AHCA Florida Health Care Connections (FX)

T-5: Technical Architecture Documentation

Version: 300

Date: June 24, 2022

Author: The SEAS Vendor

Submitted To: AHCA FX Program Administration Team

Revision History

DATE	VERSION	DESCRIPTION	AUTHOR
7/1/2018	001	T-5: Technical Architecture Documentation initial draft version	Mike Griffiths, David Rothschild
7/30/2018	002	T-5: Technical Architecture Documentation updates to Agency review comments	Mike Griffiths, David Rothschild, Paul Moore
8/8/2018	003	T-5: Technical Architecture Documentation updates to Agency review comments	Mike Griffiths, David Rothschild, Paul Moore
8/9/2018	004	T-5: Technical Architecture Documentation updates to Agency review comments	Mike Griffiths, David Rothschild, Paul Moore
8/22/2018	100	T-5: Technical Architecture Documentation approved and baselined version	Sean Gibbs
6/14/2019	101	T-5: Technical Architecture Documentation annual update including MES to FX terminology changes and process to introduce module specific services	David Rothschild
7/24/2019	101	T-5: Technical Architecture Documentation annual update. Updates to Agency review comments	Mike Griffiths, Barry McConnell
7/25/2019	200	T-5: Technical Architecture Documentation approved annual update	Carol Williams
1/29/2020	201	T-5: Technical Architecture Documentation Quarterly refresh version	Mike Griffiths Steve Ruszkowski
2/3/2020	225	T-5: Technical Architecture Documentation approved quarterly update	Eric Steinkuehler
6/10/2021	226	T-5: Technical Architecture Documentation draft refresh version updates made: ▪ Updated Section 1 boilerplate language and made minor standardized grammatical edits throughout ▪ Updated Section 3.4.1 and 3.4.2 with the IS/IP Vendor being responsible for development of the Application Service Registry ▪ Added Section 5.4.4 Cloud Interconnectivity Solution ▪ Updated Section 6.1 FX Project Level Technical Capability Matrix	Steve Ruszkowski Francisco Acevedo
7/16/2021	227	T-5: Technical Architecture Documentation draft refresh version addressing Agency review comments	Steve Ruszkowski
7/20/2021	250	T-5: Technical Architecture Documentation approved final	Carol Williams

DATE	VERSION	DESCRIPTION	AUTHOR
5/26/2022	251	T-5: Technical Architecture Documentation draft refresh updates: ▪ Updated with minor grammatical updates and standardization throughout ▪ Updated breadcrumb paths to reflect the FX Hub per DET #502	Carol Williams
6/24/2022	300	T-5: Technical Architecture Documentation approved final	Carol Williams

Modifications to the approved baseline version (100) of this artifact must be made in accordance with the FX Artifact Management Standards.

Quality Review History

DATE	REVIEWER	COMMENTS
7/2/2018	Scott Mildenberger	Quality review for draft submission
8/2/2018	Sean Gibbs	Quality review for final submission
8/8/2018	Sean Gibbs	Quality review for final submission
8/9/2018	Sean Gibbs	Quality review for final submission
6/14/2019	Carol Williams	Conducted QA review
7/24/2019	Paul Moore	Quality review for response to Agency comments
1/29/2020	Eric Steinkuehler	QC review
6/4/2021	Carol Williams	Conducted quality review
7/16/2021	Carol Williams	Conducted quality review
5/23/2022	Carol Williams	Conducted quality review

Table of Contents

Table of Exhibits

Table of Strategic Topics

SECTION 1 INTRODUCTION

1.1 BACKGROUND

The Florida Agency for Health Care Administration (AHCA or Agency) is adapting to the changing landscape of healthcare administration and increased use of the Centers for Medicare and Medicaid Services (CMS) Medicaid Information Technology Architecture (MITA) to improve the administration and operation of the Florida Medicaid Enterprise. The current Florida Medicaid Enterprise is complex; it includes services, business processes, data management and processes, technical processes within the Agency, and interconnections and touchpoints with systems necessary for administration of the Florida Medicaid program that reside outside the Agency. The future of the Florida Medicaid Enterprise integration is to allow the Agency to secure services that can interoperate and communicate without relying on a common platform or technology.

The Florida Medicaid Management Information System (FMMIS) has historically been the central system within the Florida Medicaid Enterprise; functioning as the single, integrated system for claims processing and information retrieval. As the Medicaid program has grown more complex, the systems needed to support the Florida Medicaid Enterprise have grown in number and complexity.

The Medicaid Enterprise System (MES) Procurement Project was re-named Florida Health Care Connections (FX) in the summer of 2018. FX is a multi-year transformation to modernize the current Medicaid technology using a modular approach, while simultaneously improving overall Agency functionality and building better connections to other data sources and programs.

1.2 PURPOSE

The purpose of the *T-5: Technical Architecture Documentation* is to document and establish the framework for:

- Business Services
- Technical Services
- Application Architecture
- SS-A Technical Capability Matrix (TCM)

Exhibit 1-1: Technical Architecture Components shows the flow and interrelationship between the technical architecture elements. The alignment and optimization of each technology component result in a solution set (the physical implementation of the technology architecture and services) that support FX modules.

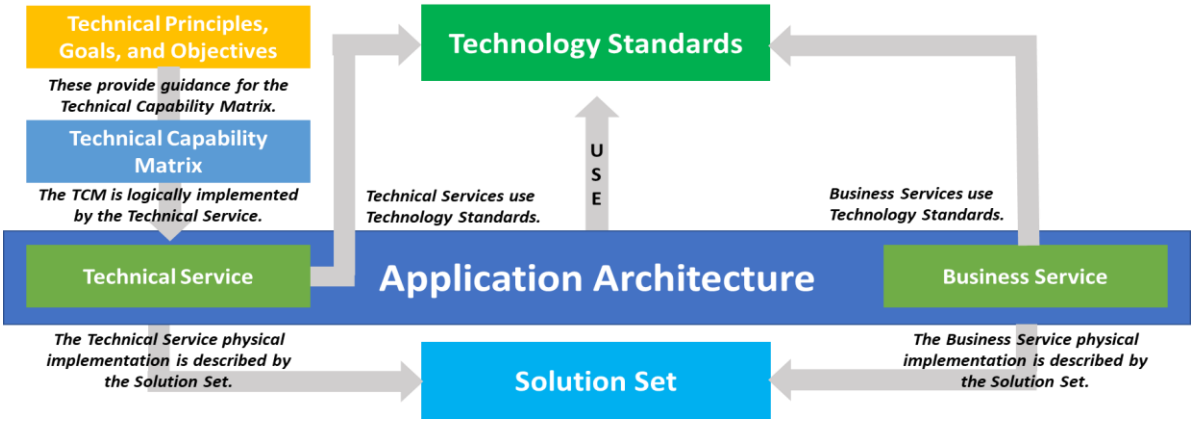

Exhibit 1-1: Technical Architecture Components

The FX Technical Architecture aligns with the MITA 3.0 documentation including:

- Part III - Chapter 3 - Business Services
- Part III - Chapter 4 - Technical Services
- Part III - Chapter 5 - Application Architecture
- Part III - Chapter 7 - Technical Capability Matrix

The approach and strategy for technical architecture documentation also accounts for unique Agency requirements.

1.3 SCOPE STATEMENT

The *T-5: Technical Architecture Documentation* scope is to provide the approach and documentation to identify, create, and maintain business services, technical services, and the Technical Capability Matrix, as defined by CMS MITA 3.0 standards.

This deliverable is an iterative document maintained and updated as the *T-5: Technical Architecture Documentation* strategy and guidance evolves. The deliverable emphasizes modularity and interoperability across the Agency.

This iteration of the *T-5: Technical Architecture Documentation* establishes initial technical architecture guidance with emphasis on the foundational capabilities of Integration Services and Integration Platform (IS/IP) including the Enterprise Service Bus (ESB), Enterprise Data Warehouse (EDW), and modular capability implementation. Deliverable content includes:

- **Section 1 Introduction** – Outlines the background, purpose, scope statement, goals and objectives, and reference documents used to prepare the deliverable.
- **Section 2 Roles and Responsibilities** – Lists the responsibilities of each of the FX stakeholders during the design and implementation phases of the project.

- **Section 3 Business Services** – Describes the structure and components of business services and the approach to developing, maintaining, and documenting reusable business services for business functionality aligned with the MITA Business Process Model (BPM) and the MITA Business Capability Matrix (BCM).
- **Section 4 Technical Services** – Describes the structure, components and types of technical services and the approach to developing, maintaining, and documenting reusable technical services to provide a MITA 3.0 technical capability.
- **Section 5 Application Architecture** – Describes the application architecture layers and the integration of business services with technical services through service elements that the Agency configures as a service layer.
- **Section 6 Technical Capability Matrix** – Provides specific technical-capability statements with defined criteria across five (5) levels of maturity for each CMS MITA 3.0 business area.

The *T-5: Technical Architecture Documentation* is the product of current state discovery, stakeholder input, strategic analysis, program strategy, and direction about techniques and priorities to support overall improvement of the Agency's mission.

This document communicates FX Technical Architecture direction to involved stakeholders, including Agency technology leadership, executives, CMS, FX Project Owners and potential FX Project Vendors, external organizations (e.g., other State of Florida agencies), and other state Medicaid programs.

1.4 GOALS AND OBJECTIVES

- **Goal #1** – Improve Data Quality
  - **Objective #1** – Create reusable business and technical services that provide consistent data validation, edits, and transformation of data based on a single source of policy truth
  - **Objective #2** – Enable use of business and technical services throughout the Agency that create, access, and maintain data consistently in a single source of data truth (e.g., Operational Data Store (ODS) for transactional data and the Reporting Data Store (RDS), Enterprise Data Warehouse (EDW), and Data Marts that derive from the Operational Data Store.)
  - **Objective #3** – Create business and technical services that perform real-time processing and enable real-time data access
- **Goal # 2** – Improve recipient and provider experience
  - **Objective #1** – Enable processing consistency across system, program, and organization boundaries
  - **Objective #2** – Enable complete and consistent experience data collection for FX business processes
  - **Objective #3** – Enable no-wrong door processing to increase access and information timeliness and accuracy

- **Goal #3** – Reduce FX Total Cost of Ownership (TCO)
  - **Objective #1** – Reduce duplication of custom code embedded in applications that performs same business or technical processing
  - **Objective #2** – Simplify testing of established services
  - **Objective #3** – Enable service versioning to reduce change management complexity
- **Goal #4** – Encourage Service Reuse
  - **Objective #1** – Establish a registry that allows potential service consumers to identify existing or planned services that can be reused
  - **Objective # 2** – Simplify the ability to identify, create, and reuse business and technical services
- **Goal #5** – Develop, document, and implement the necessary processes to maintain and update the *T-4: Technical Management Strategy*, *T-5: Technical Architecture Documentation*, and *T-6: Technology Standards*
  - **Objective #1** – Maintain and update the FX Projects Repository (FXPR) and FX Hub with the electronic materials, including any revised documentation, as FX evolves

1.5 REFERENCED DOCUMENTS

The following documents provided reference and guidance in the creation of this deliverable.

- Medicaid Information Technology Architecture (MITA)
- National Institute of Standards Technology (NIST) NIST 800 145
- Centers for Medicare and Medicaid Services MITA Application Architecture
- Architecture and Infrastructure Committee, Federal Chief Information Officers Council's *"Enabling the Mission: A Practical Guide to Federal Service Oriented Architecture"*
- National Association of State Chief Information Officers (NASCIO) State CIO Top Ten Priorities for 2018
- National Institute of Standards Technology (NIST) Cloud Computing Reference Architecture
- Organization for the Advancement of Structured Information Standards (OASIS) Reference Model for Service Oriented Architecture 1.0
- Strategic Topic Inventory

Note: OASIS is an international nonprofit computer-industry association for the development, conjunction, and adoption of e-Business and Web Services standards.

1.6 STRATEGIC TOPIC INVENTORY

This document provides guidance on many Technical Architecture Strategy topics. In the development of this deliverable, the SEAS Vendor created a Strategic Topic Inventory tool used to develop and communicate the Agency's direction on a wide range of Technical Architecture Strategy topics. The tool organizes topics into a hierarchical taxonomy based on logical groupings in areas of interest to strategic, programmatic, technology, and program management domains.

The Strategic Topic Inventory has many features to present and communicates a spectrum of strategic direction options considered across the spectrum of time for a specific topic. A summary chart can dynamically display the strategic direction for a specific topic across the time spectrum from current state direction to direction for future years. The Strategic Topic Inventory includes a field documenting a summary analysis that describes the context and considerations that influenced the defined strategy for each specific topic.

Extracts of the topic-specific summary chart from the Strategic Topic Inventory tool are included throughout this document to communicate strategy and direction for many of the important Technical Architecture Strategy decisions that are important for FX stakeholders to understand.

Over the course of FX, the SEAS Vendor will continue to define and elaborate strategic direction on many Technical Architecture strategy topics. The SEAS Vendor intends to continue to use the Strategic Topic Inventory tool as a discussion, approval, and communication vehicle for defining Technical Architecture strategy direction as topics arise.

Exhibit 1-2: Strategic Topic Inventory Item Sample below shows a screen capture example of a populated strategic topic.

Area:	Service Delivery Offerings and Assets			**Description:**				
Category:	Data Modeling			Who performs conceptual data modeling for the FX Conceptual Data Model?				
Sub-Category	Conceptual, Logical, Physical Data Modeling							
Topic:	Who performs conceptual modeling							
Importance:		**Strategy Status:**						
Displaying Row:	462							
Strategic Direction		**Current**	**2018**	**2020**	**2022**	**2025**		
SEAS vendor			X	->				
EDW Vendor				Coordination with SEAS Vendor	->			
Module Vendor								
TPA Vendor		FMMIS,DSS						
AHCA Systems (e.g. IT, HQA, ...)		X						
Analysis:	The SEAS vendor is accountable and contractually responsible for conceptual and logical data modeling. The SEAS vendor will coordinate with the EDW vendor to coordinate data services implementation issues and logical to physical modeling activities.							

Exhibit 1-2: Strategic Topic Inventory Item Sample

The SEAS Vendor develops and maintains this Microsoft Excel-based tool that resides as a document in the FXPR.

SECTION 2 ROLES AND RESPONSIBILITIES

Exhibit 2-1: Roles and Responsibilities identifies the primary stakeholders and responsibilities involved with this deliverable.

ROLE	RESPONSIBILITY
SEAS Vendor	▪ Identify business services ▪ Identify technical services ▪ Identify technical capabilities ▪ Create and Evaluate Business Service Request(s) ▪ Create and Evaluate Technical Service Request(s) ▪ Create and Evaluate Project Level Technical Capability Matrix (PLTCM) Request(s) ▪ Propose new, updates to (not including fixes), and retirement of technology service components to the FX Technology Standards Committee ▪ Maintain the Application Service Registry (ASR) forms ▪ Maintain the SS-A Technical Capability Matrix (TCM) ▪ Maintain the Project Level Technical Capability Matrix (PLTCM)
Subject Matter Expert (SME)	▪ Evaluate Business Process Matrix / functions to identify new or changing business services Identify and propose new business services to the SEAS Vendor (SEAS Vendor works with SME to enter the Business Service Request Form and facilitates evaluation through the FX Technology Standards Committee approval process)
FX Technology Standards Committee	▪ Review proposed business services and technical services ▪ Receive and process recommendations to approve or deny business services and technical services ▪ Review Project Level Technical Capability Matrix (PLTCM) entries
External Organizations	▪ Review and may align technology solutions to FX Technology Architecture service standards ▪ Submit service request forms to the SEAS Vendor, which reviews and then forwards the requests to the IS/IP Vendor
FX Project Owners	▪ Identify and understands FX Application Service Registry entries ▪ Provide project-specific Application Service Registry entries ▪ Provide updates to the Technical Capability Matrix at the request of the Agency ▪ Review and align technology solutions to FX Technology Architecture service standards
IS/IP Vendor	▪ Evaluate Business Service Request(s) ▪ Evaluate Technical Service Request(s) ▪ Maintain the ASR ▪ Implement Business Service Request(s) ▪ Implement Technical Service Request(s)

ROLE	RESPONSIBILITY
EDW Vendor	▪ Evaluate Business Service Request(s) ▪ Evaluate Technical Service Request(s), which include Data Services Requests ▪ Implement Business Service Request(s) ▪ Implement Technical Service Request(s), which include Data Services Requests

Exhibit 2-1: Roles and Responsibilities

SECTION 3 BUSINESS SERVICES

A business service is the modular implementation of a business process that enable processing consistency and reuse across system and organization boundaries.

3.1 BUSINESS SERVICE OVERVIEW

This section provides guidance defining:

- What is a business service?
- Business service approach to processing and use
- Business service development
- Business service registry purpose and use
- Information managed about business service items

3.1.1 BUSINESS SERVICE DEFINITION

Business services are a set of business activities used by multiple parties or different systems, such as a different module, provider, the Agency or another organization. The business area and business processes (the actual workflow) provide the context for a business service. A business service supports a specific business process or functionality, as defined by MITA 3.0 Part III Technical Architecture, Chapter 3 Business Services.

A business service performs all the necessary logic, calls to get data, and service access to support a business process. The logic in a business service can include calls to other business services, data services, technical services, business rule services, and usually includes custom processing logic. Most business services establish a transaction boundary or complete unit of work for database updates. In some business services embedded in other business services, the data update processing may defer to a higher-level business service.

Methods are parameters passed to a service that communicates the type of processing the services is to perform. Business services usually have custom methods (e.g., enroll provider) specific to the business process as opposed to a standard set of methods.

3.1.2 BUSINESS SERVICE BENEFITS

The use of business services benefits the Agency enabling:

- **Modularity** – Modularity is the extent to which and the ability to divide a system into smaller modules. Smaller pre-written and pretested modules reduce the development and testing effort to implement and maintain systems. A business service is an implementation of modularity that allows multiple system implementations to perform business processing consistently without affecting the rest of the enterprise. An example of modularity is a business service that performs postal address validation and standardization that could be used across multiple business areas and modules.

- **Interoperability** – Interoperability is the ability of a system to work with or use parts of another system. Business services use flexible defined data interface structures to exchange information or reuse processing logic that exists in another system or module. An example of interoperability is the ability to use health records, education, and justice systems data as part of a 360^0 view.
- **Adaptability** – Adaptability enables customizing common services to support additional or unanticipated needs. An example of adaptability is the ability to verify provider credentials through one licensing bureau for medical doctors and through a different licensing bureau for holistic doctors.
- **Extensibility** – Extensibility is the ability to add new functionality and capabilities quickly and efficiently. An example of extensibility is adding the ability to verify that a provider is qualified to perform a specific service.

3.2 BUSINESS SERVICES APPROACH

The FX Business Services Approach provides guidance for providers and consumers of business services. Guidance to providers explains the processing flow, identification, establishment, and operation of business services. This guidance includes business service definition, design and development, testing and implementation, operation, monitoring, and business service evolution. Guidance for users or consumers of business services explains identification, business service structure and provides usage insights.

The FX Business Services Approach seeks to help achieve the strategic priorities of the FX and the FX Technical Architecture goals and objectives. Specific elements of the FX Business Services Approach include:

- **Use business services to help achieve strategic priorities** – The standardization of processing enabled using business services inherently helps improve data quality, improve recipient experience, and improve provider experience. Use of business services by all systems that perform a business function allows consistent data validation, data transformation and application of business policy. The standardization enforces data quality validation and consistency in user experience for recipients and providers. When change or enhancement in business process is needed changes to the business service provides a single point of change and an easier method to perform impact analysis.
- **Use of an integrated team of business, information, and technology expertise** – Identification, design, development, and testing of reusable business services is most efficient when business service developers incorporate the insight and expertise of affected business areas, FX project capabilities, and system development disciplines.
- **Prioritize implementation and usage based on usage, benefits, and risk** – Prioritizing the implementation and usage of new business services should consider the frequency of usage, the benefits of use and the risks of using or not using a business service. Business services that are widely used are a priority because of the processing consistency and the reduction in custom code used in different systems. Business services that produce FX benefits are a prioritization factor. Examples of benefits include real time processing, improved data quality and faster business process completion.

Risks considerations are also a factor in prioritizing business services. Examples of risk considerations are the processing impacts from inconsistent processing, risks of data integrity, duplication and sequencing related errors.

- **Migrating embedded processing to use business services in existing applications** – For systems that have embedded business process logic in multiple existing applications; there are long term benefits of using business services and eliminating the use of custom logic. The benefits include enhanced functionality, consistency when business processes change, increased data quality, and standardization of business activity monitoring.

Industrialize and harden – In planning the implementation and use of business services, it is important to build services with robust validations, error-free processing, and efficient and effective exception handling. Industrialization means applying the manufacturing concepts of interchangeable and independently deployable components to application development. Hardening, within the state environment, or could be Cloud-Computing based. **Exhibit 3-1: Business Service and Technical Service Infrastructures** depicts scenarios where business services and technical services are:

- Active locally (shown as Standard Process below)
- Active in a mixed environment (shown as Hub Service below)
- Active completely in a Cloud-Computing environment (shown as Full SaaS below)

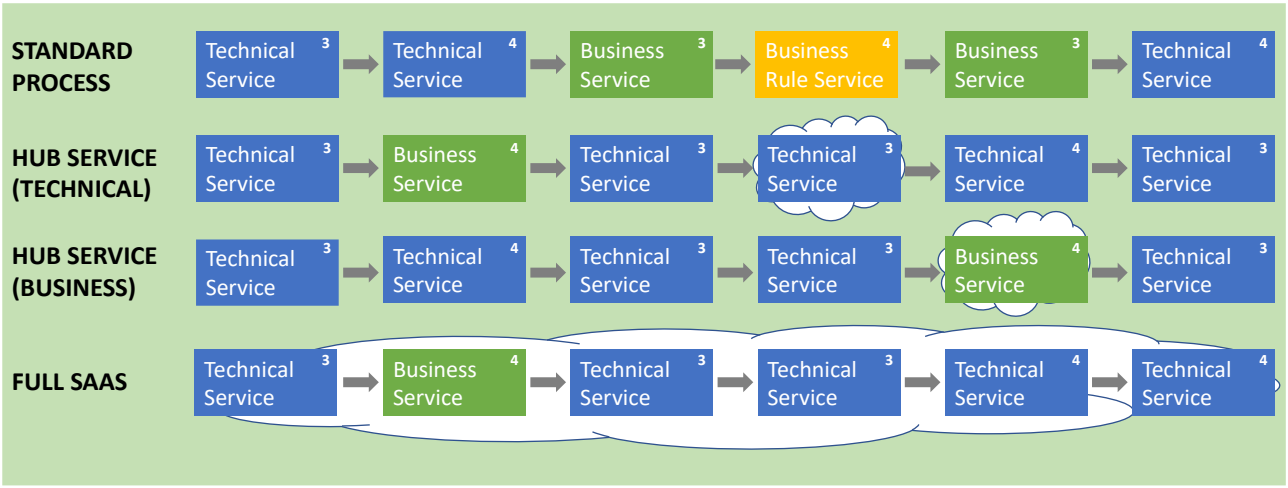

Exhibit 3-1: Business Service and Technical Service Infrastructures

3.2.1 Business Service Identification

All business services are logical extensions of some enterprise business process capability derived from the business architecture. These business process capabilities derive from the Business Capability Matrix (BCM). As part of the development of the BCM, a definition for each business-process capability for each of the five maturity levels is created. In the BCM, maturity

levels 3 through 5 expect use of business services. Based on the specific business capability, CMS may not require use of a business service for maturity levels 1 and 2.

Exhibit 3-2: Business Area, Business Rules, and Business Capability Maturity Interaction depict the relationship between a MITA Business Area, Business Rules and Business Capability Maturity.

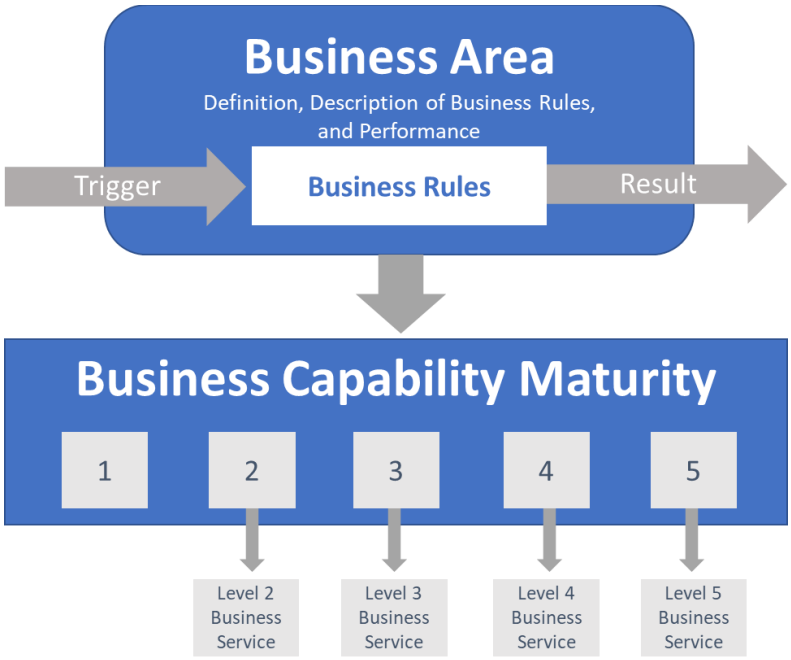

Exhibit 3-2: Business Area, Business Rules, and Business Capability Maturity Interaction

To accommodate these execution details (e.g., performance, platform, infrastructure, and software model), each business service may have one (1) or more logical descriptions of the service, and for each of these, may initiate one (1) or more deployments. The Agency defines the specific execution details for each unique deployment by the Agency or vendor-developed logical service definition. **Exhibit 3-3: Business Capability Maturity** shows this relationship.

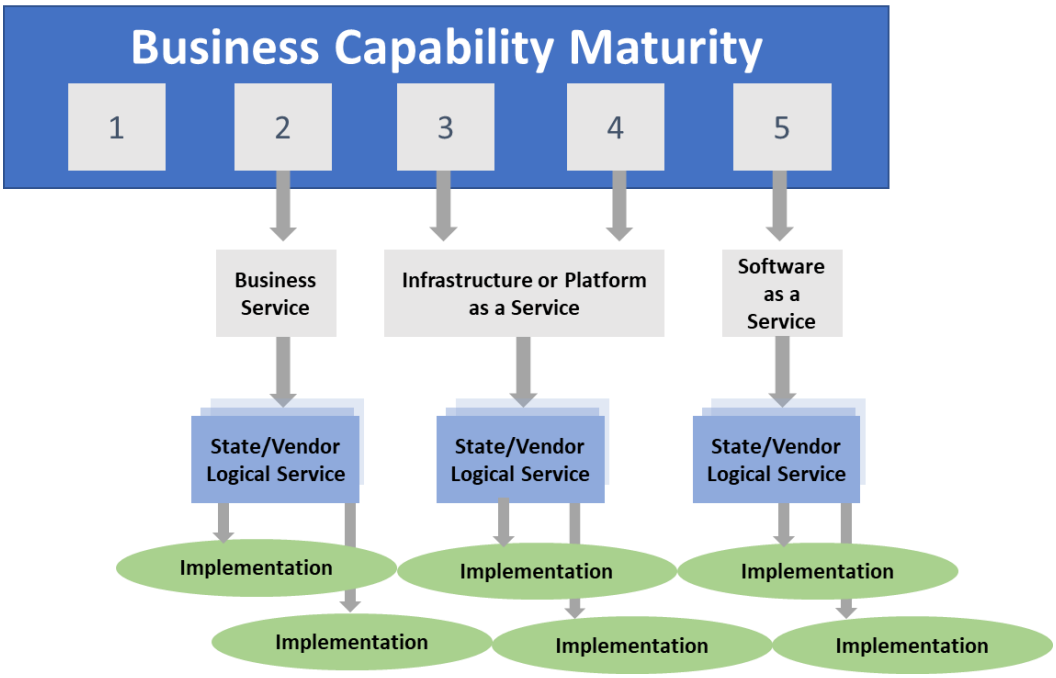

Exhibit 3-3: Business Capability Maturity

3.2.2 Business Service Solution Sets

Because FX business services are implementation-neutral, the FX Framework requires a method for documenting execution details. This means that the individual Agency or FX Project Owner does not need to recreate the service's solution. The solution sets are pattern-specific and platform and technology-dependent.

- A solution set is a deployment of an FX business service.
- **Exhibit 3-4: Relationship of Solution Sets to Business Processes** depicts solution-set mapping.
- The SEAS Vendor uses the Application Service Registry (ASR) to determine if a solution set implementation already exists for a service applicable to its specific environment.

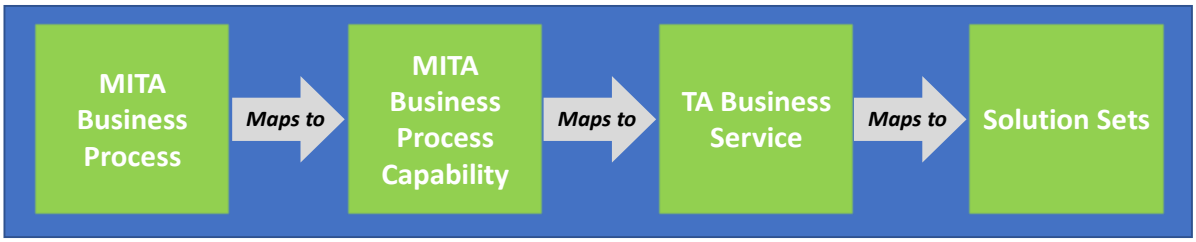

Exhibit 3-4: Relationship of Solution Sets to Business Processes

3.2.3 BUSINESS SERVICE FLOW

A business service's objective is to provide an independent version of a business process that can merge with other services to arrange composite-business processes. Business services that contain the following architectural characteristics provide this independence:

- Loosely coupled services
- The service has no predefined predecessor or successor services to an individual service
- Configuring services requires using a service contract which defines access to the service and an orchestration language
- Making changes to the orchestration instead of the service itself, enables changes to the flow of services
- Mandatory interface compatibility exists between the services

Orchestration is the process to define a flow that links several services together. The FX Business Services Approach is to achieve orchestration by using the latest version of the Business Process Execution Language (BPEL) and the Business Process Model and Notation (BPMN).

