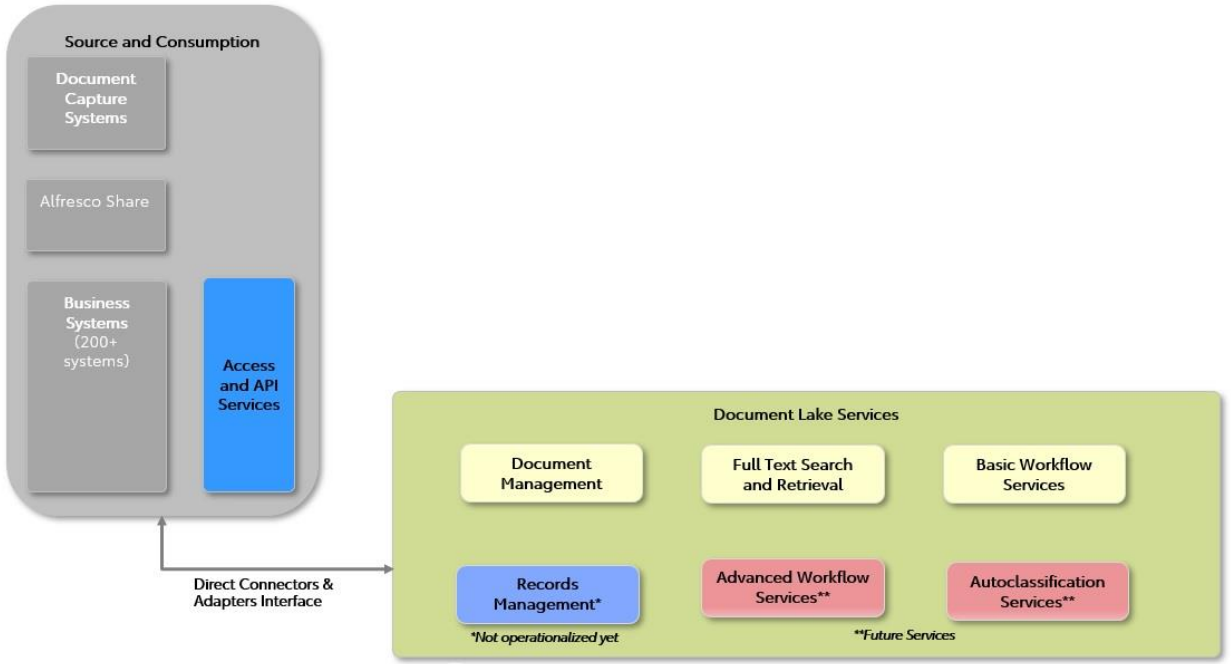
TOYOTA FINANCIAL SERVICES

Extended Document Lake Services

The “Source and Consumption” modules can leverage advanced document management services offered by Document Lake. These services will provide features and functionality beyond the simple ingress and egress of documents.

How

- Business systems (e.g. SalesForce, SAP) can provide connectors to integrate with Document Lake. These connectors will allow end users to upload and retrieve documents, trigger and participate in workflows, seamlessly transfer and update metadata, enable links between business objects and corresponding documents – to name a few.
- Document capture systems (TBD) and their connectors can integrate with Document Lake, not only to upload documents and metadata, but also can receive tasks and assignments (e.g. indexing) from Document Lake. In addition, non-OCR-Readable documents can be pushed to the Capture systems to make them OCR-Readable in order to provide full text indexing and search capabilities for those documents.
- In non-headless implementations, the primary end user application will be Alfresco Share UI. This application will allow the users to work with all relevant Document Lake Services as applicable to their use cases.



TOYOTA FINANCIAL SERVICES

Document Inventory and Content Modeling Process

Phases	Description	Deliverables	Responsibility
Inventory Documents	• Understand document types, generation sources, volume, formats, business use cases	Inventory list	• Business stakeholders • Factory SMEs • Content Modeling team
Explore Document Metadata and Security	• Gather metadata to be used for identification, retrieval, processing • If applicable, identify and define records management parameters • Evaluate security requirements	Taxonomy definition	• Content Modeling team • Factory SMEs • Business stakeholders
Build Content Model	• Build content model	Content Model definition	• Content Modeling team • Factory SMEs
Content Model Validation	• Validate the content model with Business stakeholders, Application Owners, Enterprise Architecture and Platform teams. • Realign and enhance the content model based on the feedback	Updated Content Model definition	• Content Modeling team • Factory SMEs • Business stakeholders

RACI Chart

Tasks	Business Stakeholders	Factory SMEs	Enterprise Document Lake	Enterprise Architects
Inventory Documents	R	C	A	I
Explore Document Metadata and Security	C	A	R	I
Build Content Model	I	A	R	C
Content Model Validation	C	A	R	I

Enterprise Document Inventory and Content Modeling is a foundational step in building the Document Lake platform.

The process and resulting Content Model allows the creation of identifiable containers by which documents are classified and categorized. The model lays the basis for the inter-relationships of the document types and applicable records management processes and procedures.

Acceptance Criteria

- Ensure that the document type identities always maintained the uniqueness.
- Model will be reviewed to ensure the completeness and model capture of requirements.
- Ensure the model follows consistent naming standards.
- Model has been arranged for readability and the definitions are clear, complete and accurate.
- Ensure the model metadata match the actual data.

How

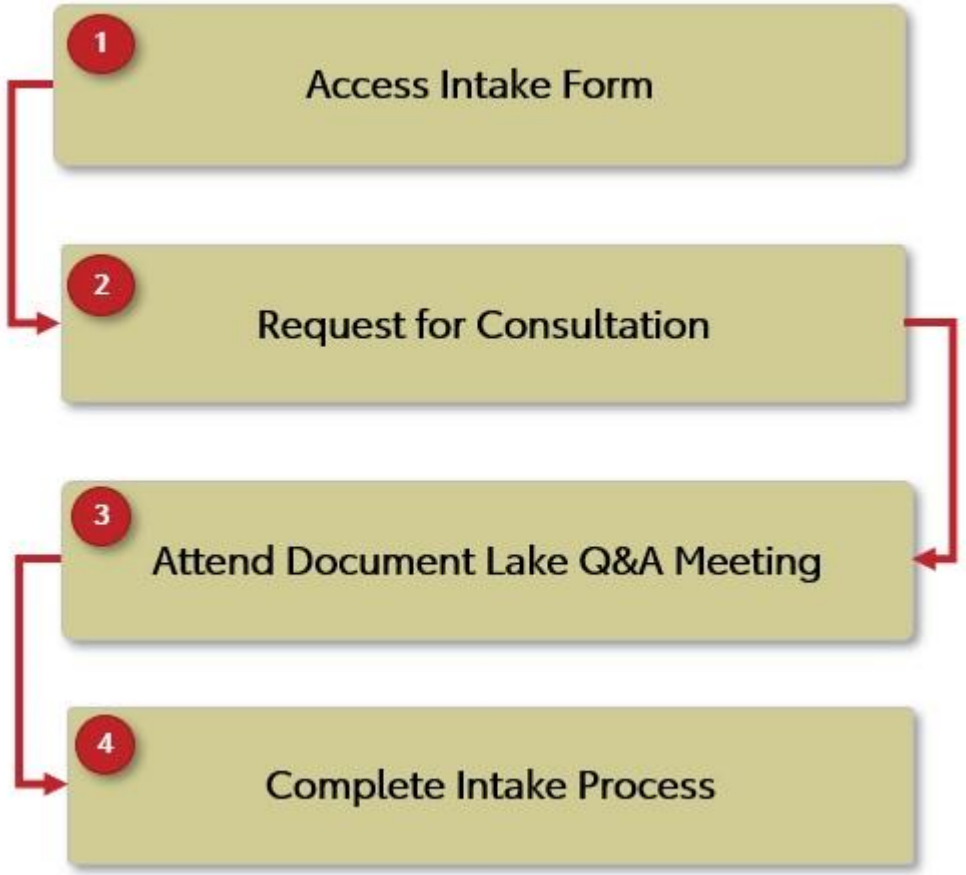
- All factories will participate in the intake process.
- All Factories will ensure document structures support their business area
- All Factories will collaborate with Document Lake Factory to preserve enterprise content model by capturing knowledge in explicit form for future project to use.
- All physical, logical data models and document lineage have to be identified & documented (source/target) to provide a strong understanding of the document requirements.

TOYOTA FINANCIAL SERVICES

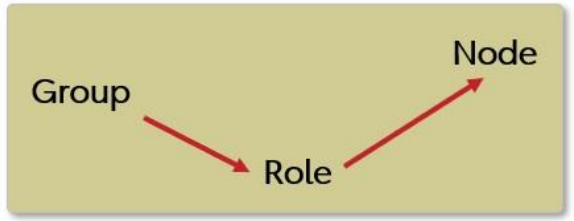
Document Lake Intake Process

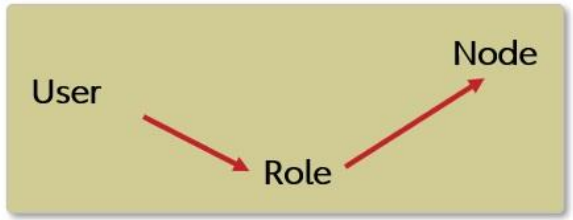
Document Lake Intake Process consist of the following steps:

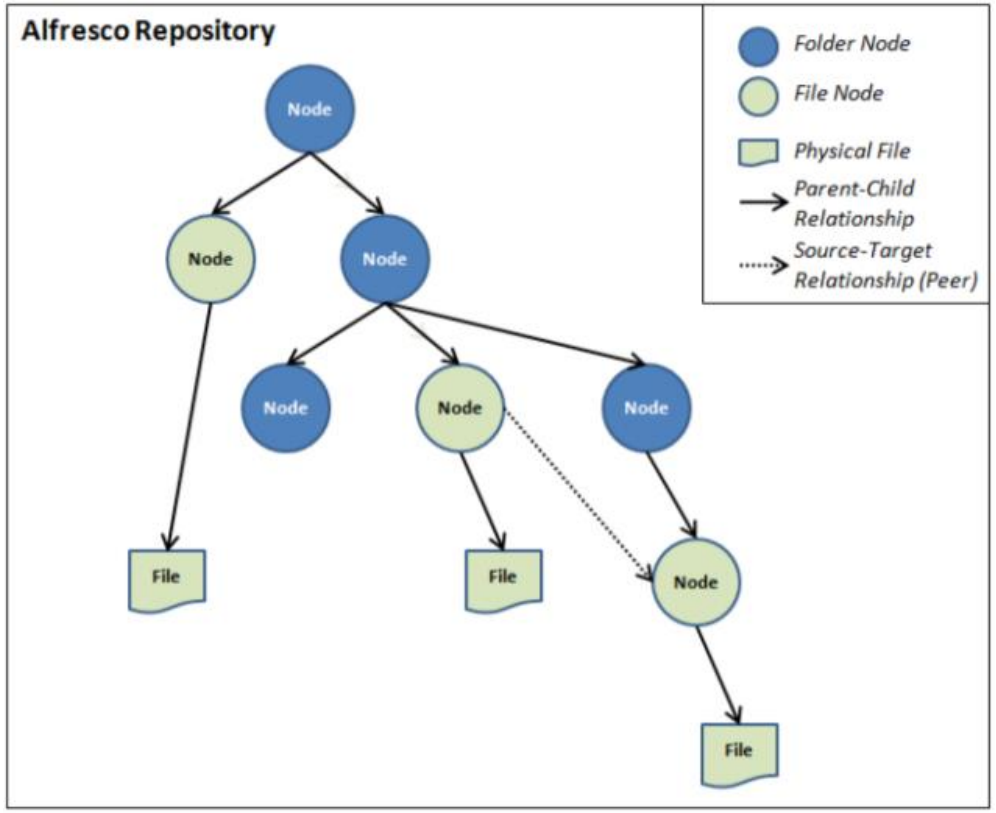
1. Access intake form here.
2. Request for consultation in completing the form – DocumentLakeFactory@internal.Toyota.com
3. Attend bi-weekly DocumentLake Q&A meeting (TBD)
4. Complete intake process.



Recommended Best Practices from an Implementation Point of View – By Alfresco







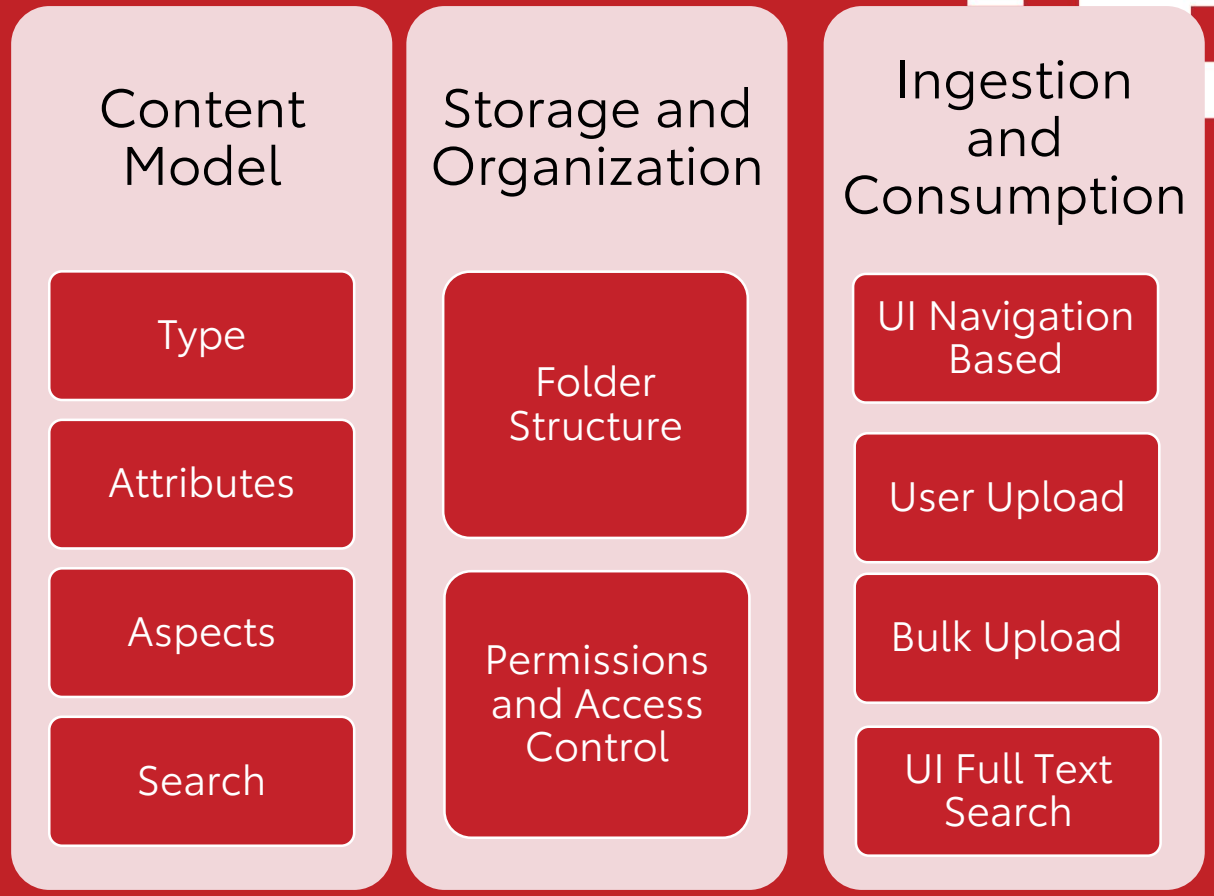
1. Limit Security Groups Hierarchy to 5
    I. When possible, limit the max user group hierarchy depth to 5
2. Number of Sites
    I. It has relative influence on performance
    II. Keep the number of sites below 5000
3. Limit the folder hierarch depth
    I. It has an impact when browsing and performing document actions under a certain folder
    II. For better user experience, keep the folder depth < 5 levels
    III. When possible, limit the max depth of a folder hierarchy to 15 levels
4. Configure Alfresco to automatically distribute (Plan for folder allocation) the documents into a folder structure
5. Clean the trashcan
    I. It can lead to performance gains especially for long-lived repositories
6. Disable thumbnails and documents previews (If you are not making any user of thumbnails and document previews)
7. Enable full-text index on the documents when required
8. Content Modelling:
    I. Do not change Alfresco out-of-the-box content model
    II. Add properties that you really need as it is hard to remove properties later
    III. Use aspects when you can, they are flexible

