Connected Health Framework Architecture and Design Blueprint

A Stable Foundation for Agile Health and Social Care

Part 3 – Technical Framework

Second Edition Published March 2009
Rev April 2011

Microsoft

Part #098-106615

Contents

Figures

Introduction to Part 3

In *Part 2 – Business Framework* of this guide we discussed the service-oriented approach to defining a core set of business components, each addressing a major subject area and offering a range of services that can be "orchestrated" to enable and support the wide range of healthcare business processes. The resulting set of business components and service definitions for citizen-centric care constitutes our Business Pattern for Health and Social Care.

This *Part 3 – Technical Framework* is focused on the technical architecture aspects, starting with advice on *Addressing Common Architectural Challenges*, and then describing the typical set of infrastructural services likely to form the foundation of e-Health and e-Care solutions at any level – *Reference Architecture* (page *44*).

As pointed out in *Part 1*, The Business Pattern and the Reference Architecture focus on different aspects of the system and represent distinct viewpoints, but they align very closely, as shown in *Figure 1*.

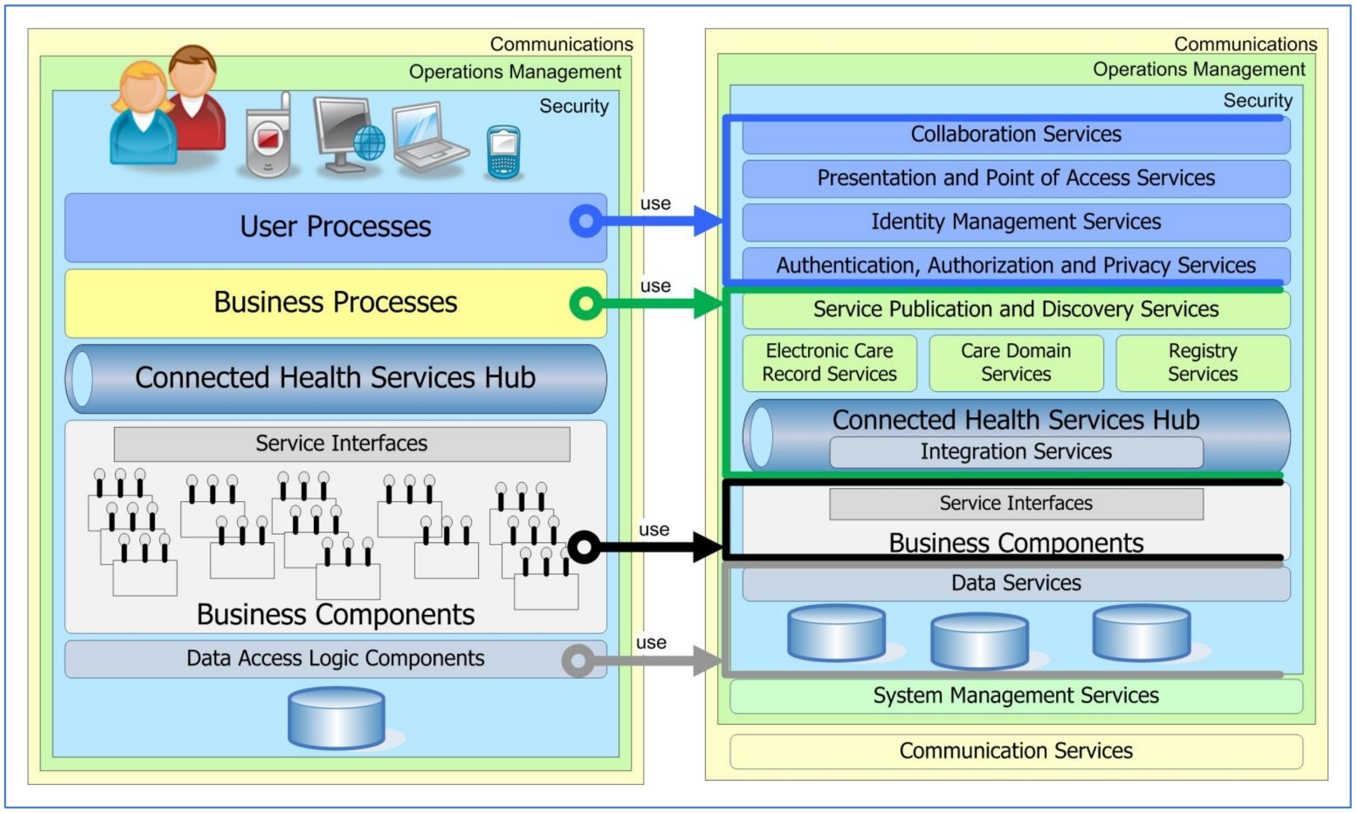

Figure 1. Alignment of the Business Pattern and the Reference Architecture

Figure 2 shows the Business and Technical Frameworks within a combined schematic, aligned around the Connected Health Services Hub.

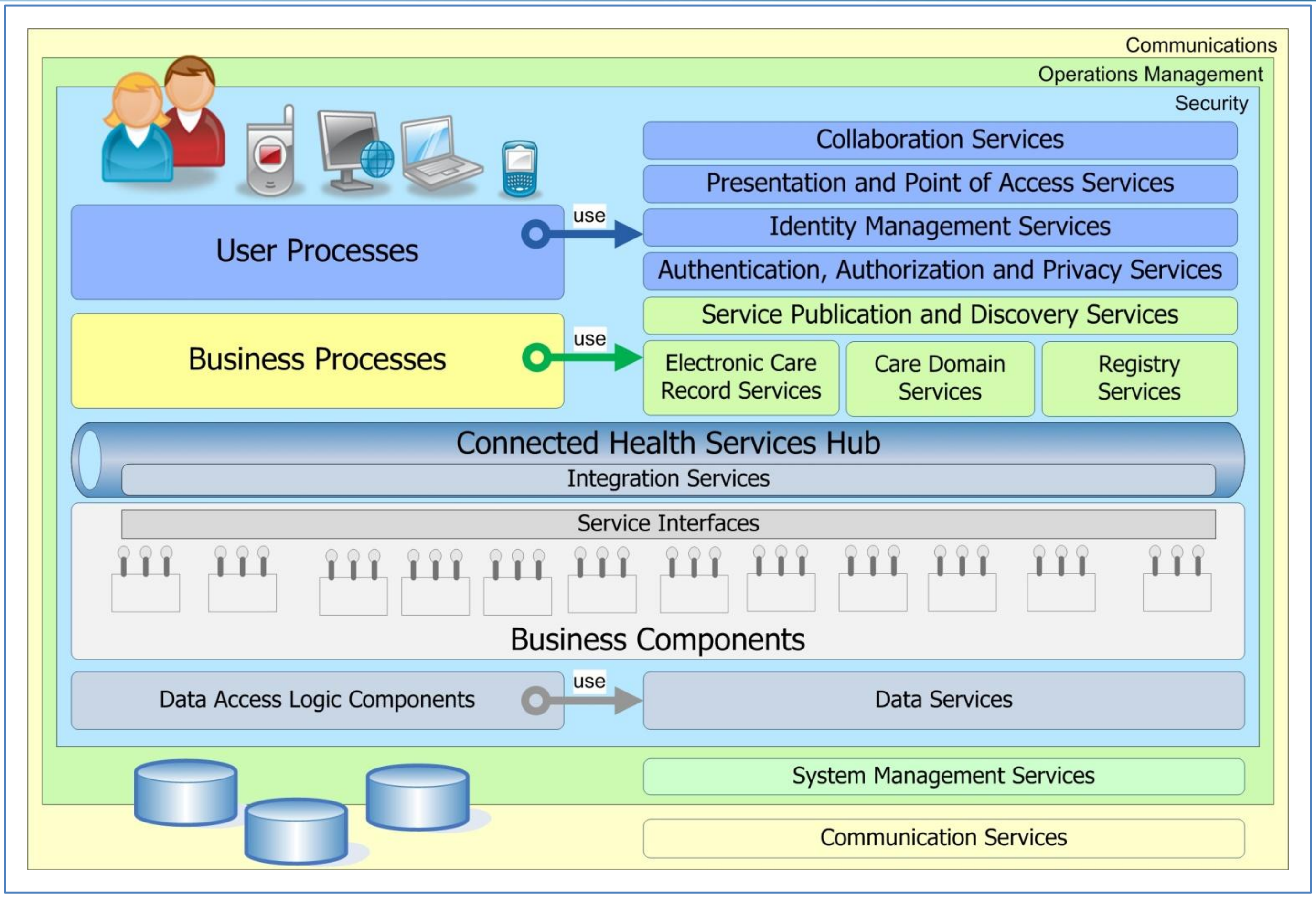
Communications
Operations Management
Security
Collaboration Services
Presentation and Point of Access Services
Identity Management Services
Authentication, Authorization and Privacy Services
User Processes
use
Service Publication and Discovery Services
Business Processes
use
Electronic Care Record Services
Care Domain Services
Registry Services
Connected Health Services Hub
Integration Services
Service Interfaces
Business Components
Data Access Logic Components
use
Data Services
System Management Services
Communication Services