Orchestration defines the successor and predecessor services to a specific service. The developer of a business service is responsible for developing the BPEL and the BPMN orchestration and submitting those to the service registries as part of a business service solution set. After placement in the FXPR and service registry, the BPEL and the BPMN orchestrations are available**.**

Exhibit 3-5: Example of Process Orchestration Modification shows the insertion of a technical service into the orchestration stream.

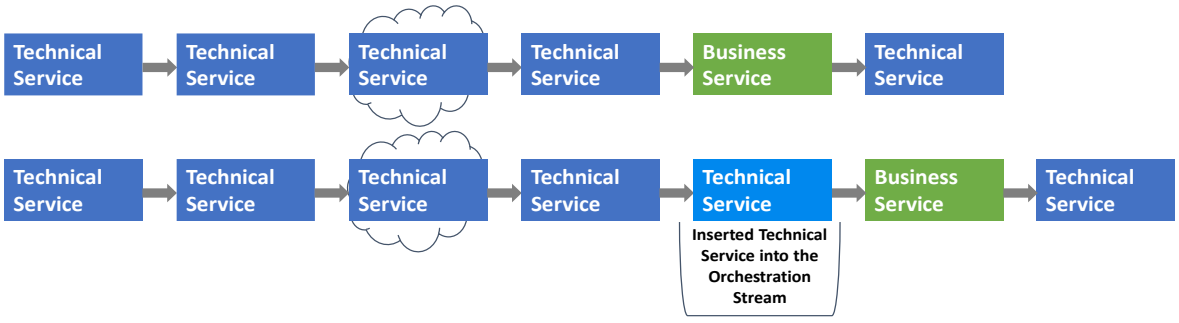

Exhibit 3-5: Example of Process Orchestration Modification

3.2.4 Business Service Construction Layers

The approach to building business services is much like an assembly line. The components of the business service are assembled from multiple smaller items. Each of the smaller items may be reused in other business services. In this context, business services are constructed by calling other business services, data services, or business rule services. Each of these other services are constructed to facilitate reuse. This allows for the construction of large grained, also referred to as coarse grained, business services without the usual risk and cost. Large grained refers to the amount of functionality included inside the service. The larger the service the less likely that it will be reused and the more likely that it will be subject to change over time. Large grained services have the advantage that they reduce complexity for user interfaces and batch processes because one use of the service executes large amounts of functionally. In this approach, assembling pretested blocks of reusable code, greatly reduces the total cost.

Exhibit 3-6: Business Service Construction Layers shows the assembly of components of a business service into a solution set (the deployment of a business service).

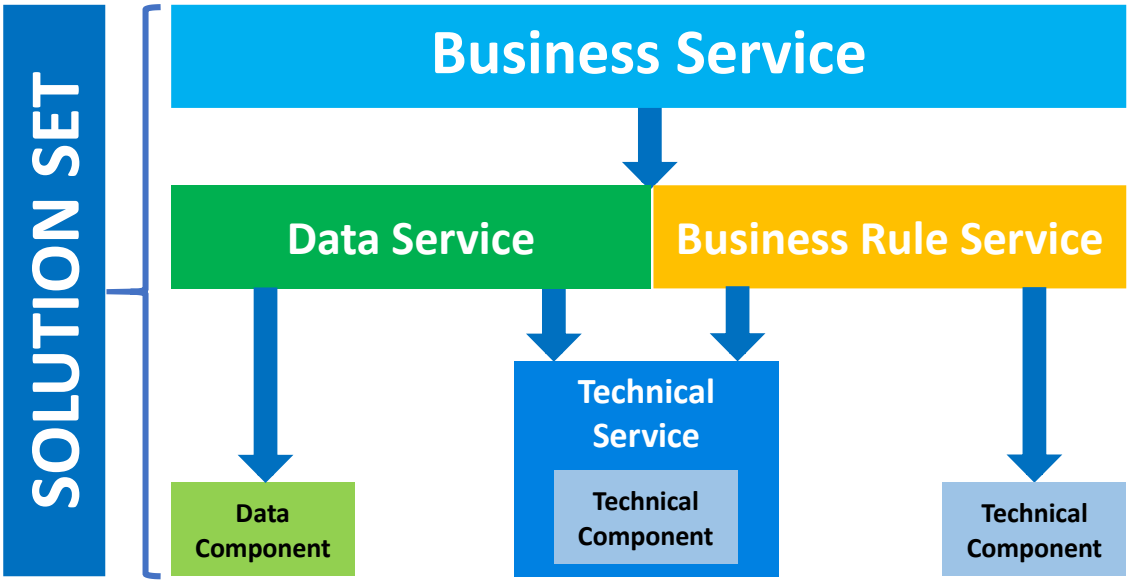

Exhibit 3-6: Business Service Construction Layers

3.3 Business Service Development

Business service development is the high-level process to identify, document, and approve a business service need that a Business SME, SEAS Vendor, or FX Project Owner identifies during project design**.**

Exhibit 3-7: Business Service Request Workflow Process shows the steps to create a Business Service Request.

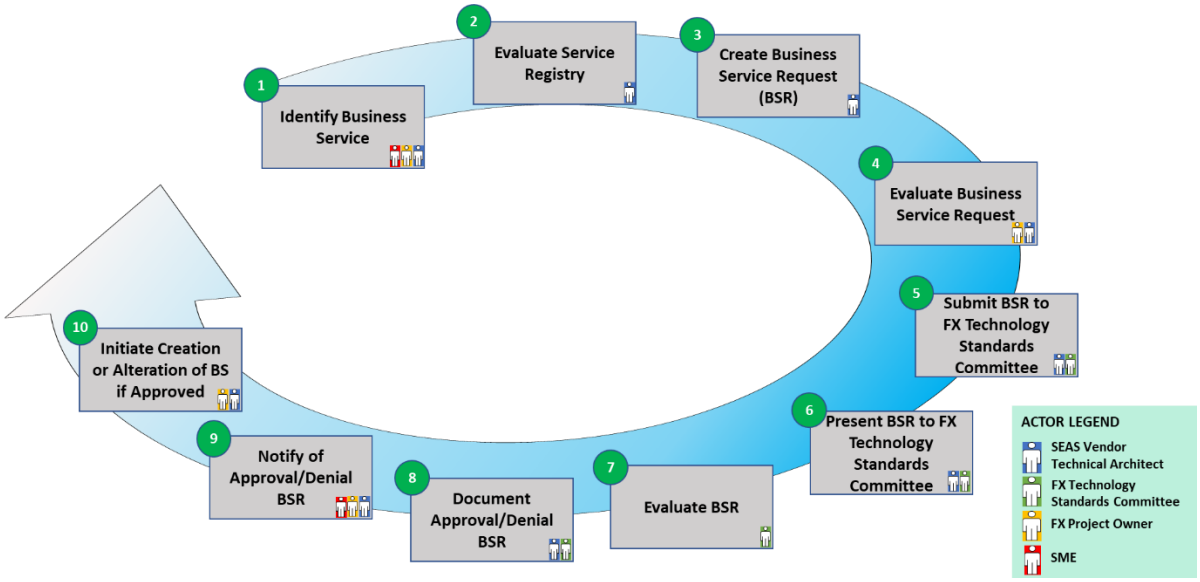

Exhibit 3-7: Business Service Request Workflow Process

The steps to create a Business Service Request are:

1. **Identify the Business Service** – The SEAS Vendor, FX Project Owner, or SME identities a business process that justifies consideration for automation as a reusable business service. The first step in developing a business service is determining the MITA business process the service enables, and the MITA Maturity Level of the associated business process. After identifying the specific business process, identify other applications and systems that use the MITA business process.
2. **Evaluate the Service Registry** – The SEAS Vendor reviews the Application Service Registry to find similar or related business services and associated documentation. Based on its findings, the SEAS Vendor evaluates and proposes a course of action (e.g., use an existing business service, modify an existing business service to support FX business needs, create a new business service, do not use a business service).
3. **Create Business Service Request** – If the SEAS Vendor course of action is to use a new or modified business service, the SEAS Vendor completes the Business Service Request form. The SEAS Vendor documents all the pertinent research and answers all the applicable questions.
4. **Evaluate Business Service Request** – The SEAS Technical Architect works with the FX Project Owner to document additional details about the Business Service Request.
5. **Submit Business Service Request to FX Technology Standards Committee** – The SEAS Technical Architect submits the Business Service Request to the FX Technology Standards Committee and then schedules presentation to the FX Technology Standards Committee.

6. **Present Business Service Request to FX Technology Standards Committee** – The SEAS Technical Architect presents the Business Service Request to the FX Technology Standards Committee.
7. **Evaluate Request** – The FX Technology Standards Committee reviews the request and determines the next action. Possible next actions are to recommend approval to move forward, complete denial, or request changes, and resubmission of the request.
8. **Document Approval/Denial for Business Service Request** – The SEAS Vendor documents the results of the FX Technology Standards Committee's evaluation.
9. **Notification of Approval/Denial for Business Service Request** – The SEAS Technical Architect provides official notice to the Stakeholders to the request indicating whether the request is approved or denied via email.
10. **Initiate Creation or Alteration of Business Service** – If the request is approved, the development or alteration of the business service can be scheduled with a SEAS Project Coordinator. If necessary, the SEAS Technical Architect returns to step 1 to modify the request for future review. If the denial is permanent, the SEAS Technical Architect will mark it as such and document the reason(s) for the denial.

3.4 Business Services Registry

The Business Services Registry manages the list of business services accessible through the ESB or other technical gateways. The Business Service Registry contains information that defines the service instances and their locations. The IS/IP Vendor solution and involvement is to establish a Business Service Registry supporting the ESB as an FX foundational capability and make Business Services discoverable. Although the initial release of the Business Service Registry is for the FX, the goal is to share or publish Business Services Registry entries to a federal registry.

To manage the development of a business service and its components (e.g., technical service, business rule services, etc.), the FX solution will also establish an ASR. When business services are ready for deployment, they will be implemented by the IS/IP Vendor.

3.4.1 Application Service Registry

The ASR is software used to track the available services. Unlike a Service Registry, which is often a component of an ESB used to discover services, an ASR is a database of internal application services and components that are used to create a business, technical, data, and business rule service. The ASR is a necessary part of an Application Architecture that requires and supports reuse. This FX Application Architecture will likely generate ten thousand or more objects in the ASR. The Business Service Registry will include several hundred services. The reuse of the objects in the ASR will likely average around two (2) reuses for each of the ten thousand objects. In comparison to existing FMMIS architecture, this reuse would be considered massive and requires automation to properly support impact analysis.

ASR provides the need-to-know specifics to conduct impact analysis. To maximize the usefulness of the ASR the FX strategy is to populate entries for all services of the Agency and cross-Agency services in the ASR.

Most commercially available registries are tailored to the ESB or other similar product(s). The capabilities of the service registry provided by the IS/IP Vendor will determine if the ASR needs to be custom developed.

The ASR to support the FX must offer the following functionality:

- **Full Life Cycle Support –** The ASR must be able to accept proposed usage from the initial design before any code is written.
- **Versioning –** The ASR must be able to accept and manage multiple versions of a single object.
- **Impact Analysis –** The ASR must be able to determine which objects are impacted by proposed changes. This can help when determining when versioning is required because of the impact of a proposed change.
- **Searching –** The ASR must be able to support the search for existing objects. As the number of objects grows, this becomes a more difficult task. Keywords and interface details can help with the search.
- **Ownership –** The ASR must identify the owner of every object. Owners are necessary to validate requested changes to services, agree to resources needed to change services, and coordinate the timing of service implementation and modification.
- **Project Tracking –** The ASR must track project changes as they effect all objects in the ASR.

The most important quality of the ASR must be its accuracy. Errors will lead to distrust, which then leads to usage issues. To ensure the ASR works, it will be an active part of the development process. The detail design process will integrate the search for existing and registration of new and proposed services. The ASR provides the structure chart for new development. Architects and developers update the ASR during development as the detail design changes. Automated tools will exist to verify the ASR during the promotion process to ensure the accuracy of the ASR.

The ASR must be able to extrapolate changes at any level into to the Dynamic Link Library (DLL), executables (EXEs), ESB objects, Web Application Programming Interface (APIs), mobile applications, and any other software deliverable that is part of the enterprise.

The ASR also offers an opportunity to collect additional data about the enterprise software process. Collecting data about development times, software changes and their costs, reuse statistics, and potentially performance data and failure rates, can expand the understanding of process problems.

The IS/IP Vendor shall be responsible for the development of the ASR.

3.4.2 SERVICE PARTS

Service parts are characteristics of any type of service (e.g., business, technical, business rule). The service parts characteristics include service name, service definition package, service

contract, purpose, business logic, constraints, use cases, solution set, structure diagram, performance standards, test scenarios and test cases, and map to MITA data models.

Service parts or links to the parts reside in the ASR. Supplemental content such as diagrams, logic, use cases will be stored in a location independent of a specific FX Project Owner. The location could be the FXPR document library, an Agency-level application life cycle tool, or a service repository provided with the IS/IP Service Registry.

This directory will contain folders by Service Type. For each Service Type that has artifacts, there will be a directory corresponding to the Service Name. Under each Service Name will be a folder named with a three-digit version number (that follows the document naming conventions). Artifact files should include the service part followed by the artifact type. Artifact types should use the FX Project Life Cycle (FXPLC) artifact names, where applicable. The artifact directory is currently unpopulated and will be populated for each project as needed.

3.4.3 Service Repository

The service repository is a store for web service objects and artifacts, particularly those used in the operation of enterprise service bus processing. The service repository is a component of the IS/IP that integrates with the ESB and Integration Platform service registry. The types of content included in the service repository can include parameters, message flows, and executable components used in processing.

3.4.4 Application Service Registry Maintenance Procedures

In collaboration with FX Project Teams, the SEAS Technical Architect creates or updates entries in the ASR List. They will collaborate with FX Project teams in making these changes. The ASR is a SharePoint list in the FXPR and maintained by the SEAS Technical Architect. SharePoint access control policy for the ASR defaults to read and create access for all FX team members and restricts edit authority to the SharePoint group *SEAS Technical Architect*.

Attachment G – How to Maintain the ASR List is a Word document that describes the procedures to maintain content in the Application Service Registry List (i.e., FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5)).

3.4.5 Application Service Registry Strategic Topic

Strategic Topic 3-1: Application Service Registry describes the Agency strategy for use of an Application Service Registry tool.

Application Service Registry	Current	2018	Timeline 2020	2022	2025
Module / System provider preference	X		Begin migration to Agency-wide ASR		->

Application Service Registry	Current	Timeline 2018	2020	2022	2025
Module / System Specific Implementation of Mandated ASR COTS product					
Use Optional		2019			
Use Mandated – Division-wide ASR					
Use Mandated – in Agency-wide ASR			All Services exposed on IS/IP	->	
Use Mandated – Cross-Agency ASR				Reassess or Pilot Use	

Analysis

Most ESBs contain a service registry, but this is not sufficient for the needs of the Agency. A tool is needed to track the governance of services. For this to occur, all parts of the enterprise need to centralize their different services into one location to allow for the greatest possible reuse. This tool needs to support the new complexities of the Application Architecture.
These complexities include:

1. Preventing duplication
2. Tracking availability
3. Determining Impact Analysis
4. Support for Multiple active versions of a service
5. Service Ownership
6. Project Estimating
7. Service Governance

Strategic Topic 3-1: Application Service Registry

3.5 Business Services Parts

Business services parts are the characteristics that describe a business service. Business service parts include service name, business service definition package, service contract, purpose, business logic, constraints, use cases, solution set, structure diagram, performance standards, test scenarios and test cases, and map to MITA data models.

The SEAS Vendor documents the business service parts based on input from the Business Request Form and associated business analysis. The FX Project Owner documents the Business service parts that reflect the design, development, testing, and implementation artifacts for the business service before development, coding and testing the business service and associated solution set.

The following sections describe each business service part's content, format, and sample artifact, if available.

3.5.1 Business Service Definition Package (BSDP)

A complete Business Service Definition Package (BSDP) contains the following parts:

- **Service Name** – The Service Name should match the Business Process name to avoid confusion.
- **Configuration Data** – Configuration data includes the following:
  - **Framework** – Defines the FX Logical Data Model (LDM).
  - **Deployment** – Defines state-specific logical and physical-service definitions.
  - **Version** – Indicates the configuration data's release or development number.
  - **Activation Date** – Indicates the date of the release package.
- **Service Contract Development** – Service Contract Development includes the following elements:
  - **Purpose** – A short one- to two-sentence description of the service's function(s), which comes from the business process definition and BCM.
  - Documentation of the service's functionality and capability.
  - List of any service constraints.
  - Identification of common service requirements and candidate areas for business service adaptability and extensibility.
    - Adaptability enables the customization of common services to meet the needs of a specific state. An example would be verifying provider credentials to one (1) licensing bureau for medical doctors and a different licensing bureau for natural doctors.
    - Extensibility enables states to add new functionality to common services to meet their specific needs, while still meeting MITA goals and objectives. An example is adding the ability to verify that a provider qualifies to perform a specific service.
  - **Formal Interface Definition** – Definition of the formal interface requires using the business-process triggers, the results, and the FX LDM. The definition also documents the use of the service's interfaces and operations. Prior to design and development, this is an informal textual description (e.g., template) of the service's interface. As design occurs, the formal interface definition evolves to be a Web Service Definition Language (WSDL) interface description.
- The WSDL includes the following elements:
  - **Interfaces** – An interface includes a function library (or a module or class) in a traditional programming language, such as WSDL, which describes the interfaces (e.g., connection points), that a MITA business service exposes.
  - **Operations** – Much like a function in a traditional programming language, an operation associates a message-exchange pattern with one (1) or more messages. A message-exchange pattern identifies the sent and received

message sequence and detects those who send and receive the messages logically. An interface groups together operations without any commitment to transport format.

  - **Messages** – Messages define the data communicated with the service. These messages are the service's input triggers or output results, and the messages use the W3C XML Schema to describe them. The message is a combination of a MITA header and a standard payload. If possible, the payload message uses a standard format (e.g., Health Level Seven International (HL7) or ASC X12).
  - **Parts** – Parts define attribute level definitions of elements within a WSDL and are based on a state- or vendor-specific solution set.
  - **Services** – Services are a list of functions performed.
  - **End Points** – End points defines the physical implementation where the service can be accessed and is based on a state- or vendor-specific solution set.
  - **Bindings** – Bindings provide a map of the actual protocol used for the messages.
  - **Types** – Types are data contained within a message.
  - **Documentation** – Documentation is a free-form text document service.

- **Use Case** – Documents the main success path, alternate path(s), critical failure conditions, and scenarios using Unified Modeling Language (UML).
- **Solution Set** – Based on a state- or vendor-specific technical implementation, which is a deployment of a FX business service. An example of solution set is a deployment for a specific technical platform.
- **Business Logic** – Describes the logic the service performs, and the behavior of the non-transparent service. Prior to design and development, this can be free-form text from the business process definition. During design and development of the service, the business rules specify the business logic.
- **Performance Standards** – Performance standards derive from the performance metrics that the business service definition specifies based on a state- or vendor-specific solution set.
- **Test Scenarios, Data, and Cases** – Used to validate service-contract compliance; these elements are based on a state- or vendor-specific solution set.
- **Map to FX Data Models** – Identifies the use of logical data model classes and elements used by the service.

3.5.2 SERVICE CONTRACT

Describes the expected behavior of the interface and the security and privacy constraints on the service. The following are examples of interface behavior patterns:

- One-way only receives or outputs data (e.g., report generator).
- Involved parties negotiate a specific Service Level Agreement (SLA).
- Two-way receives and sends data. Two-way traffic has two (2) other attributes:

- Initiator defines who initiates the interface, either the service (e.g., request for information) or the outside client (e.g., inquiry).
- Processing characteristic defines the relationship between the input and the output:
  - **Point-of-Sale (POS) transaction** – A real-time transaction (e.g., a pharmacy POS) that features a constrained response time and high reliability.
  - **Online transaction** – An inquiry on a provider that features a more relaxed response time while still allowing for conversation-type human interaction.
  - **Batch** – Typical batch processing constraints.
  - **Asynchronous** – No constraints on processing times but response and coordinating data are required.

3.5.3 Business Logic

Describes what business processing occurs within the service. Business Logic provides the underlying rationality, functionality, and capability that the service provides. Initially, the business logic is free-form text or template driven, which an application developer defines and codes to incorporate into a business-process orchestration step, and possibly integrates into a rules engine. This field only applies to business services.

3.5.4 Constraints

List of any service restrictions.

3.6 Business Service Example

This section highlights the business service approach, development approach, processing and business parts documentation concepts described in this section using a representative real-world example. The context for this example is the business service, **Add Provider**, which is the last activity of the MITA Business Process of **Enroll Provider**.

The **Add Provider from Enrollment** business service example provides complete context for validating all business rules related to a single provider. This business service represents a single business service to support the complete addition of a Provider record or a complete rollback if an error is determined.

The design of this large business service uses other small business services. Doing so supports potential reuse and simplifies the overall process. For example, the **Add Provider from Enrollment** business service requires processing to add the provider address. This business service reuses the **Maintain Provider Address** business service processing performed during provider maintenance and potentially during several other business services. There are several similar variations to the design approach in this example.

The vendor would begin by creating an initial design version of the BSDP. This would include a proposed structure chart like the structure depicted in **Exhibit 3-8: Add Provider from Enrollment Business Service Structure Chart**.

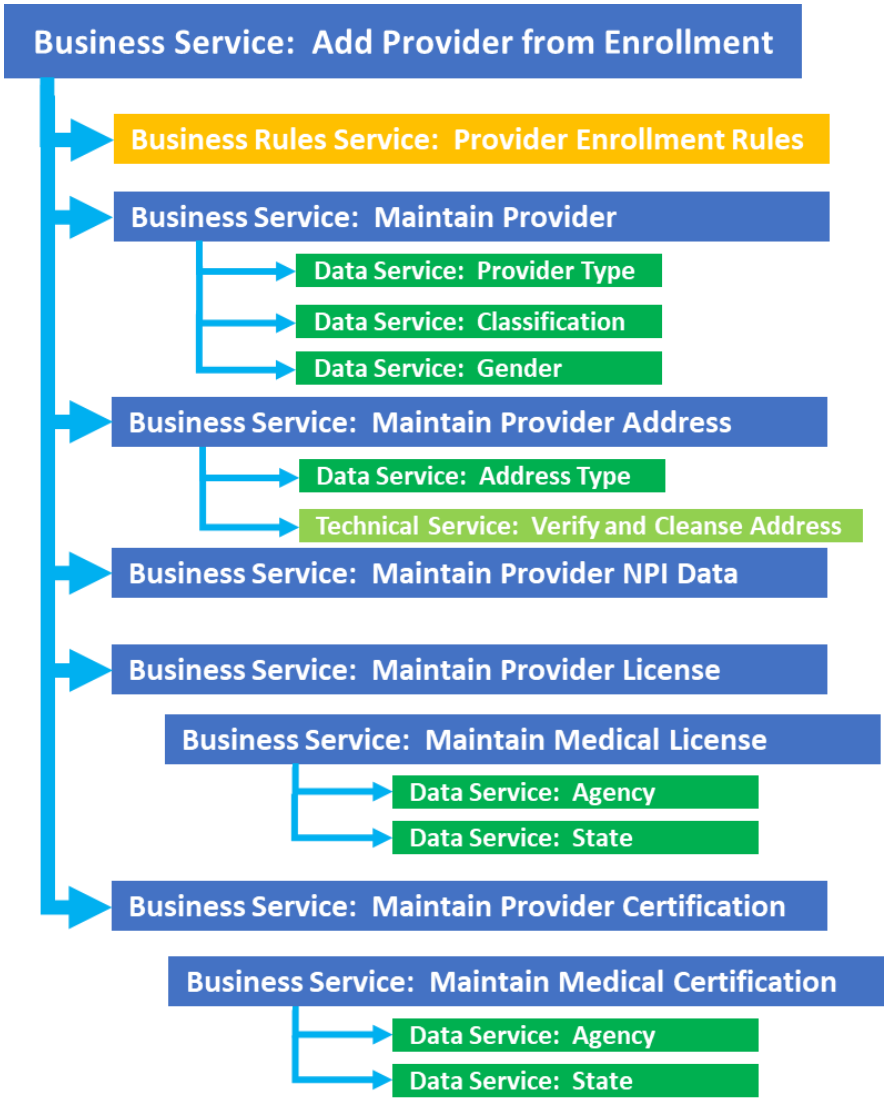

Exhibit 3-8: Add Provider from Enrollment Business Service Structure Chart

The FX Project Owner selected to implement this business service then analyzes the ASR to see what services already exist or need to be developed. In this example as seen in **Exhibit 3-9: Add Provider from Enrollment Business Service Structure Chart Analyzed**, all the lower-level Services exist except for four services. The FX Project Owner would develop each of these new services (business, technical and data services) to enable the functionality of the overall

business service. As the system matures, the submission of service requests should decline because most services will already exist.

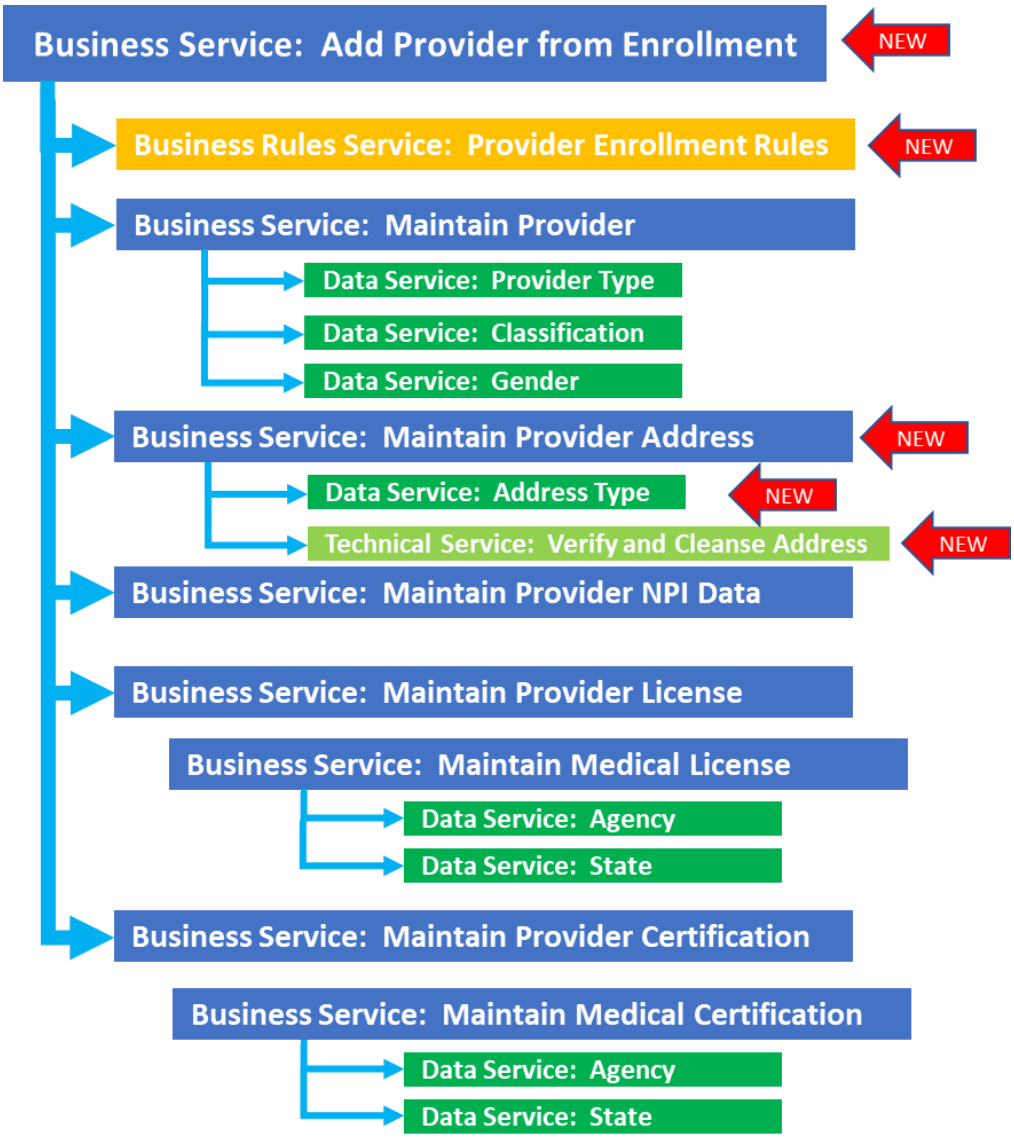

Exhibit 3-9: Add Provider from Enrollment Business Service Structure Chart Analyzed

The path below leads to an example of the Business Service Request Form the FX Project Owner would fill out for the **Add Provider from Enrollment** and the **Maintain Provider Address** business services. There are more examples of business services in the ASR located in the FXPR.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment A Business Service Request Form Example.