TOYOTA FINANCIAL SERVICES

Doc Lake's Internal Design Considerations:

- Google like search – Ability to filter further with facets filter
- Simple API based access for Document Lake
- Document storage mechanism within the Document lake is completely transparent from the consuming applications
- Acceptable SLAs - Performance
- Permissions based Document Access Control
- Auditing to track actions taken on documents.
- Version Control to track edits to documents and recover old versions
- Records Management governing out of compliance documents deletion
- Standardized egress and ingress integration patterns
- Separations of Concerns

Separation of Concerns

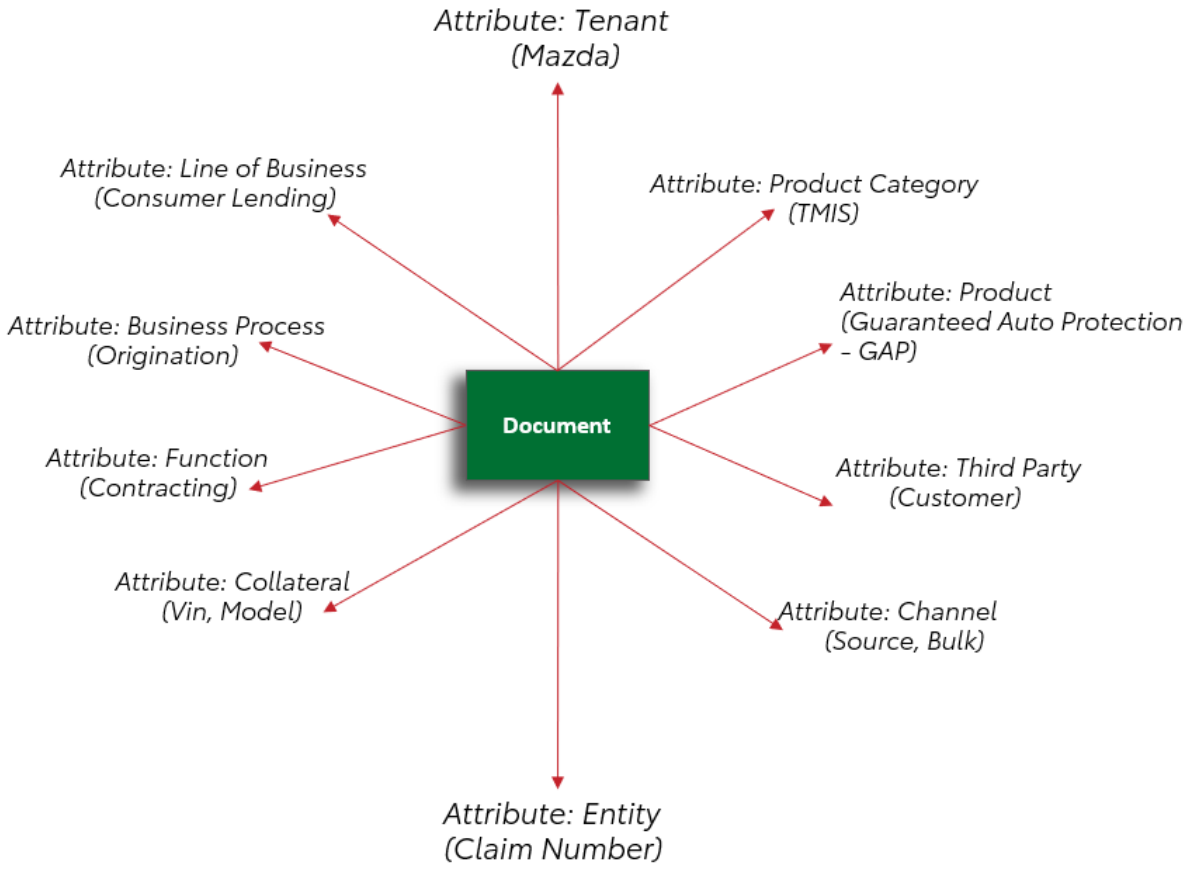


TOYOTA FINANCIAL SERVICES

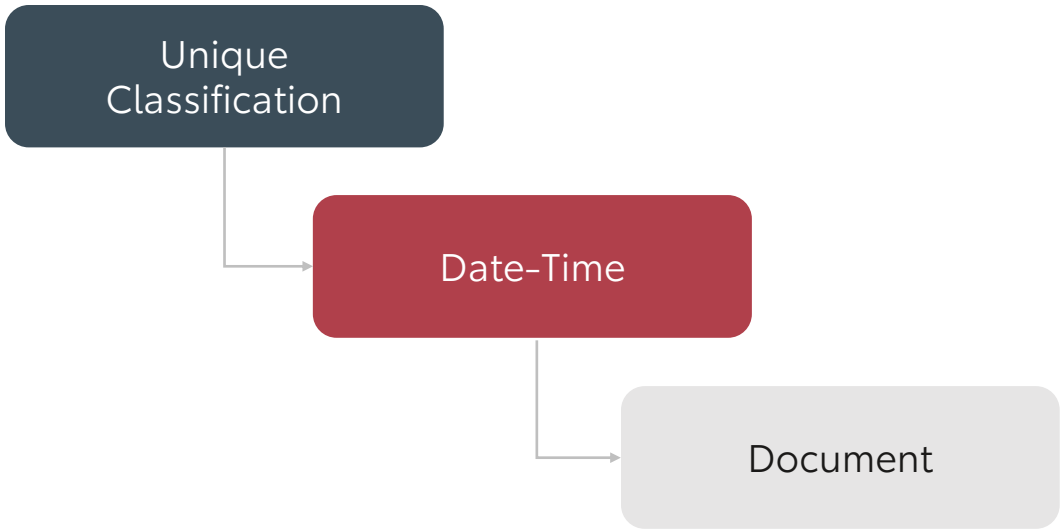
Doc Lake's Internal Design Considerations:

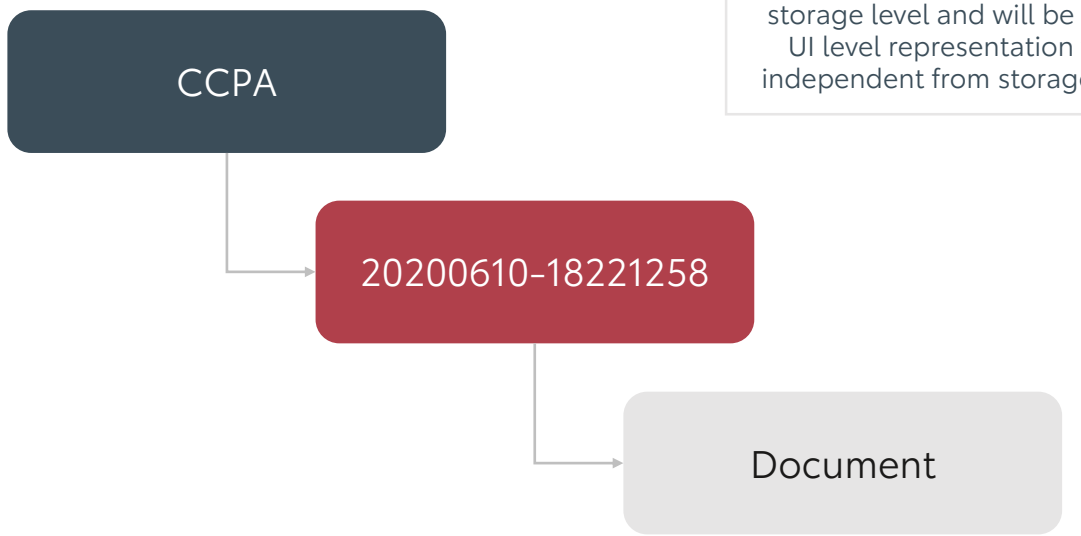
1. Making documents the center point of the design
2. Association of relevant characteristics as attributes to documents
3. Keeping storage folder structure as flat as possible
4. Identify a unique business classification/categorization that this document can be associated with and use it as the folder name to store the document
5. Use the folder name along with other attributes in your search criteria to limit the search results
6. Align Indexing with business needs for improved search performance
7. Separate documents by retention timelines
8. Know document destruction schedules

Storage & Retrieval



Document Centric Storage Model & Example





This folder structure is at the storage level and will be generic. UI level representation will be independent from storage model.

```
{
 "query": {
 "query":"(ANCESTOR:'workspace://SpacesStore/349eaa20-7d79-46a4-99cd-a786e02488d3' and
                    lm:ContractNumber:* 70400100000010001 * AND NOT lm:Restricted:true)"
 },
 "include":[
 "properties"
 ]
}
```

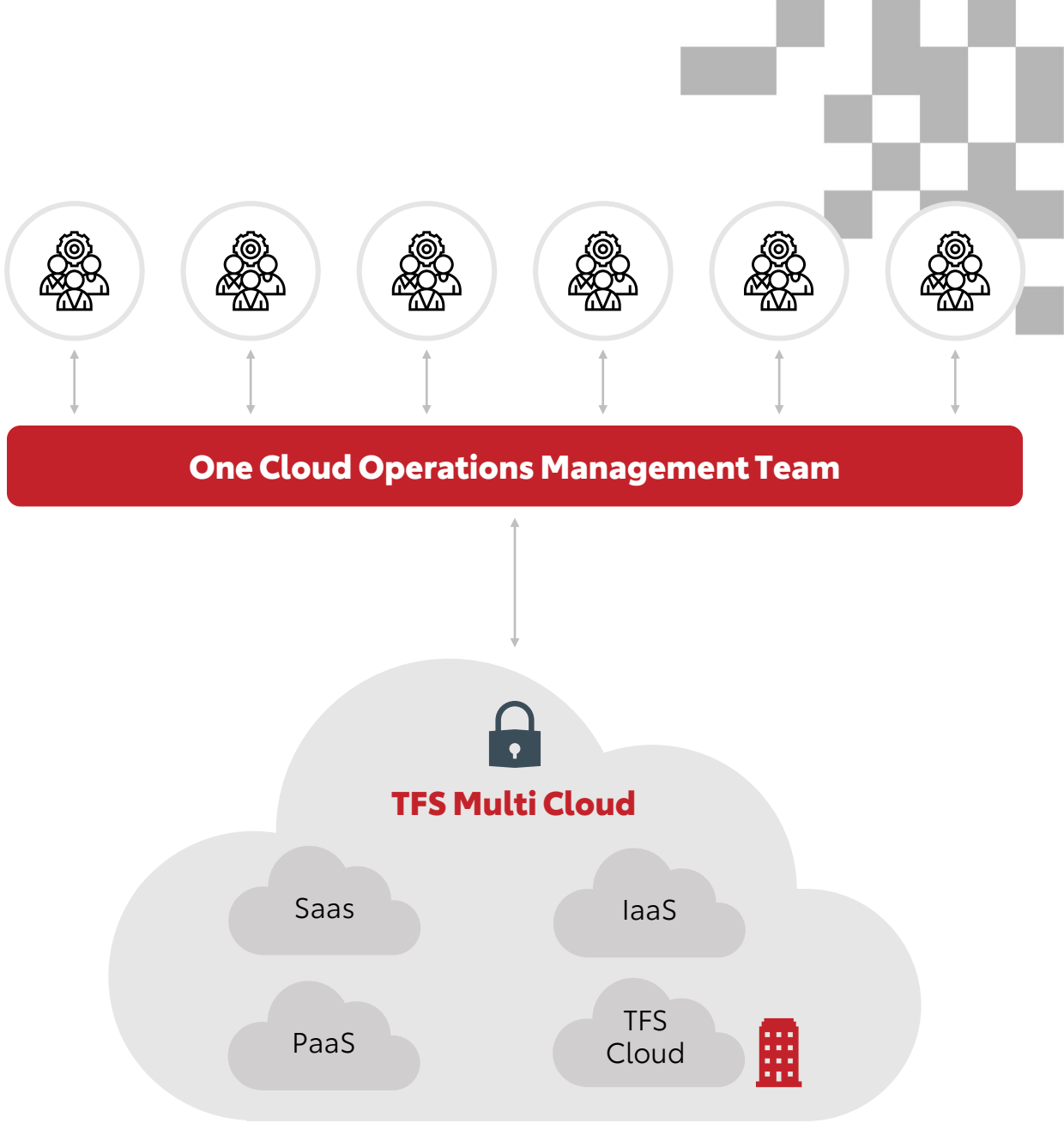
- Red section of the query will limit the search ONLY to the documents in “CCPA” folder
- Performance improvement by orders of magnitude

TFS Cloud

Factory Cloud Enablement through Unified TFS Cloud Operations

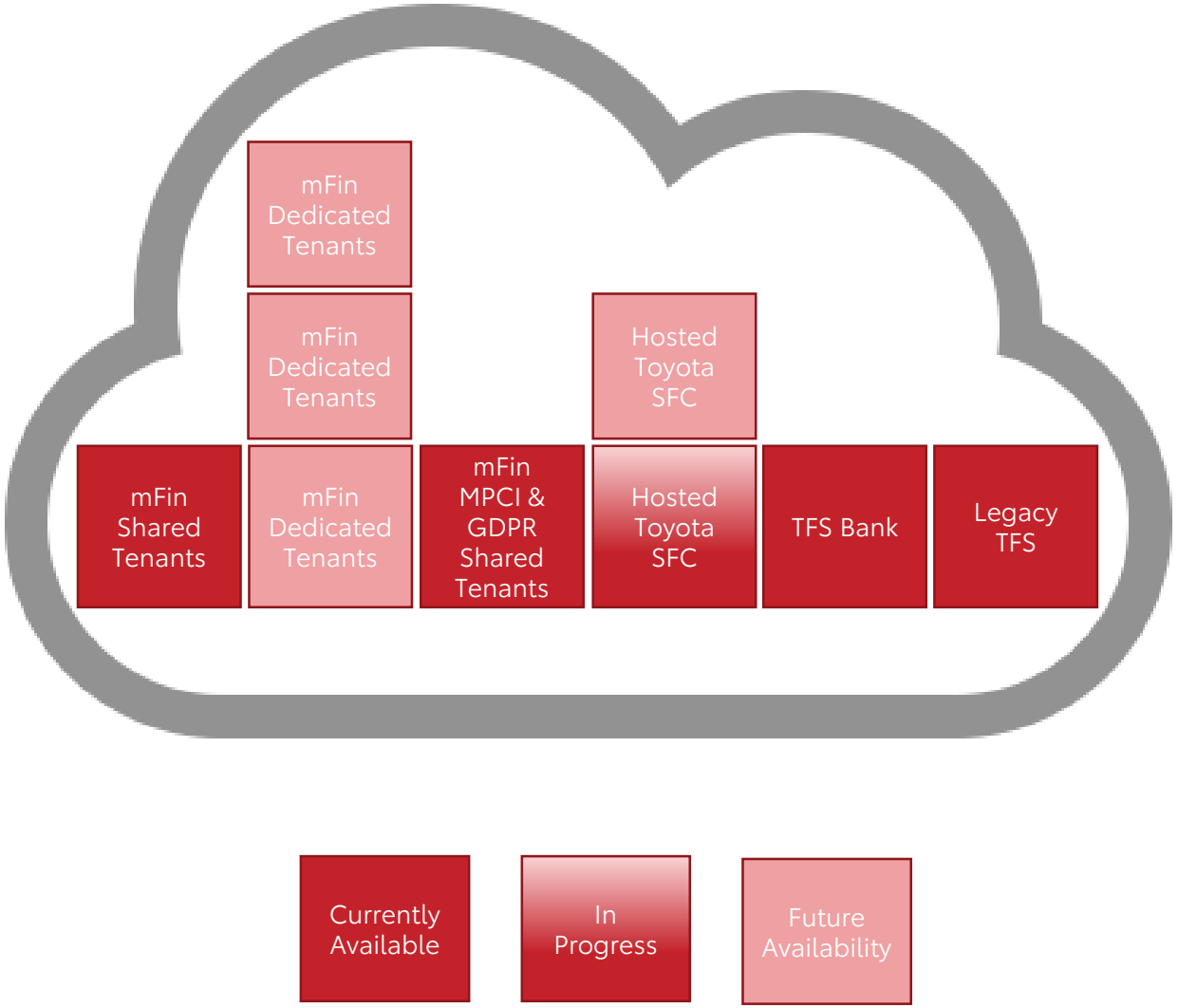
Unified TFS Data Center Management

- Multi-cloud approach with a single environment management operations and team
- Enabling a Cloud-like experience to Factories and other stakeholders, no matter where the infrastructure or data storage resides.
- Enhancing business agility by designing for future workloads and technology assets in cloud
- For Factories, our data center resources will behave as just another cloud provider, happens to be managed by us.
- For Factories, the agility, stability, security and quality matters not the physical location of the resources.
- **Process Shift:** Moving from manual infrastructure processes and waterfall steps, to automated environments and an agile automated self-service culture.



