KAISER PERMANENTE®

Enterprise Data Access

Architecture Patterns Guide

Enterprise Architecture

Chief Technology Office

Kaiser Permanente

Document Version:	v1.4
Date:	2016-06-15
Author:	Bobby Thomas
	Chris H. Lee

Table of Contents

1 Introduction 3
2 Data Source Selection 3
2.1 Data Access Patterns Guide 3
2.2 Data Directly from Source 4
2.3 Data from Master Data Management (MDM) 5
2.4 Data through Operational Data Store (ODS) 7
3 Example Scenarios for Data Access 9
3.1 Sourcing Data for Personalization 9
3.2 Data Flow In and Out of Landing Zone (Big Data) 12
3.3 Enterprise Master Patient Index (EMPI) 15
4 Additional Related Topics 16
4.1 Mixed Storage (Polyglot Persistence) 16
4.2 Data Masking for Loading Data into Public Cloud 17
4.3 Pre-computed Aggregates 17
5 Appendix 18
5.1 Related Documents 18
5.2 Glossary 18
5.2.1 Analytical Data 18
5.2.2 Business-Oriented Data Landscape 18
5.2.3 Cross-Reference Data 18
5.2.4 Enterprise Information Model 18
5.2.5 Master Data 19
5.2.6 Master Data Management (MDM) 19
5.2.7 Metadata 19
5.2.8 Operational Data Store 19
5.2.9 Reference Data 19
5.2.10 System of Engagement (SOE) 19
5.2.11 System of Intelligence 20
5.2.12 System of Origin (SOO) 20
5.2.13 System of Record (SOR) 20
5.2.14 Transactional Data 20
5.3 Contributors 21

Legal Notices

1 Introduction

This document goes into detail around choosing a source system for data as well as recommendations for special scenarios.

This document is intended to help inform solution engineers and consultants to make the best solution decisions in their own projects.

The Enterprise Data Reporting Stores Architecture Patterns Guide is a recommended pre-read for this document. The aforementioned guide introduces basic concepts around data stores such as data marts, data warehouses, and operational data stores. We recommend users first familiarize themselves with the above document before covering more details this document.

2 Data Source Selection

This selection guides the solution designer in choosing how to reach needed data.

2.1 Data Access Patterns Guide

Given that the same data is available in the originating system as well as in other systems such as a Meta Data Management (MDM) system or an Operational Data Store (ODS), the table below outlines some decision factors.

	DATA ACCESS PATTERN		
Decision Factor	Data Directly from Source	Data from MDM	Data through ODS
Real-time data	✓		
Authoritative reference		✓	
Multiple source feeds joined		✓	✓
Source abstracted from consumer		✓	✓
Further transformed data			✓
Flexible deployment / colocation			✓

The next sections detail criteria and considerations for each pattern.

2.2 Data Directly from Source

This pattern forgoes intermediate data stores and goes directly to data-originating system, often referred to as the System of Record (SoR). The SoR may provide web service APIs or a data mart for data consumers.

PATTERN ATTRIBUTE	CONSIDERATIONS
Data Type	• Original data typically following the source system's data model
Data Freshness	• Consuming applications require most current information for all operations
Business Complexity	• SoR contains business logic that may be challenging to replicate in an MDM/ODS.
Security Management	• Security policies may prohibit replication of data • SoR may be accessed in ways conforming to compliance guidelines
Trade Offs	• Increased load on source system • Increased security management complexity on source system • Networking may introduce latency impact for non-colocated use • Limited API's might lead to chatty sessions to support aggregating data
Advantages	• May be considered for projects with tight timelines and low adoption targets • May be used to implement prototypes and stage larger investments
Related Patterns	• Cache-aside pattern may be used to complement this pattern
KP Implementations	• EPIC Web Services

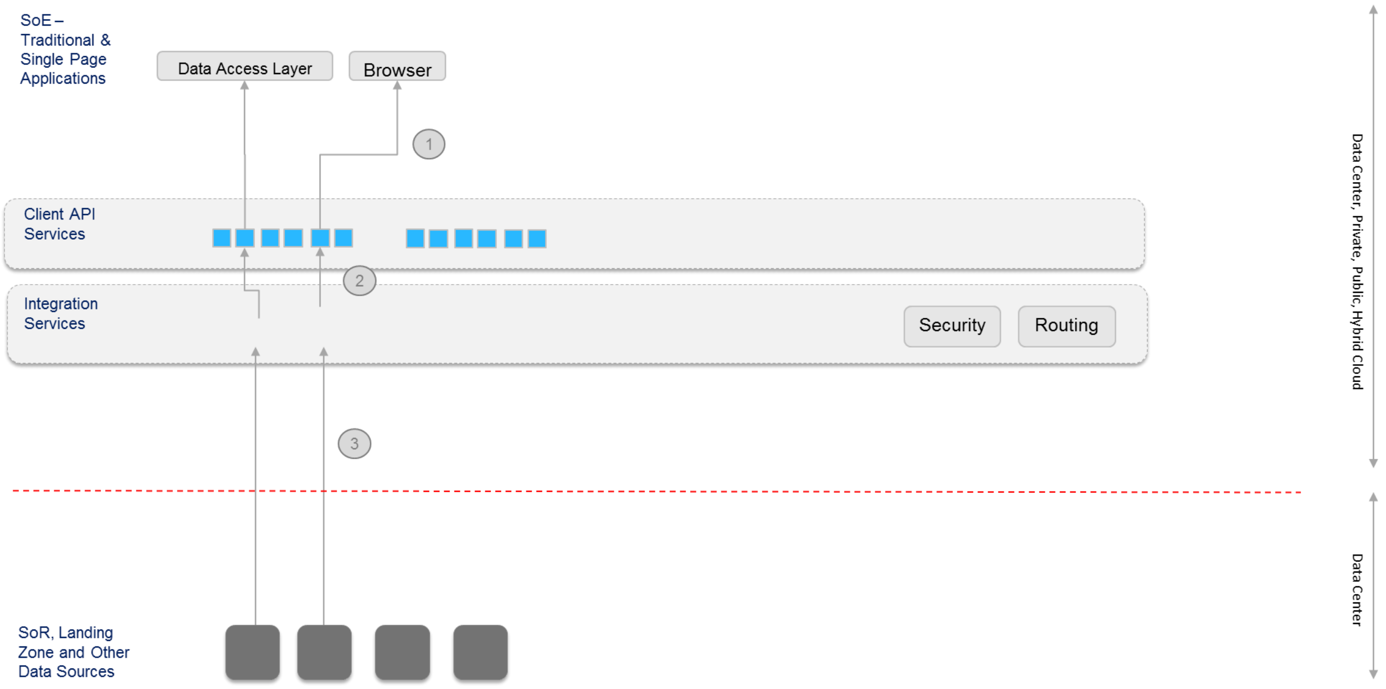

SoE – Traditional & Single Page Applications
Data Access Layer
Browser
1
Client API Services
2
Integration Services
Security
Routing
3
SoR, Landing Zone and Other Data Sources
Data Center, Private, Public, Hybrid Cloud
Data Center

Figure 2: Direct access to SoR

Diagram Sequence:

1. Browser or Data Access Layer used to invoke a Client API
2. The Client API layer in turn looks up the Integration Services end point
3. The Integration layer invokes the SoR API and retrieves information

2.3 Data from Master Data Management (MDM)

This pattern takes advantage of an MDM system that contains data that has gone through data quality checks, curation, and validation. The MDM system may be considered the authoritative source for such data.