Figure 2. Connected Health Framework Joined-Up Architecture

The presentational business requirements of the user interface and user processes are facilitated using technical capabilities of the Identity Management, Privacy and Security and Presentation and Point of Access Services provided in the Technical Framework. Similarly, the business process requirements are facilitated using the Connected Health and Social Care Services Hub and the technical capabilities of the Service Publication and Location, eHR, Health Domain, Registry, and Integration Services provided in the Technical Framework. The data access requirements are supported by the Data Services capability.

Business Services, identified and componentized in the Business Framework, can now operate upon the platform provided by the Technical Framework and combine through the Connected Health and Social Care Services Hub to satisfy the business requirements of the Health and Social Care domains.

We recognize the need for agility made possible by having a stable foundation. The Connected Health Framework helps support that agility by separating the more volatile user and business processes from the more stable business and data services, the "join" between the agile and stable worlds being provided by the Connected Health Services Hub (*Figure 3*).

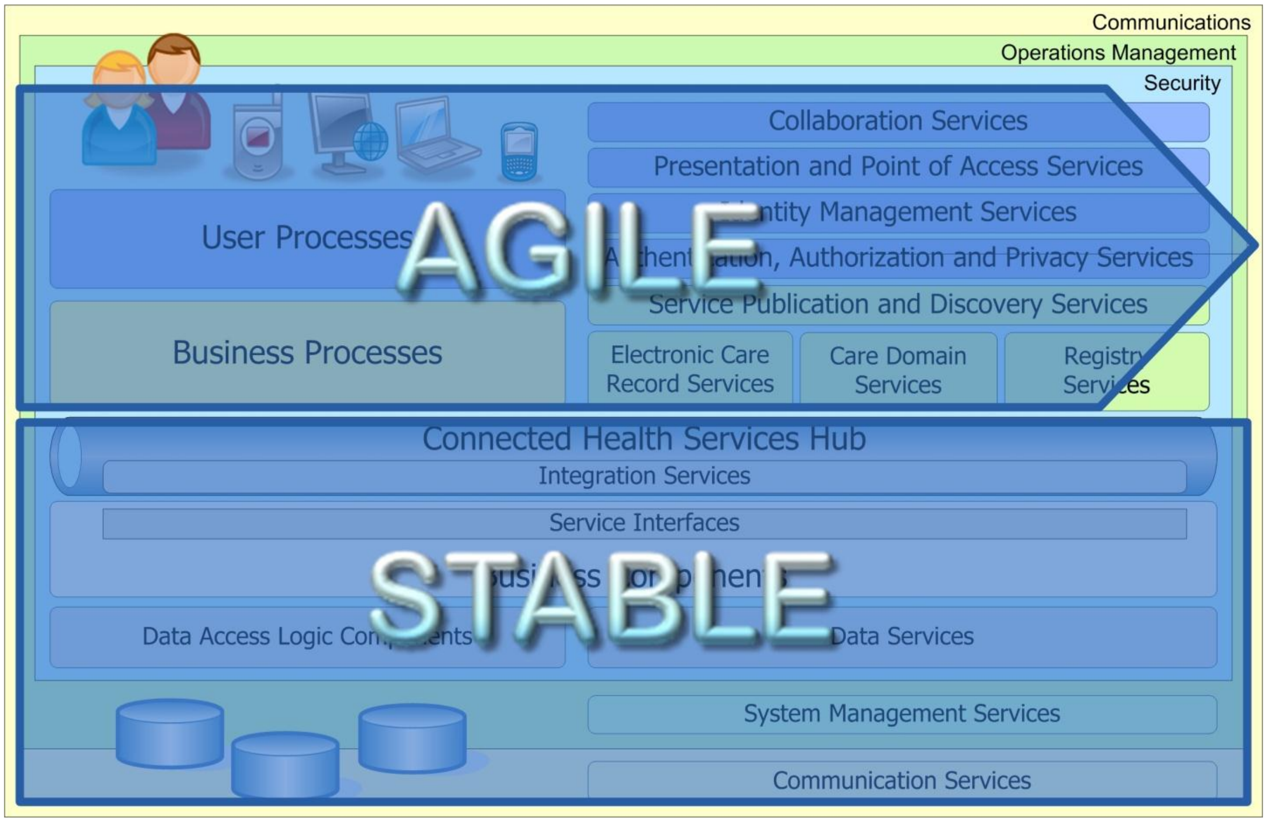
Communications
Operations Management
Security
Collaboration Services
Presentation and Point of Access Services
Identity Management Services
Authentication, Authorization and Privacy Services
Service Publication and Discovery Services
User Processes
AGILE
Business Processes
Electronic Care Record Services
Care Domain Services
Registry Services
Connected Health Services Hub
Integration Services
Service Interfaces
Business Components
STABLE
Data Access Logic Components
Data Services
System Management Services
Communication Services