TOYOTA FINANCIAL SERVICES

TFS Cloud Landscape



TFS Cloud is the combined environments across multiple regions and hosting technologies (AWS, TFS Data Centers and VMC).

Where should your application be hosted within the TFS Cloud?

Considerations

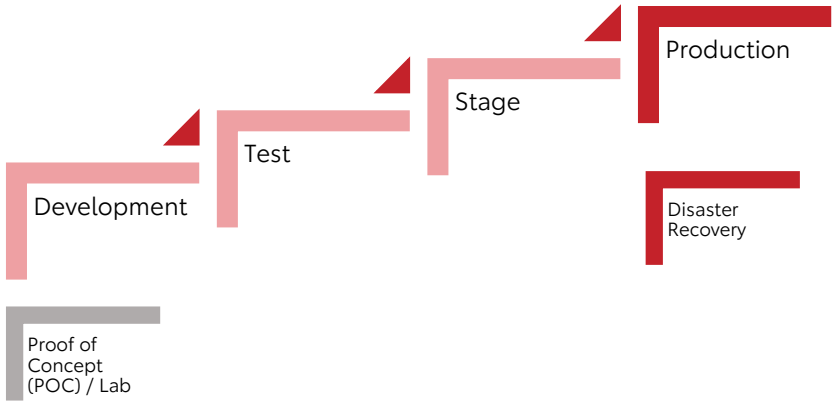
- Tenant isolation needs and external regulatory requirements are the primary indicators.
- Regional needs and proximity to users and supporting systems can further refine where to host systems within the aligned part of the TFS cloud.

Solution

Below are the primary environments within TFS cloud and should considered the default for all new deployments aligned with mFin requirements.

- **TFS Bank**
  - Hosting TFS Bank Applications
- **mFin Shared**
  - Hosting mFin compliant applications for tenants that can use a shared tenant environment
- **mFin Dedicated (future)**
  - Hosting mFin compliant applications for tenants requiring dedicated environments
- **Hosted Toyota SFC**
  - Dedicated environment for partner SFCs.

TFS Cloud Environments



Environment	Used for?	Where?
Proof of Concept (POC) / Lab	Early product or idea evaluation and experimentation	Isolated POC environment; no TFS system connections or data
Development	Active development and unit testing	Sub-production environment; Dev/Test VPC
Test	Exploratory and User Experience testing	
Stage	Prod-like environment for comprehensive testing including UAT, End to End, and Performance	Sub-production environment; stage VPC
Production	Live environment for use	Production environment
Disaster Recovery	Production-equivalent environment for emergency failover	Production/DR environment

What environments should I use for different development, testing, and deployment tasks?

Considerations

- How to coordinate with other teams on shared activities such as End to End testing?
- How to test new technologies and ideas that may go beyond what current systems support?
- How to balance resources and associated cost?
- How to enable innovation and development with speed, as well as strong confidence in workloads supporting business operations?

Solution

Multiple environments can be provisioned to support all necessary activities in enabling the business. From experimentation, to development and testing, to ensuring stability and high availability to business-critical functions, TFS' Cloud Platform provides multiple environments to support each use case.

FAQ for Proof of Concept (POC) Environments

1 How do I get a POC environment?

POC environments are currently manually provisioned and can be requested by Cloud Engineering Service Request/Pitstop meeting.

2 How is POC different than Dev?

POC environment is completely isolated from existing TFS Production & Sub-Production environments. Factory teams will have more direct access and control over the cloud infrastructure and services. However, they won't be able to cannot connect or integrate with other TFS applications (Prod & Sub-Prod).

3 So I can do anything I want in a POC environment?

You have control over the Cloud instances and services that you provision in the environment but are still using company resources and are responsible for appropriate use and managing cost.

4 How long can I use a POC environment?

POC environments are temporary environments meant for evaluation and experimentation use cases. Typically for 30 days but can extended up to 90 days upon request to the cloud team.

5 How do I get my data into my POC environment?

Company data cannot be transferred into a POC environment. Factories are encouraged to use dummy/mock-up data to evaluate. POC Environments are frequently destroyed, it should not host any critical data, applications, code or configs.

6 How do I use my CI/CD pipeline in my POC environment?

Currently enterprise services are not available within, however please use the similar guiding principals in the POC environments as you would in traditional environments (Prod & Sub-Prod).

7 Why can I see and control so much more in a POC environment than my Dev environment?

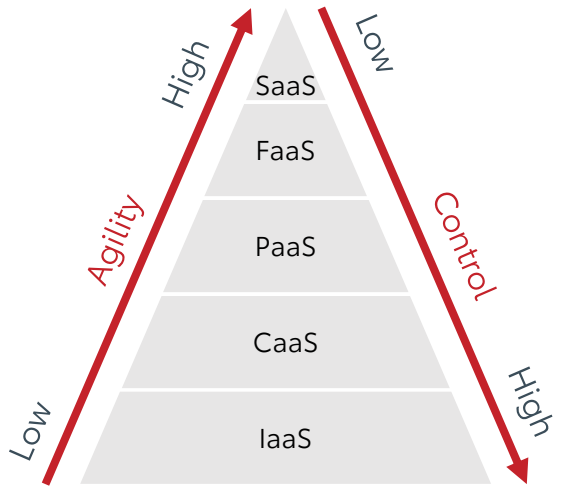
As with all digital products, features and capabilities are being iteratively improved. POC environments allow more direct control and access at the cost of losing enterprise services. Future changes will allow for expanded control with supporting guardrails in other environments.

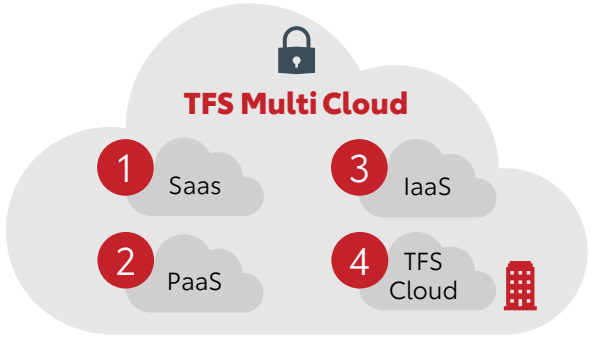
8 Is POC environment shared or dedicated to each factor?

Currently Shared, moving towards dedicated.

TOYOTA FINANCIAL SERVICES

What Cloud should I target for my product?





Cloud selection preference

Consequences:

Speed – Business gets quicker access to solutions.

Acceleration – Your team's choices will accelerate our move to the cloud

Simplicity – Your team can focus on business needs and differentiation rather than upgrading and sustaining multiple on-site vendor technologies.

Satisfaction – Modern cloud platforms will allow quicker rollout of better self-service functions.

What cloud operating environment should I choose for my product?

Acceptance Criteria

- The cloud operating environment and the application must meet the requirements of the customer
- The goal is to focus on delivering features and business value and not to spend too much time or money creating, uplifting, scaling, or patching the infrastructure
- Select a cloud solution which your team is capable of supporting and does not limit growth of TFS as a digital company

How

- Follow the simple cloud first prioritization for your team

SaaS=>FaaS=>PaaS=>CaaS=>IaaS

- SaaS = "Software as a Service"
- FaaS = "Function as a service"
- PaaS = "Platform as a service"
- CaaS = "Container as a service"
- IaaS = "Infrastructure as a service"

THESE ARE GENERAL RULES ONLY!

TOYOTA FINANCIAL SERVICES

Software as a Service (SaaS)

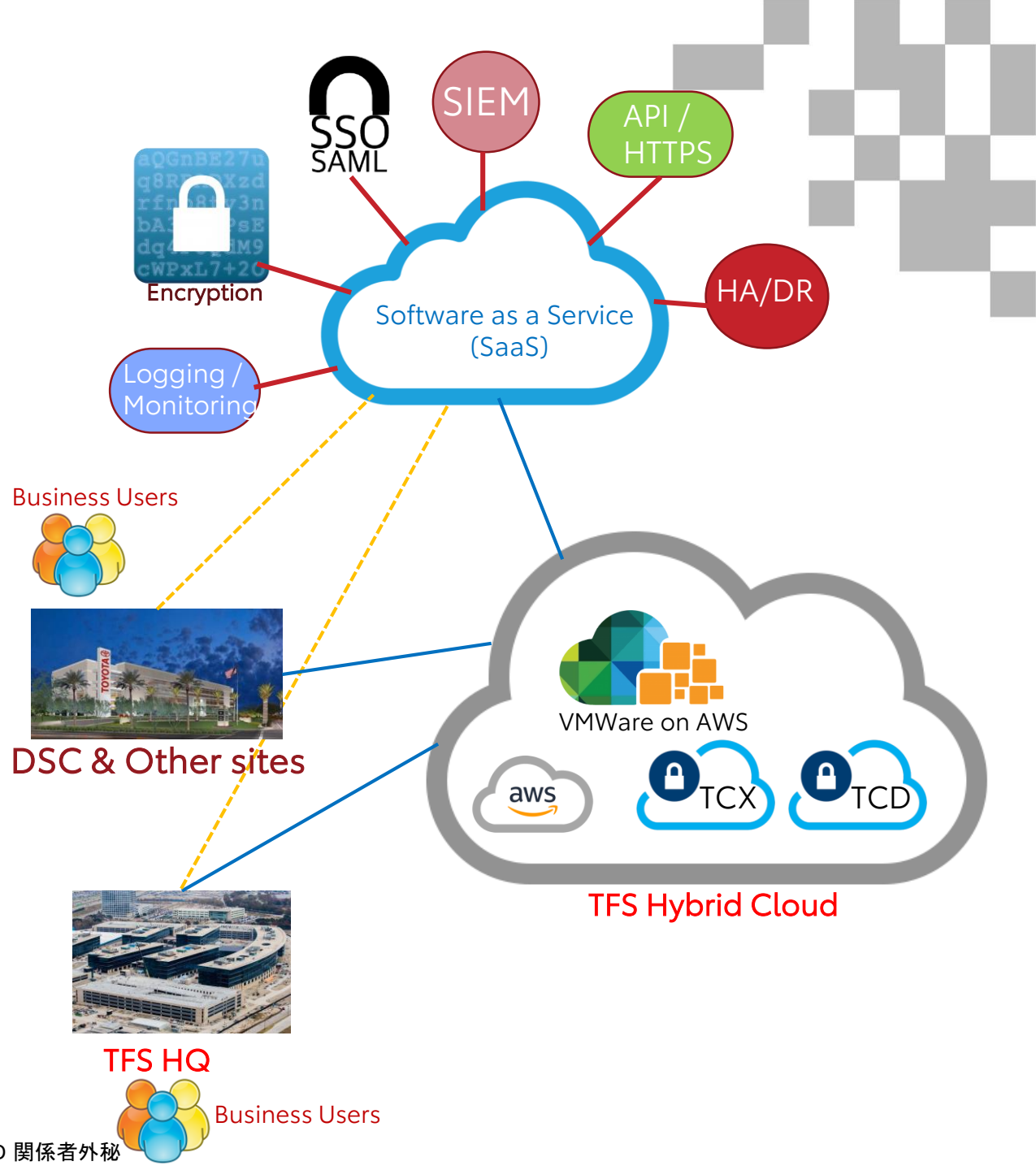
Factory needs to quickly deploy a solution to meet business requirements, is easy to use, scales and doesn't require inhouse expertise to build, manage and maintain.

Considerations

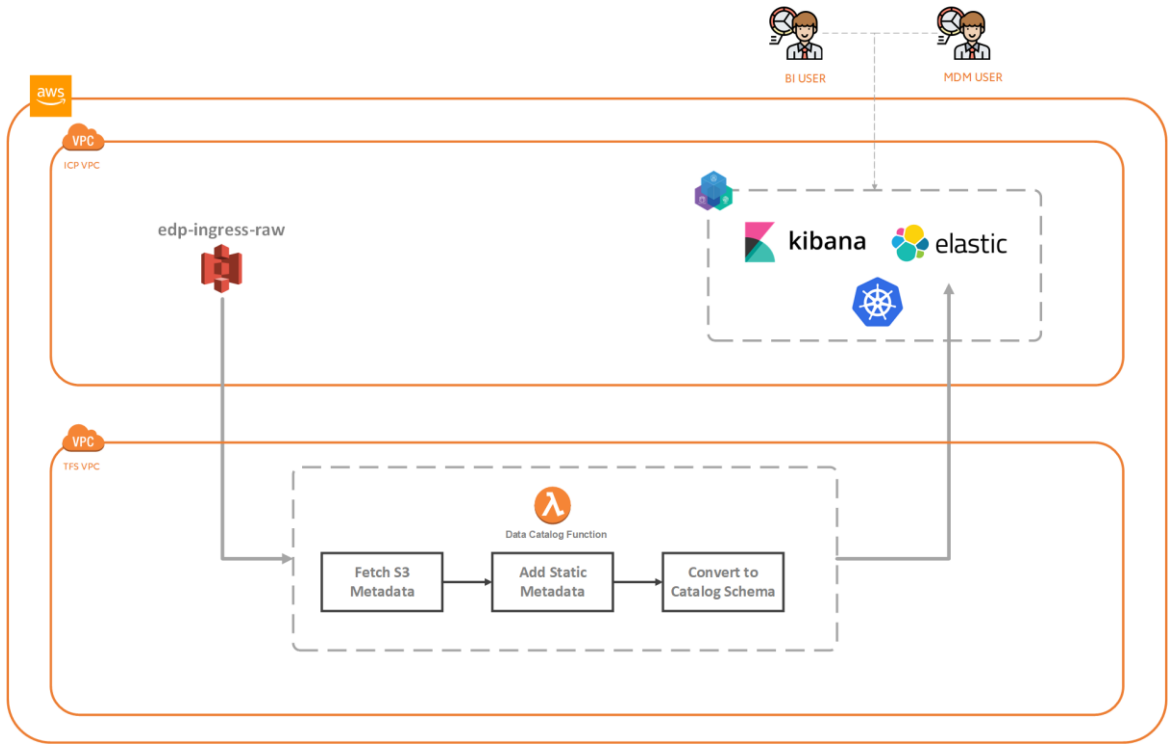
- Select SaaS solutions that have a deep and growing customer base, active user group and can integrate with the TFS technology ecosystem.
- Cost should be equal or less than building a custom solution, with rapid onboarding and minimum upfront cost (pay as you go/grow).
- It should scale up and down aligned with business growth.
- It must meet performance, availability and security requirements mandated by TFS (METER).