PATTERN ATTRIBUTE	CONSIDERATIONS
Data Type	• Shared business entities used across functions • Data already prepared, aggregated, de-duplicated • May also include cross-references to other systems.
Data Freshness	• Mixed requirements for consistency across the enterprise. Some applications can operate with the eventual consistency paradigm, while others require high level data consistency for mission critical operations • Applications require access to most current and relevant information • Data policies and processes in place to resolve operational issues including data conflicts and inconsistencies in a timely manner
Business Complexity	• MDM system provides highest consistency and harmonization across the enterprise by centralizing and defining authoritative data records • Data requires standardization, matching, de-duplication, identity & semantic resolution and correlation across systems and organizations
Security Management	• Sensitive data may require to protections in consumer data stores
Trade Offs	• May require special consideration to support applications that are not co-located • May require additional security measures to support cloud-based deployments
Advantages	• Enterprise-wide standard approach to ensure consistency of master data • Supports multiple implementation styles to meet diverse needs and scenarios • Can be supplemented with other patterns to support enterprise needs
Related Patterns	• Cache patterns (see Data Caching document referenced in the Appendix) can be used to complement this pattern • Coupled with SoR systems to extend data access across the enterprise
KP Implementations	• IBM Initiate • EPP

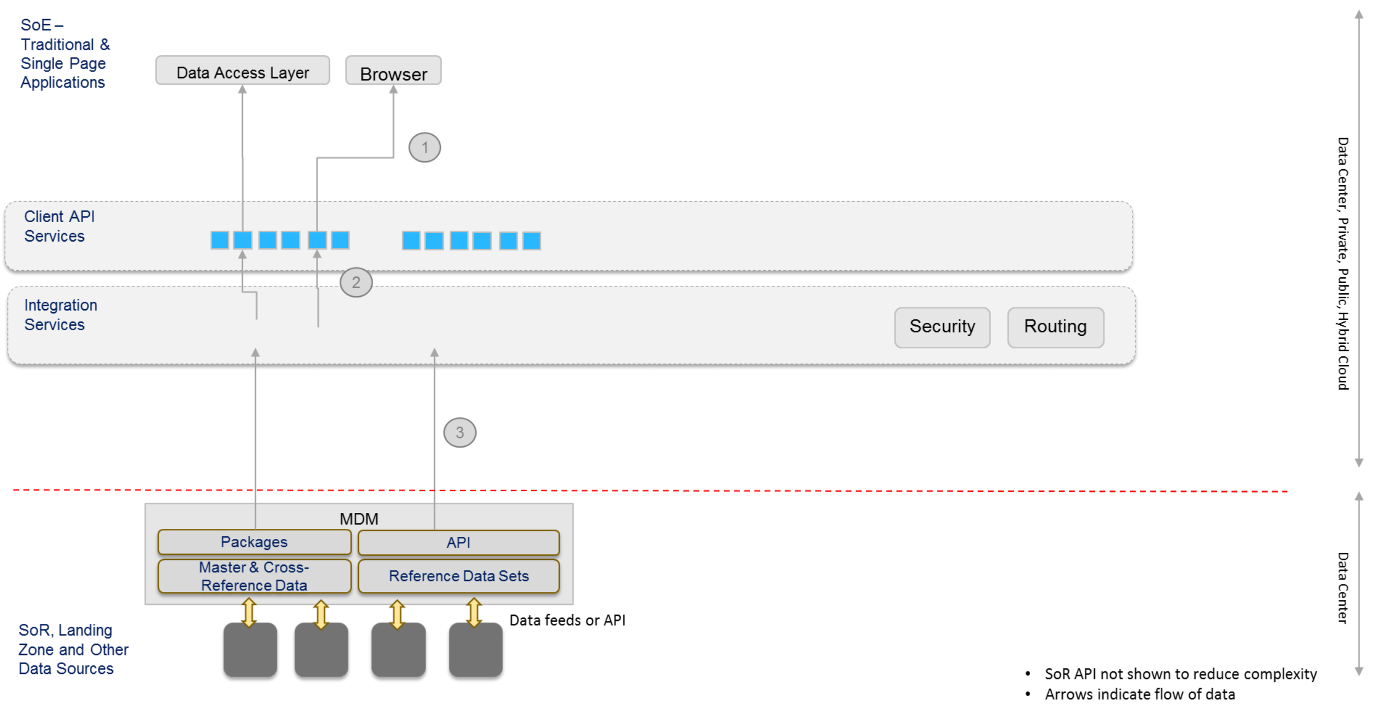

SoE – Traditional & Single Page Applications
Data Access Layer
Browser
1
Client API Services
2
Integration Services
Security
Routing
3
MDM
Packages
API
Master & Cross-Reference Data
Reference Data Sets
Data feeds or API
SoR, Landing Zone and Other Data Sources
SoR API not shown to reduce complexity
Arrows indicate flow of data
Data Center, Private, Public, Hybrid Cloud
Data Center

Figure 3: Accessing data in MDM

Diagram Sequence:

1. Browser or Data Access Layer used to invoke a Client API
2. The Client API layer in turn looks up the Integration Services end point
3. The Integration layer invokes the MDM API and retrieves information. The MDM system is fed data from SoR and other data sources. Either batch pulls by the MDM or event feeds from the SoR keep the data synchronized between the MDM application and SoR. The frequency of feeds varies across systems.

2.4 Data through Operational Data Store (ODS)

Data may also be staged in an intermediate relational database with an ODS. The ODS may combine multiple sources and be located close to the consumer application. Low-latency data transfer technologies may also be used to refresh the ODS.