SECTION 4 TECHNICAL SERVICES

4.1 TECHNICAL SERVICES OVERVIEW

Technical Services derive from business service requirements and provide the underlying technical functionality to enable business processing. This section provides guidance defining

- What is a technical service?
- Technical service approach to processing and use
- Technical service development
- Technical Service Registry purpose and use
- Information managed about technical service items

This section follows the content organization of the business services section above. In addition to describing standard technical services, the section also describes two specialized technical service implementations:

- Data Services
- Business Rule Services

The business services section above describes the overall structure and processing flow of business services and general strategy for all types of services. Significant content in this section aligns with and reiterates guidance provided in MITA 3.0 Part III Technical Architecture, Chapter 4 Technical Services. The *Technical Services* section builds on MITA and FX business services guidance providing additional approach and technology architecture strategy guidance specific to technical services.

4.1.1 TECHNICAL SERVICE DEFINITION

Technical services are a set of technical functions used to support business service requirements. The context of a technical service considers the Technical Service Area (TSA) and further definition by Technical Service Classifications (TSC).

4.1.2 BENEFITS OF TECHNICAL SERVICES

The use of technical services provides the following benefits:

- **Reuse** – Multiple programs, systems and organizations can share technical services. Each time an additional system reuses a technical service, it provides additional returns on the service's original investment cost. An example is security access and controls, which is a needed technical service for every application. By writing this service once and reusing it for all applications, stakeholders reduce the overall development time and complexity.

- **Cost** – A technical service written once and maintained in one location reduces the overall development and maintenance costs. An example is in security access and controls; costs increase when each application writes its own security access and control modules.
- **Consistency** – When multiple applications share a technical service, the processing results are highly consistent. For example, using the security access and control services, users benefit from using the same password to access multiple systems. Another example is performance standards technical services that measure performance in the same manner. Use of technical services allows processing comparisons between organizations or within a single organization over a period of time.
- **Flexibility** – Widespread use of technical services makes it possible for systems to be more responsive to change. Changes occur by improving the functionality of a single service, by using a different service to accomplish a task, or incorporating new services.
- **Broader Market Place** – The FX Technical Architecture defines technical services in general industry terms that are not proprietary to the Medicaid environment – this encourages use of technical solutions from non-Medicaid environments in the MITA Medicaid Enterprise.

FX technical services supports the Agency alignment with the MITA enterprise architecture. The Technical Services Approach defines technical services as common solutions that enable Agency-specific deployments, making it possible to develop adaptable and extensible services. The Technical Services Approach strives to help meet the Agency business needs in a rapidly changing Medicaid environment.

4.2 TECHNICAL SERVICES APPROACH

The Technical Services Approach provides the guidance and specifics for architects and developers about how to design a technical capability that follows MITA-standards. This section's focus is to present a Technical Service Approach that meets the following criteria:

- Focuses the technical assessment process on the business needs
- Provides a method to identify technical services and applications available to move between maturity-capability levels
- Determines the technical assets currently available in the state's infrastructure that can support future business needs
- Reflects the technical needs, in conjunction with business service solution sets and assessments, which will satisfy the Agency's *To-Be* configuration

The definitions used in the SS-A Technical Capability Matrix (TCM) provide the state with a technical-capability assessment tool for the State Self-Assessment (SS-A).

The Technical Services Approach provides guidance for providers and consumers of technical services explaining:

- Technical Services Identification
- Technical Services Area (TSA)
- Technical Services Solution Sets
- Technical Services Flow
- Technical Services Components
- Technical Architecture
- Technical Components

The Technical Services Approach seeks to help achieve the strategic priorities of FX Project Owner and the FX Technical Architecture goals and objectives. CMS MITA version 3.0 provides significant foundational guidance for FX technical services. Specific elements of the Technical Services Approach beyond MITA guidance includes:

- **Align with industry standard technical service interface and message protocols** – Many technical services rely on commercial products or services that enable reusable technical processing (e.g., calendaring, address standardization, document management). Often category standards exist to allow interoperability of these commercial products. There are often multiple commercial products in a specific category. While the commercial products may offer interfaces that extend industry standards or proprietary interfaces, the tradeoffs of extending beyond category interoperability standards need careful evaluation.
- **Intelligent use of technical service wrappers** – technical services will require a service wrapper whether implemented by ESB and integration platform or as a code-based wrapper that provides independence from vendor specific technology services or product interfaces. This allows consumers to use the same technical service even if processing is directed to another processing service.
- **Strategic licensing of commercial software or services** – Usage of technical services often exceeds expectations. If underlying commercial products support technical services, FX will seek licensing that is best value. Typically, that means avoiding transactional arrangements or licensing tied to a user population, specific application, or specific business unit. Licensing issues can be enough of a factor to drive use of open source solutions.
- **Controlled use of technical service templates** – Templates can help architects and developers to standardize and reuse implementations of technical design patterns. The FX approach related to templates is to use with caution. Templates that provide the discretion for processing to be changed may allow bypass of required technical architecture processing.

The above elements of the Technical Services Approach will evolve and expand in subsequent versions of the *T-5: Technical Architecture Documentation* as FX technical services maturity grows.

4.2.1 TECHNICAL SERVICE IDENTIFICATION

All technical services are the logical extension of some enterprise technical function capability derived from the Technical Architecture. CMS defined technical function capabilities during the creation of the SS-A Technical Capability Matrix (TCM). As part of the development of the SS-A TCM, CMS created maturity level definitions for each technical capability. In the SS-A TCM, technical capability maturity levels 3 through 5 expect creation and use of technical services.

Exhibit 4-1: Technical Function, Technical Rules, & Technical Capability Maturity Interaction depicts the relationship that a MITA technical service exists per technical function, per maturity level.

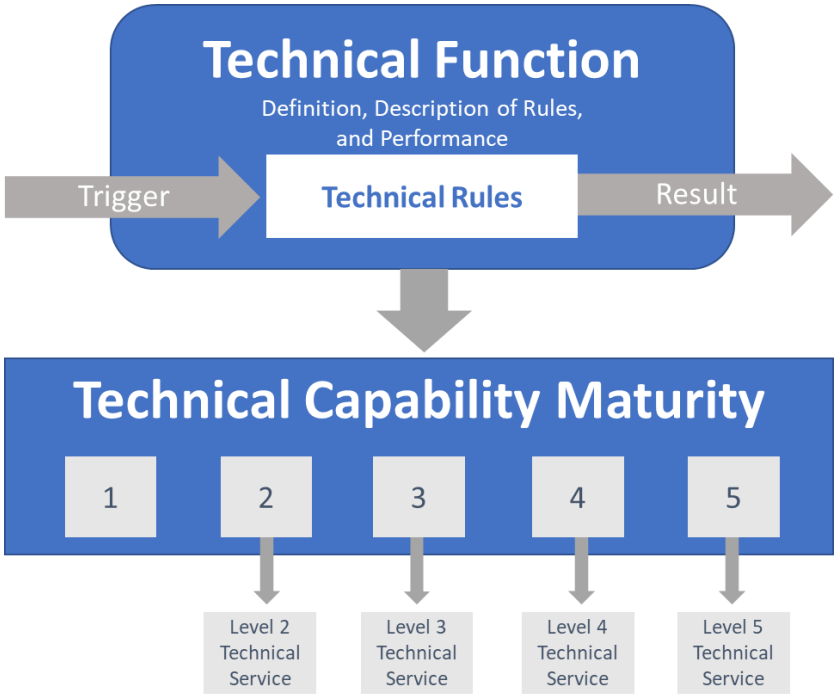

Exhibit 4-1: Technical Function, Technical Rules, & Technical Capability Maturity Interaction

4.2.2 TECHNICAL SERVICE AREA (TSA)

A Technical Service Area (TSA) is a piece of software that executes a generic-IT capability, contains a defined interface for its call, performs a defined function corresponding to the capability, and returns defined results. The MITA-standard Technical Services Framework consists of three (3) TSAs:

- **Access and Delivery** – The Access and Delivery TSA encompasses design drivers and enablers, such as Web-browser connectivity, language support, Customer Relationship Management (CRM) data, forms, and reporting services. The Access and Delivery functions directly affect the Agency's staff, the public, providers, and all other

stakeholders. The span of coverage of the services offered will tend to change over time as the demands and technology needs of the users evolve.

- **Intermediary and Interface** – The Intermediary and Interface TSA contains drivers and enablers, including process orchestration, workflow, and relationship-management functionality. The ESB capabilities handle the Intermediary services (also known as *middleware*). The Interface services tie to connectivity functions of external organizations that require a connection.
- **Integration and Utility** – The Integration and Utility TSA includes design drivers and enablers, such as solution stacks, database access layer services, scalability functionality, application-versioning capability, and verification-type utility services. These core-service components will be a combination of unique services and a set of reusable services. The layering of the types of services requires a combination of services for an end-user to retrieve data. The Interface and Integration TSA handles this orchestration of events.

Exhibit 4-2: Technical Service Area Conceptual Diagram depicts the components of the MITA Technical Service Area, which includes Access and Delivery, Intermediary and Interface, and Integration and Utility.

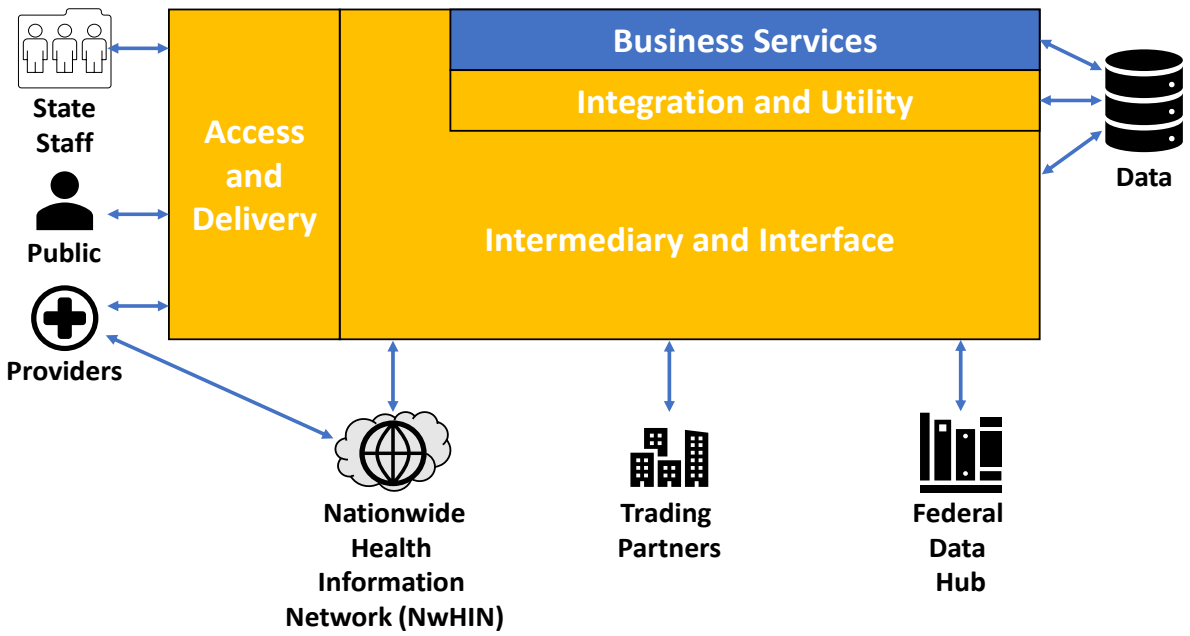

Exhibit 4-2: Technical Service Area Conceptual Diagram

4.2.2.1 ACCESS AND DELIVERY

The Access and Delivery TSA contains five (5) Technical Service Classifications (TSCs), which include Client Support, Business Intelligence, Forms and Reporting, Performance Measurement, and Security and Privacy.

Exhibit 4-3: Access and Delivery Technical Service Classifications, shows each TSC and examples of the type of functionality contained within a technical service. For example, language support and usability are services within Client Support, while authentication and encryption are services within Security and Privacy.

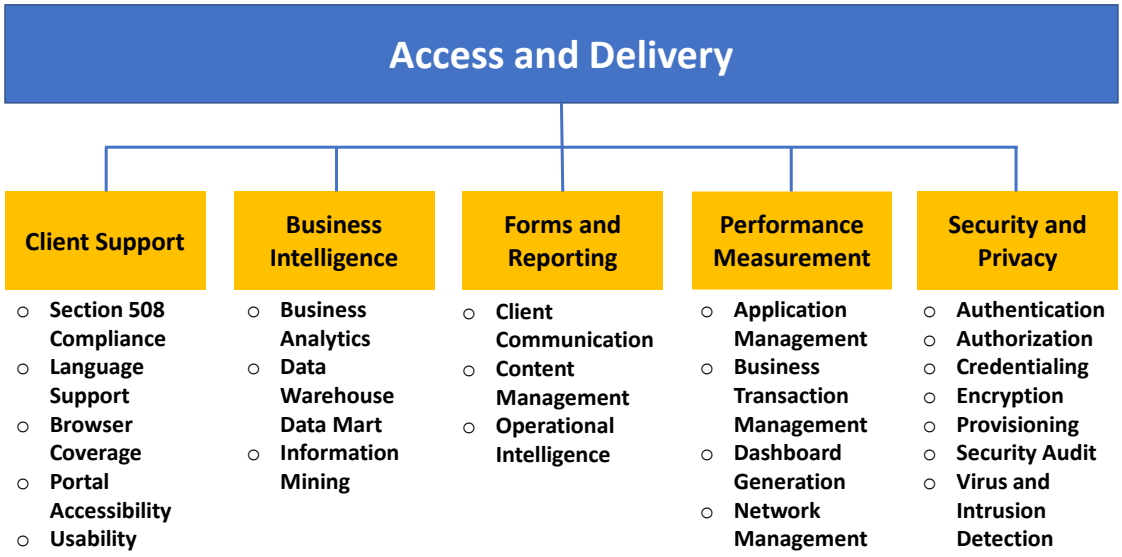

Exhibit 4-3: Access and Delivery Technical Service Classifications

These Access and Delivery technical services commingle in a composite manner with Intermediary and Interface services and Integration and Utility services, to provide full functionality required to satisfy the end-users' needs. The necessity to leverage proven-technology standards plays a major role in including the varying devices used by the public, the state's staff, and the provider community. Service Level Agreements (SLAs) are particularly important, especially regarding the necessity to support a request-and-forward community. Concise application tuning for influencing minimal throughput for these services should be part of the creation process of these application services.

A key component of this TSA includes using standardized security and privacy mechanisms. The protection of data assets – from both internal and external users – is a high priority for the FX. Because of this concern, vendors are to use established role-based security functions. Implementing a role-based access is a primary solution. In addition, common practice is to use a layered method to safeguard the state's assets to satisfy the various types of security and privacy needs. Additional security and privacy details, including standards information, are in the *T-6: Technology Standards* deliverable document located in the FXPR at FX Hub > Standards & Plans > Category: Technology.

4.2.2.2 INTERMEDIARY AND INTERFACE

The Interface and Integration TSA contains five (5) TSCs, including Business Process Management, Relationship Management, Data Connectivity, Service-Oriented Architecture, and System Extensibility.

Exhibit 4-4: Interface and Intermediary Technical Service Classifications depicts these TSCs and provides potential solutions and standards examples. For example, business analytics are contained within Business Process Management, while ESB services of the Integration Platform are within Service-Oriented Architecture.

The interactive activities associated with the FX intermediary services require an effective middleware-solution set. The Intermediary and Interface TSA combines the generally predetermined interface activities with the transactional roles performed by the intermediary services. The ability to interact with the primary FX data stores requires a strong tie to security and privacy services.

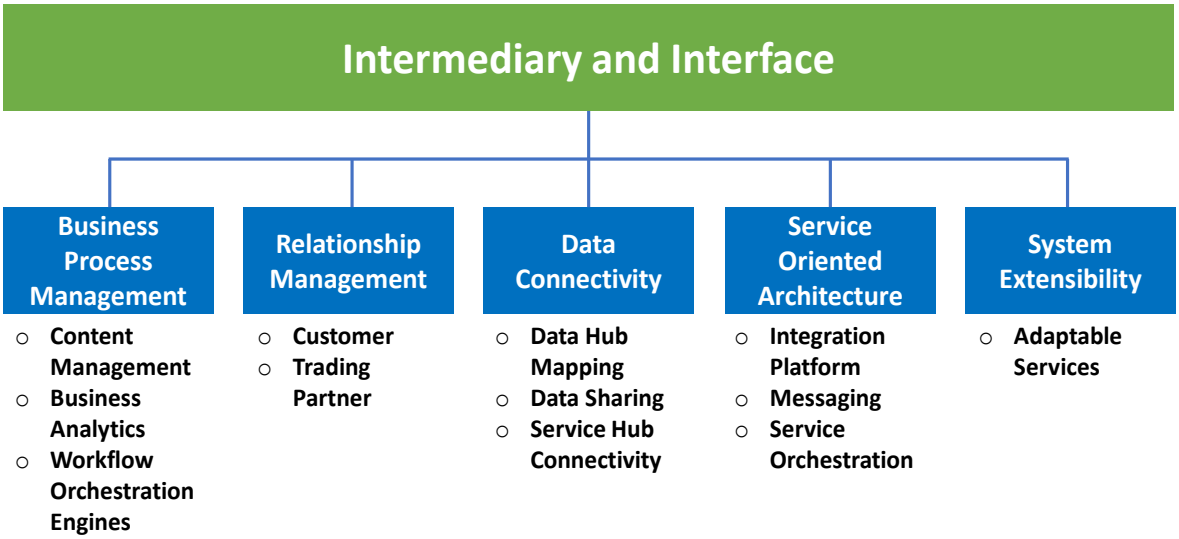

Exhibit 4-4: Interface and Intermediary Technical Service Classifications

4.2.2.3 INTEGRATION AND UTILITY

The Integration and Utility TSA contains five (5) TSCs, which include Configuration Management, Data Access and Management, Decision Management, Logging, and Utility. **Exhibit 4-5: Integration and Utility Technical Service Classifications** depicts these TSCs and provides examples of potential solutions and standards. Utility services are core services that perform a specific-Medicaid function beyond defined business services. Other general functions, such as event logging and configuration management, are part of this TSA's integration activities. One critical internal activity to the FX relates to the decision-management events used by claims processing and many of the other operational-management processes.

The Integration and Utility TSA classifies these functions, which link closely to the business services.

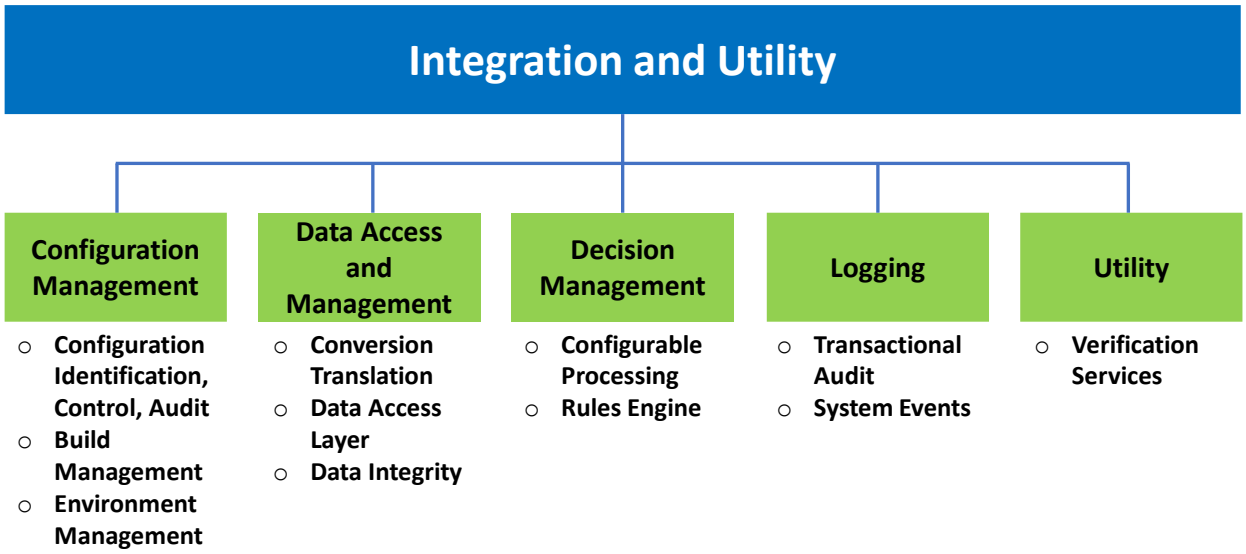

Exhibit 4-5: Integration and Utility Technical Service Classifications

4.2.3 TECHNICAL SERVICE SOLUTION SETS

The Technical Services Approach is for technical services to be implementation neutral. The FX Framework will require documentation of execution details; doing so eliminates the state and FX Project Owner's necessity to recreate the solution for each technical service. Solution sets are the physical deployments of technical services. The solution sets are pattern-specific and can be platform- and technology-dependent.

A technical service's solution set entails the Agency's implementation-specific Technical Service Definition Package (TSDP). The state's TSDP provides the technical service's specifications that the Agency is deploying. The technical service's system designer is responsible for producing the TSDP for a solution if one does not already exist in the Application Service Registry.

Exhibit 4-6: Conceptual Relationship – Technical Function to Solution Sets shows solution-set mapping.

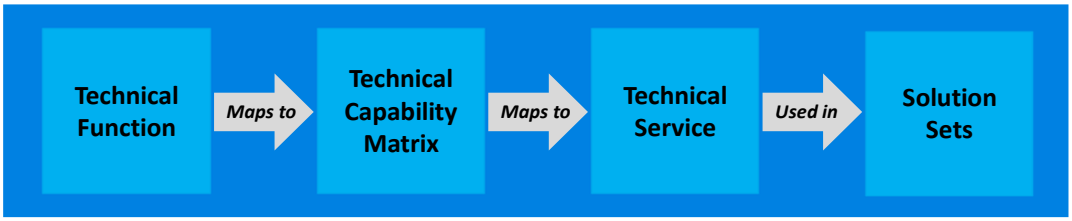

Exhibit 4-6: Conceptual Relationship – Technical Function to Solution Sets

4.2.4 TECHNICAL SERVICES FLOW

The technical services' objective is to provide an independent version of a technical capability that can merge with other services to form composite business processes. Using the following architectural concepts enables this independence:

- Services should be loosely coupled
- No predefined predecessor or successor services to an individual service exist
- Services are configured using a service contract and an orchestration language
- Changes to the flow of services are achieved by making changes to the orchestration, not to the service itself
- Mandatory interface compatibility between the services exists

4.2.5 TECHNICAL ARCHITECTURE

Exhibit 4-7: Service Orchestration Using the Conceptual Layout depicts an orchestrated flow of services using the conceptual service layout. This illustration does not provide details regarding the feedback loop but does show a simple combination of services within the service areas.

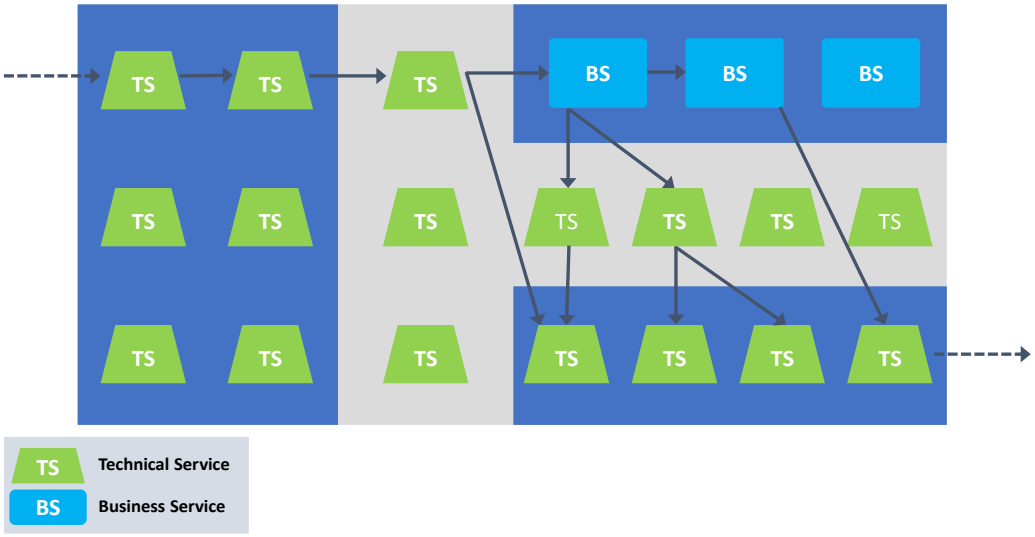

Exhibit 4-7: Service Orchestration Using the Conceptual Layout

Arguably, the biggest benefit of using an SOA is the ability to add or replace services with other services without affecting any of the other connected services or requiring their modifications, provided the replacement has a matching interface signature.

Exhibit 3-5: Example of Process Orchestration Modification illustrates adding a technical service into a previously defined orchestrated process. Using the service characteristics listed previously enables this achievement.

The service contract defines access to an individual service using WSDL. The process to define a flow that links several services together, called *orchestration*, typically uses Business Process Execution Language (BPEL) or the Object Management Group's (OMG) Business Process Model Notation (BPMN).

Service Orchestration defines successor and predecessor services to a service. Because this orchestration bases itself on definite-deployment specifics, these details are a part of the (ESB or workflow processing.

Exhibit 4-8: SOA Composite Service depicts a composite service that combines a business service with two technical services.

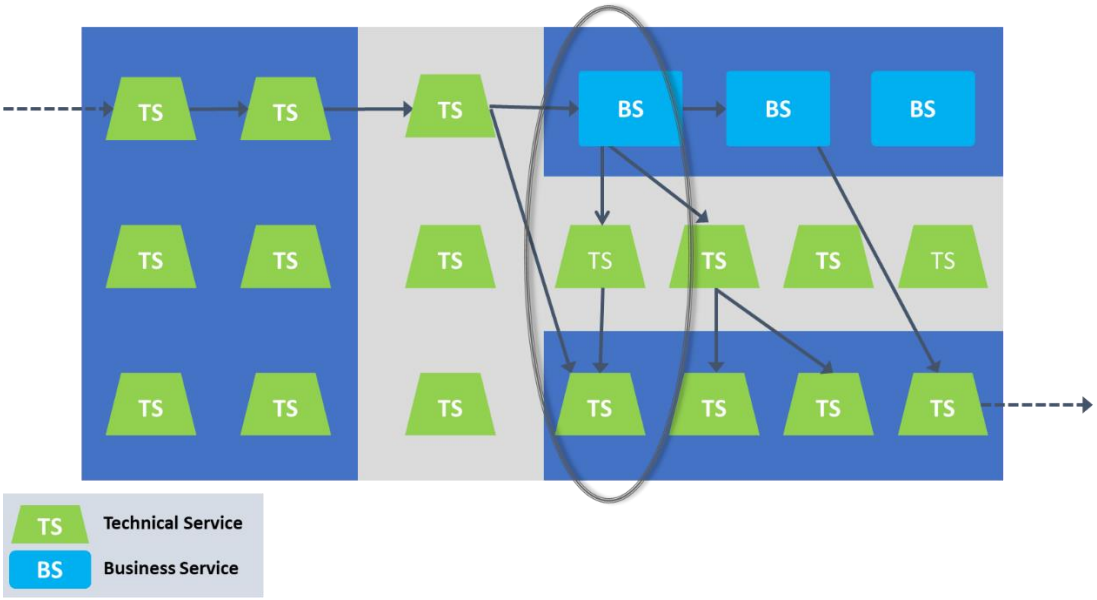

Exhibit 4-8: SOA Composite Service

The Federal Chief Information Officers Council's document A Practical Guide to Federal Service Oriented Architecture, contains additional relevant SOA-specific information. Many sections of the document are directly applicable to the MITA initiative. This document provides the following definition of a Service-Oriented Architecture:

> "Service-Oriented Architecture (SOA) is the body of standard design and engineering processes, tools, and best practices that leverage the modularity and composability of services to support business objectives. SOA deepens and extends the Enterprise Architecture and defines the implementation of the architecture in terms of its technical approach."

Exhibit 4-9: Federal SOA Framework illustrates the necessary gathering of interdependent aspects to achieve a primary MITA objective of creating a target Service Oriented Enterprise.