Solution

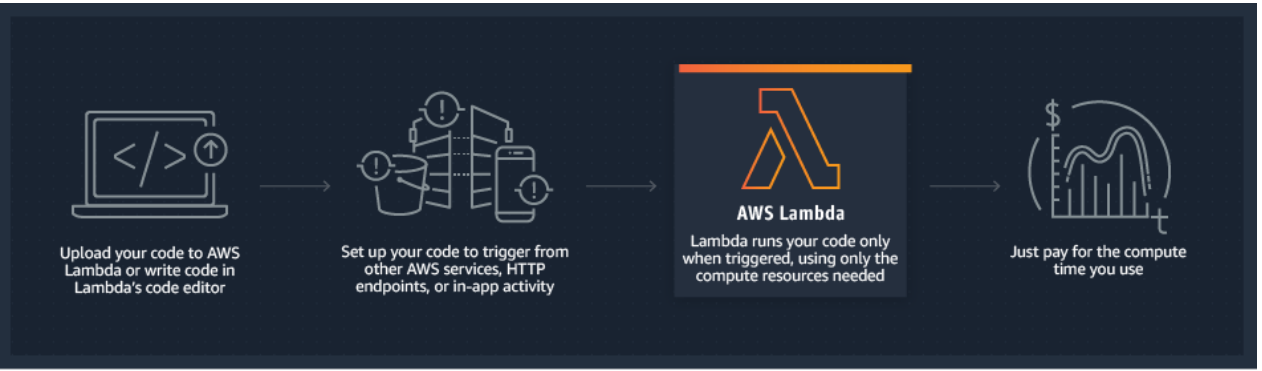
Buy or consume SaaS solution, many factories/teams uses various SaaS solutions such as Salesforce, ServiceNow, defi, Workday, Taulia, Office365, RouteOne, Fiserv Premier, etc.



Function as a Service (FaaS)



Example : EDP using Lambda to update S3meta-data



What solution is recommended to quickly develop a scalable microservice which can trigger on events and requires no infrastructure to be provisioned and managed?

Considerations

- The micro service execution is short lived and doesn't need to run all the time.
- It only triggers based on defined events with logic encapsulated in the Microservice.
- The function is stateless with state stored in external datastores such as RDS or S3. (Using functions has a learning curve and factories will need to develop the expertise to use this capability).

Solution

Use AWS Lambda, it is highly efficient and scalable. Several factories/applications such as Zuul, Digital Shop & Buy, EDP are using Lambda today. AWS Lambda is not ideal for long running batch jobs and it has short execution time of less than 15 minutes.

Platform as a Service (PaaS)

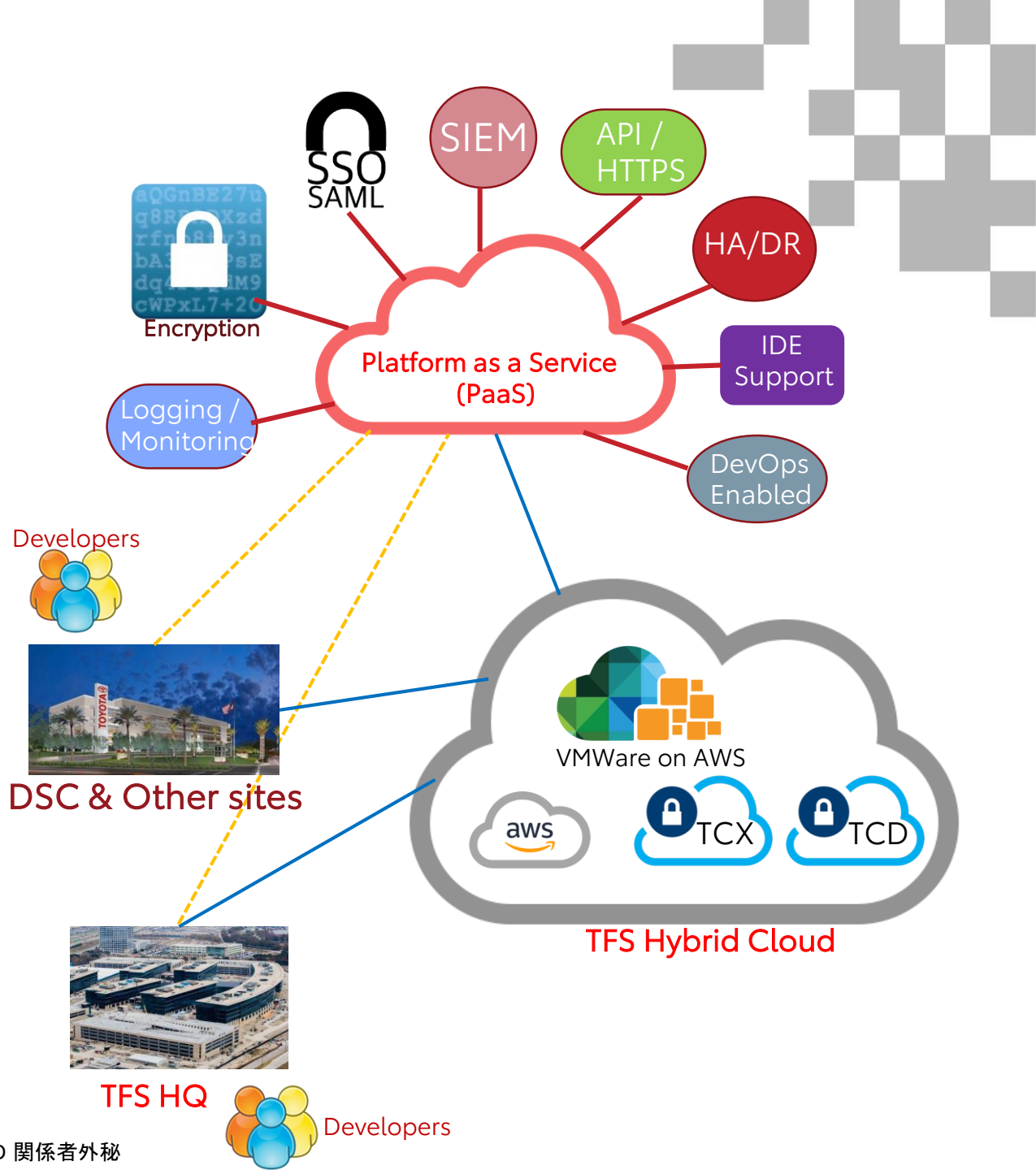
What is recommended for quickly building a solution that allows my factory to focus on the development, deployment and management of business applications; removing the need to manage the underlying infrastructure overhead such as resource procurement, capacity planning, software maintenance and patching?

Considerations

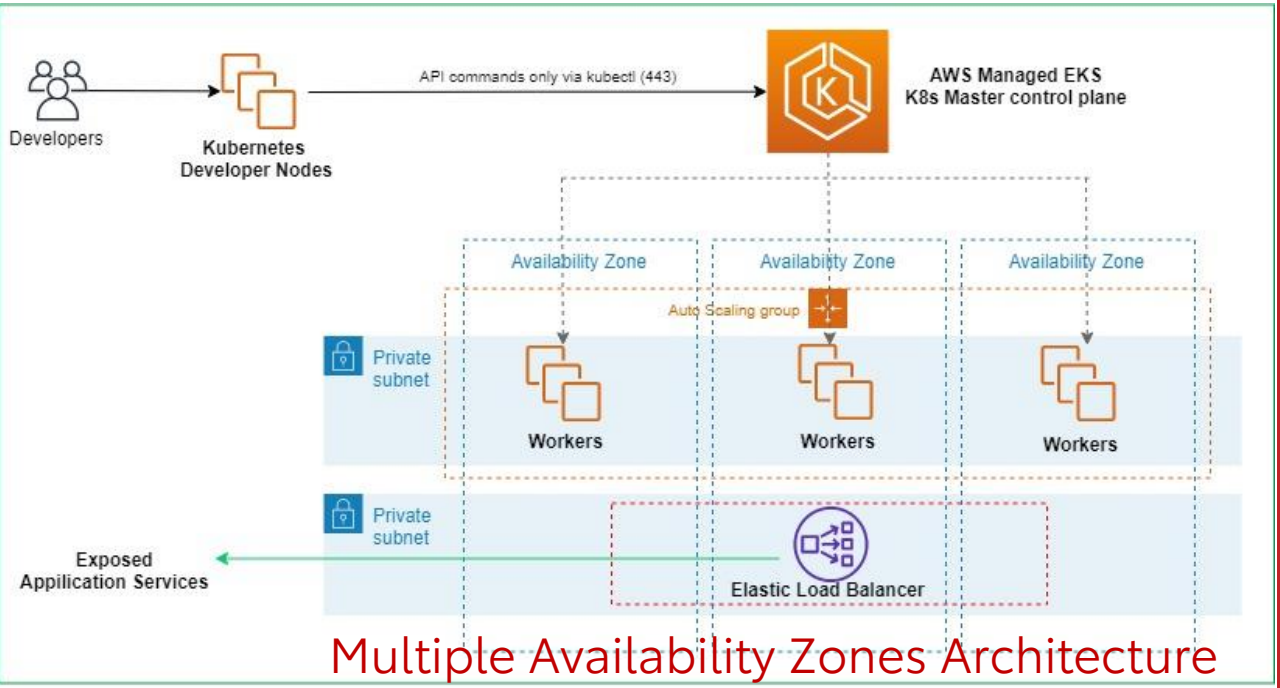
- Application is self-contained and has no external dependencies (OS and Filesystem).
- It can be developed locally and pushed to the cloud.
- The application itself is stateless and state is stored in external data stores such as RDS or S3.
- The application is also capable of scaling horizontally.
- Build it when you need and destroy it when it's not needed (no unused infrastructure).

Solution

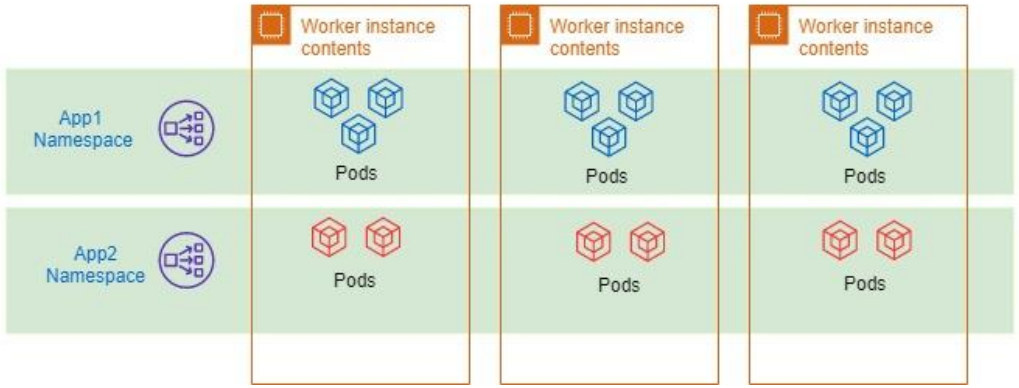
Use AWS Elastic Beanstalk. This service can be manually provided now and will be rolled out as a self-service capability in FY21 Q2



Container as a Service (CaaS)



Multiple Availability Zones Architecture



Soft Multi-Tenancy with namespaces

What solution is recommended to quickly develop & deploy containerized applications or microservices that are highly scalable, available and isolates my application or Microservice from others?

Considerations

- It should have a very low operational overhead and integrate easily with other cloud native services.
- The solution must have a built-in orchestration engine to automatically manage load & availability
- Application or Microservice is capable of running in a docker container, service(s) only exposed through a load balancer and supports multiple instances/replicas.
- Target is a set Microservices, not a monolithic app and is capable of scaling horizontally.
- It must be immutable and stateless (state is stored externally such as RDS or S3). Leveraging the 12-factor application development framework is recommended and factory is expected to have inhouse Docker & Kubernetes developer expertise.

Solution

Use AWS EKS, it is DevOps enabled platform, provides multi-tenancy using name-space isolation. It is available as self-service in TFS SNOW catalog, it can be deployed in minutes.

Infrastructure as a Service (IaaS)

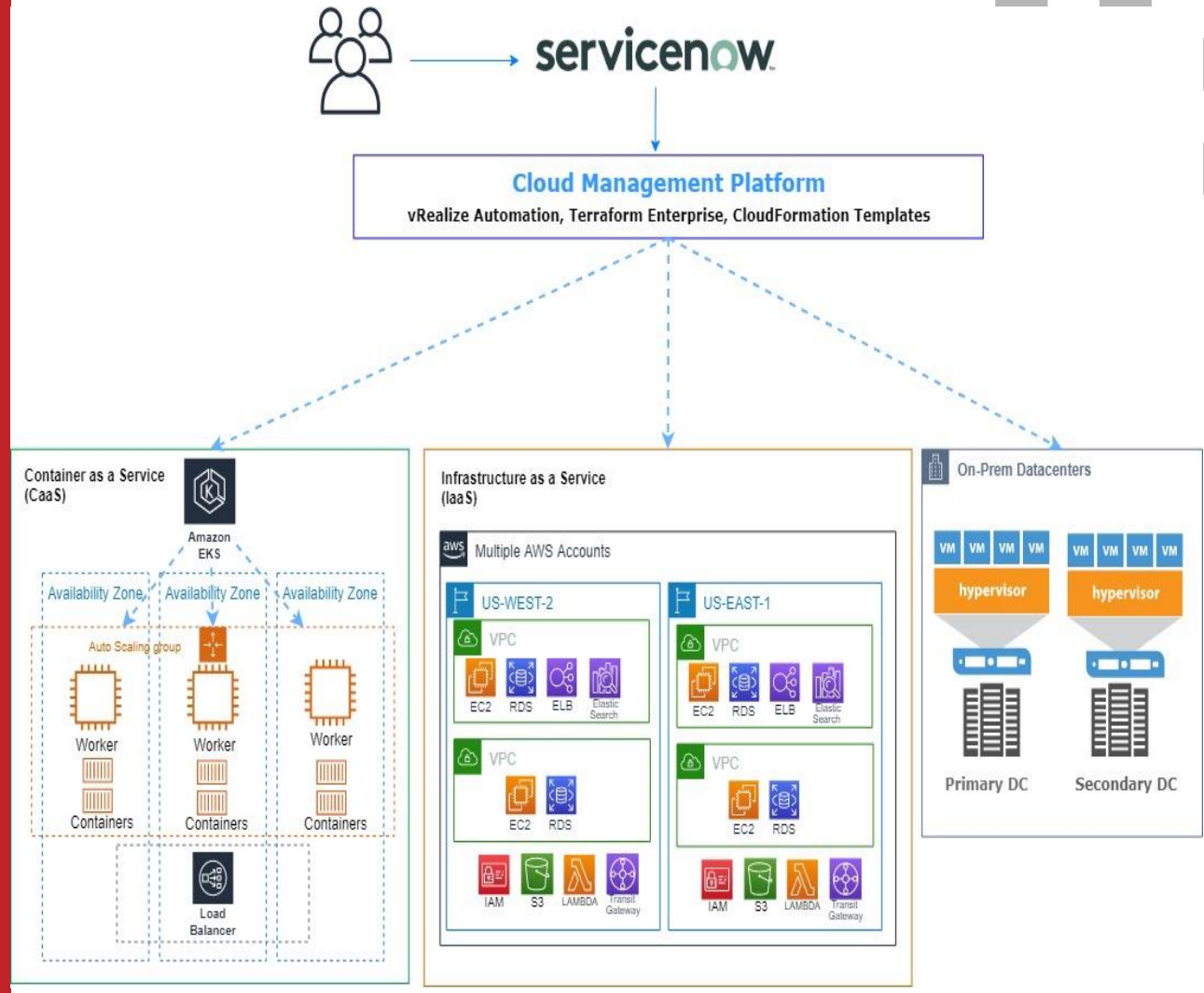
How does a factory deploy a COTS, traditional or non-cloud native application to meet a business requirement where the application stateful and only runs MS Windows or Linux OS platforms, can't be containerized, and requires customization and control of the underlying operating system?