PATTERN ATTRIBUTE	CONSIDERATIONS
Data Type	• Aggregated, translated, and cross-referenced datasets from multiple sources • Common business data that may be used across functions
Data Freshness	• Data mostly static and applications can tolerate synchronization latency • Data changes are scheduled and have a formal release cycle • Data policies and processes in place to resolve operational issues including data conflicts and inconsistencies in a timely manner
Business Complexity	• Data typically enriched and transformed upon identifiers and code sets
Security Management	• Read only access to static information / datasets • PHI and Sensitive information may need additional security protection
Trade Offs	• May result in data replicated across multiple stores • In the absence of an MDM solution, rules for data consistency and quality are replicated across applications, potentially leading to operational challenges • Data synchronization via batch ETL introduces significant latency • Usage should be limited to accessing and staging caches of lookup data
Advantages	• May have performance advantages over existing options for data sources • Can be used to implement prototypes and stage larger investments
Related Patterns	• Cache patterns can be used to complement this pattern • Coupled with SoR systems to extend data access across the enterprise
KP Implementations	• KP.org WPP database

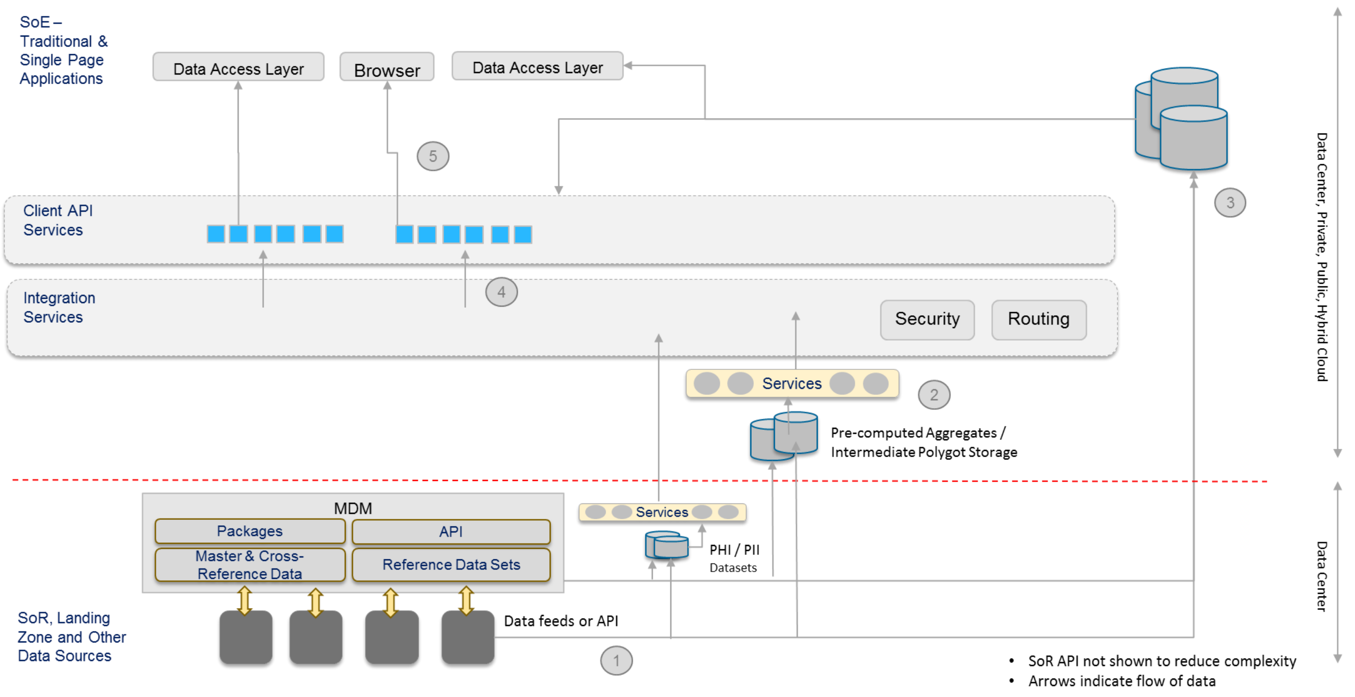

SoE – Traditional & Single Page Applications
Data Access Layer
Browser
Data Access Layer
5
Client API Services
3
Integration Services
4
Security
Routing
Services
2
Pre-computed Aggregates / Intermediate Polygot Storage
Data Center, Private, Public, Hybrid Cloud
MDM
Packages
API
Master & Cross-Reference Data
Reference Data Sets
Services
PHI / PII Datasets
Data Center
SoR, Landing Zone and Other Data Sources
Data feeds or API
1
SoR API not shown to reduce complexity
Arrows indicate flow of data

Figure 4: Staging data in an ODS for an application

<u>Diagram Sequence:</u>

1. The MDM system is fed data from SoR and other data sources either batch or events keeping the data in synch between consumer application and SoR. The frequency of feeds varies across systems. The Data from MDM or from SoR is used to feed an ODS. It can also be used to store data in a NoSQL kind of repository. The frequency of these feeds vary.
2. The staged data is exposed as Services
3. The staged data / ODS can also be exposed directly using Data Services
4. The Client API or the Integration layer invokes these data or business services as appropriate
5. The Client API is in turn invoked directly using a Browser (SPA) or via a Data Access layer.
6. PHI/PII related information which needs to securely housed can be loaded into a staged ODS.

3 Example Scenarios for Data Access

This section addressed particular scenarios when applications require interacting with additional systems for data.

Many of the examples and diagrams refer to Systems of Engagement, which include customer-facing applications hosted on the cloud or on-premise.

3.1 Sourcing Data for Personalization

Accessing Profile and Personalization (PnP) data is usually implemented via an MDM data access pattern. The MDM implementation should be the only source of the “golden” profile. Here are the patterns recommended:

1. The general guidance is to centralize access and host and serve PnP from the MDM for both reads and writes.
2. Pre-packaged applications or performance requirements might require that the application itself contain and manage parts / partial pieces of PnP in a local store. Consequently, the recommendation is to leverage MDM tooling to synchronize data, resolve conflicts, and create a “golden” record. This pattern can also be implemented using a read-through / write-through cache pattern. Cache-aside patterns may also be applicable.
3. Data may also be served via scheduled batch-based ETL. While necessary in some cases, this is the least preferred option since it introduces latency in data refreshes.

	Criteria / Considerations
General Characteristics	• Frequent read and write of data managed via MDM implementation • MDM receives data from multiple source applications for parts of PnP data • High levels of data quality and consistency required • Privacy protection required
Data Freshness	• Frequent data updates
Trade Offs	• Changes to PnP structure need to be coordinated with MDM updates • High demand of PnP data requires scaling up MDM • Use of ETL introduces latency
Advantages	• High data quality through MDM
Related Patterns	• Cache patterns can be used to complement this pattern
KP Example	• EPP