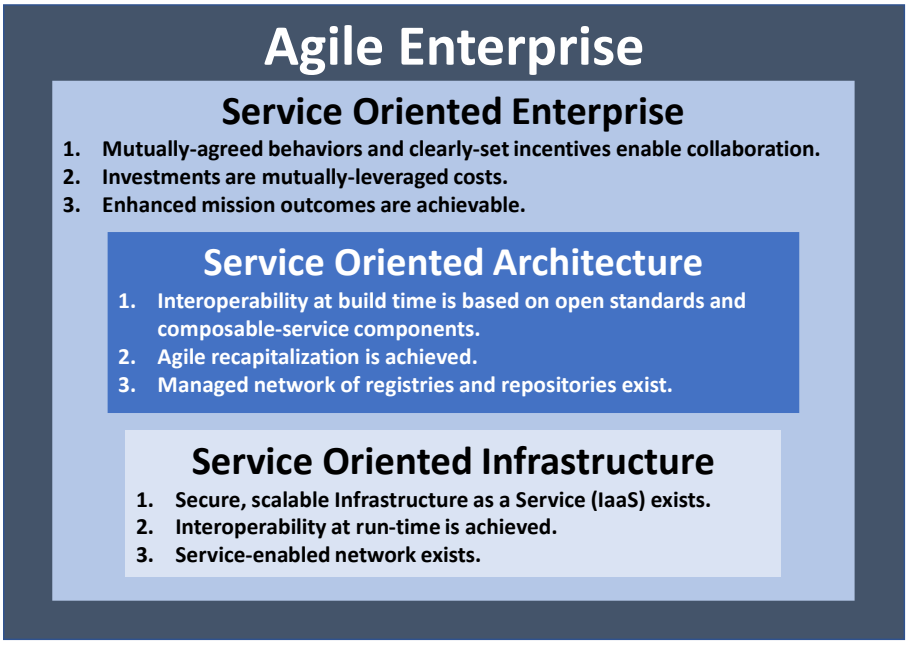

Exhibit 4-9: Federal SOA Framework

4.2.6 TECHNICAL COMPONENTS

Technical Components are small-grained functions with no business significance. They assist with processing functions like common validation, common formatting, and standard database responses. Any block of non-business code that repeats inside business code is a potential candidate as a Technical Component addition. All purchased technical functions must include a Technical Component wrapper. This wrapper should generalize the interface to minimize the impact if that purchased Technical Component is replaced in the future.

Versioning is a critical function of Technical Components. Many of these Technical Components will be reused hundreds, if not thousands of times. Any future need to modify or replace this object could make a huge impact. Versioning is required to minimize the impact and allows different groups operating on different schedules to make changes.

4.3 TECHNICAL SERVICE DEVELOPMENT

Technical service development is the high-level process used to identify, document, and approve a technical service that a FX Project Owner discovered. **Exhibit 4-10: Technical Service Creation Workflow Process** illustrates the steps required to create a technical service. The process to develop a technical service is like the process to develop a business service.

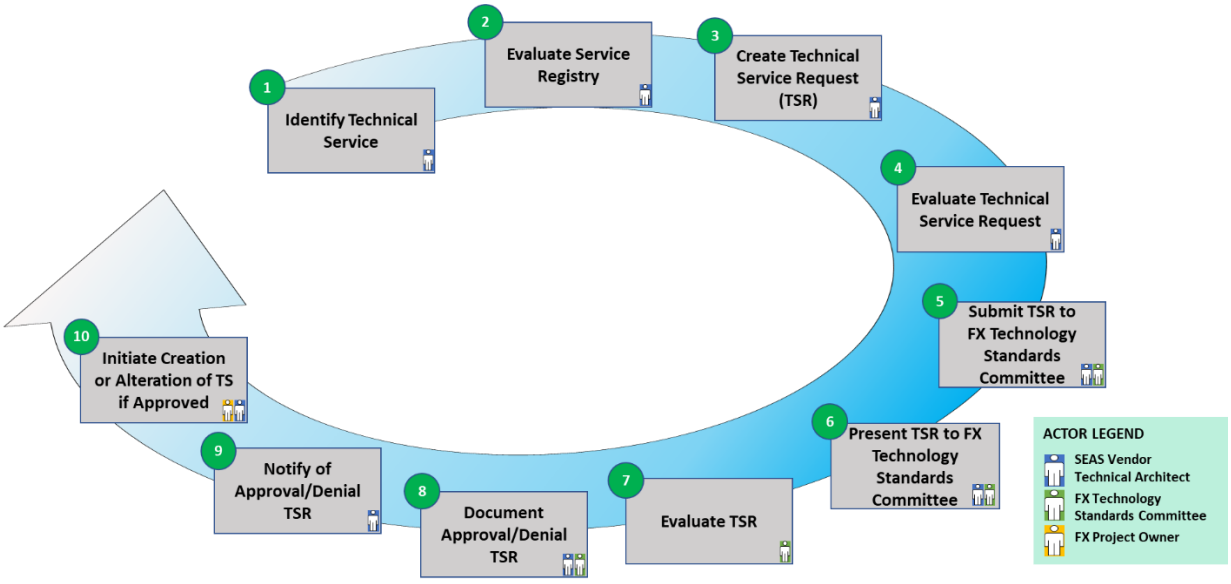

Exhibit 4-10: Technical Service Creation Workflow Process

Follow the steps below to create a technical service:

1. **Identify the Technical Service** – The SEAS Vendor or FX Project Owner will identify the technical service for development using the Project Level Technical Capability Matrix (PLTCM).
2. **Evaluate the Service Registry** – The SEAS Vendor will use the Application Service Registry to determine whether the technical service exists. If the SEAS Vendor does not locate one, the SEAS Vendor will determine if a free or purchasable technical service is available. Based on its findings, the SEAS Vendor proposes a course of action and documents those details in the Technical Service Request.
3. **Create Technical Service Request** – The SEAS Vendor begins to complete the Technical Service Request form, adding all the pertinent research details.
4. **Evaluate Technical Service Request** – The SEAS Vendor works with other FX Project Owners to add additional details to the Technical Service Request.
5. **Submit Technical Service Request to FX Technology Standards Committee** – The SEAS Vendor submits the Technical Service Request to the FX Technology Standards Committee.
6. **Present Technical Service Request to FX Technology Standards Committee** – The SEAS Technical Architect presents the Technical Service Request to the FX Technology Standards Committee.
7. **Evaluate Request** – The FX Technology Standards Committee reviews the technical service request and determines the next action. **Note:** Data service requests and business rule services requests are lower-level internal services that should not need governance oversight. The expectation is FX Technology Standards Committee approves most requests to move forward; however, FX Technology Standards

Committee could also deny the request, or request changes and resubmission of the request.

8. **Document Approval/Denial for Technical Service Request** – The SEAS Vendor documents the results of the FX Technology Standards Committee's evaluation, and then the FX Technology Standards Committee approves the write-up.
9. **Notification of Approval/Denial for Technical Service Request** – The SEAS Technical Architect provides official notice to the FX Project Owners and stakeholder organizations indicating the request's approval/denial status via email.
10. **Initiate Creation or Alter Technical Service Request** – If the request is approved, the development or alteration of the technical service can be scheduled with a FX Project Owner via the SEAS Project Coordinator. If necessary, the SEAS Vendor and SEAS Technical Architect returns to step 1 to modify the request for future review. If the denial is permanent, the SEAS Technical Architect will mark it as such and document the reason(s) for the denial.

The first step in developing a technical service is determining if the service already exists. The FX Project Owner will use the Application Service Registry to determine whether the technical service exists. If the FX Project Owner does not locate one, the vendor will determine if a free or purchasable technical service is available. Only after the vendor exhausts all options will a new technical service be created. Some organizations that might be able to provide technical services include the National Medicaid Electronic Data Interchange (EDI) Healthcare (NMEH) workgroup, and CMS.

- If a technical service does exist, the system designer looks at the associated metadata to determine whether to adapt or extend the service.
- If a technical service does not exist that can be used, then the stakeholder creates the technical service and Technical Service Definition Package (TSDP).

The process to create a TSDP is like creating a Business Service Definition Package (BSDP). The TSDP development activities are below:

- **Technical Service Name Development** – register the technical service in the Application Service Registry.
- **Service Contract Development** – This process involves the following elements:
  - Document the purpose as a short description in one or two sentences of what the service does, and where it exists in the business service requirements, technical area principles, and the PLTCM.
  - The technical service developer designs and develops the formal interface definition, using the required triggers, results, and LDM data model, to document the interfaces and operations used by the service.

A Solution Set is a combination of resources that documents or describes a specific implementation of a technical service. Solution sets map to technical services to assist the Agency with implementation planning.

4.4 TECHNICAL SERVICES REGISTRY

A Technical Services Registry is a specialized service registry or segmentation of a service registry in Service Oriented Architecture that contains information that defines the design, development, and use of technical services. Most architects and developers refer to them as Services Registries in Service Oriented Architecture because they store both business and technical services. Section 3.4 *Business Services Registry* explains that there will be one registry implemented by the IS/IP Vendor and an ASR to track all services.

4.5 TECHNICAL SERVICE PARTS

This section defines the individual parts of a technical service. In the MITA Framework, a technical service contains the same metadata as a business service, except for the service definition package.

4.5.1 TECHNICAL SERVICE DEFINITION PACKAGE (TSDP)

A complete Technical Service Definition Package (TSDP) contains the following parts:

- **Service Name** – This is the title of the service component used by the Technical Service Definition Package (TSDP), which is a MITA-defined set of metadata describing the service. The service contract information describes the interface's expected behavior and the service's security and privacy constraints. Examples of interface-behavior patterns include:
  - A one-way interface only receives or outputs data, often referred to as *fire-and-forget*. This is simply another name for one-way messages (a message-exchange pattern where a service sends a message without expecting a response). *Fire-and-forget* provides the loosest of all possible couplings; for example, regarding email messages, the Sender need not know anything about the Recipient, including how many email messages were sent, or what happens with the email messages. Achieving this one-way interaction must be performed via some form of guaranteed delivery.
  - A two-way interface receives and sends data. This type of two-way traffic has two other attributes:
    - The originator defines who/what initiates the interface, which is either the service (e.g., a request for information) or the outside client (e.g., an inquiry),
    - The processing characteristic defines the relationship between the input and the output, such as the following:
      - **Point-of-Sale (POS) Transaction** – A real-time transaction (e.g., a pharmacy POS) that features a much-constrained response time and provides high reliability.
      - **Online Transaction** – An example of an online transaction is an inquiry about a provider; this transaction type features a more-

relaxed response time, while still allowing for conversation-type human interaction.

- **Batch** – Typical batch-processing constraints exist.
- **Asynchronous** – No constraints are placed on processing times, but asynchronous transactions require response and data coordination.

- **Purpose** – The technical area's principles and PLTCM depict the objective.
- **Business Logic** – Initially, the logic is free-form text or template driven. Application developers will eventually define and code logic to incorporate it into orchestration steps and standardized business rule definitions. Business rule services ensure the rules engines are used to separate technical business rules from core programming and should provide information about the change control process that manages the development and implementation of these technical business rules. Service owners should schedule changes to technical business rules on a regular period and on an emergency basis.
- **Constraints** – Lists the limitations of the service (e.g., loose authentication control).
- **Formal Interface Definition** – Explains the services triggers and results. It also documents the interfaces and operations the service uses in a Web Service Definition Language (WSDL).
- **Use Case** – A use case provides a list of actions or event steps that defines the interactions between a role and a system used to achieve a goal. The system architect or designer documents the use case in free-form text and supports the text with the OMG's Unified Modeling Language (UML) Model before developing the technical services.
- **Solution Set** – Maps the solution set(s) currently available to execute this service. Solutions sets are previously defined groupings of technical services that are operational.
- **Structure and Activity Diagrams** – These diagrams graphically depict the business logic performed by the service and interconnects the solution set(s). The system architect or designer models the diagram in OMG's Business Process Model and Notation (BPMN) application. Using a modeling tool that is based on UML notation produces an UML diagram as part of the solution set(s).
- **Performance Standards** – Define the service's anticipated performance standards to ensure all stakeholders measure the same metrics in the same manner.
- **Test Scenarios and Test Cases** – Validate documentation of service-contract compliance.
- **Maps to MITA Data Models** – Maps a data trace used by the technical service to the Logical Data Model (LDM). Generally, technical services use non-business specific data, such as routing and format data. This is an enhancement of the mapping done for the business service. The XML schema defines the data in motion. American National Standards Institute (ANSI) Accredited Standards Committee (ASC) X12 positional

transactions and World Wide Web Consortium (W3C) XML Schemas are examples of defined data in motion.

4.6 TECHNICAL SERVICE EXAMPLE

The following example is an extension of the Business Service Request example found in **Exhibit 3-9: Add Provider from Enrollment Business Service Structure Chart Analyzed**. During the process of analyzing the Business Service Request form, a new technical service, Verify and Cleanse Address was identified.

The path below leads to an example of the Technical Service Request Form filled out for the Verify and Cleanse Address technical service. Ultimately, this information will reside in the ASR, which is discussed in Section 3.4.1 – *Application Service Registry* of this document. There are more examples in the ASR of other technical services.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment D Technical Service Request Form Example

4.7 DATA SERVICES

Data services are another form of technical service, which acts as a data access layer as defined by MITA 3.0. They are wrappers for one or more Data Components, which business services or other data services call. Data services provide a standard set of methods but can also contain custom methods. Constructing data services uses a standard set of rules and pattern templates.

Composite data services represent a grouping of larger-grained data services, comprised from a collection of finer-grained data services or complex access to multiple Data Components. The expectation of reuse of composite data services is not as high and thus, they should only be created when a specific need is determined.

The data service's code includes all the necessary functionality to maintain an Entity, which includes all foreign-key validation and all business rules. This requires a standard set of methods, which includes the following:

- **Read** – This method will determine which of the other Read methods below to use based on the data provided.
- **ReadByID** – Read the Entity by Random ID and its related Foreign Key Entities.
- **ReadByKey** – Read the Entity by Key(s) and its related Foreign Key Entities.
- **Add** – Insert the Entity after calling **ValidateAdd** with no critical errors.
- **TermAndAdd** – This method is only available for effective-dated entities. The method will terminate the currently effective row and add a new row with an open-ended Termination Date. Insert the Entity after calling **ValidateAdd** with no critical errors.
- **Update** – Update the Entity after allowing for partial data provided and calling **ValidateUpdate**.

- **Delete** – Delete the Entity after calling **ValidateDelete** with no critical errors.
- **ValidateAdd** – Validate that the data service add scope contains no critical errors and report any errors. The validation scope within the data service excludes business rules. This scope is primarily required fields, valid values for fields, and that all Foreign Keys values are valid. This method provides data in the standard error record to support use in UI level processing.
- **ValidateUpdate** – Validate that the data service update scope contains no critical errors and report any errors. The validation scope within the data service excludes business rules. This scope is primarily required fields, valid values for fields, and that all Foreign Keys values are valid. This method provides data in the standard error record to support use in UI level processing.
- **ValidateDelete** – Validate that the data service delete scope contains no critical errors and report any errors. The validation scope within the data service excludes business rules. This scope is primarily to prevent any row's removal that is a foreign key value in another table. This method provides data in the standard error record to support use in UI level processing.

The goal is to create predictable data services that can be consumed with little, or no research required. Documentation of a data service beyond its methods and properties is limited to error occurrences.

Data services are passed a database transaction, and rollback or commit activity occurs at a higher-level.

Some data service methods will be constructed in advance, but composite data services (those that act on more than one Entity), should be constructed only when a business need is identified. Only some methods should be automatically constructed for a data service. Historically, the "build it and they will come" method has been used to determine which methods to always construct, but experience has proven this approach expensive. Conversely, the "build it when they come" method creates too many delays. A compromise approach appears to be the "build it if it is likely they will come in the next 12 months" method. This decision will be made in conjunction with the FX Technology Standards Committee review.

4.7.1 Data Services Benefits

A data service defines a standard interface and data access functionality and maintenance required, which aligns the shared factors of the Agency and development with FX projects. A data service allows the following features:

- **Modularity** – Data services decouple the data access layer from the business service layer. Additionally, the data service layer creates a building block with a high potential for reuse. Composite data services are higher-level building blocks that group data services at a larger-grained level. When reuse of these higher-level blocks is expected, then the creation of them is cost effective. Detailed design should expose the required data services.

- **Interoperability** – Many business services and other data services will use many data services. As new business services arise, they will be able to consume an existing data service without any change(s) made to that data service.

4.7.2 DATA SERVICES APPROACH

The objective of data services is to provide an independent version of the data-maintenance process. These mostly predictable services create the foundation for the assembly of business services. Data services contain the following architectural characteristics:

- Loosely coupled data services from the actual data layer that SQL calls
- Data services have no predefined predecessor or successor data services
- Mandatory interface compatibility needs to exist between the data services

4.7.2.1 DATA SERVICES FLOW

Exhibit 4-11: Data Services Flow illustrates the different components and flow of a data service.

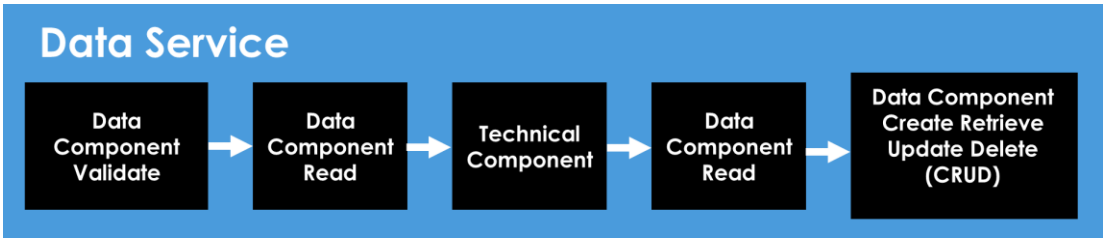

Exhibit 4-11: Data Services Flow

4.7.2.2 DATA SERVICES CONSTRUCTION

Data services commonly evolve from the LDM and the PDM. Supported methods are determined by known-usage requirements and expected near-future environments. The development process should be very mechanical because doing so supports the predictability aspect of this effort.

4.7.2.3 DATA COMPONENTS

Data Components are where the connections to the PDM exist. Inbound and outbound data uses the LDM format, whereas the internal Data Component code uses the PDM format; doing so provides the loose coupling necessary to separate the application from the PDM.

Data Components are exclusively *Create Read Update Delete* (CRUD) actions; however, exceptions exist for PDM entities from where their attributes are derived. This might necessitate special methods other than the CRUD process. The CRUD process must support the **READ by** Key and by ID. For entities that use Effective and Terminate dates, the methods need to support

something, such as **READ EFF**, where the date supplied to the read must be within the Effective and Terminate date range. Shortcut methods may be required for the effective-dated entities that terminate the previous row before adding a new row.

4.7.3 DATA SERVICES DEVELOPMENT

Data service development is the high-level process used to identify, document, and approve a data service that the FX Project Owner identifies during project design. Data services are identified from the bottom up using the LDM and PDM**.**

Exhibit 4-12: Data Service Creation Workflow Process illustrates the steps required to create a data service.

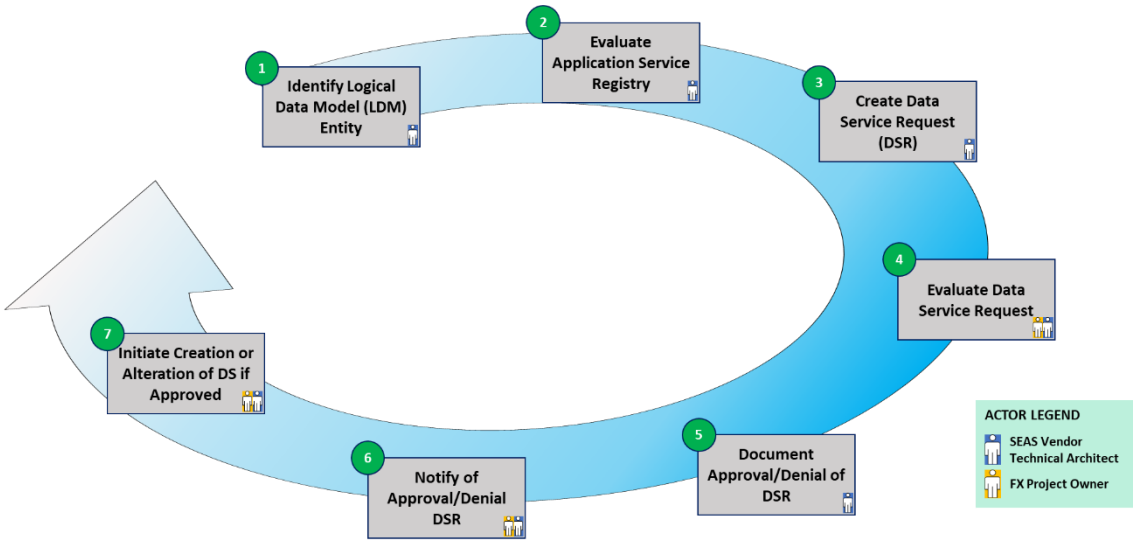

Exhibit 4-12: Data Service Creation Workflow Process

Use the following steps to create a data service:

1. **Identify Logical Data Model (LDM) Entity** – The SEAS Vendor identifies an LDM Entity to have the data service created.
2. **Evaluate the Application Service Registry** – Based on findings, the SEAS Vendor proposes a course of action and documents it in the Data Service Request form.
3. **Create Data Service Request** – The SEAS Vendor completes the Data Service Request form, adding all the research details.
4. **Evaluate Data Service Request** – The SEAS Vendor works with the FX Project Owners and stakeholder organizations to confirm further details and approve/deny the Data Service Request.
5. **Document Approval / Denial of Data Service Request** – The SEAS Vendor Technical Architect documents the approval or denial of the Data Service Request.

6. **Notification of Approval/Denial for Data Service Request** – The SEAS Vendor Technical Architect provides official notice to the FX Project Owner and stakeholder organization indicating whether the request is approved or denied.
7. **Alter Data Service Request** – If the request is approved, the development or alteration of the data service can be scheduled with a FX Project Owner via the SEAS Project Coordinator. If necessary, the SEAS Vendor returns to step 1 to modify the request for future review. If the denial is permanent, the SEAS Technical Architect will mark it as such and document the reason(s) for the denial.

Due to the nature of reusing data services, the development order of the data services can be critical. Further discussion about this issue can be found in Section 5.3.3 – *Software Factory*. At a high-level, data services for Foreign Key Entities must be completed first. To maximize development efficiency, Lower-level data services need to be completely developed and tested before constructing the higher-level data services. Without proper planning and tracking, numerous restarts of development efforts will slow down development efforts.

Modular-reusable code adds the requirement to track and manage code dependencies. Reworking code adds time and complicates determining when the task is completed. When performed correctly, modular reuse enables high levels of efficiencies. Each data service needs to be delivered on time and be fully tested with predictable methods and error handling.

4.7.4 DATA SERVICES PARTS

This section defines the individual parts of a technical data service. In the MITA Framework, a technical data service contains the same metadata as a business service, except for the service definition package.

4.7.4.1 DATA SERVICES DEFINITION PACKAGE (DSDP)

A complete DSDP contains the following parts:

- **Service Name Development** – This name should match the LDM Entity process name (Provider, Provider Composite, etc.).
- **Configuration Data** – Configuration data includes the following:
  - **Framework** – Defines the FX LDM.
  - **Version** – Provides the release or development number of the configuration data.
  - **Plan Development Start Date** – The day development is planned to begin.
- **Service Contract Development** – Involves the following elements:
  - Purpose is a short one- to two-sentence description of the data service's function(s).
  - Documentation of the data service's provided functionality and capability.
  - List of any data service constraints.
  - Identification of common data service methods.

- **Business Logic** – Describes the logic performed by the data service that is beyond the standard methods.
- **Performance Standards** – Provides an estimate of system load by method.
- **Test Scenarios, Data, and Cases** – These are used to validate data services functionality.
- **Map to FX Data Models** – Data traces used by the service to the FX LDM.

4.7.5 DATA SERVICE EXAMPLE

The following example is an extension of the Business Service Request example found in **Exhibit 3-9: Add Provider from Enrollment Business Service Structure Chart Analyzed.** During the process of analyzing the Business Service Request form, a new data service, Address Type was identified.

The path below leads to an example of the Data Service Request Form filled out for the Data Service Address Type. Ultimately, this information will reside in the ASR discussed in Section 3.4.1 – *Application Service Registry* of this document. There are more examples in the ASR of other technical services.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment B Data Service Request Form Example

4.8 BUSINESS RULE SERVICES

A business rule service is another type of technical service designed to provide a single validation point in a loosely coupled format. Business rule services automate and optimize decision making processing. Business rule services execute business rules, perform data validation and constraint checking, make business decisions and make human task routing decisions. Business rule services are frequently invoked in workflow processes for decision making steps. Business rule services can also be invoked in composite applications. Business rule services frequently use a business rules engine allowing specialized business rule processing including decision tree processing, decision matrices, dynamic access to supplemental data used in decision making, processing natural language rules logic, and providing user consumable documentation of the basis for decisions.

Exhibit 4-13: Business Rule Service Interaction shows their interaction with system services and components. Historically, business rules logic exists throughout the Data Access layer, Service layer, and User Interface layer. Business rules logic was often repeated and frequently conflicts either directly, or as they are related to the perspective of the layer.

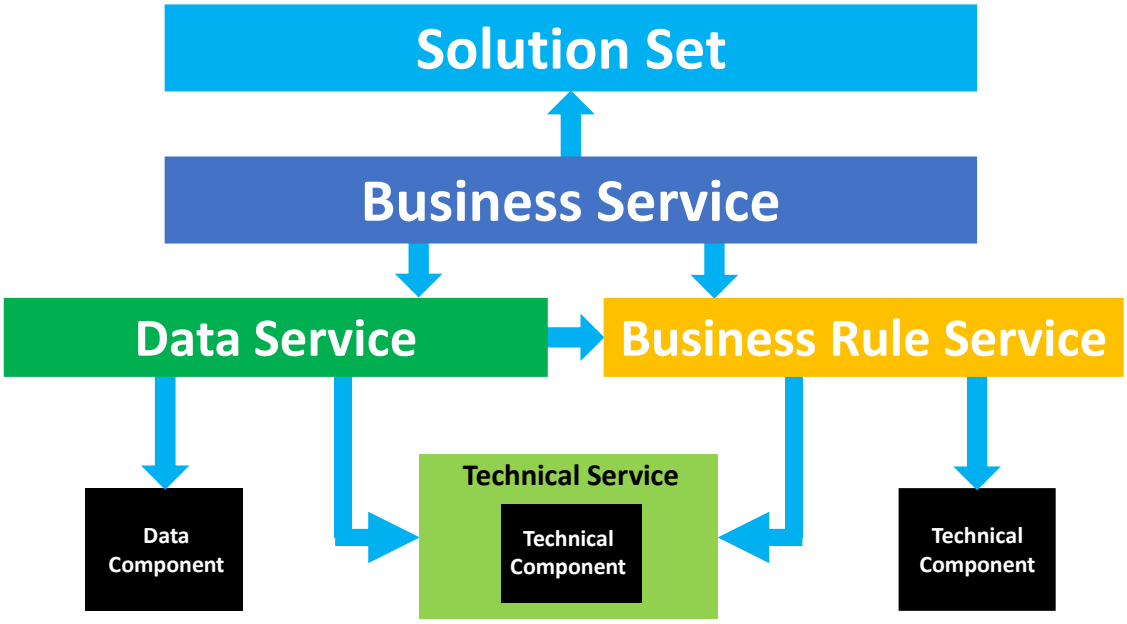

Exhibit 4-13: Business Rule Service Interaction

Two situations have led to this conflict problem. First, the inline nature of the existing code structures prevent reuse. Data structures in the Data Access, Service, and User Interface (UI) layers are all different. Data structures in the UI are most often tied to the details of the UI, making it even more tightly coupled. Second, the skill sets used in the different layers lead to separate coding styles and techniques. UI fields are usually supplied as text and their domains must be converted, whereas other layers are domain specific. In this approach, the UI layer will still need to translate to the domain; however, the actual business rule logic can be written against a domain-specific value. An example of this is that a web page's text box must be converted to a number, but the Business Rules Engine must validate only if the number is within the acceptable range, and not necessitate doing both.

The rise of using Business Rules Engines has brought about a few changes that can greatly improve this process. First, Business Rules Engines have decoupled the inbound interface by using an XML blob to pass the data to the Business Rules Engine. Second, by performing something similar with the outbound interface, the Business Rules are almost completely decoupled. The development of a strong and standard error-process interface for communicating errors discovered by the Business Rules Engine allows the various layers to adapt and retrieve the information that they need for their processing, without interfering with the data needed for the other layers.

The biggest advantage is that the Business Rules promotion and testing can be implemented independent of the other application code. Proper versioning can also provide additional layers of protection. Decoupling allows for single coding and reuse of almost, if not all, Business Rules.

This does not occur without some new opportunities; primarily, this is a new approach. With anything new requires a learning curve on how best to organize and structure the code within the business rule service. The good news is that most of business-rule coding is normally simple. Although the volume of complex business rules is not excessively high on a percentage level, the complex rule group's importance can be paramount. One should assume that there would be some extra costs associated with these complex rules and will require learning the events to uncover any weaknesses. As the FX evolves, new approaches will emerge.

The actual construction of the business rule service's internals is not pre-determined. Business Rules, in their simplest form, are simply a list of validations, some of which are nested, and some that are not. Some Business Rules can be coded in the IF/ELSE format. Some Business Rules require extensive calculations, whereas others require more complex data gathering. Because of the introduction of Business Rules Engines and the use of naming standards, the possibility exists that the business rule service code can become dynamic by retrieving a list of Business Rules Engine rules, which are based on a naming standard. Adding rules to the Business Rules Engine could change the list's rules that are dynamically retrieved. This could have a huge impact on *pulling* rules dynamically, when their production impact is not what was expected.

4.8.1 Business Rule Services Benefits

A business rule service defines a standard interface and functionality for data validation in the context of the LDM. Two main benefits are:

- **Modularity** – The business rule service is decoupled from both the Data Access Layer and the Data Service Layer. Additionally, the Business Rule Service Layer creates a building block with high reuse potential.
- **Interoperability** – Many business services will use many business rule services. As new business services develop, they will be able to consume an existing business rule service without necessitating any changes to that business rule service.