Considerations

- Many COTS & traditional applications don't scale very well and don't provide native high-availability, the resiliency and performance must be provided by the application.
- AWS frequently brings down EC2 instances for maintenances, use the multiple-available zones architecture and expose application services through load balancers to achieve the desired performance and availability

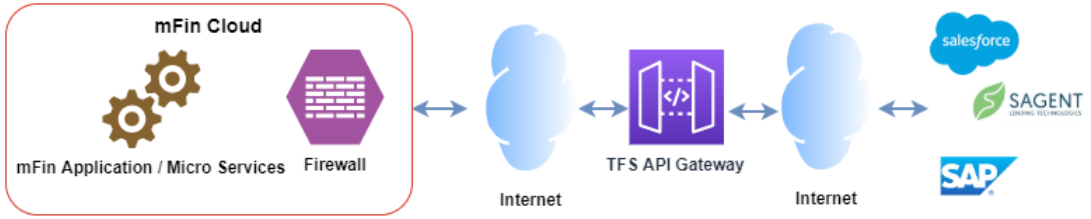
Solution

Consume AWS EC2 (IaaS) service, it is available as self-service in TFS SNOW catalog. This capability is also available in TFS Cloud where the application needs to integrate with other TFS Cloud applications and requires low latency or there are licensing/compliance requirements to build it in the TFS Cloud.



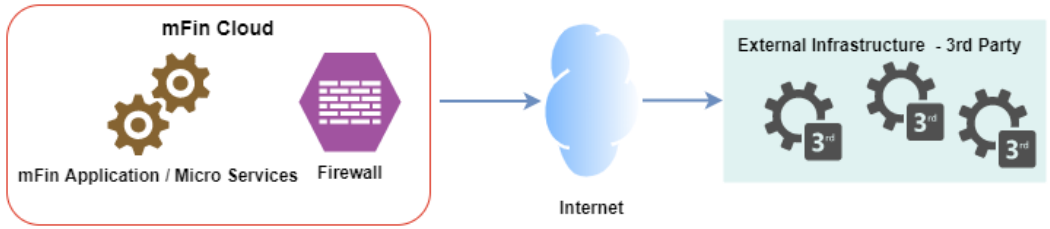
Enabling Self-Service & Speed of provisioning

mFin Cloud Connectivity



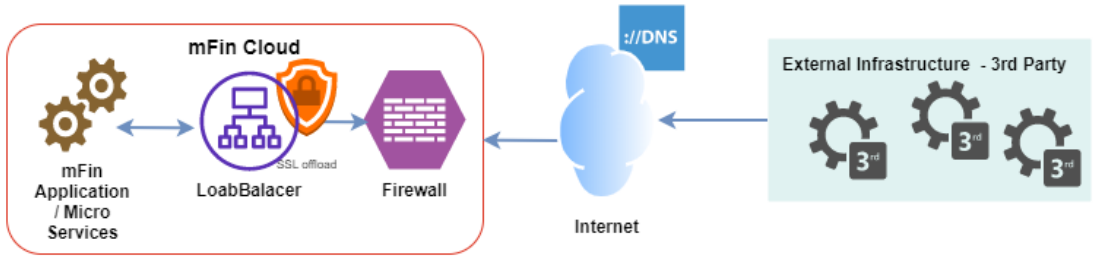
API to API Connectivity:

- Use TFS API Gateway



Non-API Communication:

- Typically ssl connectivity.
- IP whitelisting/MFA may be required.
- mFin Systems/Applications initiating the connection.
  - Request firewall access over the internet
    - Firewall Service Request



Business partners / 3rd party services connecting to mFin Systems:

- Expose service on the load balancer to proxy the requests to the backend servers
  - Load Balancer Request Catalog
- Obtain the SSL certificate
  - SSL Certificate Request
- Create the DNS entry's
  - Local and Global DNS Service Request
- Request firewall access to allow the traffic between mFin and 3rd party services
  - Firewall Service Request

mFin Cloud Security

Most cloud security incidents are due to customer configuration issues and not the cloud provider. How does a factory ensure that their cloud solution is secure and complies with Information Security policies, standards and guidelines?

Considerations

- Access Controls
  - Integrate with TFS IAM tools
  - Least Privilege Access Models (RBAC)
- Application & Data Security
  - Secure Code
  - Secure API (HTTPS)
  - Data in transit and Data @ Rest Encryption (AES256)
- Network Security
  - Private End Point
  - TLS 1.2 Connectivity
- Security Monitoring
  - Code & Vulnerability Scan
  - SIEM Integration

Solution

- Cloud providers adopt a "shared accountability" model for cloud security, providing customers with the tools to secure their solutions while holding customers accountable for securing their assets.
- For detailed guidance, follow TFS InfoSec Playbook for Cloud Security

TFS Cloud Standards

Operating Systems	Microsoft Windows Server, Redhat Linux and Amazon Linux
Database (Relational)	AWS PostgreSQL RDS
Database (NoSQL)	AWS DocumentDB, MongoDB*
Kubernetes/Container Platform	Amazon EKS
Load Balancing	AWS Elastic Load Balancer (ELB), AVI
Block Storage	AWS Elastic Block Storage (EBS)
Object Storage	AWS Simple Storage Service (S3)
Shared Storage	AWS Elastic File System (EFS)
Functions (Serverless)	AWS Lambda
Machine Learning	AWS SageMaker
Big Data	AWS Elastic Map Reduce (EMR)

Environment Provisioning : Frequently Asked Questions

	API Question	Response
1	Are you saying that our standard is that we must create microservices on EKS on TFS as a standard?	No. We are showing the available options. If your team does not have experience developing applications with microservices please stop and use another option.
2	Can I use EFS (filesystem) on EKS, if my product/application requires shared filesystem?	Yes, this capability is currently in pilot and few application teams are testing in sub-prod. We plan to roll this capability in production by Q4 FY20 .
3	I am running my application in the cloud, does it mean I have high availability and disaster recovery capability out of the box?	No, you have to design and build your application in a way achieve the high availability and disaster recovery.
4	My COTS application only supports Microsoft SQL database and it will not work with PostgreSQL, What is my option?	SQL RDS instance can be provided in such situations where the product doesn't support cloud native databases
5	Can I build any instance size with self-service provisioning?	No, instance are provided as t-shirt size (small, medium and large). If you application demands more or less resources, instance size can be change as a day-2 activity.
6	Is AWS Elastic BeanStalk available today as a Self-Service?	Not currently available. It will be available as self-service in FY21. If you need AWS Elastic Beanstack, Cloud team can manually provision it.

TOYOTA FINANCIAL SERVICES

How to provision mFin Cloud Environments

Build it yourself

Self-Service (CaaS, DBaaS & IaaS) provisioning
➔ *ServiceNow (https://tfs.service-now.com)*
➔ *Service Catalog*
➔ *IT Only Requests*
➔ *Cloud Infrastructure Services*

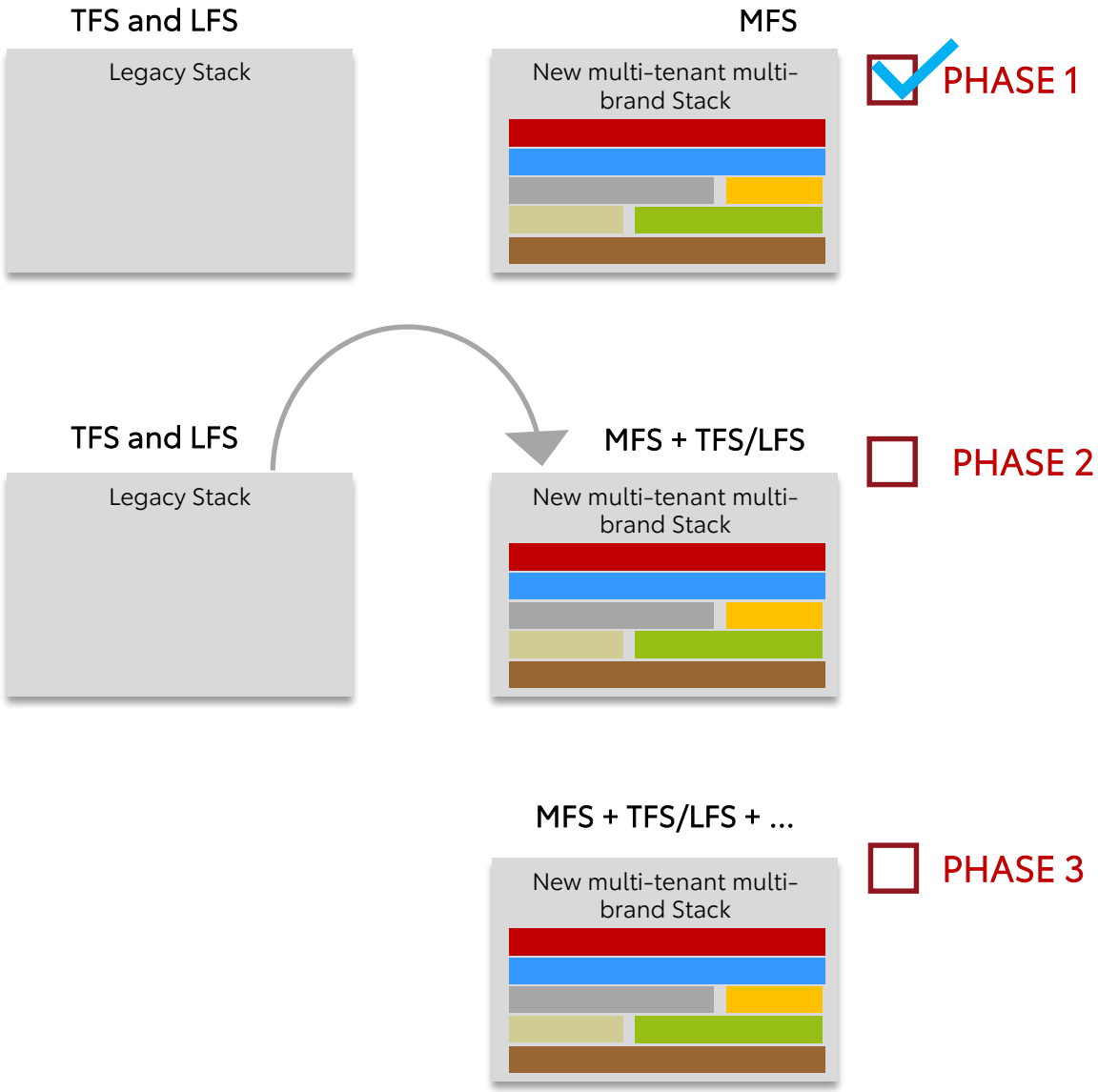
Factories can build the following using self-service

- *Windows VM*
- *Linux VM*
- *Kubernetes/Containers*
- *PostgreSQL RDS*
- *DocumentDB (NoSQL)*
- *Object Storage (S3 Bucket)*
- *DNS Entries*
- *jFrog Artifactory Repository*

Need help with Environment Builds	Daily Pit-Stop / Open House / Office Hours
Intake / Backlog Prioritization	Weekly ETAO intake Meeting

Bringing it all Together: “mFin" in Practice

Modernization Approach



April 2020

Introduce new mFin design based multi-tenant technology stack for the first tenant – MFS.

2020+

Introduce new tenant to new mFin Stack for TFS. Start moving TFS capabilities, systems and data to TFS tenant in mFin Stack

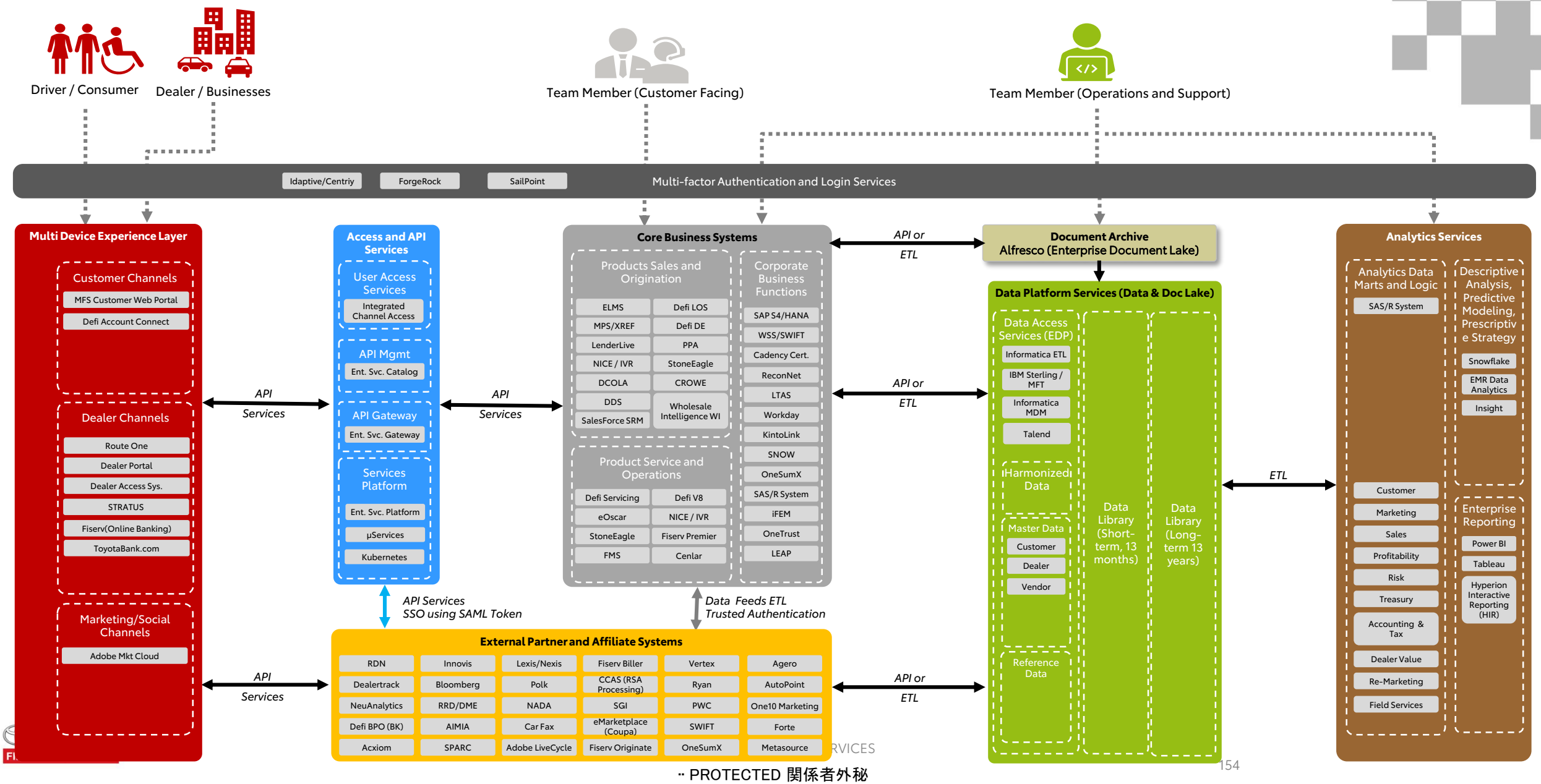
2023+

Complete transition of retained old stack systems to mFin multi-tenant approach. Decommission Legacy systems.