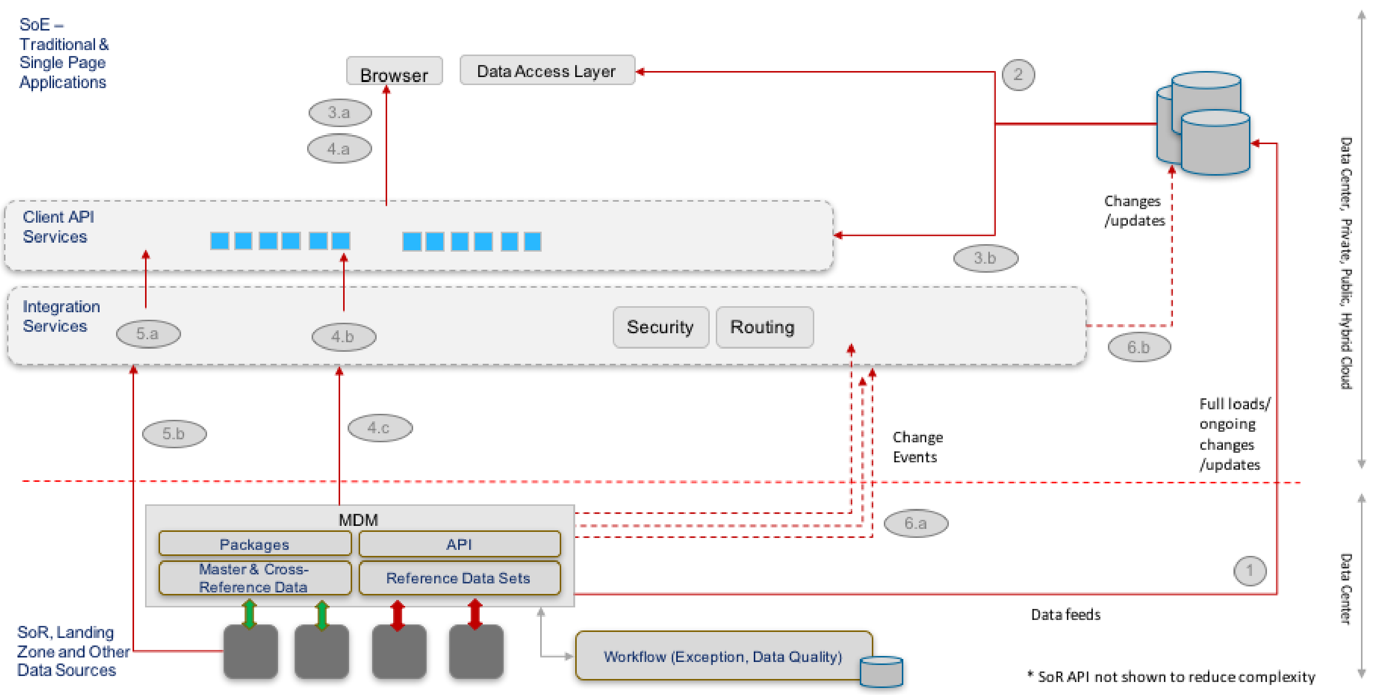

SoE – Traditional & Single Page Applications
Browser
Data Access Layer
2
3.a
4.a
Client API Services
3.b
Changes /updates
Integration Services
5.a
4.b
Security
Routing
6.b
Data Center, Private, Public, Hybrid Cloud
5.b
4.c
Change Events
Full loads/ ongoing changes /updates
MDM
Packages
API
Master & Cross-Reference Data
Reference Data Sets
6.a
1
Data Center
Data feeds
SoR, Landing Zone and Other Data Sources
Workflow (Exception, Data Quality)
* SoR API not shown to reduce complexity

Figure 5: Access PnP information from SoE

Diagram Sequence:

1. The MDM system is fed data from various SoRs and other data sources either batch or events keeping the data in synch between SoE and SoR. The frequency of feeds varies across systems. There are workflows to manage data quality and consistency within the MDM.
2. The staged data can also be exposed directly using Data Services. The Client API or the Data Services layers invokes these data services as appropriate. This access pattern is preferred if the MDM does not provide real time API's that are scalable.
3. The staged data can also be exposed directly using Business Services via a Browser (3a) and Client API layer (3b). This access pattern is preferred if the MDM does not provide real time API's that are scalable.
4. The Data from MDM can also be accessed via the Browser (4a), which is then routed through the Client API layer (4b) and then via the Integration Layer accesses the MDM API (4c). This access pattern requires the MDM system to meet all functional and non-functional needs of the accessing application.
5. In some cases, the underlying Application might contain information that is not housed in the MDM system or in the staged data. In these cases the Application can invoke the Client API (5a) and directly access the SoR API through the integration layer (5b). This access pattern requires the SoR system to meet all functional and non-functional needs of the accessing application.
6. Changes to the underlying data in SoR are synchronized with the MDM system. The MDM system also published changes using a Pub/Sub model (6a) which is then used to update the staging data (6b).

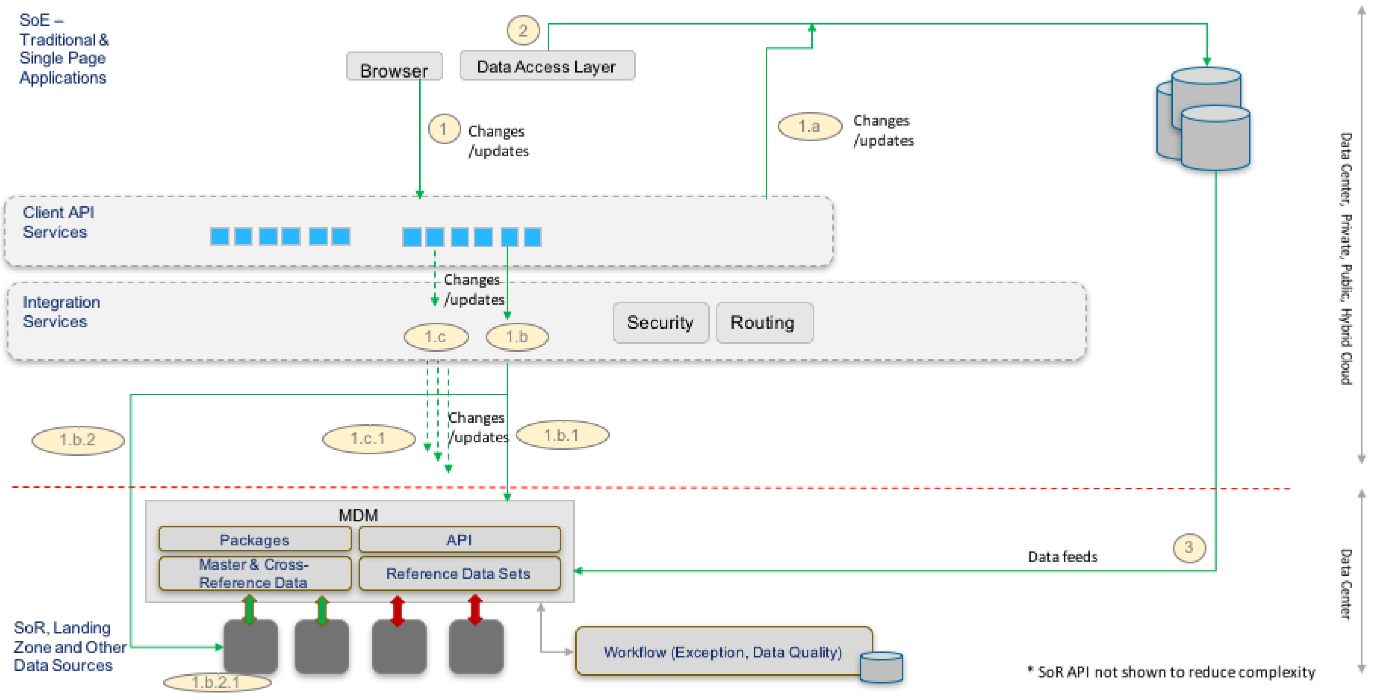


Figure 6: Updating PnP information from SoE

Diagram Sequence:

1. Update are made from the Client API services. These changes are propagated to the Integration services (1b) and then directly to the MDM system (1b-1) or the SoR (1b-2, 1b-2-1). The Client API can also updated the Staged data directly (1a). The changes from the application are / can be published to the MDM system using a Pub/Sub model (1c and 1c-1). The MDM system is synchronized with SoR in both cases to ensure data consistency. The frequency of synchronizations varies across systems. There are workflows to manage data quality and consistency within the MDM.
2. The staged data can also be exposed directly using Data Services. The Client API or the Data Services layers invokes these data services as appropriate. This access pattern is preferred if the MDM does not provide real time API's that are scalable.
3. The staged data can also be exposed directly using Business Services via a Browser (3a) and Client API layer (3b). This access pattern is preferred if the MDM does not provide real time API's that are scalable.
4. The Data from MDM can also be accessed via the Browser (4a), which is then routed through the Client API layer (4b) and then via the Integration Layer accesses the MDM API (4c). This access pattern requires the MDM system to meet all functional and non-functional needs of the accessing application.
5. In some cases, the underlying Application might contain information that is not housed in the MDM system or in the staged data. In these cases the Application can invoke the Client API (5a) and directly access the SoR API through the integration layer (5b). This access pattern requires the SoR system to meet all functional and non-functional needs of the accessing application.
6. Changes to the underlying data in SoR are synchronized with the MDM system. The MDM system also published changes using a Pub/Sub model (6a) which is then used to update the staging data (6b).

3.2 Data Flow In and Out of Landing Zone (Big Data)

The KP Landing Zone (LZ) uses Big Data / Hadoop technologies to ingest raw and produce consumable data. This section introduces some examples only; for more information please see the Landing Zone Reference Architecture document (linked in Appendix)

	Criteria / Considerations
General Characteristics	• High volume and variety of data. • Data may be loaded in real time. • Access to raw and refined data sourced from internal and external systems • Access to information for data analysis and discovery • Access to information for quick aggregation and correlation
Data Freshness	• High frequency of data changes • Business requires low latency access to raw information and changes • Business requires access to data that is more or less static and applications have a tolerance for latency in synchronizing data changes • Data policies and processes in place to resolve Operational issues - like data conflicts and inconsistencies - in a timely manner
Trade Offs	• May need to verify compatibility with consumer applications and analytical tools
Advantages	• Ability to process the largest datasets • Huge capacity and flexible capabilities enable sourcing information from across the enterprise
Related Patterns	• Cache patterns can be used to complement this pattern
KP Example	• Landing Zone (Big Data)

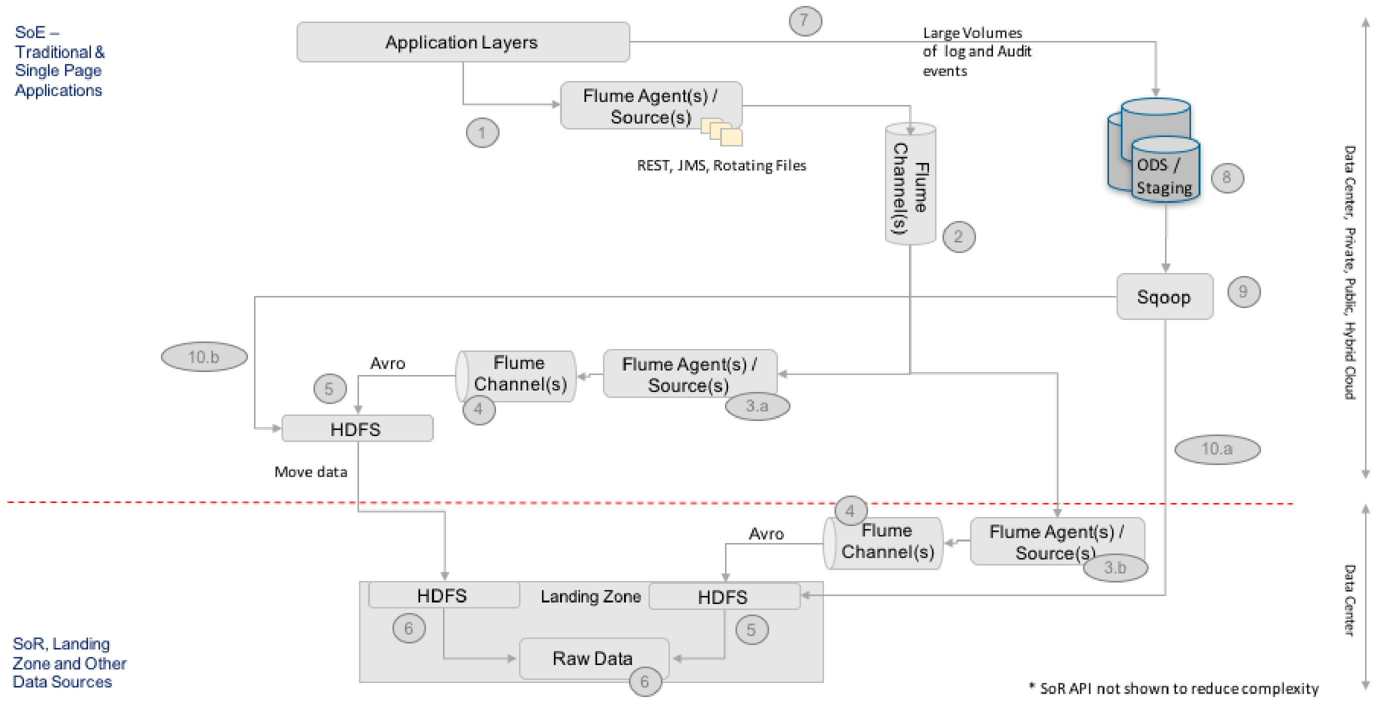

SoE – Traditional & Single Page Applications
Application Layers
7
Large Volumes of log and Audit events
Flume Agent(s) / Source(s)
1
REST, JMS, Rotating Files
Flume Channel(s)
2
ODS / Staging
8
Sqoop
9
10.b
Avro
Flume Channel(s)
Flume Agent(s) / Source(s)
5
4
3.a
HDFS
Move data
10.a
4
Avro
Flume Channel(s)
Flume Agent(s) / Source(s)
3.b
HDFS
Landing Zone
HDFS
6
5
Raw Data
6
SoR, Landing Zone and Other Data Sources
* SoR API not shown to reduce complexity
Data Center, Private, Public, Hybrid Cloud
Data Center

Figure 8: Loading Application Data into the LZ

Note: Flume is used to illustrate the pattern. Flume can be replaced with a technology approved by KP (ex: Splunk) as appropriate.