4.8.2 Business Rule Service Approach

The objective of business rule services is to provide an independent version of the validation inspections associated with the addition and maintenance of data. Business rule services contain the following architectural characteristics:

- They are loosely coupled from the actual data layer that SQL calls
- No predefined predecessor or successor business rule services exist
- Mandatory interface compatibility exists between the business rule services

4.8.2.1 Business Rule Services Flow

Exhibit 4-14: Business Rule Services Flow illustrates the flow from the business service to the data service to the business rule service, which then flows to the technical service, data component(s), and technical component(s).

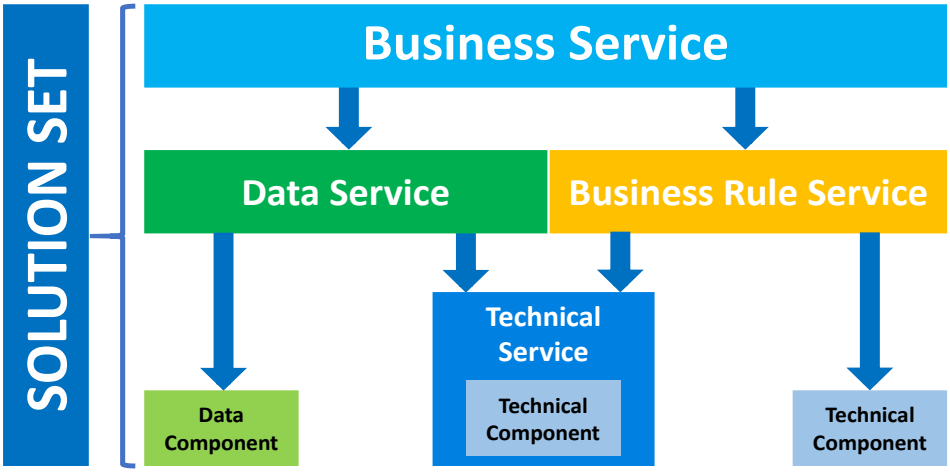

Exhibit 4-14: Business Rule Services Flow

4.8.2.2 BUSINESS RULE SERVICES CONSTRUCTION

Business rule validation ranges from simple, required, or optional assessments, value ranges, and permitted values, to complex calculations and data verifications. The specific internal construction of business rules code cannot be specified. To some extent, this could be referred to as *an art*, meaning that skill is required but learning the skill is by using instinct or experience, rather than learning by facts or rules. Using a general approach to this problem and performing peer reviews will likely result in the best-constructed structures over time.

The primary objective of Business Rule validation is to identify as many known issues in a single pass, without complicating any of these problems with consequential compound issues. An example of this is a simple error occurring, such as when an age is not provided in a required field, resulting in another error indicating that the age must be in a range. Adding the second error provides no additional value and could complicate the error analysis with unnecessary volume. This becomes a judgement call when the real-world examples involve multiple fields. The key point is to ensure some form of exit strategy is created, while progressing through the various rules.

The secondary objective of Business Rule validation is to attempt to make the rules processing dynamic, especially when regarding the Business Rules Engine's rules.

Following is a simple example of this approach, which by doing so, leverages the *Change by Addition and Deletion* principle. For this example, the business rule service is for *Member*. Assume there is a naming standard to retrieve all the business rules applicable to Member <*1.**> where * indicates the release version.

Example:

Initial Rules Engine Rules (RER) (Format is <*Type Version*>. Release is <*BRS 1.1*>.)

1. BRS-RER – Member 1.1 – (Age >10 <21)
2. New Rules
3. BRS-RER - Member 1.2 – (Gender M Age >8 <26)
4. BRS-RER - Member 1.2 – (Gender F Age >10 <21)
5. Deleted Rules
6. BRS-RER - Member 1.1 – (Age >10 <21)

The most important aspect to this approach is the ability to change logic and decision-making processing through configuration without needing to modify programming language code, rebuild executable logic, and redeploy executable modules. This is particularly applicable when regarding high-profile activities, such as outward-facing and high-volume systems (e.g., Claims and Provider Enrollment). The rules service changes would still have to be tested and deployed but without the detailed programming language complexities.

Without the ability to envision the look of the final product, the only way to work through the complexity of this problem is to create a structure chart/decision tree. Many of the Business Rules Engine authoring tools support this approach, and with training and experience, one will discover a successful process.

4.8.3 BUSINESS RULE SERVICES DEVELOPMENT

Business Rule Service development is the high-level process used to identify, document, and approve a business rule service that the FX Project Owner identifies during project design.

Exhibit 4-15: Business Rule Service Creation Workflow Process illustrates the steps required to create a business rule service.

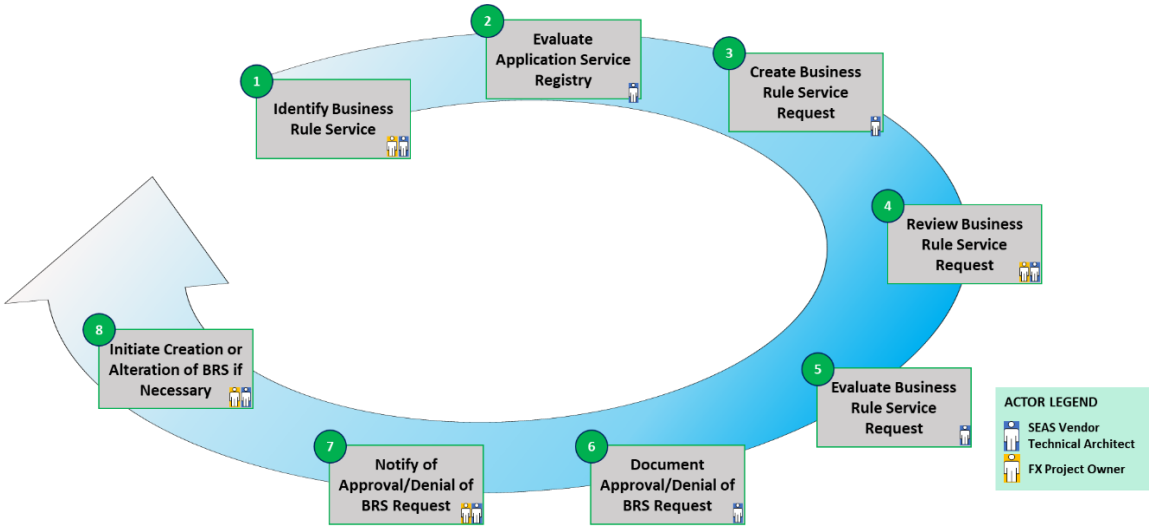

Exhibit 4-15: Business Rule Service Creation Workflow Process

Follow the steps below to create a business rule service:

1. **Identify Business Rule Service** – The first step in developing a business rule service is determining the service's scope and target. Two general possibilities exist; first, the business rule service could target the requirements of a single LDM entity. The second possibility is that the business rule service's scope could be the collection of logically grouped LDM entities.
2. **Evaluate the Application Service Registry** – Based on findings, the SEAS Vendor proposes a course of action and documents that information in the Business Rule Service Request Form.
3. **Create Business Rule Service Request** – The SEAS Vendor completes the Business Rule Service Request form, adds all the research details.
4. **Review Business Rule Service Request** – The SEAS Vendor works with FX Project Owners and stakeholder organizations to add additional details to the request.
5. **Evaluate Request** – The SEAS Technical Architect reviews the business rule service (BRS) request and approves or denies the request. The SEAS Technical Architect may seek approval input from the Agency Technical Architect or SMEs depending on complexity of the BRS.
6. **Document Approval / Denial for Business Rule Service Request** – The SEAS Vendor Technical Architect documents the approval or denial of the BRS Request.
7. **Notification of Approval/Denial for Business Rule Service Request** – The SEAS Technical Architect provides official notice to the FX Project Owners and stakeholder organizations communicating request approval or denial.
8. **Alter Business Rule Service Request** – If the request is approved, the development or alteration of the BRS can be scheduled with a FX Project Owner via the SEAS Project Coordinator. If necessary, the SEAS Vendor returns to Step 1 to modify the request for future review. If the denial is permanent, the SEAS Technical Architect will mark it as such and document the reason(s) for the denial.

4.8.4 Business Rule Services Parts

This section defines the individual parts of a BRS. In the MITA framework, a business rule service contains the same metadata as a business service, except for the service definition package.

4.8.4.1 Business Rule Services Definition Package (BRSDP)

A complete Business Rule Services Definition Package (BRSDP) contains the following parts:

- **Business Rule Service Name** – The BRS name should match the Business Process name (e.g., verb or noun) to avoid confusion.
- **Configuration Data** – Configuration data includes the following:
  - **Framework** – The framework defines the FX Logical Data Model (LDM).

- **Version** – Indicates the configuration data's release or development number.
- **Plan Date** – The day of the planned release.

- **Business Rule Service Contract Development** – BRS contract development involves the following elements:
  - Purpose is a short one- to two-sentence description of the business rule service's functionality, which is derived from the LDM.
  - Documentation of the business rule service's provided functionality and capability.
  - A list of any service constraints.
- **Formal Interface Definition** – Definition of the formal interface requires using the FX LDM and documents the service's interfaces.
- **Business Logic** – Business logic describes the soundness performed by the service and the behavior of the non-transparent service. Initially, states use free-form text from the Business Process definition; however, the business rules specify business logic as they are incorporated in the enterprise.
- **Performance Standards** – Performance standards derive from the performance metrics specified in the business rule service's definition and are based on the Agency or vendor-specific solution set.
- **Test Scenarios, Data, and Test Cases** – These testing components validate service-contract compliance and are based on the Agency or vendor-specific solution set.
- **Map to FX Data Models** – Data traces used by the service to the FX LDM.

4.8.5 Business Rule Service Example

The following example is an extension of the Business Service Request example found in **Exhibit 3-9: Add Provider from Enrollment Business Service Structure Chart Analyzed.** During the process of analyzing the Business Service Request form, a new business rule service, Provider Enrollment Rules was identified.

The path below leads to an example of the Business Rule Service Request Form filled out for the new service. Ultimately, this information will reside in the ASR discussed in Section 3.4.1 – *Application Service Registry* of this deliverable. There are more examples in the ASR of other technical services.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment C Business Rules Service Request Form Example

4.9 Technical Service Strategic Topics

This section presents the FX strategic direction on multiple topics related to technical services. Each topic considers a spectrum of alternatives over the dimension of time.

The topics in this section primarily provide guidance to FX Project Owners about the planned use to technical services and whether services are defined, managed, and used at the

application, division, agency, or cross-agency levels. For most technical capabilities, the direction is to establish Agency level reusable technical services.

The primary exception is workflow and work management where the strategic direction is to allow different implementation but enforce use of BPMN standards to enable cross-application interoperability of workflow processing.

Strategic Topic 4-1: Correspondence Generator describes the Agency strategy for systems use of correspondence generator processing solutions.

CORRESPONDENCE GENERATOR	CURRENT	TIMELINE 2018	2021	2022	2025
Module / System specific implementation of vendor's preference	X	->	Exception only for New Systems		
Module / System implementation of Agency specified of COTS product or solution					
Reusable FX Provided Service			Required for New Projects	Begin migration from system specific solutions	Complete migration from system specific solutions
Reusable Cross-Agency Service				Pilot as Optional for New Projects	

ANALYSIS

The benefits of modules using a common correspondence generation service are primarily consistency of presentation format, style, and layout. A common service can reduce complexity for vendors and the costs of duplicated infrastructure and software.

If the correspondence generation and correspondence release are combined, the use of a common service could also provide synergies such as consolidation of content from multiple sources into an aggregated communication. In this model, system providers would provide snippets of content to the correspondence generation service and the communication service would manage the assembly and dissemination. Intelligence in the correspondence generation and correspondence release can also perform intelligent release of content to reduce support spikes.

The drawback of using a common correspondence generation service is that some COTS products and module vendors may produce proprietary or module specific correspondence formats that are optimized for their module/functional area. These formats may not be supported by a common service.

FMMIS has an existing Letter Generator that performs some of these functions. At a minimum, it needs to be generalized and more methods added to be a universal technical service.

Typically, the correspondence generation service would not store, archive, and deliver correspondence. These functions would be a responsibility of the correspondence release service.

Strategic Topic 4-1: Correspondence Generator

Strategic Topic 4-2: Document Management Upload and Storage describes the Agency strategy for systems use of a Document and Management and Storage processing service.

Document Management Upload and Storage	Current	2018	Timeline 2020	2022	2025
Module / System specific implementation of vendor's preference	None				
COTs created Documents		Required (2019)	->		
FX Projects				Required for New Projects	->
Reusable Agency Provided Service				Required All New Projects Begin Migration of Existing Systems	->
Reusable Cross-Agency Service				Pilot as Optional for New Projects	

Analysis

Trying to create a 365 view of an entity's correspondence is difficult in the current Agency enterprise structure. Centralizing the correspondence upload and storage method will make it easier for the creation of a 365 view of an entity's correspondence. The EDW Vendor has implemented DocuEdge as the enterprise content management tool.

Strategic Topic 4-2: Document Management Upload and Storage

Strategic Topic 4-3: Document Management Image Display describes the Agency strategy for systems use of a common Document and Management Image Display processing service.

Document Management Image Display	Current	2018	Timeline 2020	2022	2025
Module / System specific implementation of vendor's preference			Optional with approval		

Document Management Image Display	Current	2018	Timeline 2020	2022	2025
Module / System implementation of Agency specified COTS product or solution			Optional with approval		
Reusable Service Use – FX Projects				Required for all New Projects	
Reusable Service Use – Agency				Required for all New Projects	
Reusable Service Use – Cross-Agency					Pilot as Optional for New Projects

Analysis

All aspects of the enterprise need to display documents sent to and received by various entities. Centralizing and using a common image display service reduces the cost of such services.

Strategic Topic 4-3: Document Management Image Display

Strategic Topic 4-4: Address Normalization describes the Agency strategy for systems use of an Address Normalization service.

Address Normalization	Current	2018	Timeline 2021	2022	2025
Module / System specific implementation of vendor's preference		Optional with approval (2019)			
Module / System implementation of Agency specified COTS product or solution		Optional with approval (2019)			
Reusable FX Provided Service		Optional (2019)			
Reusable Service Use - FX Projects			Required for all New Projects		
Reusable Service Use – Agency				Required for all New Projects	
Reusable Service Use – Cross-Agency					Pilot as Optional for New Projects

Analysis

Address Normalization	Current	2018	Timeline 2021	2022	2025
A common service can reduce complexity for vendors and the costs of duplicated infrastructure and software. Currently FMMIS uses a COTS product service to cleanse incoming addresses.					

Strategic Topic 4-4: Address Normalization

Strategic Topic 4-5: Report Storage and Retrieval describes the Agency direction for systems use of a Report Storage and Retrieval service.

Report Storage and Retrieval	Current	2018	Timeline 2020	2022	2025
Module / System specific implementation of vendor's preference	X			Exception only Begin migration of existing	Module specific systems retired
Module / system provider implementation of Agency specified COTS product,					
Reusable Agency Provided Service Agency-wide shared service Use Mandated - FX				All New or Replacement Systems Projects	
Use Mandated – Agency-wide				All New Projects	
Analysis					

Analysis
A common service can reduce complexity for vendors and the costs of duplicated infrastructure and software. Currently FMMIS uses a COTS product for Report Storage and Retrieval. An enterprise-wide common solution would be applicable across the Agency.

Strategic Topic 4-5: Report Storage and Retrieval

Strategic Topic 4-6: Error Handling describes the Agency strategy for systems use of an Error Handling Framework.

Error Handling	Current	2018	Timeline 2020	2022	2025
Module / System specific implementation of vendor's preference			Required for all ESB Services		

Module / system provider implementation of Agency specified COTS product,			Required for all ESB Services		
Use for existing projects		Optional (2019)	X		
Reusable FX Provided Service			Required All New Projects		
Reusable Agency Provided Service				Required, All New Projects exposed though the IS/IP	
Reusable Cross-Agency Service					Required, All New Projects exposed though the IS/IP

ANALYSIS

Common Error Handling is a part of all ESB services so that the use of these services is simpler. To achieve true modularity in a Service Oriented Architecture a common error handling framework is essential. The FX Application Architecture (Application Architecture) is employing this same Common Error Handling for all components of the Application Architecture. The IS/IP Vendor has implemented the Informatica AddressDoctor tools as the enterprise address validation solution.

Strategic Topic 4-6: Error Handling

Strategic Topic 4-7: Service Versioning describes the Agency strategy for requiring version capabilities in services.

SERVICE VERSIONING	CURRENT	TIMELINE 2018	2020	2022	2025
Service Versioning implementation is Module / System provider preference	X		Required for all services exposed through the IS/IP		
Service provider concurrently supports one version of a service. New service versions always require changes by service consumer (e.g., specify version)					
Service providers concurrently support multiple versions of a service (e.g., beta, multiple releases)			Preferred, especially if there are many service consumers		

SERVICE VERSIONING	CURRENT	2018	TIMELINE 2020	2022	2025
Service consumers have ability to automatically use the latest compatible version of a service			Capability to be available using IS/IP		
Service versions support additions or changes to interface data structures without requiring changes by service consumers except for new mandatory fields			Preferred		
Services support service consumer specific versions of processing				Support as needed driven by business requirement	

ANALYSIS

Historically only a single version of a piece of code exists. When that code is changed, all code affected by the change must change as well. Moving to massive reuse means that the earlier approach does not always work. Versioning supports the concept of changing by adding. Changes to Version 1 are made by copying Version 1 and naming it 1.1. Programs that need the new version are changed to use the new 1.1 code and programs that do not need the changes are left on the old version.

The goal of using services is that they can be shared across the enterprise. This sharing creates problems when the services must be changed. Some changes do not support the use of prior versions, but many do. Versioning means that more than one version of a service is available at the same time. This allows consumers of the service to migrate to new versions on a different schedule. It also means that multiple versions of a service must be supported if someone is using them. Management of the multiple version migration schedules is necessary to ensure that the support costs are minimized. In some cases, service consumers will be required to use newer versions of code.

Strategic Topic 4-7: Service Versioning

Strategic Topic 4-8: Rules Engine Usage describes the Agency strategy for systems use of a rules engine for system decision processing and automation.

RULES ENGINE USAGE	CURRENT	2018	TIMELINE 2020	2022	2025
Module / System specific implementation of vendor's preference	In Rule - FMMIS	->	Exception only for New Systems		
Module / System implementation of Agency specified COTS product or solution					

Rules Engine Usage	Current	Timeline 2018	2020	2022	2025
Reusable Agency Provided Service			Required for New Projects	Begin migration from system specific solutions	Complete migration from system specific solutions
Reusable Cross-Agency Service				Pilot as Optional for New Projects	

Analysis

Vendors and solution providers use different strategies to store and manage business rules and perform rule-based logic in systems. Many implement business rules in custom code. Some systems that implement custom code will parameterize factors and reference data that change frequently to reduce maintenance. There are also many commercial rules engine options that range from simple table-based policy to products that accept natural language (text) of policy and automated processing based on the language.
There are few industry standards about business rules that allow business rule content to be reusable in different rules engine products.

The FX direction is to enable cross-enterprise use of business rules for validating data or making rule-based decisions and calculations. In effect, the Agency is seeking a single source of policy truth so that different systems and different users operate from the same set of business rules. This direction implies standardization, consolidation, and centralization of business rule processing. An alternative solution would be to have processing rules implemented as discrete services that are coordinated by the ESB invoking required services.

There are many synergies in standardizing the location and capabilities for the business rules processing including decision-making consistency, improved data quality, reuse of data validations, reduced testing complexity, etc.

A challenge to getting to a single source of policy truth is that module vendors use a wide variety of business rules management strategies. Many are proprietary (e.g., embedded custom code). Vendors may consider their processing strategy for business rules proprietary and pushback on using an external rules engine service.

COTS products that use their own rules engine solution should be accountable to maintaining the same rules in a reusable enterprise rules service or solution so that if / when a change is required, that vendor does not have an advantage and there is not loss when they leave. All modules will be required to export their module specific business rules to the Enterprise Business Rules Engine provided by the IS/IP Vendor.

The use of a reusable enterprise rules service becomes the single source of truth for business policy and business rules implementation in systems. It is important that the implementation of policy in the enterprise rules service provides transparency to the rule and decision calculation, enable support of questions about rule-based decisions or calculations and testing processes ensure correctness of decisions and calculations from implementation and as business policy and rules are maintained in the enterprise rules service.

Strategic Topic 4-8: Rules Engine Usage

Strategic Topic 4-9: Workflow Standard describes the Agency strategy for systems use of a workflow standard.

Workflow Standard	Current	Timeline 2018	2020	2022	2025
Workflows and business processes within a service or system	Module / System provider preference		Use of BPEL or BPMN preferred		
Workflows or business processes across FX systems or modules	BPEL or BPMN are preferred		BPMN preferred if workflow is an external event		
Workflows or business processes across Agency boundaries		BPMN Preferred for New Projects	->	BPMN Preferred All New Projects Begin Migration of Existing Systems	->

Analysis

There are two industry standards used in workflow tools, Business Process Execution Language (BPEL) and Business Process Model and Notation (BPMN). BPMN is a higher-level tool that will generate the lower level BPEL that is required by the workflow software.

Workflow tools and ESBs are similar but each supports a slightly different situation. ESB allows for the orchestration of fully automated tasks that complete quickly. This allows for the creation of large grained business processes. Workflows support the orchestration of long running tasks that often involve human interaction and software execution. Both products are needed in most enterprises.

Workflow products ultimately use BPEL at the execution layer. Business Process Model and Notation (BPMN) is a visual version of what the BPEL does. BPMN solves the problem of modeling and orchestrating business processes, integrating people and systems. BPMN is a graphical representation for specifying business processes in a business process model. Business Process Management Initiative (BPMI) developed BPMN, which has been maintained by the Object Management Group since the two organizations merged in 2005. Today tools exist that allow for the creation and maintenance of the workflows at the BPMN level and generate the BPEL needed for the execution. BPMN is preferred to bring the definition and maintenance closer to the business user.

Strategic Topic 4-9: Workflow Standard

SECTION 5 APPLICATION ARCHITECTURE

Application Architecture describes how the application's various parts work together to provide a business solution. The integrated solution includes middleware, which provides the ability for application integration of reusable components.

5.1 APPLICATION ARCHITECTURE APPROACH

The FX Application Architecture (Application Architecture) approach is a component of the Technical Architecture (TA) that provides guidance to application architects and developers that build software applications. The FX Application Architecture approach reiterates and extends the guidance of CMS MITA 3.0 Part III Technical Architecture Chapter 5 Application Architecture and direction for applications to use Service-Oriented Architecture (SOA). Layered application architectures that use SOA integrate business services with technical services through a service layer. Using SOA as an integrating framework allows services to be platform and technology independent and interoperable. The FX Application Architecture seeks to provide the structure that connects technology-independent services through a platform-dependent service interface that can operate optimally in any environment deployment. The predominant focus of the FX Application Architecture is on effective application processing as consumers and providers of the types of business, technical, data, and business rule services described in previous sections. This emphasis on services, integration and interoperability in the application architectures will help FX projects build and operate applications that achieve the strategic priorities of the Agency.

This section explains:

- Application Design Principles and Patterns
- Application Architecture
- Application Architecture Key Components
- Security and Privacy
- Services and Infrastructure Interaction

5.2 APPLICATION DESIGN PRINCIPLES AND PATTERNS

FX processing supports large numbers of users and stakeholders with complex business rules and business processes. This processing relies on application software built from diverse components. This software supports large transactional workloads and must process to support the ecosystem of direct and interrelated systems. Applications must also adapt to rapidly changing business demands. Scalability, correctness, stability, and extensibility are the most important concerns in the architecture of applications and the overall enterprise system.

Many challenges exist in creating application software systems that will meet the demands of the Agency. The FX applications require a high degree of quality, reliability, and functionality while performing responsively and cost effectively. Business process and policy change and increasing complexity is occurring at a rapid and growing rate. Supporting the needs of the

Agency, enterprise and overall ecosystem in this environment is a driver for FX applications, modules, and systems to follow sound development principles and design patterns.

The Agency will require FX Project Owners to follow the Application Architecture Design Principles and Patterns as they relate to business services, business rule services, technical services, and data services.

5.2.1 APPLICATION ARCHITECTURE DESIGN PRINCIPLES

Following a set of well-defined application development design principles improves the quality, consistency, and overall cost effectiveness of the FX application environment. The FX Application Architecture design principles are widely used concepts for building a sophisticated system to enable future expansion.

The primary FX application design principles are:

- **Data Normalization** – Because data duplication leads to errors, there is a strong incentive to establish Single Source of Truth (SSOT) entities to achieve the goal that each fact be a single non-decomposable unit, where these facts are independent of all other facts. When a data change occurs, the expectation is only one data location requires modification.
- **Factoring** –This principle is like Data Normalization but refers to application code. Well-planned architectures segment specific code functions or behaviors. At runtime, these appear as separate groups or layers, where each of these layers represents a level of abstraction or domain.
- **Automatic Propagation** – Entails the need to maintain accuracy and consistency through disseminating changes in data or code across a disparate environment. This means that when it is necessary for performance sake, to duplicate data or application code to maintain consistency and correctness, the update of these facts is automatic at construction time.
- **Minimize Functionality** – If an application component exists that meets requirements, reuse that component or service wherever possible. Doing so provides the benefits of needing less code to write, verify, and maintain. It can also reduce memory and runtime resources. To support widespread reuse, an **3.4.1 Application Service Registry** (ASR) will track the inventory of reusable components and services. The Application Service Registry will support search and registration activities. Managing change to reusable components and supporting concurrent use of multiple versions of services reduces the operational complexity of change to services and reusable components. Designers and developers use the ASR to conduct impact analysis across all development and production environments. Identifying both direct impacts and secondary impacts all the way to the business service level simplifies management of shared components reducing the coordination effort. Likewise, the ASR can help track use of previous service and component versions to reduce the overall maintenance efforts.
- **Construct Layers** – To construct an extensible system, the construction process involves using intermediate layers, which can act on the data received from higher layers of the Application Architecture. These intermediate layers act like virtual machine

engines that handle the processing of a specific function in a separate session. This allows data to define the specific functionality, which enables the layered components to be reusable.

5.2.2 TECHNICAL ARCHITECTURE DESIGN PATTERNS

There are two types of Technical Architecture Patterns relevant to the FX Technical Architecture: Application Architecture Patterns and Software Design / Coding Patterns.

5.2.2.1 APPLICATION ARCHITECTURE DESIGN PATTERNS

Application Architecture design patterns are a reusable solution to a commonly occurring problem within software design. The larger the application the more complex the patterns can become. Identifying and leveraging these application architecture design patterns improves quality, productivity, and reliability. Using application architecture design patterns across the enterprise creates momentum and buy-in to identify, use and improve the application architecture design patterns. Architects, developers, and external vendors will use the application architecture design patterns to establish consistency of components.

The FX is most interested in adherence to Industry-standard application architecture design patterns for FX projects. The SEAS Vendor will monitor compliance with application architecture design patterns. Compliance evaluation will span the system development life cycle confirming compliance at design, development, testing and implementation stages. The exact procedures for monitoring compliance of application architecture patterns will be developed with the first FX project that includes processing with industry application architecture patterns. In general, architecture pattern compliance assessment will strive to minimize impact to the FX Project Owner while still confirming compliance.

The first application architecture design patterns developed specific for FX projects will use the integration principles and patterns that result from the IS/IP Project. Integration patterns will standardize usage of the enterprise service bus and other integration components. After implementation of the integration patterns, FX projects will use these patterns to comply with FX Integration standards. The specifics of this procedure will be defined when the IS/IP is installed, and each system or module begins to use IS/IP capabilities.

Examples of application architecture design patterns that are appropriate for FX projects to use include:

- Integration Services / Integration Platform Related
  - ESB Patterns
    - Request / Response
    - Message Exchange Patterns (e.g., one way/fire and forget, fan in, fan out, asynchronous response, …)
    - Publish / Subscribe
  - Master Data Management

- Master Data Services
- Master Data Synchronization
- Master Data Replication

› Application Patterns

- Error Handling
- Job Scheduling
- Data Validation

- Enterprise Data Warehouse Related

› Data Architecture

› Operational Data Store

- Reporting Data Store
- Data Mart
- Data Warehouse Use
- Data discovery

› ETL

- Batch ETL
- Near Real Time ETL

The SEAS Vendor will work with the IS/IP Vendor and other FX Project Owners to define the specifics of these and additional application architecture design patterns for the above and other areas of FX Application Architecture.

Use of application architecture design patterns saves time and reduces risk. Use of application architecture design patterns will help the Agency, SEAS Vendor, IV&V Vendor, and IS/IP Vendor with project management of development projects. Use of design patterns helps manage system development projects in three ways. First, work estimation metrics (hours), associated with the use of design patterns, can be leveraged once an actual work effort baseline is established and recorded. Second, vendor competencies and schedule efficiencies can be measured and compared to existing work in implementing design patterns. Third, many application architecture design patterns when combined with relevant metadata can generate up to 80 percent of code needed for implementation. Standard architecture patterns implemented as generated code provide processing consistency with the remaining 20 percent service specific relating to items like error message text and service specific edits. For example, code that processes new transactional records may perform input validation, check submitter security and authority, check for transaction duplication, create application log records, trigger business events and trigger notifications. The processing logic would be the same for many business transaction types with each transaction type specifying transaction specific validations, data service targets and error messages.