Figure 3. Connected Health Framework – A Stable Foundation for Agile Health and Social Care

Addressing Common Architectural Challenges

This section of the guide discusses the most common architectural challenges faced by e-Health solutions, areas that require special consideration (and are sometimes, at least initially, overlooked), some of the available options, and the recommended approach to addressing these challenges. The focus is on the major issues specific to e-Health solutions, beyond those commonly encountered in other large-scale electronic delivery and integration projects.

This section discusses the technical architectural challenges at a high level, in order to develop an appreciation of the complexities, issues, and available options—irrespective of the specific technologies and implementations. The section *Reference Architecture* starting on page *44* contains further recommendations on how to address these challenges.

Flexibility and Agility

Being able to cope effectively with changes, and adapt to accommodate new requirements, is essential for most systems. This is even more important for e-Health solutions, where the pace of growth and the volume of unknown factors at the start mean that the rate of change is typically much higher. Likewise, the need to provide backward compatibility with systems and versions deployed previously is even greater.

Why Flexibility and Agility Are Essential

E-Health solutions provide a platform that must support a growing range of services. Unlike many commercial systems, where the range and variety of services and delivery channels—both existing or anticipated—is reasonably well-defined and stable, an e-Health integration platform has to cope effectively with changes as they arrive, anticipated or not, and preferably without redesign or disruption of existing services.

The common pattern seen in many countries is that e-Health solutions typically start small as a “pilot” or “proof” and involve a few carefully selected services, usually based on their visibility, urgency, and readiness to deliver quickly. After a successful initial phase, the ambitions and the appetite for delivering more services online grow very rapidly, and the expectation is that the platform already in place should accommodate these seamlessly.

Changes, such as enablement of new services, new authentication providers, and new delivery channels, are not rare occasions that must entail major disruption and be possible only through new releases of the e-Health platform. Instead, they should be part of a normal and mainstream usage scenario, adequately supported by the architecture and design of the system, tools, and procedures.

Ideally, the application of such incremental changes to the running system will be possible without downtime, as the implementations in some countries and in other areas like e-Government have already demonstrated.

Architecting for Flexibility

Achieving this higher level of flexibility is an important factor driving the architecture of the solution. To accommodate major change scenarios such as adding new services, new message types, changing validation methods, or reconfiguring routing rules—preferably without code changes—the whole solution requires management through configuration parameters and data updates as required.