Diagram Sequence:

1. Data from Applications are fed to Flume Agents
2. Data is then fed into Flume Channels
3. Data then flows into
    a. PATH A : Flume Agents (3a) and Flume Channels (4b) to be stored as Avro files on HDFS (5)
    OR
    b. PATH B: Remote Flume Agents (3b), Flume Channels (4) and then stored to HDFS files (5).
4. Path A is preferred wen there are large amounts of data that need to cross networks. Path B tends to be very chatty.
5. Application data can also be stored into an ODS for fast writes (ex: Cassandra) – 7
6. The staged data can then be exported using Sqoop (9) and written to local HDFS files (10b) or written to a remote HDFS file store (10a). The first approach works best for large scale loads.
7. The local HDFS files and then moved to remote HDFS file storage (6) and then loaded into the Raw Data Landing Zone (6).

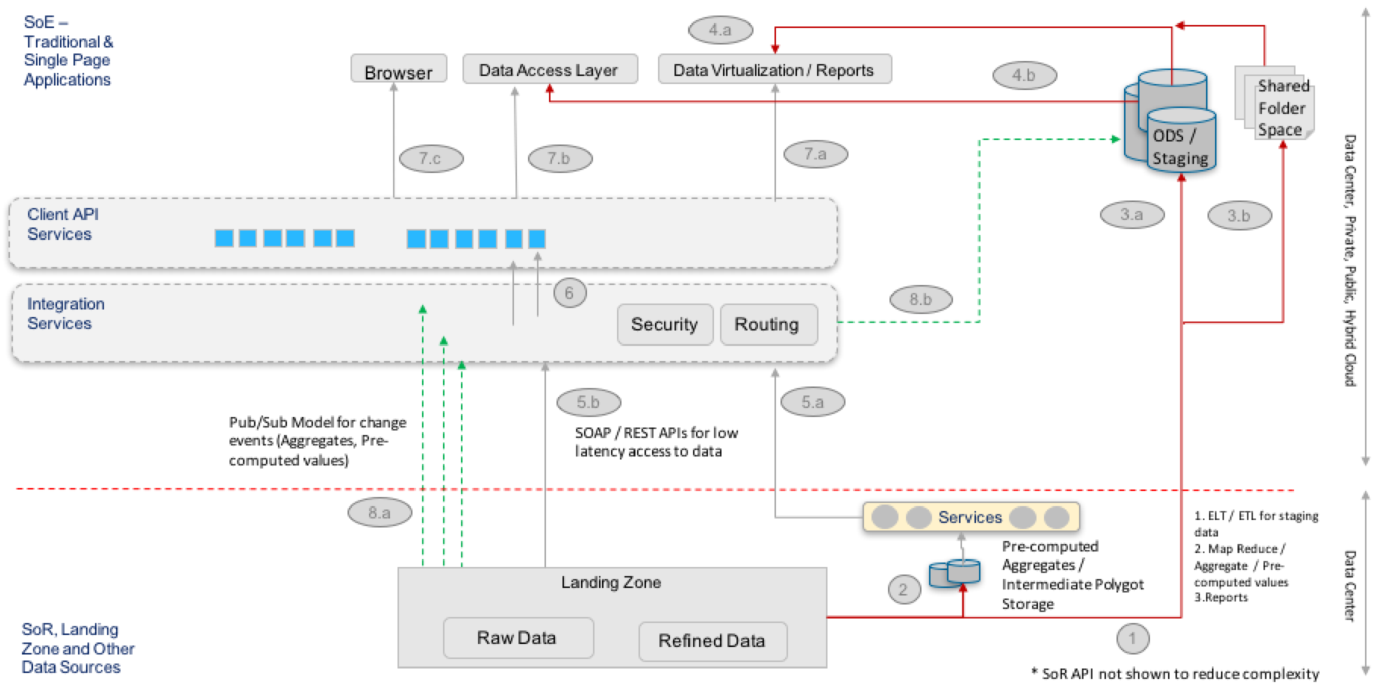

SoE – Traditional & Single Page Applications
Browser
Data Access Layer
Data Virtualization / Reports
4.a
4.b
Shared Folder Space
ODS / Staging
7.c
7.b
7.a
3.a
3.b
Client API Services
Integration Services
6
Security
Routing
8.b
Data Center, Private, Public, Hybrid Cloud
Pub/Sub Model for change events (Aggregates, Pre-computed values)
5.b
SOAP / REST APIs for low latency access to data
5.a
8.a
Services
Pre-computed Aggregates / Intermediate Polygot Storage
1. ELT / ETL for staging data
2. Map Reduce / Aggregate / Pre-computed values
3.Reports
Data Center
Landing Zone
Raw Data
Refined Data
2
1
SoR, Landing Zone and Other Data Sources
* SoR API not shown to reduce complexity

Figure 9: Loading Reporting Stores with LZ Data

Diagram Sequence:

1. Data from Landing Zone (Big Data) can be staged into a staging area.
2. Pre-computed aggregates can be created / loaded into intermediate polyglot storage.
3. The staged data can also be exposed directly using an ODS (3a) or via Shared folders (3b).
4. The staged data can also be exposed directly using a data virtualization / reporting mechanism (4a) or via a data access layer (4b).
5. Services can be built on top of pre-computed aggregates (5) or the raw data can be exposed via REST/SOAP services (5b) to the Integration layer.
6. These services can then be exposed to the client API layer (6).
7. The client APIs can be accessed via the browser (7c) or via a Data access layer (7b) or via a data virtualization layer / reporting layer (7a)

3.3 Enterprise Master Patient Index (EMPI)

While a detailed explanation and design of EMPI solution is outside the scope of this document, EMPI plays a critical role within a healthcare organization.

EMPI implementations are MDM solutions and should be regarded as such. Most implementations are registry style with focus on data cleansing, data standardization and conflict resolution workflows. CDR (Clinical Data Records) are usually not captured within the EMPI Implementation but pointers are used for in-direction. Duplicate MRN match and merge cases are explicitly handed within the EMPI system and downstream systems.

Beyond the core infrastructure within an organization, the EMPI takes on special meaning when used within an HIE environment; especially within an ACO setup. Both Federated EMPI and centralized EMPI solutions become relevant.