Bringing it all Together: “mFin" in Practice

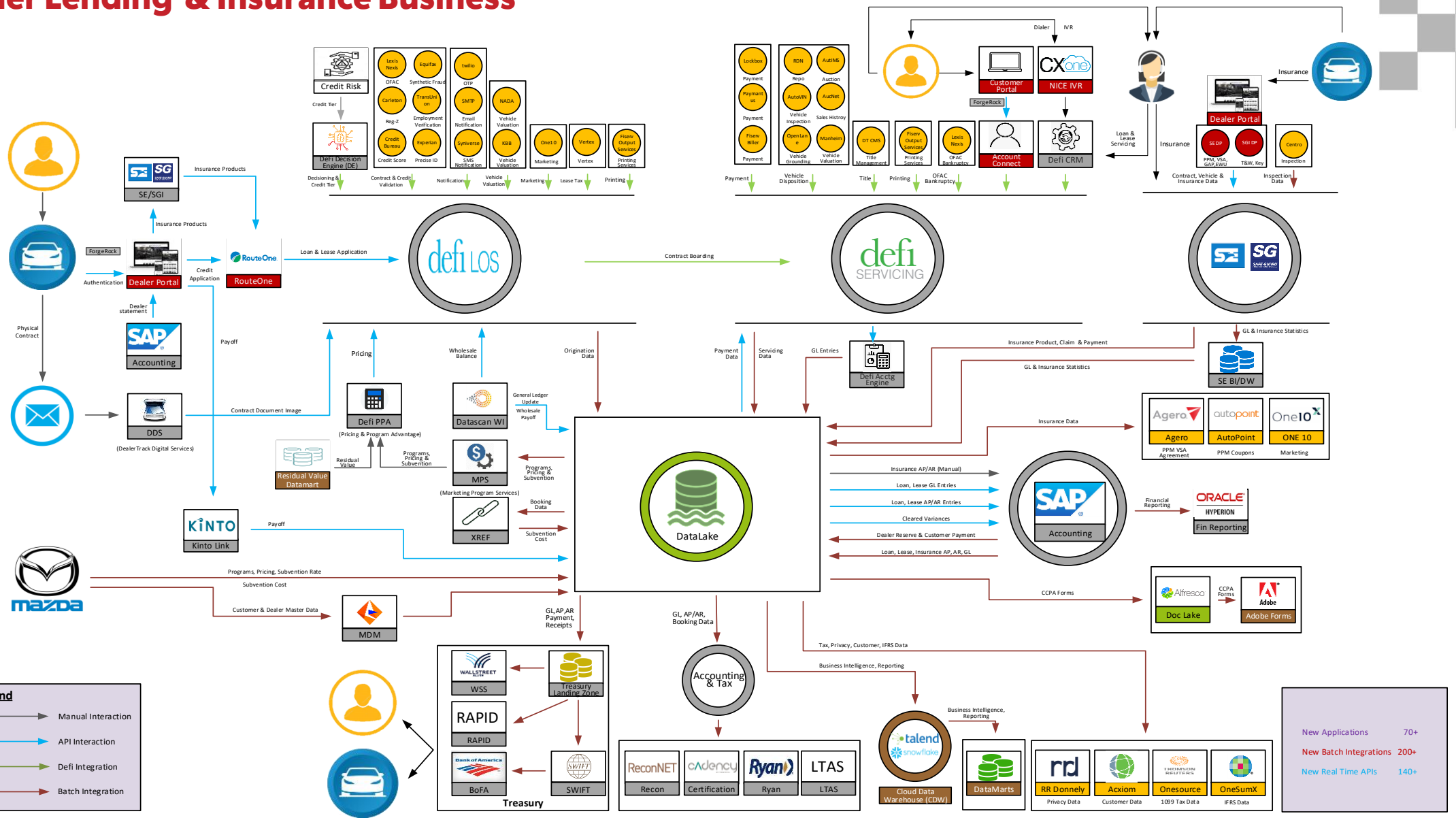
Modernization Approach: Phase 1

‘mFin’ Implementation: PHASE 1



MFS Implementation: PHASE 1

Consumer Lending & Insurance Business



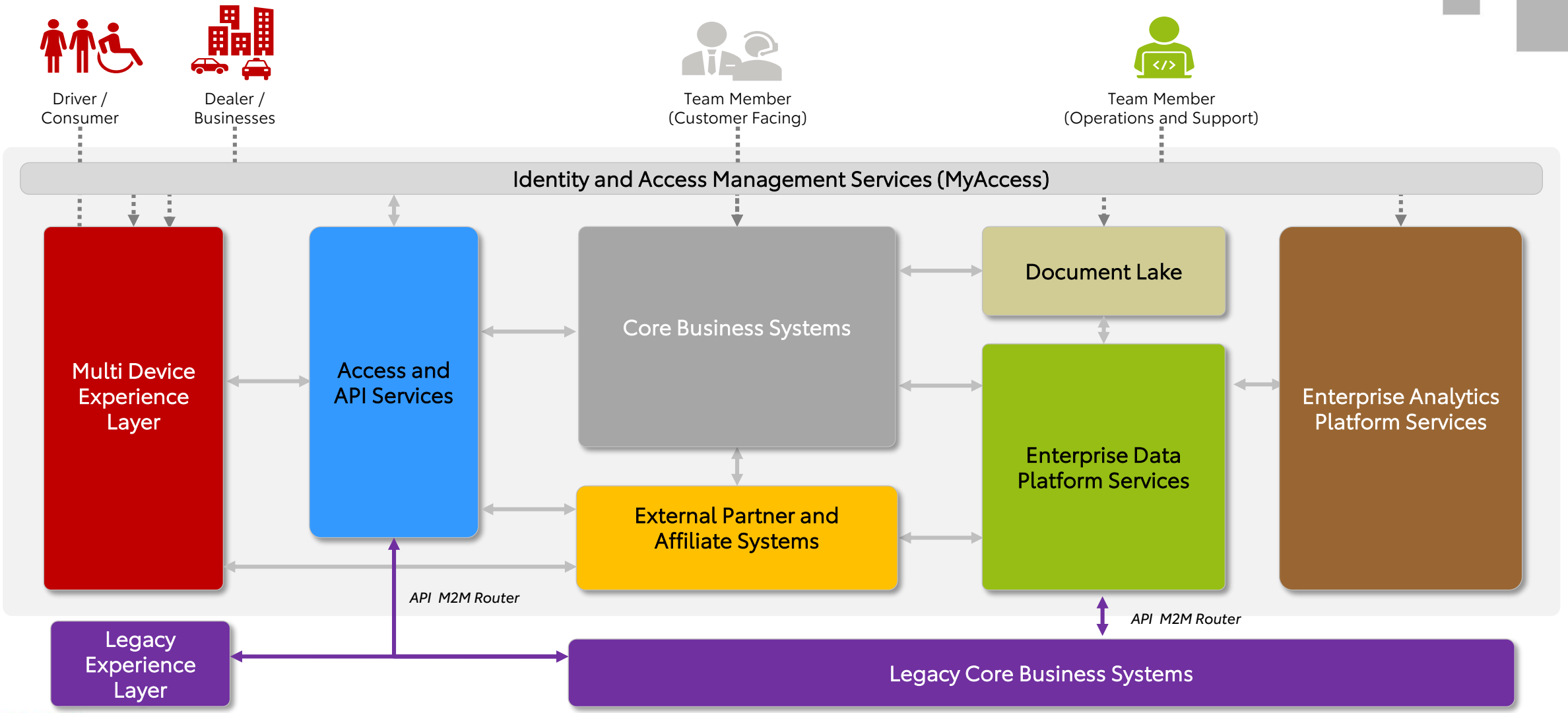
Bringing it all Together: “mFin" in Practice

Modernization Approach: Phase 2

‘mFin’ Implementation: PHASE 2

Phase 2 requires a Transition State (illustrated in purple)

One Unified Eco-System with Shared Standards and Common Platforms

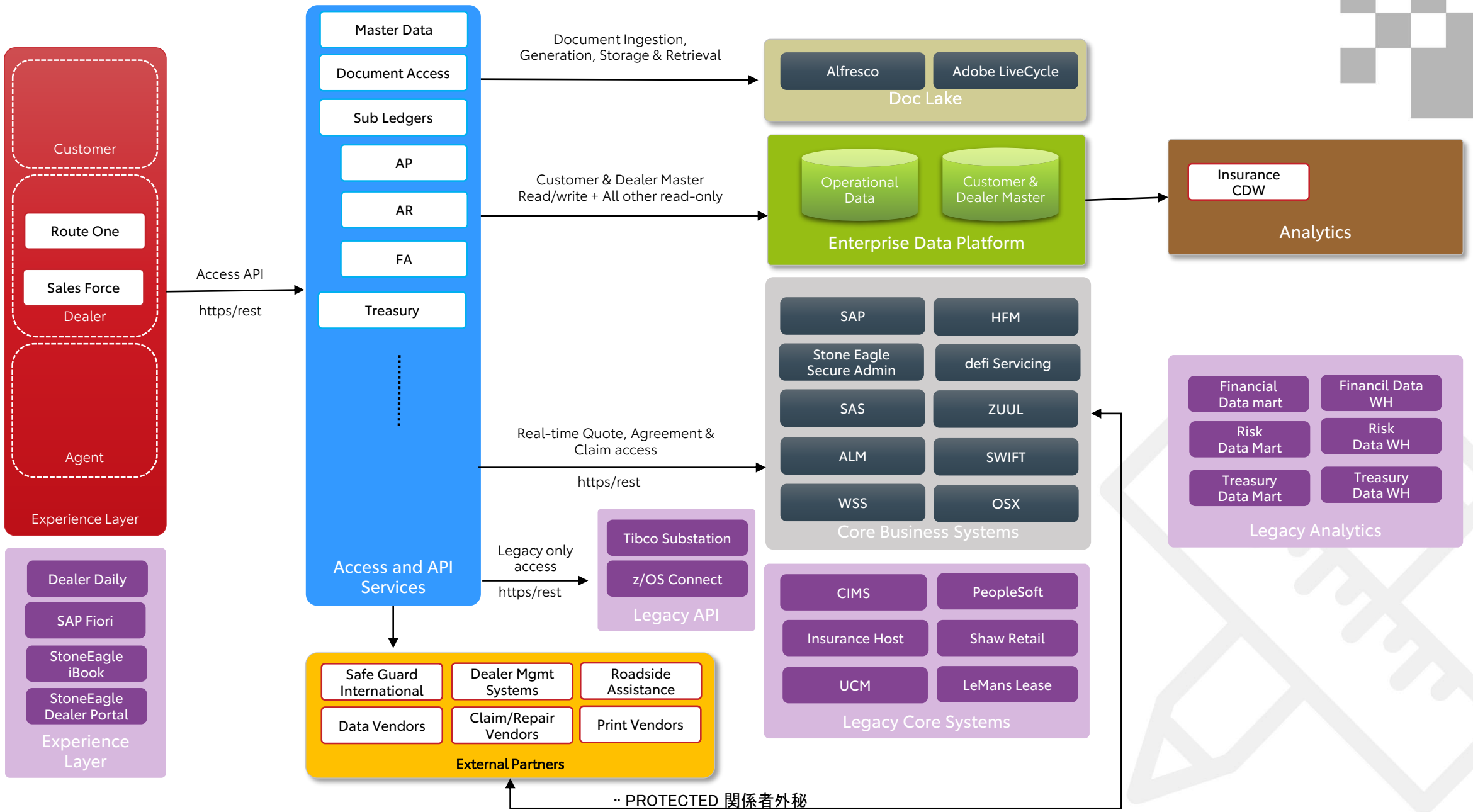


m in ‘mFIN’ stands for Multi-Tenant, Multi-Company, Multi-Brand, Multi-Country, Multi-Device, Mobility-Finance, My-Finance...

Bringing it all Together: “mFin" in Practice

Modernization Approach: Phase 2
Corporate Systems Domain

Corporate System – Logical



Bringing it all Together: “mFin" in Practice

Modernization Approach: Phase 2
Products Domain

‘mFin’ Implementation: PHASE 2

Modernization Approach: Products

New Business on New Multi-Tenant Platforms
- Originate New Loan, Lease & Insurance Business on New Platforms
- No New Business Originated on Legacy Platforms

No Legacy Portfolio Data Migration
- No High Risk "Big Bang" Data Migration Cutover Event
- No Data Migration & Reconciliation between Old and New Platforms

Runoff Legacy Portfolio & Retire Legacy Platform
- Multiple Servicing Systems In Production during Portfolio Runoff
- Remaining Loan Portfolio Migrated from Legacy to New Platforms

Single Tenant

Today

Mainframe Apps:
- **Shaw Retail**
- **LeMans Lease**
- **CIMS Claims**
- **Insurance Host**

Multi-Tenant

2023+

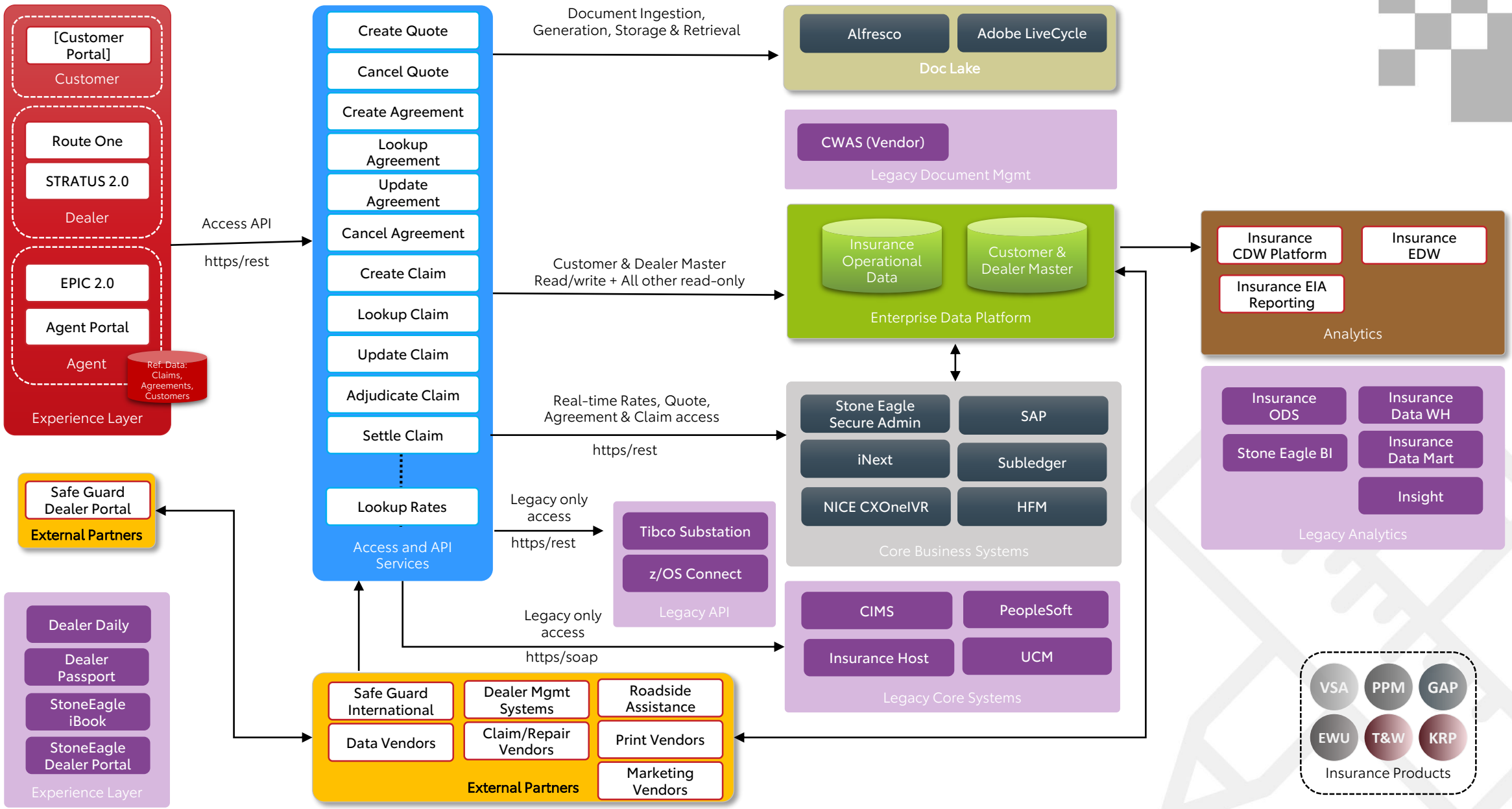
Candidate Platforms:
- **defi Servicing***
- **StoneEagle***