Adopting an architecture that assumes and caters for multiplicity at every level (nature and type of participating nodes, authentication providers, identification models, and service-specific rules) is an essential requirement. Expecting the requirements and rules for different services to vary, if not initially then at some point in the future, requires an architecture that supports per-service segregation. This minimizes the core set of common rules, message flows, and formats that every participant must agree on for the system to work. Avoiding assumptions about particular types of connectivity, security, and other logistical limitations is essential to support an expanding range of services.

User Experience and Acceptance

Ultimately, the success (or failure) of an e-Health system, regardless of its technical merits, is critically dependent on the acceptance of the end users. Ensuring that their experience with the system is easy and intuitive, the range of supported devices fits their working practices, and gains in productivity are important architectural considerations.

Providing Safe and Intuitive User Interfaces

To date, there has been no standardization for the display of important information elements, such as patient identification information and dates, in clinical applications. As healthcare becomes more reliant on electronic systems, there is a need to standardize display elements so that healthcare professionals can switch between different applications and rapidly identify the information they need, thereby increasing clinical effectiveness and improving patient safety.

A clear, consistent and simple interface enables healthcare workers to perform their jobs more efficiently by being able to find, view, and enter patient information in a predictable and easily recognizable manner. This also aids in the reduction of their cognitive load and the need for specialist training.

Pioneering work in this area, led by patient safety professionals from the National Health Service in the United Kingdom and user experience experts from Microsoft, has produced a set of user interface guidelines and reusable by application developers controls compliant with the guidance. *Microsoft Health Common User Interface* (CUI) is published and available free on http://www.mscui.net.

Choice of Thin, Rich, or Smart Client

Most requirements emanating from government and public sector organizations seem to want a browser-based solution, whereas most solutions from ISVs are rich client based. This presents something of a dilemma. The argument for the browser solution is largely economic: they can run on cheaper devices, with low maintenance overhead, and can be “locked down” to prevent unauthorized use. Rich client solutions are favored by ISVs because most of their products began life as departmental solutions, either running stand-alone or in a client/server configuration. They were designed to run on PCs that did other things too, like word processing or spreadsheets. So recently ISVs have been creating Web browser versions of their products. However, it seems that many clinicians do not like Web page displays with limited or no functionality and want displays that are rich to the extent that they support the processes they carry out. Using terminal server technology to present a server-based rich solution on a thin client device is popular but quite expensive in terms of server hardware. We have noticed in practice that the number of concurrent sessions that can run, before performance and capacity issues arise, can be fewer than predicted in design calculations.

A preferred solution is the so-called “smart” client. With this, the interface is tailored to the role of the user, providing only the functionality and data required by the user for the task he or she is currently carrying out. At one extreme this might well be a just Web page—for example, showing a busy ward nurse today's care plans for her patients—while at the other extreme, a rich client display showing a consultant physician the patient's history, current and previous test results, some research findings, possible treatment plans, and areas to enter notes and prescribe might be more appropriate.

We conclude that a range of display options should be offered; there is a place for thin, thick, and smart client options, depending on the context and functionality. The important point is that each kind of display is fuelled by the same data. This infers separation of the presentation layer of the application from the logic and data layers as we suggest later.

Managing User Processes

Recently, there has been much discussion of clinical acceptance (or the lack of it) of major health records systems. Much of this reluctance stems from a belief held by many clinicians that computer systems will not reduce their workload, but rather will *increase* it. They are fearful of increased data entry tasks, duplicated effort, and receipt of erroneous or incomplete information. It is believed that much of this distrust comes about because use of IT is not part and parcel of the routine processes carried out by healthcare professionals.

Therefore, we believe that much effort should be made to reflect the actual user process into the system dialogue. This includes not only tailoring the screen contents to the items needed at any point in the process, but preloading anticipated data and performing in-flight validation. User processes vary of course, the same task being carried out in different ways in different hospitals, for example. Personal preferences can also apply.

The imaginative and perceptive combination of device, user interface, and user workflow can produce major benefits in terms of user acceptance, data quality, and operating effectiveness.

Using Mobile Devices

Many people today use laptop computers, Tablet PCs, PDAs, Smartphones, and other portable devices in their business and personal lives. These can be of considerable value in Health and Social Care scenarios. Many care professionals visit their patients either at home or during ward rounds. The ability to have current patient data available during such encounters and to make on-the-spot transactions and in-flight data validation is a highly valuable asset.