While total adherence to architecture design patterns is not feasible, general adherence provides a widget approach to software-development estimates. Detail designs constructed from the assembly of design-pattern templates can support highly accurate project estimation. Actual development costs compared to estimates over the course of project development help determine revisions and estimates corrections. Over time, change requests will become predictable and trusted.

The tracking of development hours against standardized estimates will provide insight into the level of expertise assigned to projects by vendor(s). This supports the ability, over time, to compare vendor metrics and cost.

Most, if not all, of today's programming languages are still verbose. Even straight-forward Create Read Update Delete (CRUD) action methods can result in several hundred lines of code each. 75-80 percent of code for CRUD action methods can be generated by using well defined design patterns. Example of generated logic that can implemented consistently would be patterns to prevent duplicate entry on create processes and logical locking processing on update or delete transactions that verifies other users or processes have not updated a record between the time a record is retrieved and updated. With that level of generated code, the effort to enforce application design is a much easier task. Generation reduces coding and allows generation of unit testing cases.

5.2.2.2 SOFTWARE DESIGN / CODING PATTERNS

The Agency expects the FX Project Owners and other stakeholder organizations to follow sound software design / coding practices and use applicable software development and coding patterns in the software they develop, configure, and implement for FX projects. Initially, formal monitoring of application coding practices and use of industry coding patterns is a secondary compliance priority for the software developed, configured, and implemented for FX projects.

The SEAS Vendor will work to provide awareness of industry standard coding practices to vendors and system implementers and provide guidance as situations of practices not being followed are identified in deliverable reviews.

The primary software design / coding patterns that are of compliance concern are:

- **Concurrency related coding patterns** – Concurrency coding patterns are those patterns that ensure code supports concurrent user or system access to information. A classic example of this is the common process of modifying and updating existing data records. Inexperienced developers may simply retrieve information to a page, let the user enter modifications on the page and then perform an update of data records with the values from the page. If multiple people or systems are updating the record where one person or system updates one field and another is updating a different field, the update by one user or system can be overwritten with older data when the second user or system updates the record with values from the second user's page or system. The preferred data update pattern that supports concurrency is a stateless process of retrieving data with last update timestamp and performing conditional update to the data that had that last update timestamp. If another user updates the information after the

requestor retrieved the data, they would need to re-retrieve the data to not overwrite a more recent change by another process. This approach is preferred compared to setting database record locks.

- **Use of application audit and error handling patterns** – Most application monitoring and problem solving would occur by the FX Project Owner responsible for applications they provide. Standardizing application audit trail processes and error handling patterns improves the completeness and consistency of content application logs. Quality application log data helps monitor application quality and solve cross-application issues.

If the need arises, a code quality analysis tool can be acquired and implemented at the appropriate spot in the development process to minimize the cost of changes and maximize the identification of questionable code.

5.3 Multilayer Application Architecture Model

The FX Application architecture uses layers to separate application activities. The layers are fully described below but are there to simplify the coding and insulate changes to the system from other code. Each layer has a task, which the other layers do not share.

5.3.1 Application Architecture Services

The Application Architecture connects business services with technical services, as shown in **Exhibit 5-1: Conceptual Technical Architecture Diagram**. The Agency tailors business services to environmental needs. The Application Architecture framework defines services from the abstract level to the design level, which allows the Agency to build service interfaces as standard interfaces without dialects caused by interpretations.

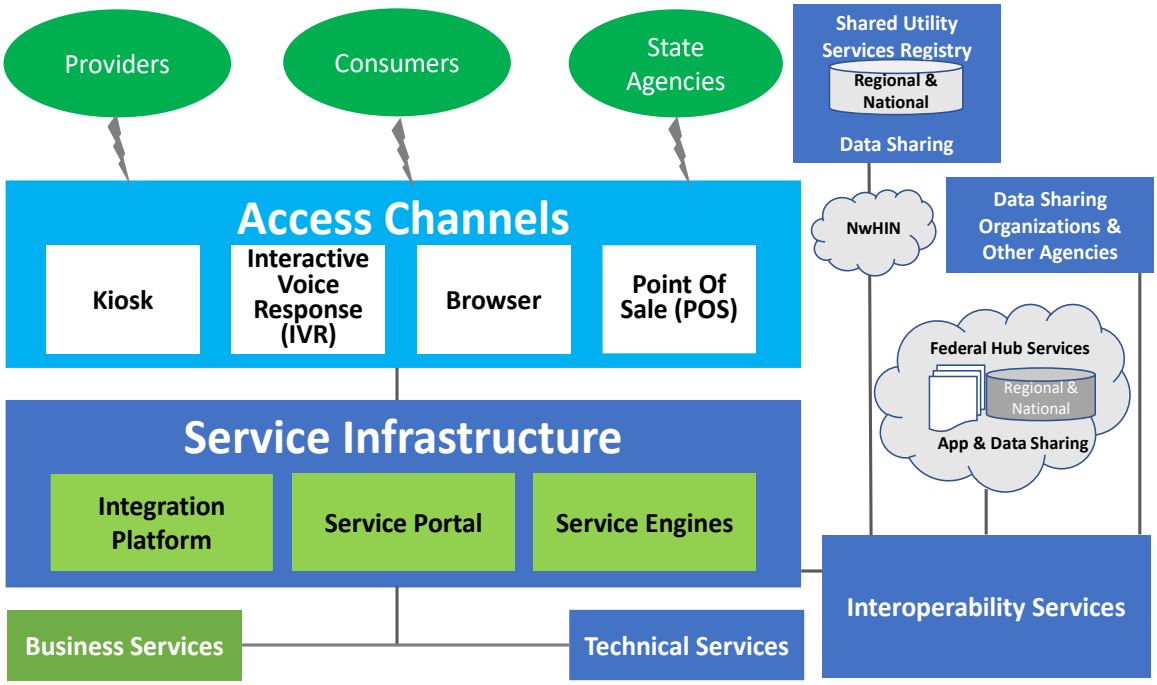

Exhibit 5-1: Conceptual Technical Architecture Diagram

The service infrastructure includes standards-based elements that use service-process integration and data sharing with other organizations and agencies. The Application Architecture framework is compatible with the Federal Health Architecture (FHA), the Nationwide Health Information Network (NwHIN), regional and national shared data sources, and the network on Regional Health Information Organizations (RHIOs). The Application Architecture framework defines a series of interoperability services based on Web Services (WS) and Extensible Markup Language (XML) message formats and protocols. The tools the Agency requires to establish interoperability, data capabilities, and other support requirements, are available to the Agency in groups using common facilities.

The following sections provide a description of the top-level Application Architecture. They describe fundamental infrastructure components, such as the ESB, Service Management Engines, infrastructure services (e.g., external data-sharing and hubs), and provide references to industry standards.

5.3.2 Components of Application Architecture

A multilayer Application Architecture model represents a combination of applications and connections to deliver services to stakeholders, as shown in **Exhibit 5-2: Multilayer Application Architecture Model**. The four (4) levels are the Access Layer, Service Management Layer, Service Application Layer, and Platform Layer.

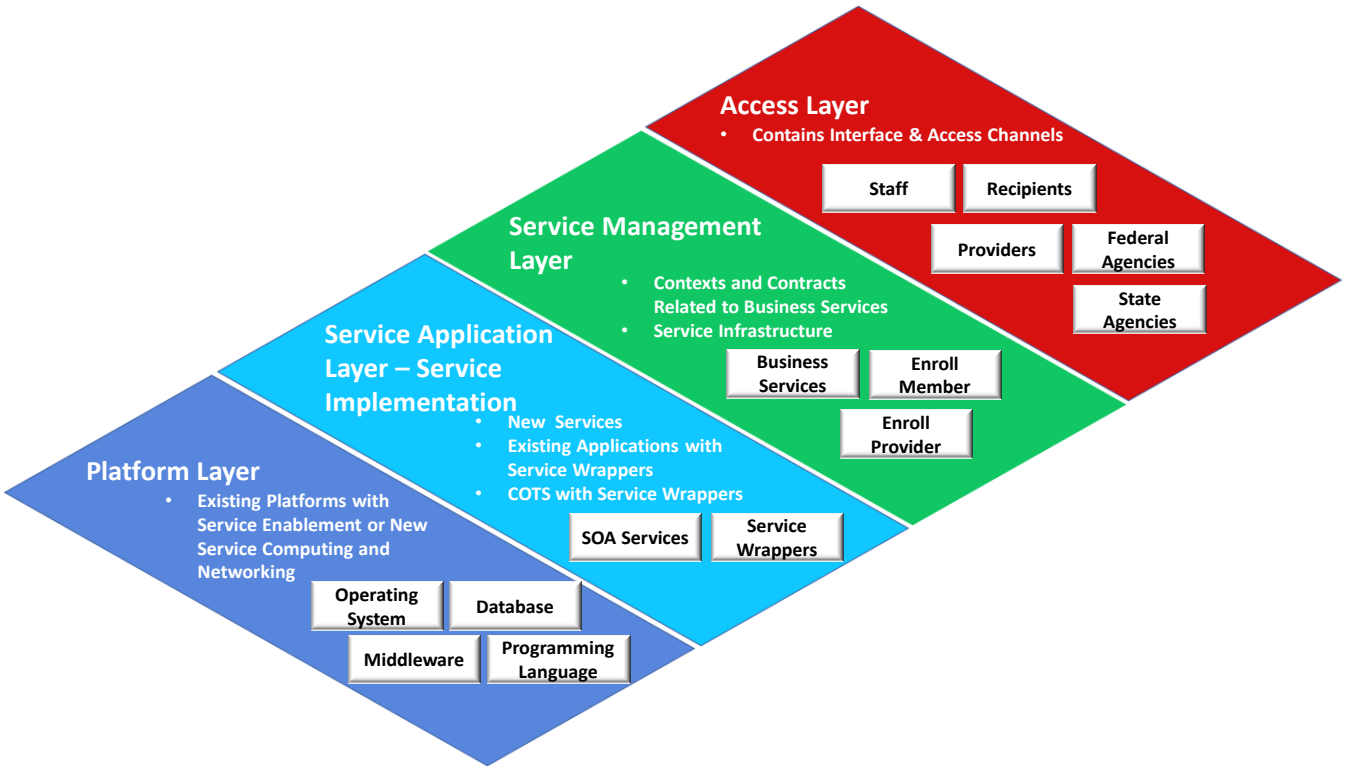

Exhibit 5-2: Multilayer Application Architecture Model

- **Access Layer** – This layer is where end users connect to the application. This is most likely through a user interface like a web page. Additionally, this could be via a web service or API accessed by an internal or external system.
- **Service Management Layer** – This layer consists of the service infrastructure, service contexts, and service contracts for each business service. All business and technical services are exposed via this layer. The Service Management Layer links to the Service Application Layer, either directly or through service wrappers.
- **Service Application Layer** – This layer consists of data and business rule services. Although the Service Application Layer consists of services, those services might be new services, wrapped legacy applications, or wrapped-COTS products. The Service Application Layer evolves incrementally as new applications are added.
- **Platform Layer** – This layer includes the software that is necessary to support the execution of the Application Layer.

5.3.3 SOFTWARE FACTORY

The FX is positioning to apply software factory concepts to the development of FX business services and applications. A Software Factory assembles software components for use in other computer software applications or components. This assembly uses a standardized software construction process. A software factory mimics manufacturing techniques and principles. Software assembly is like a manufacturing Bill of Material (BOM) process. Business services are

comprised of intermediate products, including data services, business rule services, technical services, and technical components. The software factory assembly process manages the other services used by a business service to assemble the business service to use sub-components with minimal custom integration login.

Software Factories principles reduce the cost and time associated with developing new applications. To accomplish this, the software factory assembly process integrates pre-built services and tested software components to create new business services. New business services assembled with the software factory technique cost less since the primary cost is assembly and testing of the assembled service, which can also reduce time to implementation.

Software Factories excel at producing predictable components. Predictability is key to reducing consumption time. Predictability requires the use of templates and a coding standard.

A simple Return on Investment (ROI) for a reused object is:

1. Cost New = Cost of developing, testing, and registration
2. Consumption Cost = Cost of the developer finding and consuming the existing object, and then registering the new usage
3. Times Used = Number of times reused + the original development
4. Total Reuse Investment = Cost New + (Times Used * Consumption Cost).
5. Savings = ([N]) * Cost New) – Total Reuse Investment.
6. ROI = Savings / Total Reuse Investment.

See the following example:

- Cost New = $5,000
- Consumption Cost = $1,000
- Times Used = 4= 3 (three times reused) + 1 Original Development Use
- Total Reuse Investment = 5,000 + (3*1,000) = 8,000
- Savings = ((3 + 1) * 5,000)) – 8000 = 20,000 – 8,000 = 12,000
- ROI = 12,000 / 8,000 = 150%

Actual reuse typically runs about 10-20 percent, so actual realized benefits may vary from calculated values.

Some of the primary considerations in the implementation of a software factory to build reusable services include:

- **Discovery** – The Discovery requirement is one of the most significant factors in the success of a Software Factory. A developer must be able to locate and consume a Software Factory object. Before a developer creates a new object easily and accurately,

the developer verifies the existence or absence of an object that meets or satisfies the required processing. The earlier in the object's life cycle that developers determine that an object exists, the lower the total cost to implement and operate. The capture of discovery information occurs in the planning phase after the completion of high-level design work.

- **Impact Analysis** – The advantages of software reuse are lost if a fully-automated impact analysis is not available. Inaccurate impact analysis results can cause the developer to distrust the process. This process must be able to track a changed object from its consumption all the way up the chain to the executable(s), while adding to the list of directly- and indirectly impacted objects. This includes the possibility that multiple executables may be impacted
- **Disperse Ownership** – To maximize savings, reuse is most effective when applied across the enterprise. This means that one organization may depend on services maintained by another organization. The loss of control and need for cross-unit coordination undermine the benefits of reuse. In addition, distributed ownership can result in cross-organization staffing and budgets constraint considerations that affect the timelines of proposed changes and additions.
- **Cost Distribution** – This determines the entity that pays for the cost of mandatory and desired changes to shared objects.
- **Project Schedules** – Interdependencies between organizations, project teams, or developers can create bottlenecks. Delays can occur when a developer's timeline conflicts with the revised schedule for one of the needed objects. A process keeps shared component development and maintenance aligned to agreed schedules help minimize issues as they arise.
- **Compliance** – Providing strategic guidance, supporting vendors in complying with standards and practices and assessing compliance is a responsibility of the SEAS Vendor. Review of deliverables, project artifacts, and development phase reviews are all points at which compliance with code generation development and use of factory generated and developed code occur.
- **Service Governance** – Software best practices evolve and will require changes. Predictability falls as changes to templates and procedures evolve. With enterprise reuse, changes in templates and procedures affect all organizations. The FX Technology Standards Committee or a designated work group is necessary to make the difficult decisions on what changes are or are not made.
- **Culture** – For new processes, people want to continue to do as they have in the past. Today's software-development process differs across the organization. Reuse requires commonality. Arguments can arise regarding which is the best approach. What is best for one organization may not be best for the enterprise. What is best for the enterprise must be sensitive to the impact on other organizations. Culture does not change quickly and requires upper-management support if it is to change at all.
- **Versioning** – Software Factories use a process much like manufacturing does. This process is *change by addition*. This means that changes that affect consumers are made as the release of a new version instead of as an update to an existing version. Concurrent support of a new and existing version allows beta testing, incremental

deployment, and reduced need of simultaneous implementation of change in consuming systems. To reduce ongoing support complexity, it is important to control the number of active versions of software concurrently in use. Global infrastructure or component changes can affect the enterprise. A new DBMS, ESB, language version, and many other changes, can require a widespread reverification at a minimum. When old versions are not retired, this burden becomes even higher.

- **Maintenance** – When changes are required that affect multiple organizations, there must be coordination at a detail level. Versioning can help in some, but not all situations. At a minimum, more coordination time is required.

The use of an automated Application Service Registry reduces the complexities of many of the above considerations and challenges. The Application Service Registry is most effective when it supports registration at any stage of the application's life cycle and includes all consumed and consuming objects.

5.3.3.1 SOFTWARE FACTORY STRATEGIC TOPIC

Strategic Topic 5-1: Software Factory describes the Agency strategy to apply Software Factory practices in the development of new systems, applications, and services.

SOFTWARE FACTORY	CURRENT	TIMELINE 2018	2020	2022	2025
Module / System provider preference	None	Exception with Approval (2019)			
COTs Products		Vendor uses preferred development process	->		
FX Projects		Mandatory (2019)	->		
Agency-wide		Optional	->	Mandated All New Projects Begin Migration of Existing Systems	->
Cross-Agency Systems				Pilot as Optional for New Projects	

ANALYSIS
MITA's approach requires the usage of business and technical services with their solution sets. To support maximum software reuse data services would need to be included. Business rule services are a new concept, but act to centralize recording of all business rules for reuse. The proposed Application Architecture uses an assembly approach whereby higher-level services are constructed utilizing multiple lower-level services with little or no extra code. This maximizes software reuse and minimizes duplicate code.

5.3.4 BUILDING SERVICES

To illustrate how application services are built, **Exhibit 5-3: General Structure of Business Services Layers** shows the primary components of a SOA-based business service. Each service contains Interface Components that include security attributes. For example, a service can inspect incoming messages to verify the message originator has authorization to invoke the service.

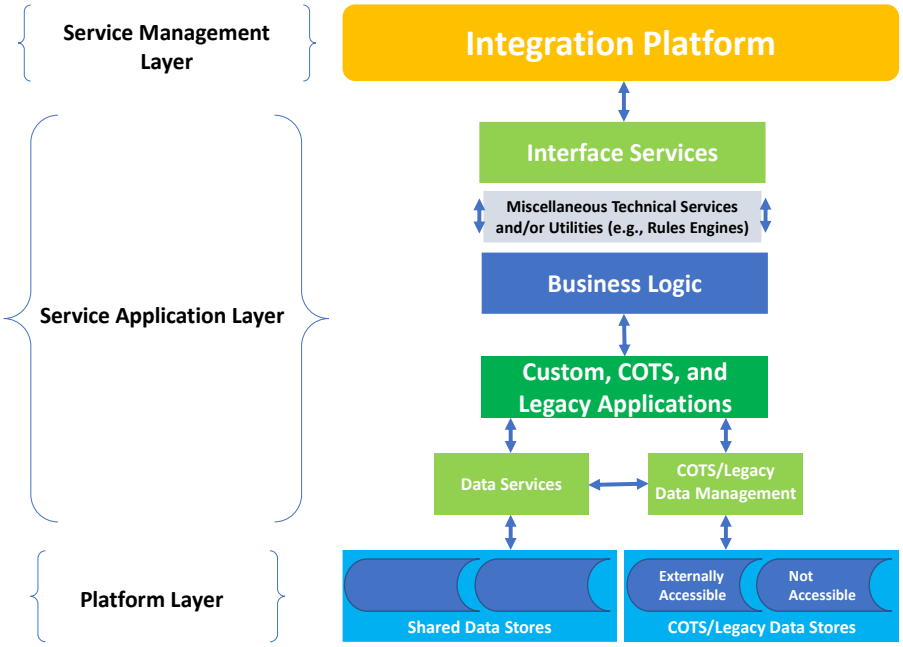

Exhibit 5-3: General Structure of Business Services Layers

The Business Logic portion of the business services examines the received message and coordinates the execution of underlying custom, COTS, or legacy applications to provide the necessary results for the received message. The business logic process accesses miscellaneous enabling technical services to perform its processing responsibilities.

The business service's applications portion accesses and uses a set of data services to provide uniform access to its data, by using standard definitions for all shared and externally accessible data in the FX. Direct access to application data stores containing private or sensitive data, whether programmatically or using integration technologies, is strongly discouraged. Allowing programs to directly access data stores:

- Requires user-level role-based security to be implemented in the DBMS (this is difficult to maintain)
- Makes impact analysis much more complex (may require scanning all code),

- Introduces the risk of inefficient data access negatively affecting performance of data access by other users or systems
- Undermines the integrity of access logs

When coded according to the FX Application Architecture, it is easy to find the code impacted by a change.

Data contained in COTS and legacy systems requires special handling. For legacy data stores, data access routines tend to be specific to the underlying technology. Data stores used by COTS packages often have proprietary data structures; data-access routines and only vendor-approved Application Programming Interfaces (API) access these data stores. Not all COTS data stores are externally accessible, not even via API. The data services provide transparent data-management services to all accessible data stores in the FX. The data-access services are platform independent; as legacy systems phase out, or technical staff restructures databases, changes to the data services do not affect the consumer.

Using SOA standard application methods, systems call services at different architecture layers. To maximize reuse across the FX, the Agency standardizes the service descriptions, invoking messages for all the services connected to the Integration Platform (IP) and as many of the lower-level technical services as possible.

5.4 Application Architecture Key Components

The FX Technical Architecture will align and adhere to the application architecture direction guidance provided in MITA version 3. The depictions of these components that follow reiterate use of the MITA application architecture frameworks, principles, patterns, and direction.

Exhibit 5-4: Service Infrastructure shows the relationship between the Application Architecture infrastructure and services. Business and technical services linkages are by service infrastructure elements using the integration element known as the Integration Platform (IP) especially for services on different platforms or systems.

The service integration and interoperability methods provide loose-connectivity and are important enablers of flexibility. Consistently using this service approach is an important part of designing the SOA. The significant components of the Application Architecture are:

- Integration Platform (IP) and Access Channels
- Service Management Engines
- Service Gateways and Mediators
- Distributed Computing and Data Access
- Framework Documents
- Interoperable Services
- Security and Privacy (S&P)

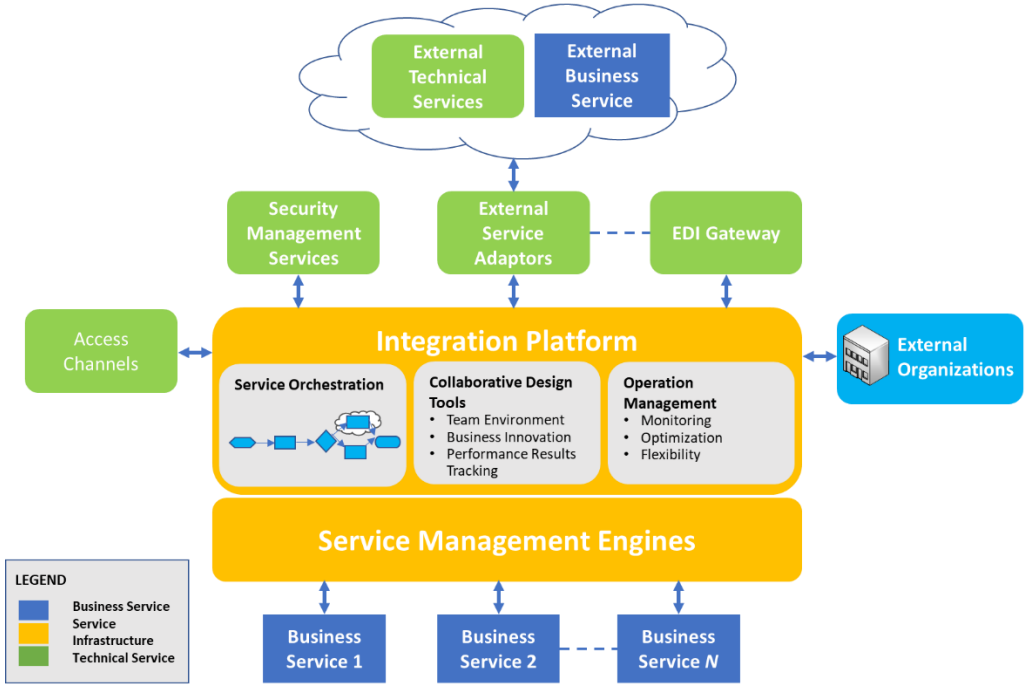

Exhibit 5-4: Service Infrastructure

5.4.1 Integration Platform and Access Channels

The Integration Platform (IP) is an infrastructure component that supports the Modularity Standard. This standard addresses the following:

- Using Software Development Life Cycle (SDLC) methodology
- Identifying, describing, and using adaptable and open interfaces
- Using Business Rules Engines for fast response to program changes
- Submitting standardized business rule definitions to future design repositories to support collaboration

Customarily, users have accessed or linked to systems using proprietary formats of individual vendors, developers, or integrators, thus making more-complex systems and hindering interoperability and integration abilities. The Application Architecture addresses this issue through implementation of Access Channel Services, shown in **Exhibit 5-5: Integration Platform and Access Channel Services**.

Access Channel Services support specific device-handling types. Each Access Channel Service handles the unique features of specific device types.

The access channel routing and administration features tie into the security boundary protection services, accesses other S&P services that support single sign-on needs, authenticates users, sets up the Role-Based Access Control (RBAC) permission, and passes a token to the IS/IP.

The access channel uses a token with the IS/IP and passes it to the business service with any additional information that relates the service message to the problem domain. The access channel can carry the related data along but often only carries a pointer to the data. The IS/IP or a technical service has the capability of logging and gathering levels of tracking information for exception handling, recovery, and the S&P audit.

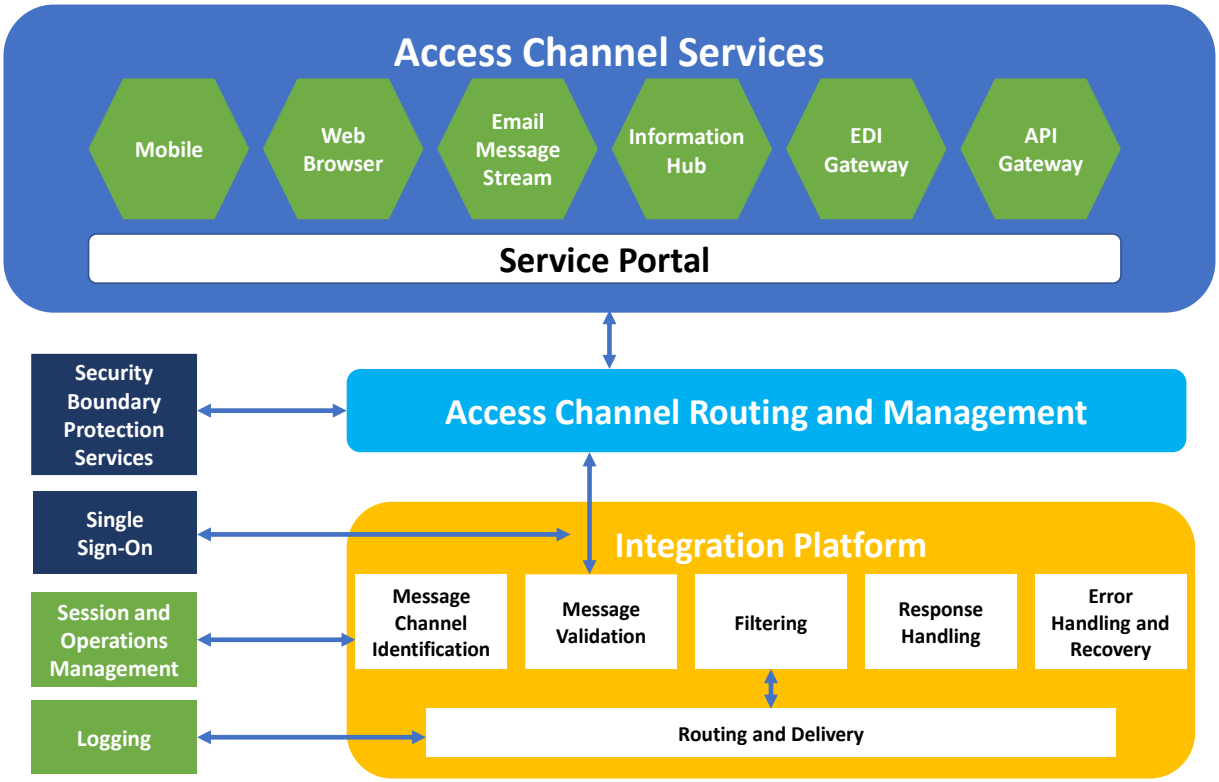

Exhibit 5-5: Integration Platform and Access Channel Services

Access Channel Services and the IP manage a range of interoperability issues across business areas to enable providing cross-organization interoperability services. An Access Channel provides the translation from the device and technology unique features, such as the screen size and the keys layout, to translate the message to a common format that the IP handles. The Access Channel routing and management capability that ties into the Security Boundary Protection Services can access other Security and Privacy services that support single sign-on needs.

Many sites use web portals to enable access to the system. These service portals require an Integration Manager to manage the human side of the workflow. Automated work queues manage user access through an assigned role. Users could play multiple roles that may require additional queues.

Service portals follow Web Services (WS) standards, such as WS-Remote Portlet standards. Service users access the services that provide a link between their work queues and other

special areas to form a service connection. Each work queue represents one end of the service end point and the business services represent the other end. The service end points, which are a Uniform Resource Identifier (URI) for service users and service providers, provide the ability to connect in a consistent method as one of the main uses of the service portal. The Application Architecture uses WS-Addressing standards for the service end points. The service portal represents a natural development of the Web portal technology and contains capabilities that enable interaction with the other service-infrastructure components. Some major capabilities of a Web portal are supporting Web browsers that understand Web Service Definition Language (WSDL) and supporting output to other service elements with service-formatted messages. The portal and some of its parts are service end points.

5.4.2 SERVICE MANAGEMENT ENGINES

Another fundamental component of the FX Application Architecture infrastructure is Service Management Engines. Service Management Engines are specialized processing software or applications that manage specialized business and technical services processing. Examples of service management engines are workflow engines and rules engines. These types of software engines enable specialized processing of workflow and activity management and rule-based decision making.