Target Platforms:
- **defi LOS**
- **Wholesale Intelligence (WI)**

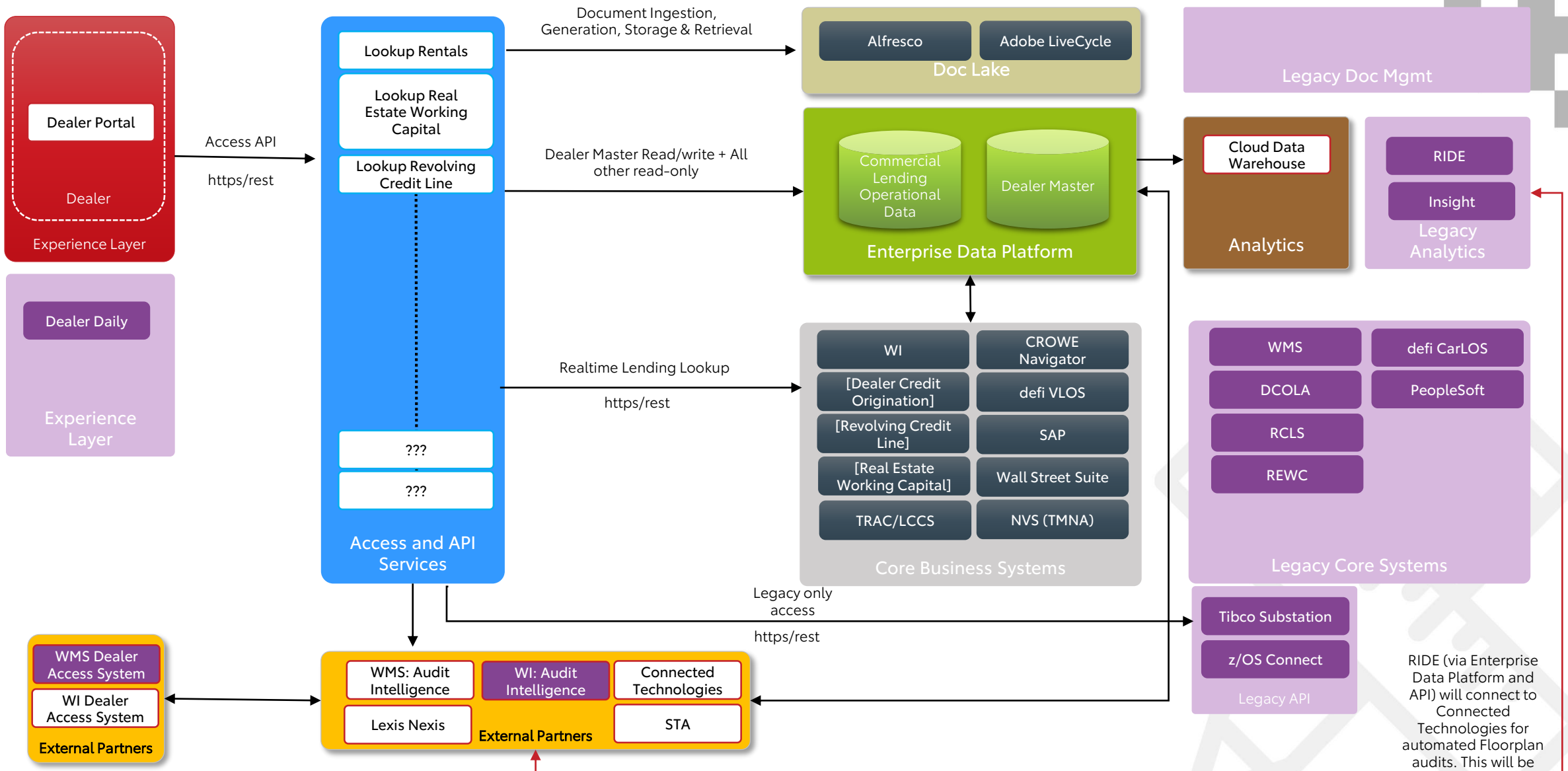
TOYOTA FINANCIAL SERVICES

*Gap analysis in process

Products – Insurance Logical



Products – Commercial Lending Logical



Bringing it all Together: “mFin" in Practice

Modernization Approach: Phase 2

Channels Domain

‘mFin’ Implementation: PHASE 2

Modernization Approach: Channels

New Business on New Multi-Tenant Platforms

- Enable Digital Customer Experience (Customer, Agent, Dealer & Interaction)
- Create Domain Driven Micro-Services and Use Across Channels for Consistent Experience
- Create Customer Universal Party Master (MDM)
- Originate New Loan, Lease & Insurance Business on New Platforms

No Legacy Portfolio Data Migration

- No High Risk "Big Bang" Data Migration Cutover Event
- No Data Migration & Reconciliation between Old and New Platforms

Runoff Legacy Portfolio & Retire Legacy Platform

- Multiple Servicing Systems In Production during Portfolio Runoff
- Remaining Loan Portfolio Migrated from Legacy to New Platforms after 42 Months

Single Tenant

Today

Mainframe Apps:

- **Shaw Retail**
- **LeMans Lease**
- **CIMS Claims**
- **Insurance Host**
- **Collections Host**
- **PeopleSoft**

Multi-Tenant

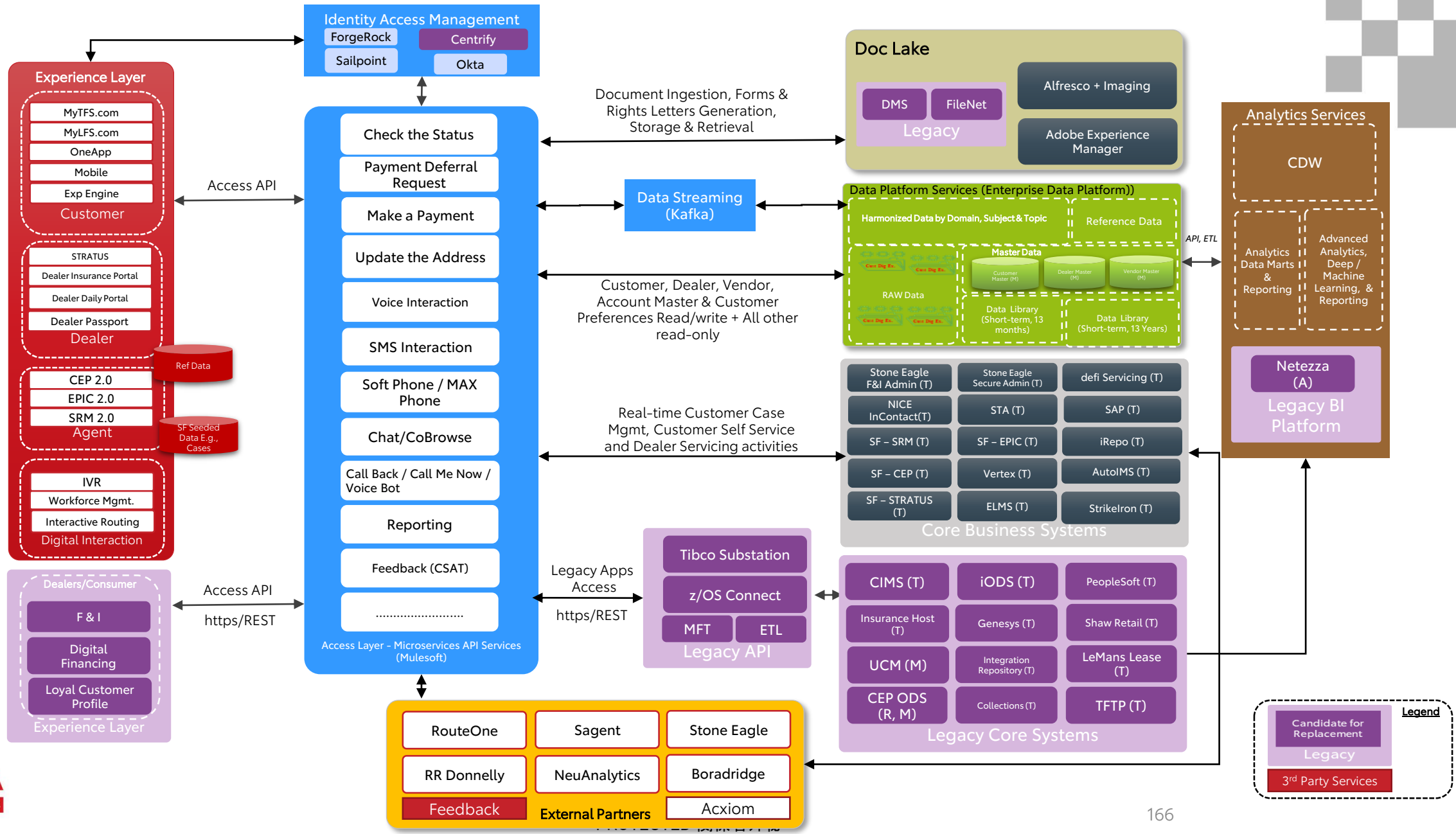
2023+

‘mFin’ Platforms:

- **Digital Experience**
  - **Customer**
  - **Agent**
  - **Dealer**
  - **Interaction**

TOYOTA FINANCIAL SERVICES

Channel Modernization M2M – Logical View

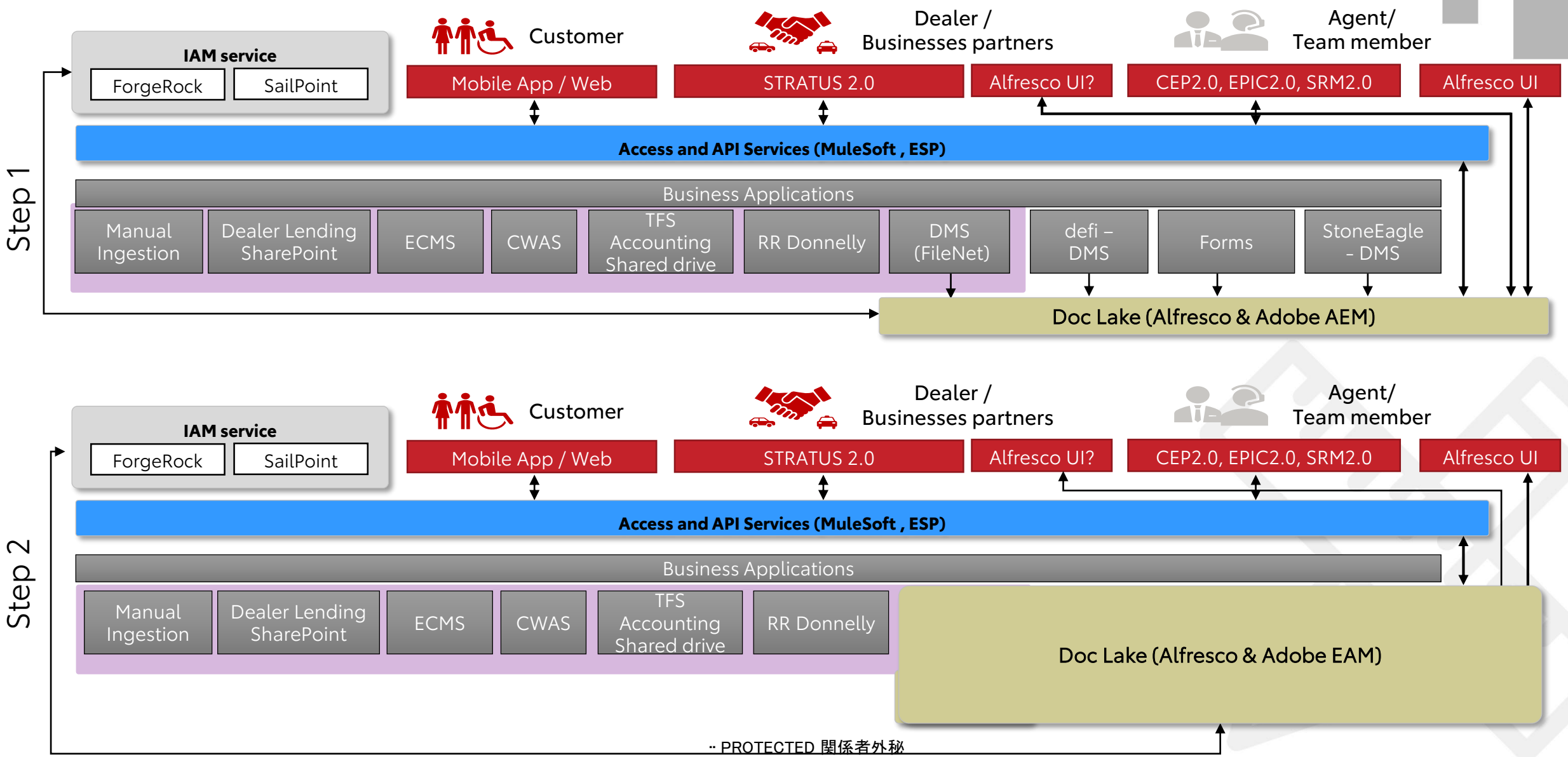


Bringing it all Together: “mFin" in Practice

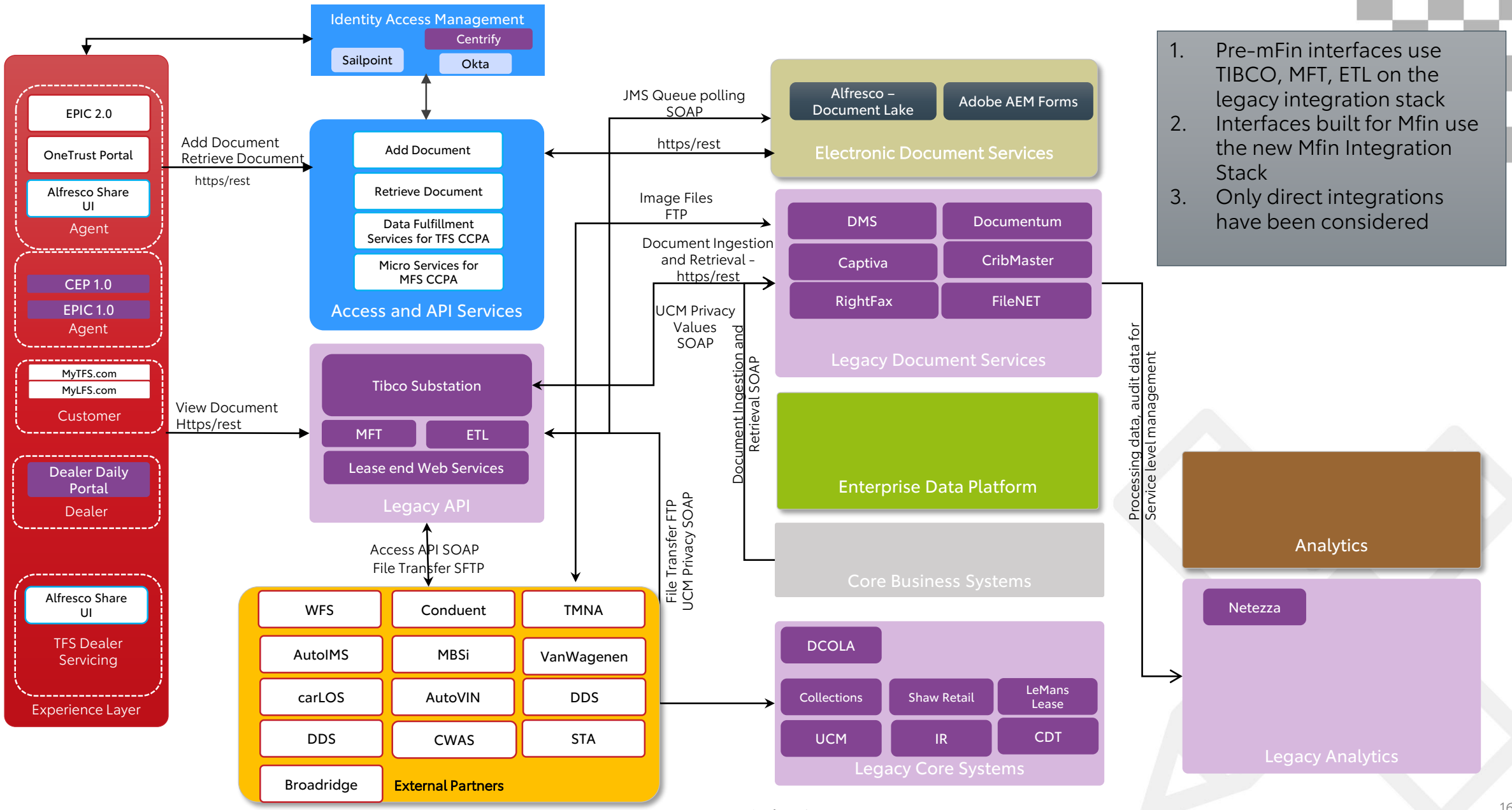
Modernization Approach: Phase 2
Expanded Value Chain Domain