However, use of these devices can present problems. First, they may require special user interfaces because the processes they support are different from in-house processes. Second, they must be secure. The loss of a device containing confidential, personal data could be embarrassing at least. Third, to be fully useful, such devices need to be connected to base for the uploading and downloading of information. However, communication facilities are not always available or reliable.

Therefore, in designing mobile applications, provision must be made for the context within which the device is being used—"in-house" (such as in the hospital, clinic, or practice); or "on the road" (in the patient's home or some external location). This means that different operating modes will be needed. For example, an "in-house" connection will be to a secure network, whereas for an "on the road" connection, public networks will be used with potential security and coverage problems. Mechanisms are needed to cover each of these situations. Typically, "on the road" issues may be handled by preloading applications and data and storing input for uploading when back at base. However, this might not apply in the unexpected situation or give the desired level of response, in which case, methods of remote data provision and secure interactions will be needed.

An issue arises when a remote session fails in mid-flight due to a loss of connectivity. The procedure then would be to freeze the session at its last synchronization point, continue “offline” and “rehydrate” the session when communication is restored. This procedure is similar to the “ActiveSync” process using Microsoft Pocket Outlook on a Smartphone or PDA. An interesting variant of this could be the scenario of a hospital consultant doing his rounds. During a patient consultation using his Tablet PC, he is interrupted by an emergency call and suspends the session immediately. Later, having dealt with the emergency, he wants to resume the session to complete his notes – but this time in his office using his desktop system. This too is a “freeze-and-rehydrate” situation, but this time picking up on a different computer with different capabilities.

Support for Multiplicity

Many aspects of e-Health services involve a solution to the issue of multiplicity, and can benefit from the consolidation and simplification provided by a unifying shared infrastructure that supports these services. The following sections highlight the main areas where such multiplicity is encountered.

Growing Range of Capabilities

E-Health initiatives typically start small, with only a few capabilities available initially, and then grow over time, aiming to incorporate a growing range of services. This presents some unique challenges stemming from the variety of types of systems, and their growing number.

The back-end systems supporting these e-Health services typically run on a wide range of platforms and use different technologies. Gradual discovery of these disparate services is the common scenario as e-Health initiatives expand and evolve over time. Designing and implementing a common infrastructure capable of effectively supporting this growing range of capabilities is a significant challenge, beyond that normally encountered in most traditional commercial systems.

An effective e-Health platform, once put in place and running in production, should be able to support the addition of new participants, new services, new transactions, and other changes to requirements seamlessly, without code changes in the core solution and preferably with no downtime affecting existing services. Adding new services should be a mainstream use case, not a rare exception—adequately catered for by the solution design.

Variety of Access Channels

There is a high probability that even when the delivery channels for the initial set of services are well defined at the onset of the project and catered for in the first implementation, requirements to support new categories of users and new access channels will emerge over time. Examples include kiosks in public areas, interactive TV, mobile devices for patient/consumer services, and the growing variety of device form factors for care professionals.

A well-architected integration platform exposing services in a consistent way should allow adding the capability to use new delivery channels through a single, one-off effort in the central hub to enable the new channel—without affecting the multitude of individual services.

Broad Support for Different Client Platforms

E-Health projects are often subject to stricter regulations and restrictions than typical commercial systems. Provision of non-discriminatory access for citizens to e-Health services from a wide variety of client platforms is often an explicit legal requirement, or at least politically desirable. This can involve different types and versions of hardware, operating systems, and browser software.

The owner of a commercial system can make a calculated trade-off decision to exclude some small proportion of prospective customers by narrowing the range of supported platforms and losing only small potential revenue, but making significant savings from the reduced complexity of the design, development, and testing for the system.

However, healthcare providers are much more constrained in this respect, and often have to bear the significant cost of supporting relatively obscure platforms in order not to exclude and discriminate against even a small proportion of their constituencies.

Creating the architecture for an e-Health integration platform with a clear understanding of the constraints and requirements for broad client platform support is very important, because satisfying these at later stages of development can be very expensive or even impossible.