In order to abstract the complexities of the EMPI systems and the workflow; consuming applications should access the EMPI systems using the API's that are supported. Applications should be ready to implement logic to handle the following scenarios, details like IHG profiles are not detailed out here. The recommendation is to not extract data and store identity information in local stores; especially due to the asynchronous nature and volatility of data.

The focus of the tables below is limited to **Patient Identifier Cross Referencing** (PIX) and **Patient Demographics Query** (PDQ) only.

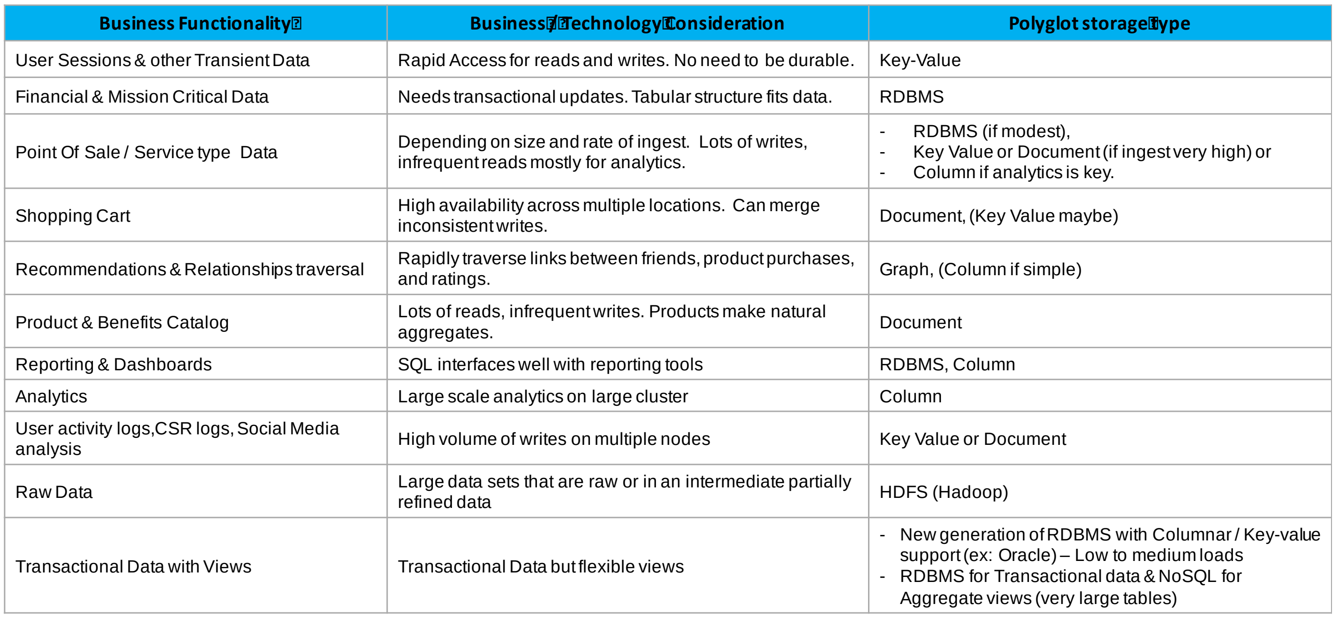
Action	Identity Algorithm	Result
Create Record	Deterministic	• New or existing EMPI ID
Resolve Identity	Deterministic / Probabilistic	• New or existing EMPI ID • Weighted likelihoods for EMPI IDs
Split Record	Deterministic	• Old and new EMPI IDs
Merge Record	Deterministic	• New EMPI ID for initial merge when no previous record exists • Existing EMPI ID for merging new data into an existing record

4 Additional Related Topics

4.1 Mixed Storage (Polyglot Persistence)

Polyglot persistence is the approach of using the best format suited to meet the needs of the consuming applications. Examples of different choices include SQL, graph, document, object, and file-based stores.

	Criteria / Considerations
General Characteristics	• Data aggregated and transformed from sources into intermediary formats for faster access and search • Data primarily used to pivot existing data sets for faster views • Different layers may be used as a fast “sink” to store data for later processing • May meet performance or storage needs that otherwise cannot be met by using one single storage layer
Data Freshness	• Source data store maintains master records; using polyglot persistence for staging is acceptable
Business Complexity	• Existing solutions do not meet QoS needs • Data is primarily used to pivot existing data sets for fast views • Is used as a fast “sink” to store data to be moved for subsequent usage
Trade Offs	• Not all technologies have full and mature ACID (Atomicity, Consistency, Isolation, Durability) implementations
Advantages	• Native scaling and performance (scale up and scale out) • Flexible storage and data models
Related Patterns	• See Pre-computed Aggregates
KP Example	• NoSQL store available in Bluemix SoE PaaS • In-memory store (Redis and eXtreme Scale) available in Bluemix SoE PaaS



Business Functionality	Business / Technology Consideration	Polyglot storage type
User Sessions & other Transient Data	Rapid Access for reads and writes. No need to be durable.	Key-Value
Financial & Mission Critical Data	Needs transactional updates. Tabular structure fits data.	RDBMS
Point Of Sale / Service type Data	Depending on size and rate of ingest. Lots of writes, infrequent reads mostly for analytics.	- RDBMS (if modest), - Key Value or Document (if ingest very high) or - Column if analytics is key.
Shopping Cart	High availability across multiple locations. Can merge inconsistent writes.	Document, (Key Value maybe)
Recommendations & Relationships traversal	Rapidly traverse links between friends, product purchases, and ratings.	Graph, (Column if simple)
Product & Benefits Catalog	Lots of reads, infrequent writes. Products make natural aggregates.	Document
Reporting & Dashboards	SQL interfaces well with reporting tools	RDBMS, Column
Analytics	Large scale analytics on large cluster	Column
User activity logs,CSR logs, Social Media analysis	High volume of writes on multiple nodes	Key Value or Document
Raw Data	Large data sets that are raw or in an intermediate partially refined data	HDFS (Hadoop)
Transactional Data with Views	Transactional Data but flexible views	- New generation of RDBMS with Columnar / Key-value support (ex: Oracle) – Low to medium loads - RDBMS for Transactional data & NoSQL for Aggregate views (very large tables)

Figure 7: Polyglot storage guide for SoE

4.2 Data Masking for Loading Data into Public Cloud

The private cloud data store contains critical and non-critical data. Often times, sensitive data needs to be shipped to a public cloud or to a 3rd party system; however, internal Identifiers, code sets, and similar sensitive data need to be especially protected when residing in external systems. Furthermore, transfer of sensitive data may be restricted by compliance regulations.

In these cases, the MDM platform is used to generate additional identifiers for the data and maintain cross-references. The logic implemented in the business layer is split into one requiring critical data and one, where critical data in **pseudonymized** form is sufficient for processing. Any correlation that is necessary occurs only within trusted boundaries and is strictly controlled. ID's should not be generated externally within local ODS stores or as part of an ETL job. Synchronization of any downstream stores to the MDM is also recommended.