Exhibit 5-6: Service Infrastructure – Service Management Engine shows the integration platform invoking the service management engines to process specific types of service requests. The Organization for the Advancement of Structured Information Standards (OASIS) publishes SOA standards, which include standards for ESB and Service Management Engines.

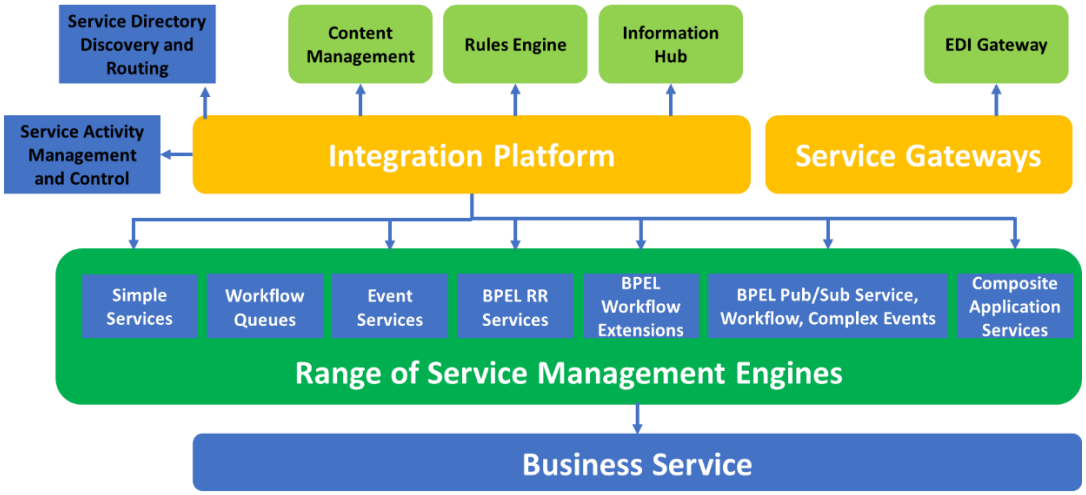

Exhibit 5-6: Service Infrastructure – Service Management Engine

Service Management Engines implement the service contracts defined in WSDL, or the more-advanced service composition and business-process management languages. These engines are diverse and can support a variety of capabilities. Different services necessitate different service-behavior needs, from simple services to complex and composite services. Depending on

Agency needs, a business service will use various service and management arrangements. The following types of service engines provide this orchestration and management:

- **Simple Services** – Service specification with a service contract, as defined in WSDL, and a simple request of a service and response
- **Workflow Queues** – Services performed primarily by users to manage incoming work requests and process messages or cases according to defined work prioritization and escalation rules. Work requests are routed to a specific user or group of users who normally handle that work item. A user or role performs actions required by the queue of messages and work items. Each user or role has a work queue.
- **Event Services** – Event services manage the delivery of event messages to several business services and people, roles, or contexts interested in a condition and change of behavior of interest.
- **Business Process Execution Language (BPEL) Engine with Request Response** – A service that triggers a Business Process, as defined in WS-Business Process Execution Language (WS-BPEL 2.0 standard), by using a triggered message in a simple request-response message pattern. The business process executes, and the locations identified in the Business Process Model receive the results.
- **BPEL with Workflow Extensions** – This is a service that combines BPEL with the ability to integrate the workflow queuing and high levels of workflow management. The Workflow Management Coalition (WFMC) standards group (Level 4) is defining a common bridge between BPEL and workflow tools.
- **BPEL Advanced** – A service that includes more advanced BPEL features (Pub/Sub, Service Plus, Workflow, and Complex Events). **Note:** A Pub/Sub model refers to the asynchronous service of message publishing/subscribing; anytime a message is sent, all subscribers receive that message.
- **Composite Application Services** – Services that address more comprehensive business processes and describes how to handle transactions, user involvement, and long-running activities. The WS Composite Application Framework standard addresses these needs.

Service Management Engines incorporate other technical services within the orchestration and workflow processes. Rules Engines and Enterprise Content Management (ECM) are promising services for usage within the Enterprise. Rules Engines allow the entry of easily configured business logic into the stream of events, while ECM services enable entry of different forms of information content in a variety of manners. For example, the decision-management methods included by rules engines provide logic, which involves the retrieval and manipulation of a database, so database integration through the Service Management Engines becomes critical. Workflows combined with an enterprise content manager will enhance record control to help comply with government regulations, such as the Health Insurance Portability and Accountability Act (HIPAA), Sarbanes-Oxley, Payment Card Industry Data Security Standard (PCI DSS), and the Federal Rules of Civil Procedure. In addition to publishing SOA standards, the OASIS group has established Content Management Interoperability Standards (CMIS) to provide uniformity among ECM offerings. These technical services are becoming mission-critical components of

the Medicaid Enterprise as the computing environment and end-user needs and expectations expand at an alarming rate.

Marketed products exist in each of the categories described above. During the next few years, Service Management Engines will be a significant aspect of designing and managing for change, which is an important aspect of the MITA goal of flexibility.

5.4.3 SERVICE GATEWAYS AND MEDIATORS

To deliver services end-to-end, the Application Architecture will provide a common set of service elements that agrees with the standards that are set, and links technologies that address changes and innovation. Those linking technologies are semi-automatic and intelligent to handle many of the common interoperability elements, as shown in **Exhibit 5-7: Service Gateways and Mediators**. The Service Gateway and Service Mediators are two service elements that currently accommodate this necessity. The Application Architecture uses bridging services to mediate differences because absolute compliance with standards is not realistic.

The Service Gateway addresses external or cross-boundary compatibilities between IS/IPs. The Service Gateway can interface with many different formats, such as Electronic Data Interchange (EDI) gateways or HIPAA translators.

The Service Mediators provide a common service contract and service-message interface that translates specific vendor offerings. Although three (3) different products may have similar capabilities and a service interface, they might differ slightly. Service Mediators handle those differences.

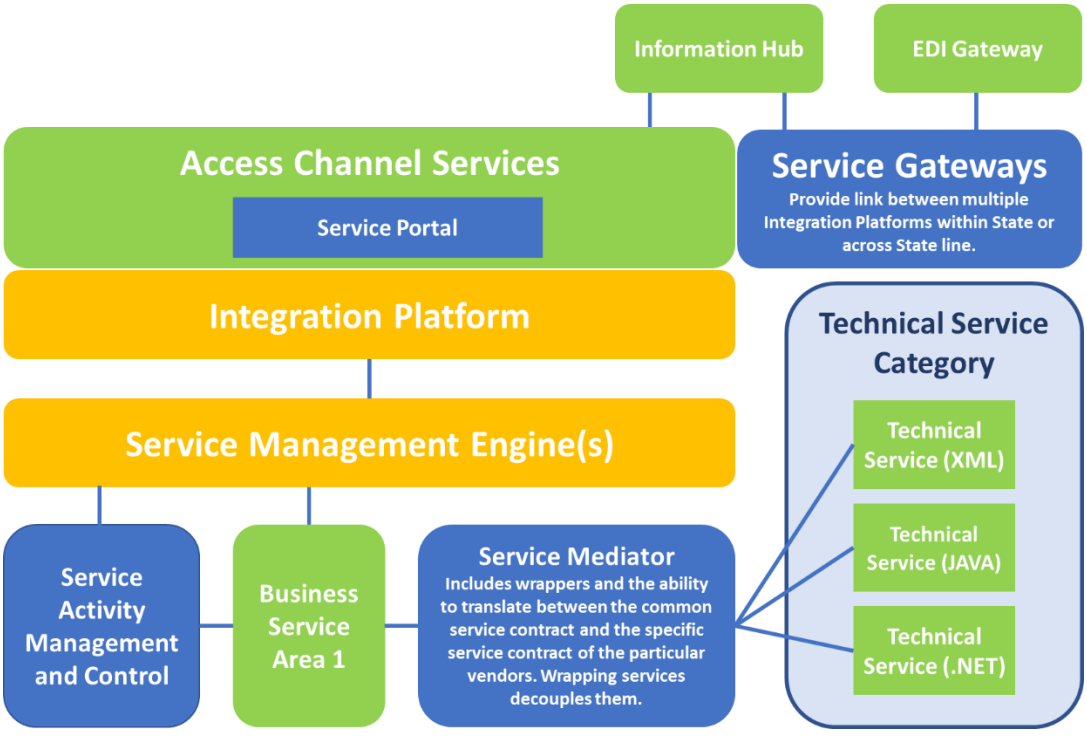

Exhibit 5-7: Service Gateways and Mediators

5.4.4 CLOUD INTERCONNECTIVITY SOLUTION

The FX solution will be comprised of several vendors, each of which deploys their solution into potentially several different cloud hosting services. Two of the initial solutions, IS/IP and EDW, are hosted on Oracle Cloud and Amazon Web Services (AWS), respectively. One of the challenges facing the FX project is interconnecting the cloud services from disparate cloud computing platforms and vendors. Establishing numerous point-to-point connections to each vendor's solution and cloud environment would be expensive, difficult to manage, and introduce latency. It also introduces the potential for issues when troubleshooting problems when they surface.

To address the connectivity issues, Equinix Network Edge was selected as the cloud interconnectivity solution. Equinix is a digital infrastructure company specializing in providing internet connections and data center solutions. It provides data centers and interconnectivity between multiple vendors and systems across a cloud-first world, serving as a single point of contact for connectivity issues. Equinix also provides private and secure connectivity to major cloud service providers (i.e., AWS, Microsoft Azure, Google Cloud, IBM Cloud and Oracle), as well as to Network Service Providers (NSP), and other mission-critical *as a service* providers. Flexibility is accomplished by allowing other vendors and organizations to join up and connect as the needs of an enterprise change over time.

Equinix's environment provides the ability for each vendor and internal Agency systems to connect to any platform through a single connection to the central Equinix integration hub. This type of connectivity allows the Agency to achieve an agile and secure digital infrastructure that delivers its mission goals at software speed. By leveraging a vendor-neutral, globally distributed interconnection platform, known as Platform Equinix®, the Agency can achieve direct and secure access to the world's largest ecosystem of clouds and networks to ensure secure, cost effective, and low-latency communications.

The diagram shown in **Exhibit 5-8: Equinix Cloud Interconnectivity Solution** below is a representation of the planned Equinix connectivity solution for EDW and IS/IP:

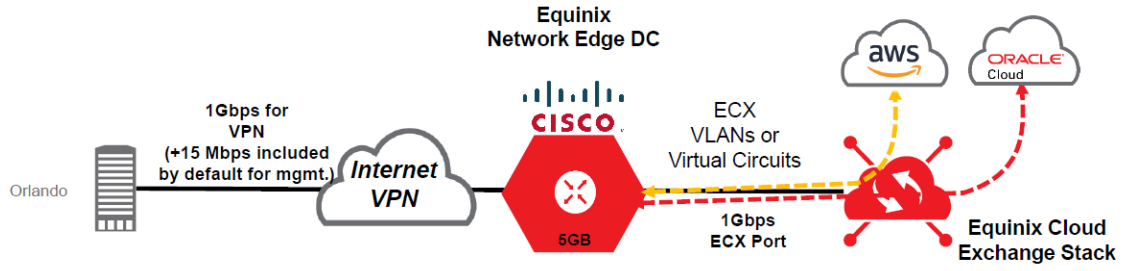

Exhibit 5-8: Equinix Cloud Interconnectivity Solution

5.4.5 Distributed Computing and Data Access

As more external private and public organizations make solutions to meet specific requirements available via Cloud Computing, the use of services external to the Medicaid Enterprise will expand. Cloud Computing could provide various types of service-oriented solutions on a Software as a Service (SaaS) basis, where stakeholders incur costs only when they invoke a solution. The Application Architecture uses a modular, flexible approach to systems development, including usage of open interfaces and exposed APIs. The Application Architecture pursues a service-based and cloud-first strategy for system development. The Application Architecture helps to identify, evaluate, and incorporate commercially or publicly available COTS or open-source solutions, and makes considerations and plans for cloud computing. *T-4: Technical Management Strategy* Section 3.2.5 has more details on cloud computing.

Interoperability between the mechanized claims processing and information retrieval systems, eligibility determination systems, and the cloud-based external service(s) may not require any integration through adapters. Other cloud-based external calls of services or solutions that are not interoperable, including services with non-conforming service contracts, or non-SOA solutions, will require some type of data or message transformation as shown in **Exhibit 5-9: Distributed Computing Services via Cloud Computing**.

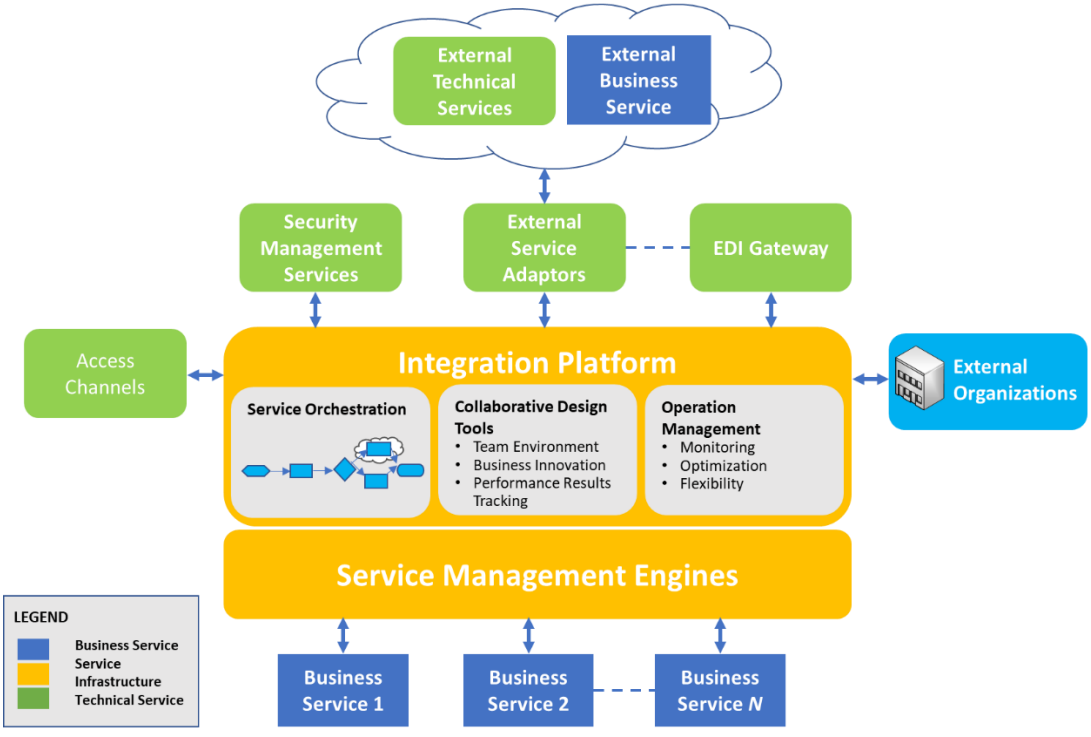

Exhibit 5-9: Distributed Computing Services via Cloud Computing

The types of services presented through Cloud Computing are extensive and vary from business services to technical services, depending on the need and application. The access to these

services, whether based on interoperability or by integration, passes through typical network security frameworks established within the enterprise architecture.

While external servers perform the processing, accessing external data stores requires a utility service or set of services to connect or transform data. **Exhibit 5-10: Data Access Services via Cloud Computing** shows the orchestration of accessing data that resides in a Cloud Computing environment. Query processing, directory management, concurrency control, and deadlock management deals within the Cloud Computing environment by the service provider. Service Level Agreements (SLAs) are a necessity to address the numerous coordination and operational details.

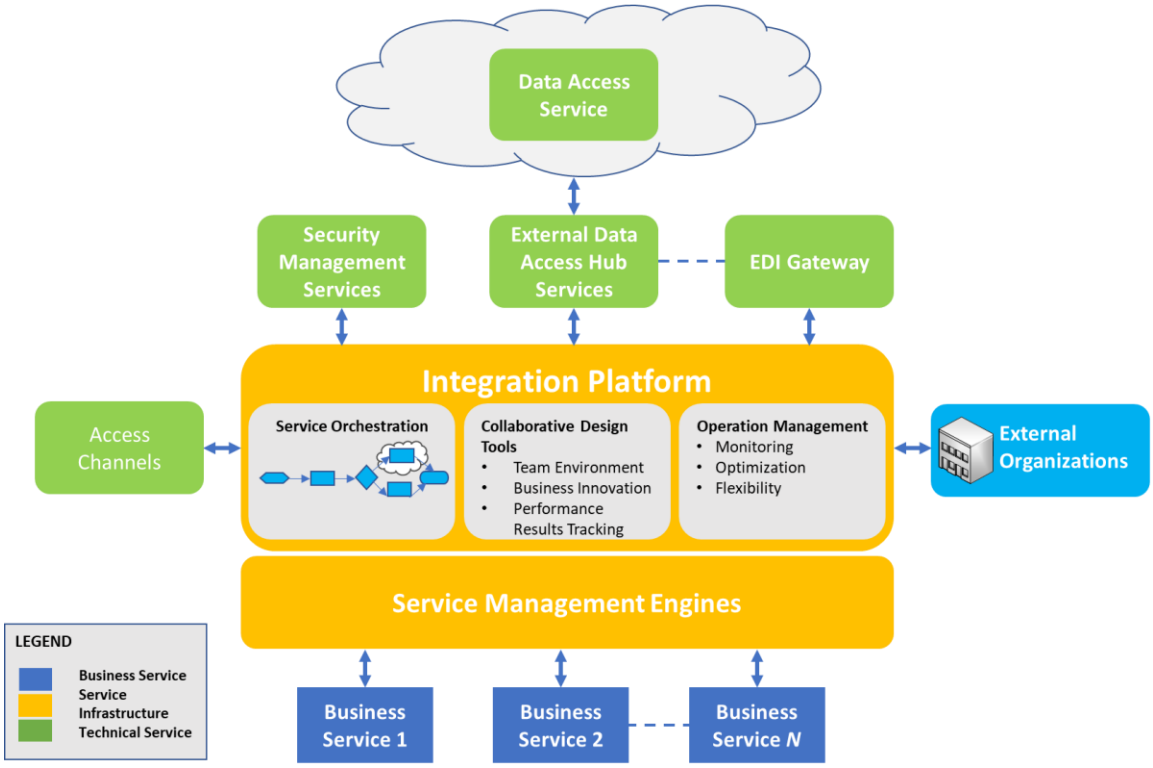

Exhibit 5-10: Data Access Services via Cloud Computing

5.4.6 INTEROPERABLE SERVICES

Interoperability is an Application Architecture goal and provides an overall solution to many of the FX business challenges. Systems must ensure seamless coordination and integration with exchanges (whether run by the Agency or federal government) and must allow interoperability with Health Information Exchanges (HIE), public health agencies, human services programs, and community organizations, which provide outreach and enrollment-assistance services. Technical challenges address business interoperability, which includes the following:

- Lack of incentives to cooperate with other states will require selling the value of interoperability benefits.

- Lack of funds for cross-organization activities may require changes to budget allocations.
- Lack of infrastructure to support interoperability and reconciliation may slow deployment.
- Legacy systems with dissimilar definitions and stovepipe systems (systems that have the potential to share data or functionality with other systems but do not) might not conform to new standards for interoperability.

The service-oriented interoperability approach provides technical enablers common ground for addressing key issues, reducing the learning curve, and allowing the Medicaid community to share architecture designs. The Application Architecture can apply hub architecture, interoperability and access channels, and utility services to meet these challenges, as shown in **Exhibit 5-11: Service Oriented Architecture.**

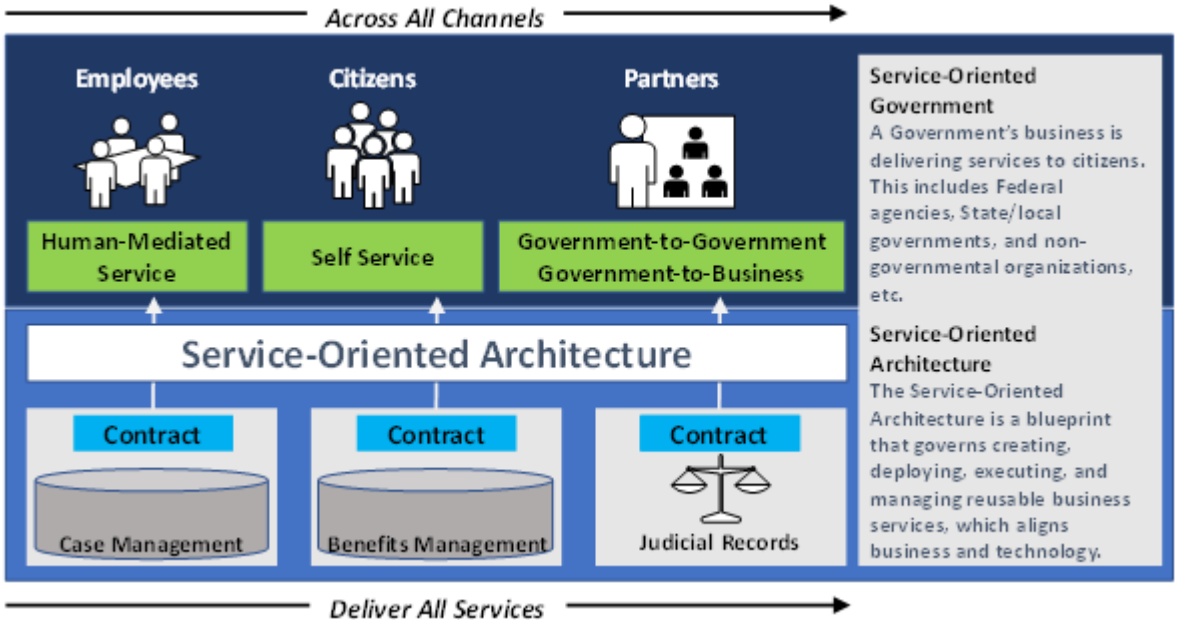

Exhibit 5-11: Service Oriented Architecture

5.4.6.1 KEY SERVICE INTEROPERABILITY ELEMENTS

The critical concepts for service interoperability include the following considerations:

- Because interoperability is a byproduct of solid programming techniques, SOA-based services are to adhere to proper design principles with the following characteristics:
  - **Standardized Contract** – Expresses the purpose, capability, and interface content quantity.
  - **Loose Coupling** – Defines dependencies between the contract, deployment, and consumer.
  - **Abstraction** – Hides as much of the service's details to preserve loose coupling.

- **Reusability** – Positions servers as enterprise resources with agnostic function context.
- **Autonomy** – Designs service logic and deploys environment impact reliability.
- **Statelessness** – Comprised availability occurs when managing excessive Medicaid information.
- **Discoverability** – Avoids the accidental creation of redundant service(s) that implement redundant logic.
- **Composability** – Complex service compositions place demands on service design.

- Use services and messaging standards for real-time, business area-to-business area, and cross-organizational communications.
- Use services to define clear processes and consistent mechanisms for system-to-system communication, the definition of communication requirements, and recommend technologies for automated responses (e.g., WS and XML protocols).
- Use services to define a common user interface that reduces complexity and buffers applications from technical details.
- Use services to define common functions and features that separate applications and design using service utilities.
- Use services to define a logical interoperability architecture (a service overlay), which is based on hub technology and communication protocols.
- Support alternative access to the same information and services, including web (human interface), internet (machine-to-machine), and others. Data, processes, and services hide behind interoperability channels and adapt to meet changing needs by using configuration files.
- Use a business-oriented service interoperability process that focuses on the business-needs perspective, based on three principles:
  - Define common semantics (for example, the meaning of a message).
  - Define common syntax (for example, the structure of a message).
  - Define a common mechanism (the method of exchanging information).
- Define a set of common service elements and adapt them through variants and extensions.
- Define service interoperability solutions that rely on common definitions for channels and utilities that are specific to business areas, which are designed with common underlying architecture and common utility components.
- Define and create virtual-access mechanisms that can be used to exchange information.

5.4.6.2 SERVICE INTEROPERABILITY MODELS

This section describes potential service interoperability models for use in the FX. Each interoperability model is based on CMS MITA guidance.

The IS/IP Vendor will have primary responsibility for technical interoperability. As such the IS/IP will be responsible for managing the interoperability portfolio and providing guidance to FX projects on the technical use and implementation of relevant interoperability models.

5.4.6.3 ACCESS CHANNEL MODEL

The Access Channel Model performs several vital functions:

- Access Channels allows the Agency to access data and information by multiple methods (e.g., mobile, wireless, and kiosks) and to interface with organizations that provide batch interaction with messages. Private-Public Partnership access might include an organization allowed access by its contract. The features and functions accessed must have clear definitions.
- Access Channels protect rights to certain information and allow information sharing through specific interoperability channels. Access Channels plan for these exchanges and provide collaborative tools. The Access Channels and Interoperability Channels include defined connectors. Connectors define alternate access approaches. Access approaches are adaptable based on policy, failure, or recovery conditions.

Exhibit 5-12: Access Channel Model Questions and Answers provide further details about the Access Channel Model.

Question	Answer
What is the importance of the Access Channel Model?	The Access Channel Model depicts multiple access channels supported by utility services. Easy data access transforms the Medicaid business. The central concept is the importance of separating access channels from interoperability channels.
Do developers understand the Access Channel Model?	System designers evaluate possible access channels and interoperability channels to make data as readily available as possible
Do developers know how to use the Access Channel Model?	System designers adopt an architecture that separates access channels from interoperability channels and use common utility services to simplify development. These utilities may be available to share within a state, among certain Medicaid systems, or nationally.
Do developers know how to refine the Access Channel Model?	The Interoperability Portfolio updates the Access Channel Model. An interoperability portfolio is a collection of services that rely on common definitions and proven SOA characteristics. The portfolio addresses both policy and technical issues regarding the secure data exchange performed by the collection of interoperable services.
Do developers know the supporting business decisions with the Access Channel Model?	New IT procurements adopt the concepts of isolating access and interoperability using utility services.

Exhibit 5-12: Access Channel Model Questions and Answers

5.4.6.4 MITA INTEROPERABILITY MODEL

The use of an ESB in the IS/IP provides a valid foundation for Technical Architecture development. This foundation enables expansion and provides a key component for establishing interoperable application services. **Exhibit 5-13: MITA SOA Framework Integration Platform Model** depicts the core ESB and IS/IP functions referenced throughout this section.

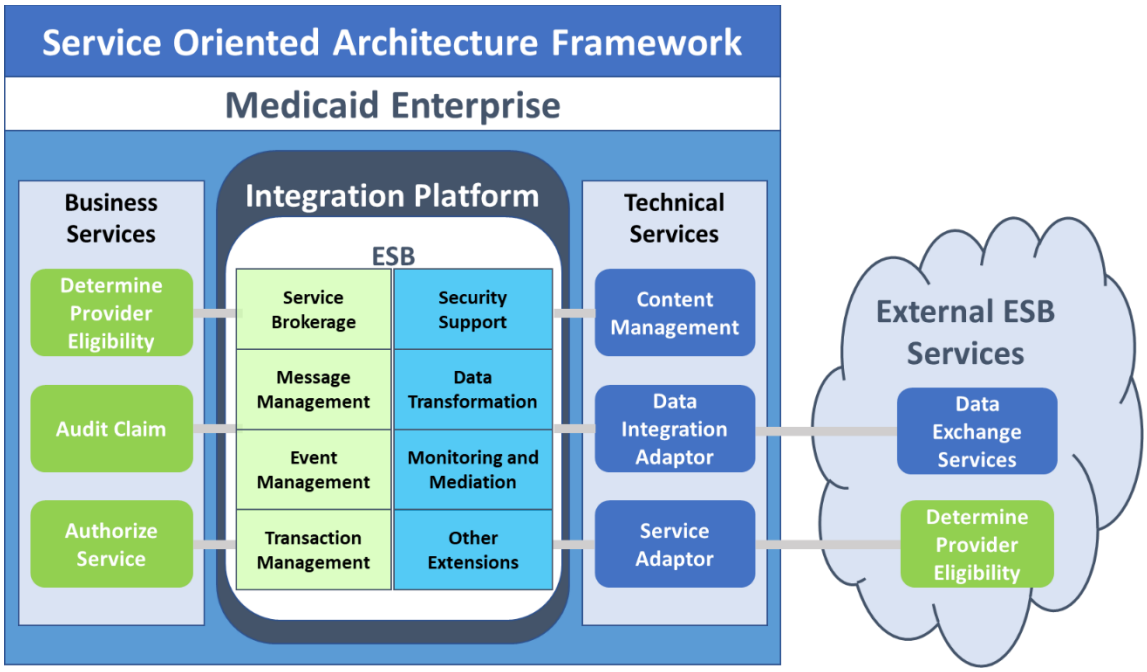

Exhibit 5-13: MITA SOA Framework Integration Platform Model

The FX Application Architecture has taken a strategic business and technical approach to interoperability, as shown in **Exhibit 5-13: MITA SOA Framework Integration Platform Model**. The Framework defines interoperability as sub-functions, topics, and types of communication, and understands that conflicts can occur when it uses common solution patterns. The Integration Platform software will offer several methods to resolve these conflicts.

5.4.6.5 LOGICAL INTEROPERABILITY MODEL

The FX Application Architecture defines and designs its interoperable concepts through a configuration, depicted in **Exhibit 5-14: Logical Interoperability Model**. This logical model is provided for future consideration though it is not planned for an initial FX Project. The Logical Interoperability Model defines hubs, virtual private network capabilities, and a common set of utility services to create the Logical Interoperability Model and addresses interoperability at many levels. The Logical Interoperability Model also addresses a minimal set of information-sharing needs in a standard way for intrastate data sharing, among business areas and with partners. The model and concepts extend based on workload, and the recovery and contingency planning needs.