Multilingual Capability and Universal Accessibility

In many countries, providing multilingual access to health services is a legal requirement, or is at least highly desirable. It is important to cater for this upfront, since it has a significant impact on the design. In general, there are three important areas to be aware of:

- Accepting the *input of multilingual data*, and using alternative scripts, pages, and validation logic (for example, for right to left written languages and languages that use special characters)
- *Core processing and storage* of data, where multilingual requirements can influence the data model, message formats, and encoding, and may require special consideration when writing code
- It may be required to design a fully *multilingual user interface*, where all text strings and messages are separated from the code for easy localization. This usually includes a requirement for alternate graphics and other content to enable full support for character-based right-to-left languages, and taking into account that the same text in different languages results in differing string lengths.

Health sites and services are often subject to very stringent accessibility requirements—more so than ordinary commercial sites where accessibility is often desirable to reach a wider audience, but lack of it may have only a limited commercial impact. For health sites and services, accessibility is usually a legal requirement, and an important political issue.

Retrofitting multilingual capability and accessibility after the design of the system is complete is difficult, expensive, and sometimes impossible. Designing for accessibility should be a primary consideration from the start.

Part 5 – References of this guide contains checklists for creating translatable applications, and a list of links to current accessibility legislation and national initiatives including the World Wide Web Consortium (W3C) accessibility guidelines, the various types of specialized screen readers and browsers that are available, tools for testing accessibility of applications, and links to other useful resources.

Exposing e-Health Services in a Consistent and Secure Manner

Exposing e-Health services through electronic (online) channels requires substantial investment and effort to meet the necessary requirements for security, availability, and reliability. In some countries, there are stringent regulations and a mandatory certification process for connecting systems to the Internet. If the electronic enablement of these services occurs independently, duplication of this effort takes place for each service. This wastes time, resources, and expertise that is not always readily available in all organizations and agencies responsible for these services.

The problem grows as the number of electronic services increases beyond the initial circle of major agencies and providers that have the size, visibility, and political influence to secure the necessary resources. Smaller organizations, such as local health authorities, clinics, and individual practitioners, often cannot afford the high initial cost of entry to offer their services electronically on their own.

Isolating the most challenging elements of online enablement, and implementing them once in a common e-Health platform shared by all services, can help ensure the high quality, consistency, and security of the resulting solutions and produce significant savings.

Handling Health Data

Health-related data for a particular individual is typically stored in disparate computer systems, operated by different care providers, in multiple locations, sometimes even in different countries. There is a major issue in knowing what data is where, how current it may be, and whether it is accurate. Assembling a consolidated record from such diverse sources (either proactively, by feeding some "master" data store, or dynamically when needed), in order to provide care professionals with the relevant complete picture and to support them in making well-informed decisions, presents significant architectural challenges. Choosing the most appropriate data topology—in the specific context and at various levels of the system—is one of the central architectural challenges and can be critical for the overall success of the project.

In this section, we analyze the typical *Requirements Related to Data*, how they apply to different *Types of Data*, and discuss possible *Data Topologies*—their advantages and disadvantages, and typical scenarios where each may be appropriate.

Requirements Related to Data

The typical requirements related to data in e-Health systems can be grouped as follows:

- *Resilience* – once stored, data must be protected from loss or damage as a result of technical problems or operational or human errors
- *Availability* – data must be available when and where needed
- *Accuracy and consistency* – data must properly reflect the "last known" state
- *Privacy* – personal data must be protected from unauthorized and inappropriate access, in compliance with the legal and privacy regulations
- *Access and audit* – access to data must be controlled and audited

It is important to distinguish the degrees to which these requirements may apply to different categories of data. One commonly seen mistake is vaguely defined, "blanket" requirements like "*no data loss*" or "*access within 2 sec*", without any details or further qualification of what kinds of data they refer to (and hence assumed to apply to *all* data), appearing in RFPs—which make the resulting systems over-engineered and more expensive than needed to meet the actual needs in the specific context. The following section provides further guidance on a more differentiated approach to requirements, depending on the *Types of Data*. After that, we look at typical *Data Topologies*, their advantages and disadvantages, and offer some recommendations for choosing the most appropriate in a specific context.

Types of Data

Data can be categorized in a number of different dimensions (by volatility, content, scope) – with their respective (and often quite different) requirements.