4.3 Pre-computed Aggregates

Dashboard views for users often contain summary level information or aggregated counts that cannot be feasibly produced when record counts are into the millions. Pre-computed aggregates are a mechanism to alleviate this problem. There are several ways to realize this pattern:

1. Pre-computed views in standard databases using relational schemas or NoSQL: pre-computed views are information models that avoid read-time aggregation of information. These views are updated and maintained as relevant data is made available. Ex: Accumulator information for individuals and family, billing summary, household information for display purposes, care team view for members. These views are created by pre-assembling information across various data sources.
    a. Pros – Easy to implement
    b. Cons – Trigger implementations add extra load especially if running large batch UPSERTS
    c. Risks – Supporting drill downs across multiple dimensions usually take heavy infrastructure and large processing windows. Full table scans may be slow, regardless of the storage engine used. Maintaining proper dimension tables, indexes and aggregate tables was painful. Supporting high dimensional OLAP in NoSQL DB's requires pre-computing an exponentially large amount of data.

2. Landing Zone (Big Data) with BigTable options such as HBase or Cassandra: distributed data transformation.
    a. Pros – Leverage Landing Zone investments to scale to very large loads

3. Aggregates using data warehouses and OLAP cubes.
    a. Pros – Leverages existing investments and skills.
    b. Cons – Warehouses usually have higher data latency when transactional applications require access to the latest information.

5 Appendix

5.1 Related Documents

This guide refers to concepts or contains content originated from the following document(s).

REFERENCE TITLE	REPOSITORY
Landing Zone Reference Architecture	EA Community (RAM)
Enterprise Data Provisioning High-Level Design Patterns	EA Community (RAM)
Enterprise Application Integration Reference Architecture and Patterns Guide	KP Architecture Practice
Enterprise Data Reporting Stores Architecture Patterns Guide	KP Architecture Practice
Systems of Engagement (SoE) Application Data Caching Architecture Patterns Guide	KP Architecture Practice

5.2 Glossary

5.2.1 Analytical Data

Subject oriented, integrated, time variant, non-volatile collections of data in support of business intelligence and other types of analytic activities.

Note: Analytical data is typically used to understand behavior and trends over time past, predict future trend and behavior patterns, and even prescribe actions that would correct or enhance trends and behaviors. Analytical data must be organized and accessible by a variety of categories, including a time period category.

5.2.2 Business-Oriented Data Landscape

A high-level data flow and lineage diagram that identifies systems and data movement between systems. The names used to designate the data that flows are familiar to the business. It depicts the current state of a system identifying where data is authored, stored, updated, deleted, and shared across applications (using CRUD indicators). A data landscape helps: 1) identify systems of record, 2) master, reference, and transactional data stores, and 3) PII/PHI/PCI. It informs planning for future state data architectures (e.g. MDM) and data governance.

5.2.3 Cross-Reference Data

A mapping of data identifiers and/or codes that serves to make the data they categorize or codify equivalent, and adds detail necessary for appropriate use. Example: A mapping of the diagnosis codes used within KP Health Connect to the Industry diagnosis codes. The KPHC diagnosis code set has additional levels and detail. Note: Commonly referred to as "Crosswalks" by business partners. May contain hierarchies.

5.2.4 Enterprise Information Model

A collection of domains that together make up the "Enterprise". The Enterprise Information Model (EIM) focuses on the business meaning or "semantics" of data, totally divorced from solution choices and implementation concerns. The EIM aligns definitions of the data into "information domains" that provide a picture of data "in context" of the business processes required to support the enterprise. It enables strategic planning, architecture, and seeds lower level models with business definitions. Note: the EIM is not an Entity Relationship (ER) Data Model. There are no keys and placement of data elements is influenced by business process.

5.2.5 Master Data

Data that is the subject of a transaction and is agreed upon and shared across the enterprise. Examples: Data about Members, Providers and Products. Note: They are the business critical "nouns" of your business.

5.2.6 Master Data Management (MDM)

Processes that control management of master data values to enable consistent, shared, contextual use across systems of the most accurate, timely, and relevant version of truth about essential business concepts.

5.2.7 Metadata

Meta-data describes the context, structure, meaning, characteristics, constraints, and possibly even origin of something of interest. Meta-data improves the usability of whatever it describes.

Examples of how meta-data can be specified:
- A flow diagram with definitions of a business process
- A description of an application system and its components
- A description of "data at rest" in a database
- A description of a SOA service with data "in-flight"
- An ETL script that documents data movement
- A "Help" directory that documents the scope, intended use and column definitions for a report

5.2.8 Operational Data Store

An Operational Data Store (ODS) is a repository of current-valued detail data, extracted and stored separately from a source application system(s) for the purpose of supporting immediate decision-making or other time-critical processing needs.

Note: Use the term ODS not for referring to the private staging of a single application but rather to a shareable data store used by multiple applications with common data needs. The consuming applications may be reporting applications or transactional systems.

5.2.9 Reference Data

Reference data is used to classify or categorize other data and dictate conformance to a prescribed set of allowed values. It may be internally defined such as a Chart of Accounts, or externally defined such as zip codes or country codes. Reference data is designed to be centrally defined, stored and governed.

5.2.10 System of Engagement (SOE)

A system that enables individuals to interact with KP and health care communities.
- Acts as the communication mechanism and relationship builder with our consumers.
- Focuses on ease of use.

5.2.11 System of Intelligence

A system that transforms data into meaningful and useful information which aids KP in making informed business decisions and providing needed business support.

5.2.12 System of Origin (SOO)

The first source where data is captured by a business function. For auditing and legal accountability purposes, this system may provide a record of KP's *initial* knowledge of the data and its characteristics for a specific area of interest.

Examples (as of June 2016):
California ACA Member = KP Connector (An externally hosted system)
California Traditional KP Member = Foundation Systems

Note: The SOO may be an internal or external system.

5.2.13 System of Record (SOR)

The definitive and singular source for data that enables business function(s). This system serves as the authoritative data source for the data and its characteristics for a specific area of interest.

Examples (as of June 2016):
Colorado Member = Common Membership
Northern California Member = Foundation Systems
Mid-Atlantic States Member = the Membership System

5.2.14 Transactional Data

Business transactions are the record of events involving the exchange of products, money, and/or data. Transactional data supports a high volume of updates while preserving consistency and integrity. It is current and changes frequently.

5.3 Contributors

Special thanks to the following contributors for their support, input, and feedback.

DOCUMENT VERSION	DOCUMENT DATE	CONTRIBUTORS
1.0	February 2016	• Kelly J. Mantione • Chris Lee
1.1	March 2016	• Dee M. Goldschmidt
1.2	March 2016	• Rosie Jergovic • KP Architecture Review Board
1.3	April 2016	• Chief Data Office
1.4	June 2016	• Chief Data Office