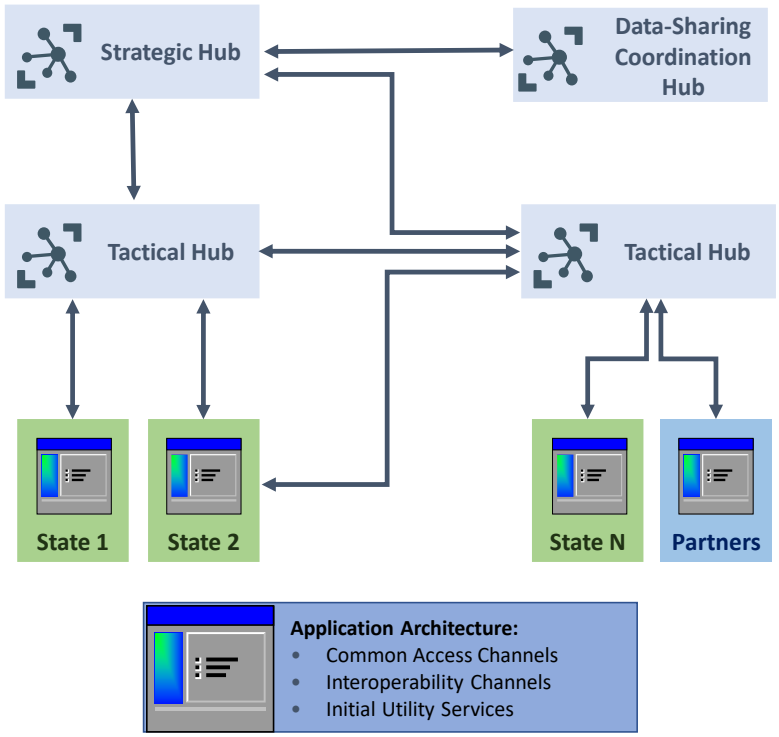

Exhibit 5-14: Logical Interoperability Model

5.4.6.6 LOGICAL HUB ARCHITECTURE

As shown in **Exhibit 5-14: Logical Interoperability Model**, the Agency can configure hubs built on standard hub architecture to address tactical, strategic, data-sharing, and coordination functions. The hub architecture consists of three (3) layers:

- **Interface Management Layer** –
  - Receives messages based on defined interoperability channels.
  - Handles all the message buffering, transport protocols, and any necessary message translation.
  - Includes any routing to the data management or the utility services layer.
  - Supports the adaptability needs or any necessary manual functions, such as special queries.
- **Data Management Layer** – Houses data stores that are either data marts or relational data models. This layer also includes a virtual data access capability.
- **Utility Services Layer** – Represents the portions of utility services that reside on the hub or server (often called the *servlet*) and provides capabilities to run on the hub, such as access to virtual models, collection, filtering, and delivery of blocks of information to the business area.

Exhibit 5-15: The Interoperability Model outlines the reasons for using an Interoperability model.

Topic	*Strategy*
Importance of the Interoperability Model	The Interoperability Model describes the business capabilities and technical functionality necessary to achieve efficient system-to-system interactions.
Understanding the Interoperability Model	System designers understand the concepts in the Interoperability Model and incorporate them into system designs.
Using the Interoperability Model	The Interoperability Model provides guidance and recommendations that support services design and development and data that the Medicaid community can share, although the Agency retains its autonomy. The Agency can follow the model to achieve cross-organizational information sharing through a common approach.
Refining the Interoperability Model	The Interoperability Portfolio updates the Interoperability Model. An interoperability portfolio is a collection of services that rely on common definitions and proven SOA characteristics. The portfolio addresses both policy and technical issues regarding the secure data exchange performed by the collection of interoperable services.
Supporting Business Decisions Made with the Interoperability Model	New IT procurements adopt these concepts of interoperability.

Exhibit 5-15: The Interoperability Model

5.4.6.7 LEVERAGING INTEROPERABILITY PROJECTS

Significant interoperability projects and investments have been made to enable federal-to-state, state-to-state, state-to-local and interoperability between many other organizations. The FX Technical Architecture seeks to leverage insights from interoperability work including:

- Projects by Intelligence agencies established infrastructure that links many organization data sources across many business areas.
- These projects address federal-to-state communications, such as those communications that relate to the Centers for Disease Control (CDC), the Internal Revenue Service (IRS), and communications concerning state grants with the Global Justice Network initiative and its definition of standard XML-based schemas.
- Interoperability and cross-boundary issues are an active area of architectural alignment the Federal Chief Information Officers Council is pursuing with the Office of Management and Budget's Federal Enterprise Architecture Program Management Office (FEA-PMO), the architecture team from the National Association of State Chief Information Officers (NASCIO), and with industry associations that support both federal and the state initiatives.

5.4.7 SECURITY AND PRIVACY

Many of the application architecture components describe Security and Privacy topics and capabilities above. For example:

- **Access Control Services** (Section 5.4.1) – Discusses coordination with security-boundary protection services, accesses other Security and Privacy (S&P) services that support single sign-on needs, user authentication, the setup of the Role-Based Access Control (RBAC) permissions, and delivery of tokens to the Integration Platform. Access Control services also discusses correlation sets passed along entirely and use of the Integration Platform or a technical service to log events and gather levels of tracking information for exception handling, recovery, and Security and Privacy auditing.
- **Service Engines** (Section 5.4.2) – Discusses how workflows combined with an enterprise content manager will enhance record control to help comply with government regulations, such as the Health Insurance Portability and Accountability Act (HIPAA), Sarbanes-Oxley, Payment Card Industry Data Security Standard (PCI DSS), and the Federal Rules of Civil Procedure.
- **Service Interoperability Models** (Section 5.4.5) – Discusses that access channels can protect rights to certain information and allow information sharing through specific interoperability channels.

5.4.7.1 SERVICE INFRASTRUCTURE OPERATION CONCEPT OF SECURITY

The Service Infrastructure Operation Concept of Security provides an overview of how the application architecture including the ESB, provided as part of the Integration Platform, protect security and privacy rights.

Exhibit 5-16: Service Infrastructure's Operation Concept of Security shows how security tokens integrate into the infrastructure. The user signs on, which starts an authentication and authorization process that consists of the following steps:

1. The security service creates a token (like an identification) to provide Security and Privacy rights, depending on the specific sign-on, and then it creates a Role-Based Action Control (RBAC).
2. The token notifies the IS/IP the message can transmit and complete the connection to the proper business service.
3. The business service initiates the correlation sets, registers the correlation token, and tracks the progress through the business service. The user can stop or pause the process and sign on the next day. If a failure happens, the recovery can identify the progress, based on the token and the corresponding correlation set.
4. Technical services and business services use the token to determine access control to business logic and data access.

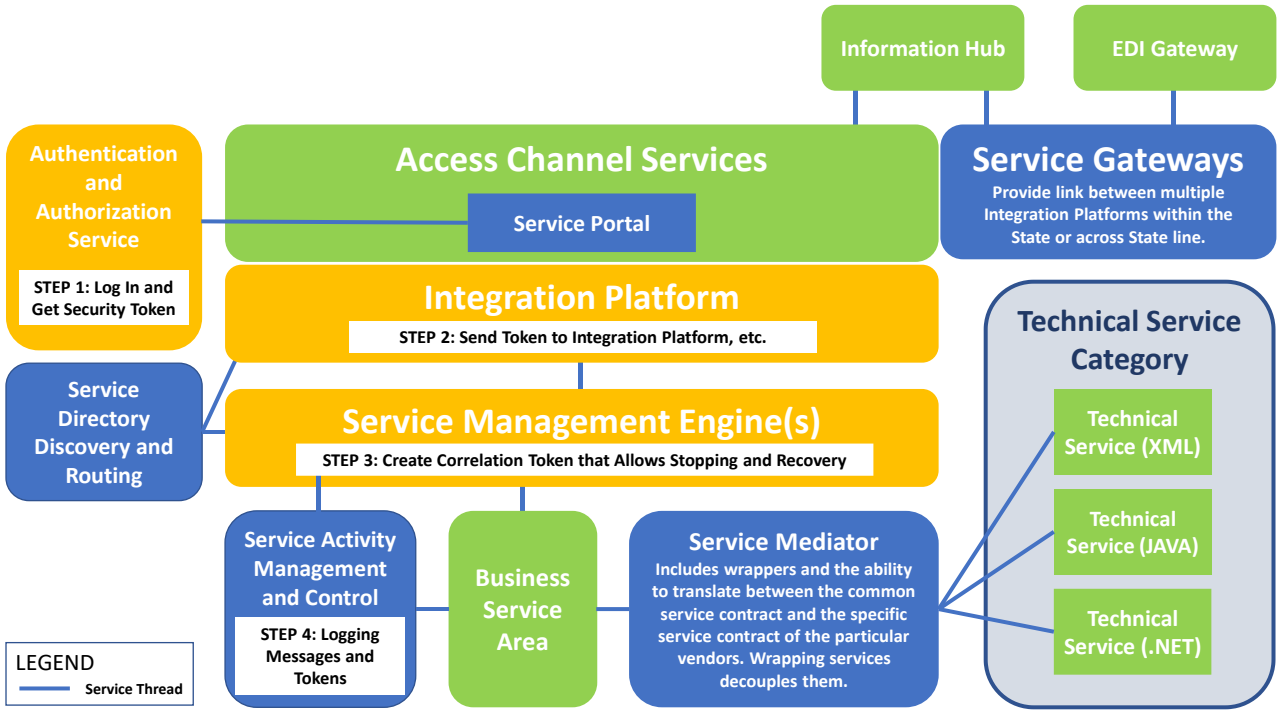

Exhibit 5-16: Service Infrastructure's Operation Concept of Security

5.4.7.2 APPLICATION ARCHITECTURE SECURITY STANDARDS

The SEAS Vendor will also leverage Security and Privacy standards, processes, and procedures in the *T-8: Enterprise Data Security Plan* deliverable document located in the FXPR at FX Hub > Standards & Plans > Category: Technology > Enterprise Data Security Plan (T-8).

5.5 SERVICE INVOCATION AND EXECUTION

Service Invocation is the process of requesting a service for execution. There are three ways to invoke a service within the defined architecture.

- **Invoke Service, Authenticate, and Correlate** – This type of invocation initiates from a user request, typically through a portal or some other access channel. Upon sign-on, the system authenticates the user and relates their user role to a given service before invocation occurs.
- **Invoke from External Message, Invoke Service, and Correlate** – This type of invocation initiates based on an external event (e.g., receipt of a message, receipt or transmission of a file, data change event) to perform a set of services in a more automated fashion, like batch EDI submissions. Designers have more flexibility when configuring these types of invocations as they can configure them based on a stakeholder's performance needs.

- **Execute Business Process Based on Service Mode and Engine** – This type of invocation is interactions between the business services themselves. This is the most prevalent type of execution in a Service Oriented Architecture.

Service Execution is the process to execute a service after a request invokes the service. The IS/IP plays a vital role in the execution of services. Here are some of the more common steps in execution of a service:

- **Step 1: Receive Message and Tokens, Route, and Manage** – The ESB receives a message, relates a token to the service type, and attaches a Security and Privacy token to the message, and routes the message to the Service Management Engine.
- **Step 2: Track Performance Limits and Fault Handling** – The ESB ensures the service follows performance polices and fault handling issues to maintain the quality of service levels.
- **Step 3: Route Message and tokens to Business Services** – The Service Management Engine routes the message and tokens to the appropriate business service.
- **Step 4: Route Response Message and Tokens** – If the business service generates a response upon completion, the ESB routes it to the appropriate service based on a predefined orchestration.

SECTION 6 TECHNICAL CAPABILITY MATRIX

The Technical Capability Matrix (TCM) is a part of the MITA framework used to define the technical capabilities of the enterprise at a specified MITA maturity level. The TCM is a key tool for conducting the State Self-Assessment (SS-A). The SS-A defines current state Medicaid Enterprise technical capabilities and develops the targeted future state of the enterprise with defined capabilities and performance standards.

The Technical Capability Matrix (TCM) provides the Technical Service Area (TSA) and the relevant Technical Service Classifications (TSC) groups with technical capability maturity statements across five (5) levels of maturity as shown in **Exhibit 6-1: The MITA Technical Capability Matrix Framework**.

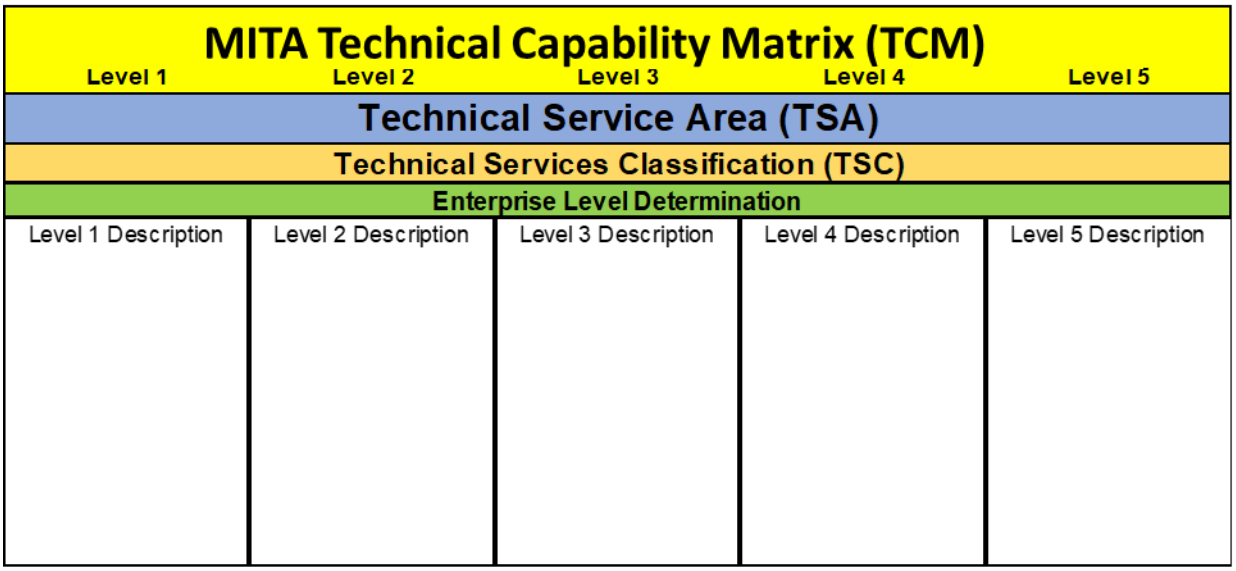

Exhibit 6-1: The MITA Technical Capability Matrix Framework

A technical capability defines a technical function at a specific MITA maturity level. Technical Architecture (TA) assigns technical capabilities to a maturity level based on the maturity level of business usage they enable. The TCM uses the following:

- **TSA** – The TSA is a technical tier that supports the secure construction, exchange, and delivery of Service Components.
- **TSC** – The TSC is a lower-level classification comprised of one or more service standards.

The MITA Framework gives direction for a basic three-tier performance monitoring structure that applies to the Technical Capability Matrix (TCM). **Note:** This also applies to the Business Process Template (BPT) and Business Capability Matrix (BCM). The measurement categories include:

- **Performance Standard** – The performance threshold(s), requirement(s), or expectation(s) that CMS expects the state to meet to evaluate at a level of performance.
- **Performance Measure** – Performance Measure is an element based on established Performance Standards and tracks past, present, and future business activity.
- **Performance Metric** – The Performance Metric measures a specific aspect of performance. A performance metric is also referred to as a Key Performance Indicator (KPI). Effective performance metrics usually focus on outcomes versus activity to encourage improvement, effectiveness, and appropriate control levels.

The path below leads to an example of the Florida specific Technical Capability Matrix (TCM), which was updated as part of the 2018 SS-A.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment E TCM 2018 SSA Example

6.1 FX Project Level Technical Capability Matrix (PLTCM)

FX seeks to improve the processes for collecting maturity information used to produce the SS-A. The current process for creating the SS-A TCM is labor intensive as it requires many sessions with the system experts to reevaluate all as-is and to-be capabilities. To lessen the impact on these experts and reduce the chance of missing enhancements made since the last SS-A, the FX seeks to use event driven collection of capability maturity data. The approach is to track the impact on capability maturity at the time of, by the people involved and in the context of FX projects that affect capabilities. This approach collects information updates to the as-is or to-be capabilities for involved participants at the time when projects are being defined and implemented. The As-Is capabilities are maintained in the SS-A, the PLTCM tool only captures To-Be maturity levels after a Project is implemented.

To document changes in the maturity of capabilities in the SS-A TCM, the SEAS Vendor and FX Project Teams will use the FX Project Level Technical Capability Matrix (PLTCM) to document how FX Project changes affect the SS-A TCM. The PLTCM will be developed at the direction of the Agency. The FX PLTCM will be a custom SharePoint list in the FXPR used to capture changes made by FX projects that will affect the SS-A TCM content. The data in this repository will be an input to periodically produce the SS-A. By collecting the technical capability enhancements in a more real-time fashion this allows for the following benefits:

- **Timeliness** – Capturing the enhancements as they happen
- **Quality** – Technical experts are defining the enhancements while they are fresh in their mind
- **Accuracy** – By mandating this as a FX Project requirement reduces the chance of missing capability maturity changes
- **Preparedness** – Project Teams can easily generate the SS-A TCM from PLTCM data
- **Reduced Cost** – Eliminates time consuming discovery sessions with the technical experts, thus shortening the SS-A TCM development life cycle

- **Reduced Dependency on Staff** – Capability impact knowledge is retained when there is staff turnover

6.2 PROJECT LEVEL TECHNICAL CAPABILITY MATRIX DEVELOPMENT

Exhibit 6-2: Project Level Technical Capability Matrix Workflow Process illustrates the steps required to create a PLTCM entry.

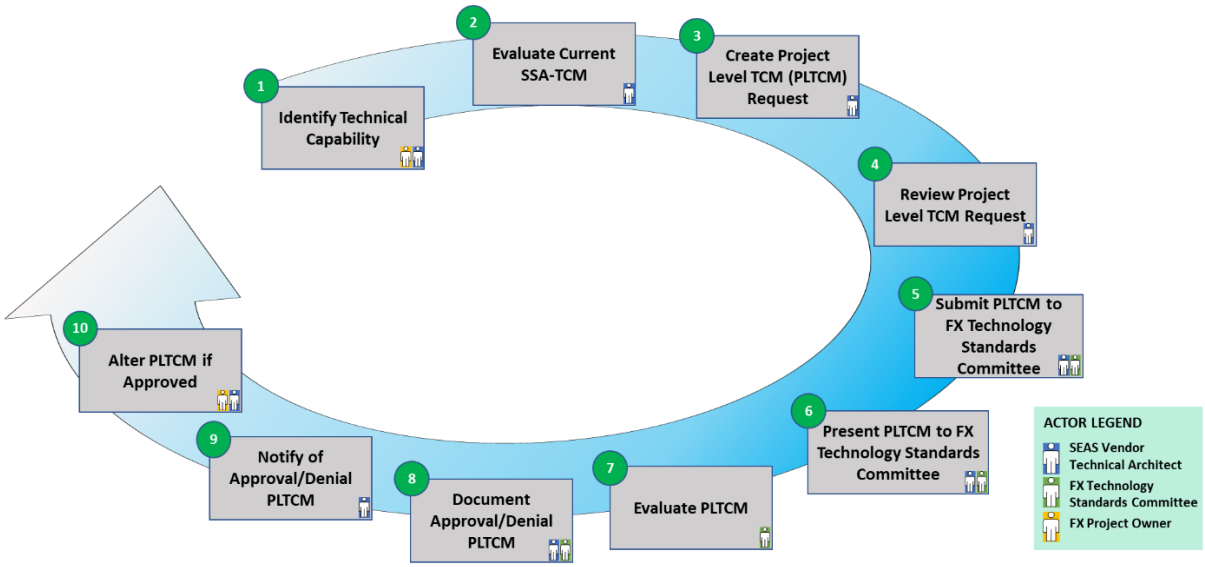

Exhibit 6-2: Project Level Technical Capability Matrix Workflow Process

Follow the steps below to create a PLTCM:

1. **Identify Technical Capability** – The FX Project Owner in coordination with the SEAS Vendor Technical Architect will identify the Technical Capability affected by the FX Project.
2. **Evaluate the Current SS-A TCM** – Based on findings, the SEAS Vendor Technical Architect evaluates the current and projected SS-A TCM maturity of technical capabilities and how they are impacted.
3. **Create PLTCM Request** – The SEAS Vendor completes the PLTCM Request form and adds all the research details.
4. **Review PLTCM Request** – The SEAS Vendor works with the FX Project Owner to confirm and add further details to the form if needed.
5. **Submit PLTCM to FX Technology Standards Committee** – The SEAS Technical Architect submits the PLTCM to the FX Technology Standards Committee and then schedules a review.
6. **Present PLTCM to FX Technology Standards Committee** – The SEAS Technical Architect presents the PLTCM to the FX Technology Standards Committee.

7. **Evaluate PLTCM** – The FX Technology Standards Committee reviews the PLTCM and determines the next action. Possible next actions are to approve to move forward, complete denial, or request changes, and resubmission of the PLTCM request.
8. **Document Approval/Denial of PLTCM Entry** – The SEAS Vendor documents the results of the FX Technology Standards Committee's evaluation.
9. **Notification of Approval/Denial for PLTCM Entry** – The SEAS Technical Architect communicates the PLTCM entry to FX Project stakeholders (procurement team members, FX Project Owner if / when selected, IV&V Vendor, IS/IP Vendor).
10. **Alter PLTCM Request** – If necessary, the FX Project Owner and SEAS Technical Architect return to step 1 and update the PLTCM entry to reflect additional technology capabilities affected and changes in projected or actual to-be capability maturity resulting from the FX Project.

6.3 FX Project Level Technical Capability Matrix Maintenance Procedures

Attachment F – How to Maintain the PLTCM List is a Word document that describes the procedures to maintain content in the Project Level Technical Capability Matrix List.

The path below leads to the current How to Maintain the PLTCM List document, reflecting any updates since the submission of this deliverable.

FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5) > Attachment F How to Maintain the PLTCM List

SECTION 7 MODULE SPECIFIC SERVICES

This section describes scenarios to manage the use of module specific services embedded in commercial system offerings and module specific services used exclusively for a specific business area.

7.1 SERVICES EMBEDDED IN SYSTEM COMMERCIAL OFFERINGS

All types of services (business, business rule, technical, data access) and APIs described in previous sections exist in systems provided as commercial offerings. Systems provided as commercial offerings include Commercial-off-the-Shelf (COTS) software, Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and other Everything-as-a-Service offerings. Services provided as commercial offerings frequently have the following characteristics:

- A vendor owns, develops and maintains the service(s)
- Service(s) contain vendor proprietary processing logic
- Service(s) use vendor proprietary data stores
- Service(s) use external proprietary data services
- Multiple consumers of the vendor use the service(s)
- Vendor controls and coordinates updates and changes
- Agency has minimal influence in service functionality, evolution or system change control

The inherent vendor responsibility and control in these types of services is both a benefit and a risk. This section provides guidance for the use of services embedded in commercial offerings to maximize benefits and minimize risk to the FX.

7.1.1 GENERAL GUIDANCE ON USE OF SERVICES AND APIS IN COMMERCIAL SYSTEMS

Below is general guidance on use of proprietary Services and APIs embedded in commercial system offerings:

- Use the IS/IP integration platform to connect to any proprietary Services or APIs. Avoid direct application calls to proprietary Services or APIs.
- Verify proprietary Services and APIs include user authentication logic and encryption processing in conjunction with the IS/IP integration platform.
- FX projects are encouraged to consider and adhere to the following recommendations related to the use of Services and APIs embedded in commercial system offerings.
- Use open industry standard services and APIs, if available from commercial system offerings.
- Use open source alternatives and wrappers to vendor specific services and APIs based on the projected risk of migration to an alternate commercial system offering.

- If open industry standards have not matured or dominant market vendors are not offering their services via open industry standards, it is acceptable to use Services and APIs of vendors that have dominant market share and sustainable market share dominance.
- Manage risk and vendor dependence by monitoring use of proprietary services and APIs.
- If possible, negotiate contractual protections to ensure continued use and vendor capacity to provide proprietary Services and APIs.
- Develop a plan for contingency services and APIs to be used if the proprietary Services and APIs are not usable.
- Minimize use of proprietary Services and APIs that act as exclusive access channels to Agency owned data.
- Monitor and comply with vendor terms of use for proprietary Services and APIs.

7.1.2 GUIDANCE ON PRIVATE / PUBLIC SERVICES EMBEDDED IN COMMERCIAL SYSTEMS

Services and APIs are normally classified as private or public (also called internal and external). Private or internal services or APIs are used to provide processing and access data within a set of systems in an organizational boundary. Public or external services and APIs are used to provide processing access to data for external parties to systems outside an organization. Typically, the majority of transactions occur using private or internal services. These services are convenient for internal developers to use to access information in a business area.

7.1.2.1 PRIVATE / INTERNAL SERVICES AND APIS

Below is guidance on use of proprietary private Services and APIs embedded in commercial system offerings:

- Minimize use of proprietary private and internal services and APIs that are subject to uncommunicated vendor-initiated change
- Restrict use of proprietary private and internal services and APIs to use primarily within the processing module
- Validate proprietary private and internal services and APIs have sufficient and adequate access controls, logging, and data protections

7.1.2.2 PUBLIC / EXTERNAL SERVICES AND APIS

Below is guidance on use of proprietary public Services and APIs embedded in commercial system offerings:

- Confirm the Service or API does not contain software bugs.
- Confirm the Service or API performs adequately.
- Confirm the Service or API is secure.

- Confirm the Service or API does not leak any private data.
- Confirm the Service or API does not use custom data formats or proprietary protocols.

7.2 Business Area Specific Services

This section provides guidance on services that are designed uniquely for use in system processing related to a specific business area. There is an obvious cost and time tradeoff in creating robust industrial public Services and APIs versus tactical specialized Services and APIs that are optimized for specific processing in a specific business area. It is often easier for designers and developers to assume Services and APIs are specialized with one purpose supporting a single use case. When additional use cases emerge, the designers will confront the challenge of rework to a previously developed Service or API or creation of another specialized business area specific Service or API.

General guidance for business area specific Services and APIs includes:

- Perform due diligence that business area specific services are indeed unique and there is minimal chance of expanded use or access
- Use standard enterprise integration patterns
- Use enterprise information exchange package definitions using enterprise attribute names and formats

SECTION 8 APPENDICES

Attachments A-H referenced below are located in the FX Repository at FX Hub > Standards & Plans > Category: Technology > Technical Architecture Documentation (T-5).

ATTACHMENT A - BUSINESS SERVICE REQUEST FORM

Attachment A – Business Service Request Form is a PDF document used to define and maintain business service entries and content in the Application Service Registry.

ATTACHMENT A - BUSINESS SERVICE REQUEST FORM EXAMPLE

Attachment A – Business Service Request Form Example is a PDF document that provides an example of how to complete the form.

ATTACHMENT B - DATA SERVICE REQUEST FORM

Attachment B – Data Service Request Form is a PDF document used to define and maintain data service entries and content in the Application Service Registry.

ATTACHMENT B - DATA SERVICE REQUEST FORM EXAMPLE

Attachment B – Data Service Request Form Example is a PDF document that provides an example of how to complete the form.

ATTACHMENT C - BUSINESS RULE SERVICE REQUEST FORM

Attachment C – Business Rule Service Request Form is a PDF document used to define and maintain business rule service entries and content in the Application Service Registry.

ATTACHMENT C - BUSINESS RULE SERVICE REQUEST FORM EXAMPLE

Attachment C – Business Rule Service Request Form Example is a PDF document that provides an example of how to complete the form.

ATTACHMENT D - TECHNICAL SERVICE REQUEST FORM

Attachment D – Technical Service Request Form is a PDF document used to define and maintain technical service entries and content in the Application Service Registry.

ATTACHMENT D - TECHNICAL SERVICE REQUEST FORM EXAMPLE

Attachment D – Technical Service Request Form Example is a PDF document which provides an example of how to complete the form.

ATTACHMENT E - 2018 SS-A TECHNICAL CAPABILITIES MATRIX EXAMPLE

Attachment E – 2018 SS-A Technical Capability Matrix Example is an Excel document that provides the Technical Capability Matrix updated by the 2018 SS-A.

ATTACHMENT F - HOW TO MAINTAIN THE PLTCM LIST

Attachment F – How to Maintain the PLTCM List is a Word document that describes the procedures to maintain content in the Project Level Technical Capability Matrix.

ATTACHMENT G - HOW TO MAINTAIN THE ASR LIST

Attachment G – How to Maintain the ASR List is a Word document that describes the procedures to maintain content in the Application Service Registry.

ATTACHMENT H - PROJECT LEVEL TECHNICAL CAPABILITIES MATRIX REQUEST FORM

Attachment H – Project Level Technical Capability Matrix Request Form is a PDF document used to define and maintain entries and content in the PLTCM.

REFERENCE TO OTHER DELIVERABLE DOCUMENTS

The deliverable referenced below is located in the FXPR at FX Hub > Standards & Plans > Category: Technology > Enterprise Data Security Plan (T-8).

The *T-8: Enterprise Data Security Plan* deliverable is a Word document that describes the Enterprise Data Security Plan for the FX Enterprise.