‘mFin’ Implementation: PHASE 2

Modernization Approach: Document Lake

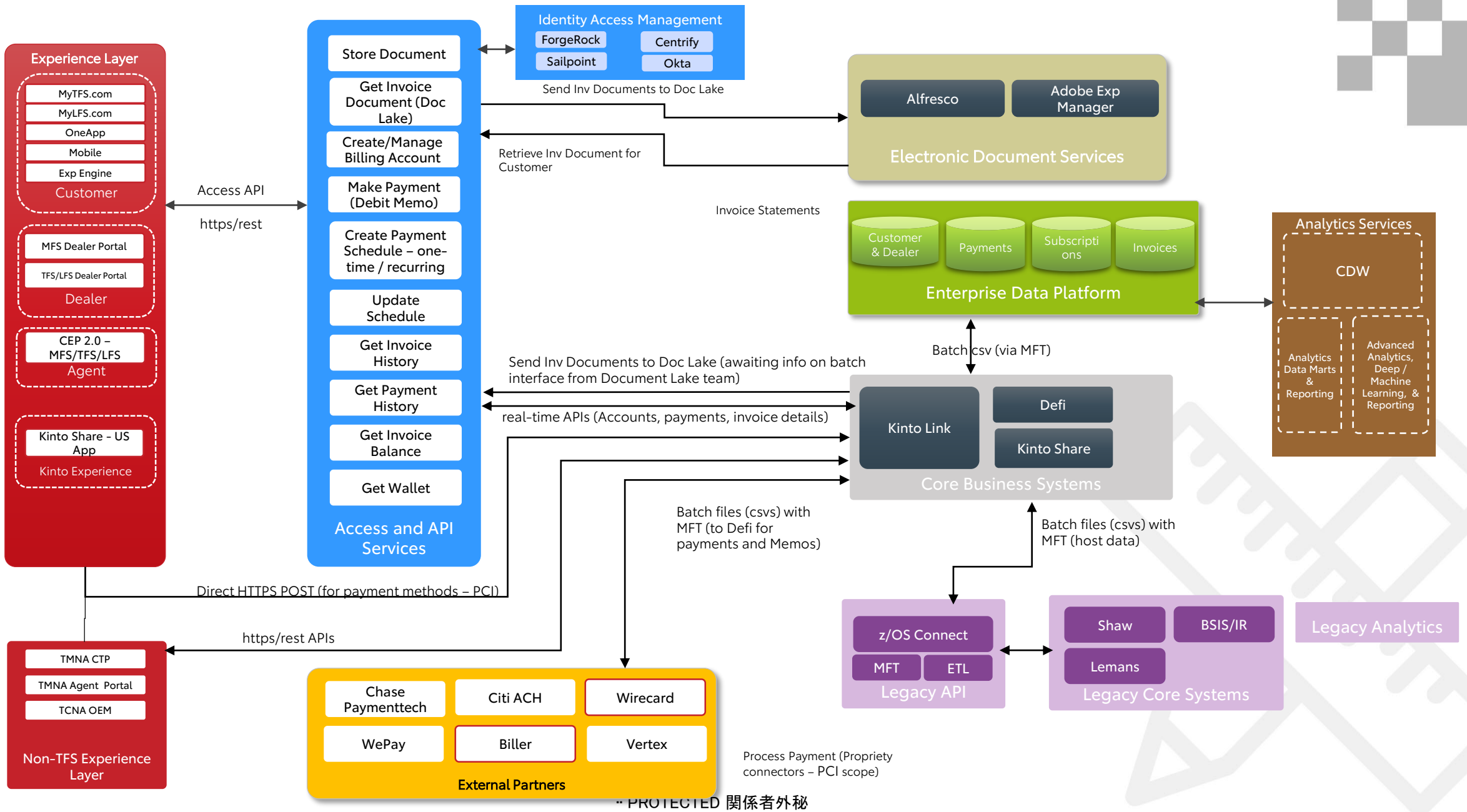


Document Lake - Logical View

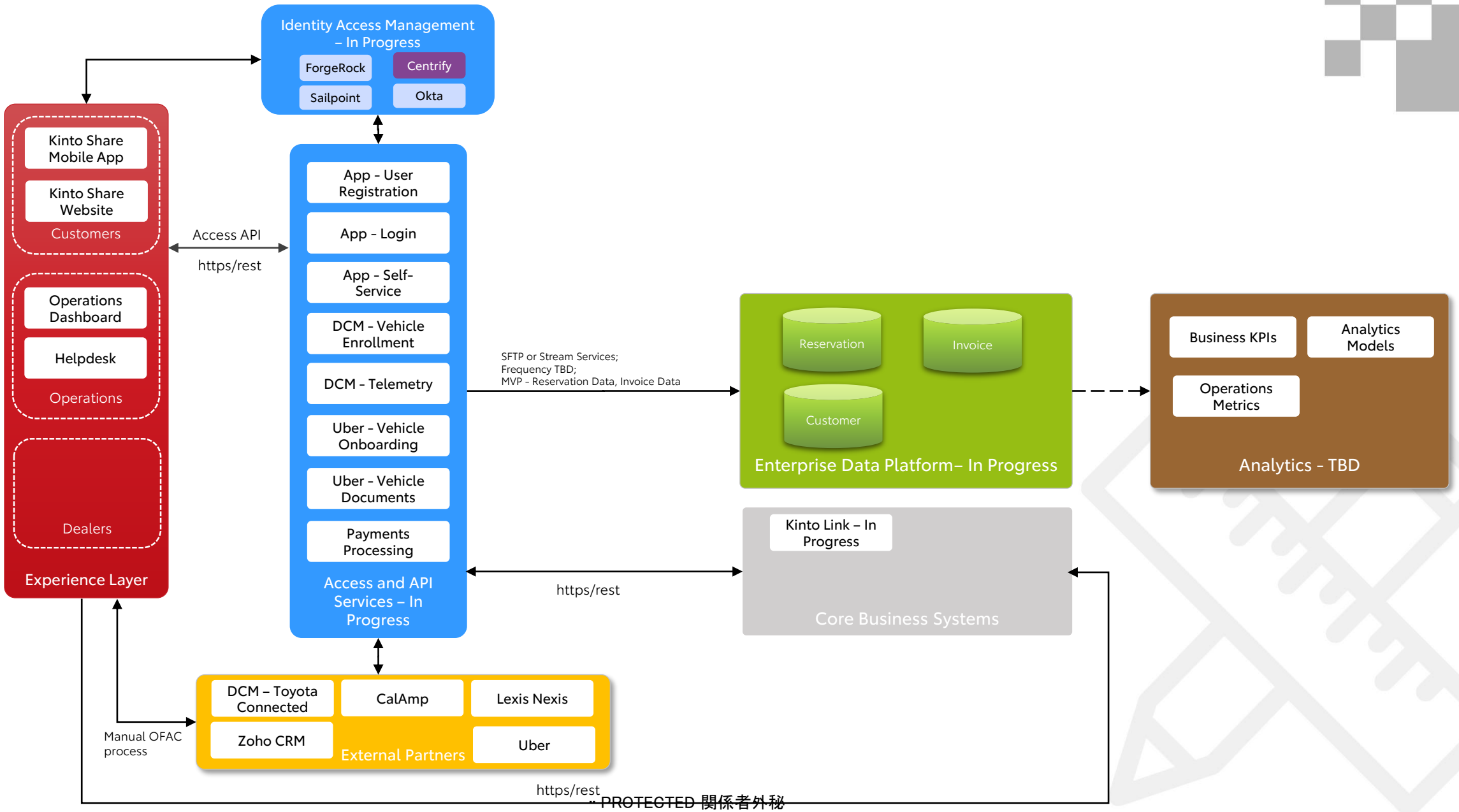


1. Pre-mFin interfaces use TIBCO, MFT, ETL on the legacy integration stack
2. Interfaces built for Mfin use the new Mfin Integration Stack
3. Only direct integrations have been considered

Kinto Link and TFS/MFS Payments - Logical View



Kinto Share – Logical View



Enterprise Architecture Team Design & Where to Get Help

Architecting with Agility

Enterprise Architecture
(Domain Architects)

Domain Technology Direction (Macro Decisions)
Enterprise Technology Platforms & Roadmaps
Integration Design Patterns
Technology Standards
.....

Application Architecture
(Product Designers)

Product Designs
Development Patterns & Practices
Non-Functional Requirements
.....

Security Architecture
(Security Consultants)

Security Platform Direction
Security Integration Designs
Security Patterns & Practices
...

Data Architecture
(Data Designers)

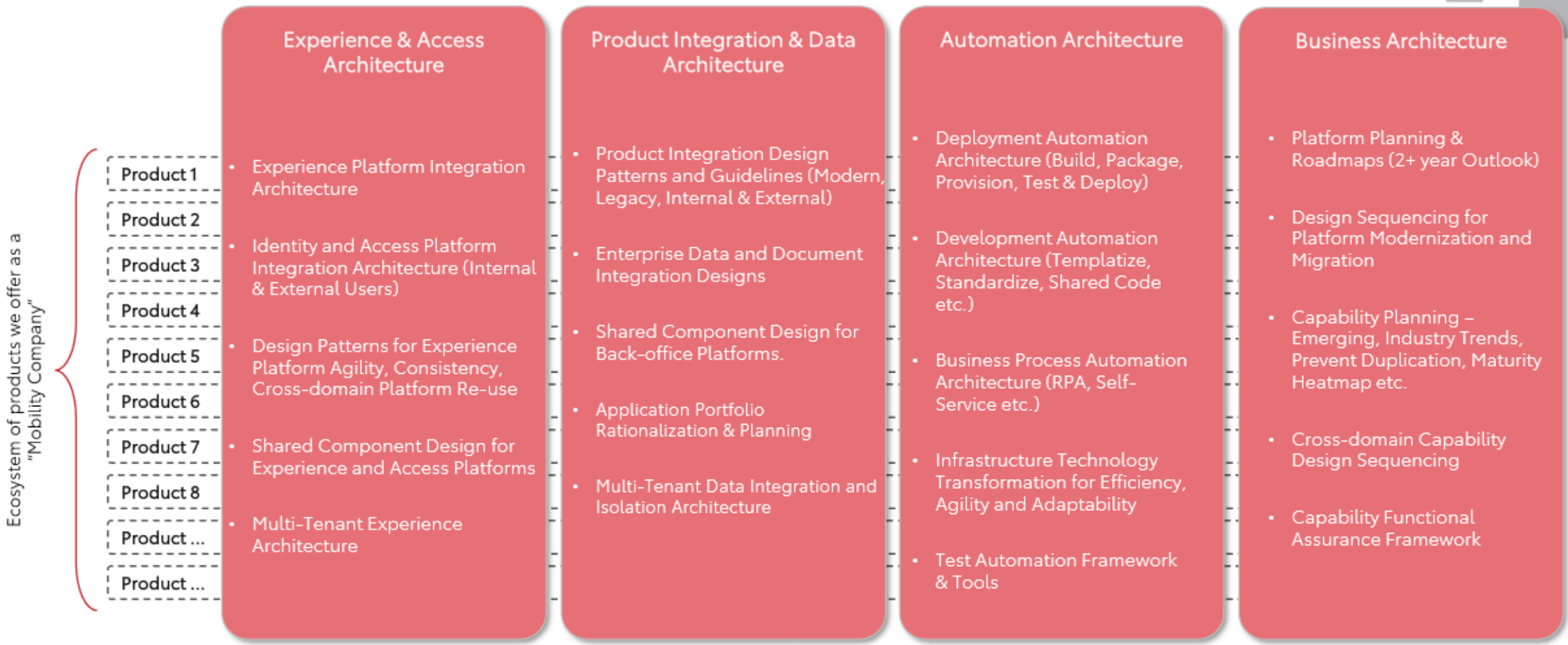
Enterprise Data Models
Master Data Management
Data Ingestion Patterns
...

Infrastructure Architecture
(Infrastructure Designers)

Infrastructure Platform Designs
New Platform POCs
...

TOYOTA FINANCIAL SERVICES

TFS Enterprise Architecture – Operating Model



Enterprise Architecture Routines

A-LAT (Architecture Leadership Action Team):

Purpose:

- Identify, discuss and resolve cross Domain and Enterprise impediments
- Escalate impediments to T-LAT or E-LAT that is outside the scope of A-LAT

Expectations for Product Designers:

- Raise impediments to alignment on the Technology Architecture and Product Design Direction
- Be prepared to discuss impediments at A-LAT, identify next steps to resolve

Impediment pre-work before presentation at A-LAT:

- What is the root cause for the impediment?
- What remediation actions have been taken to address the root cause?
- What decisions are needed at T-LAT or E-LAT or actions needed to address the root cause?
- What is the due date by which you will resolve the impediment?

Impediments should be logged in the JIRA Impediment Tracker, here in order to ensure closure of issues and escalation when appropriate.

Designers Meet:

Purpose:

- Designers Meet (DM) will serve as a forum for Designers to collaborate, learn, contribute, and obtain feedback on macro-designs, technology selections and new standards.
- Participation will ensure the understanding and alignment with mFin platform direction and standards; creating consistency, efficiency, and improved quality and digital capability in line with TFS' business goals of transforming to a digital mobility company.

Expectations for Product Designers:

- Present new technology proposals, macro-designs, and new standards to Designers Meet.
- Present necessary documents, as outlined within DM process.

DM process, artifacts, decisions, and routine minutes will be uploaded to the Designers Meet TEAMS site.

TOYOTA FINANCIAL SERVICES

‘mFin’ Architecture Playbook

Revision History

Revision History

Version	Author(s)	Section(s)	Description of Change
02222020	Squad in ToC	All	2.0 Release
03272020	Walter Campbell	Product Design	Add high level flow slide
	Daniel Waymel	Product Design	InfoSec requirements slide added
	Walter Campbell	Multi-Tenancy	Aligned intro slide with METER
	Ken Schuelke	API	New / revised content
	Walter Campbell	Revision History	New section added for revisions
05282020	Walter Campbell	Tech. Arch.	Edits to Modernization Approach
	Ken Schuelke	API	Add new content and adjust flow
	Daniel Waymel	Security	Revise / simplify existing content
	Reza Ghamsari	Doc Lake	New Section
	Mo Malik	Cloud Env.	New / revised / simplified content
	Walter Campbell	METER	Revised / simplified content
	Pankaj Devgun	'mFin' in Practice	New section, added MFS Phase II
	Walter Campbell	'mFin' in Practice	Added TFS Phase II plan
10192020	Amanda Walter	All Chapters	Reorganized and Created Chapters
	Amanda Walter	Where to Get Help	New Section – Included Designers Meet
	Raphael Mathias	Chapter 7: Integrating Enterprise Data and MDM	New / revised content
	Reza Ghamsari	Chapter 8: Integrating Enterprise Document Lake	New / revised content

TOYOTA FINANCIAL SERVICES

Version	Author(s)	Section(s)	Description of Change