By Volatility

It is essential to distinguish data depending on how frequently it may change, and the source of these changes – as this can determine the appropriate (and often different) strategies for meeting the requirements above:

- **Reference data** – infrequent changes, typically distributed periodically top-down from national or other levels. Examples include catalog of medicines approved for use, lists of care providers and their specialties, approved treatments, care pathways, and price lists. All such data can be easily cloned/replicated through the system as needed, with multiple copies existing in parallel, and some mechanism for rebuilding/refreshing (periodically or event-driven). Resilience is rarely an issue; privacy, access, and audit typically do not apply to this type of data (except for the master source).
- **Stable (cumulative, historical) data** (like snapshots of episodes of care) – once created, these typically do not change; each new item is added (appended) to the previous set. Data related to a single episode is clearly, uniquely identifiable and date-time stamped—it does not conflict or overlap with any other such item, even if replicated/transmitted and arriving out of sequence. The analogy from another industry: each bank transaction, which may originate in a different place, is uniquely identifiable – and they all eventually arrive and reconcile in a ledger or statement. The stable and independent nature of each "transaction" (such as an episode of care) makes achieving data consistency across a distributed system relatively easy. Resilience is important, especially in the time window from creation to transmission to another node – after that, redundant copies of the data allow recovery from other nodes.
- **Core (current) data** – which reflects some "current" (or "last known") state, and may be updated. The analogy is the "current balance" of a bank account, reflecting all known/processed transactions. Patient data like personal details and core medical information (allergies, current problems, and medications) are typically in this category. Satisfying requirements for this type of data is more challenging than for the other two above. Managing potentially concurrent updates to such "single core set" must be designed carefully to handle possible conflicts and prevent data loss. Ensuring accuracy (and completeness) of the core data, taking into account the latency and cost of communications, could also be a challenge.

By Content

Data can also be categorized by its content and subject:

- **Clinical** – directly related to the subject of care (observations, examinations, treatment, prescriptions). This narrow category is typically the main focus of the discussions about data distribution and topologies, with the most stringent requirements.

- **Administrative** – related to managing various care provider activities (teams, schedules, assignments, appointments). Most of the data in this category is only used within the particular provider node and rarely needs to be shared, transmitted, or consolidated at higher levels—so requirements around privacy, consistency, and access may be more relaxed.
- **Financial** – data related to financial planning and reporting of care provider entities, and their relationships with each other, payers, or other parties. This data could be detached and processed independently from the sensitive personal data, allowing for more relaxed requirements around privacy and access. Consolidation and bottom-up reporting are common in many countries.

By Scope

The applicability of requirements would also vary depending on the scope of data:

- **Internal** for a level (department, enterprise, group, region, national) or node – data that originates and is used only within that particular node. For example, individual temperature readings are not likely to be needed beyond a particular episode of care; so too are the meals eaten by a patient. All such data rarely needs to go anywhere outside of a node, and may not remain stored indefinitely. So, even when requirements like "all data is stored centrally" are proclaimed, it is never actually "all" data, and further qualification is needed. This becomes even more important for fine-grained data collected directly from various devices – as the data volumes are no longer naturally constrained by manual procedures.
- **Shared** – data that is transmitted to another node (typically up the hierarchy) to form a broader, consolidated record. In this case, it is important to define how the ownership and "master source" status are transferred, how multiple copies are kept synchronized, and so on.

Data Topologies

Choosing the appropriate topology and model for data distribution between nodes is one of the key architectural decisions when designing e-Health systems. There is no single "right" model appropriate in all cases; many factors influence these decisions at every level of the system (local, group, region, national, cross-agency). The following sections discuss the advantages and challenges presented by various models, and provide guidance on how to select the most appropriate one depending on the circumstances. These considerations would typically be applied multiple times, depending on the scope of the system, to each level of the hierarchy, and to different types of data—possibly leading to different optimal choices.

From an architecture perspective, there are two extreme models at the opposite ends of the spectrum: centralized and distributed (federated).

Centralized Data Model

In the centralized model (*Figure 4*), all data is stored centrally, and accessed by all consumers from that single source. Other subsystems may act as "data feeders", providing and transmitting data to the central store.

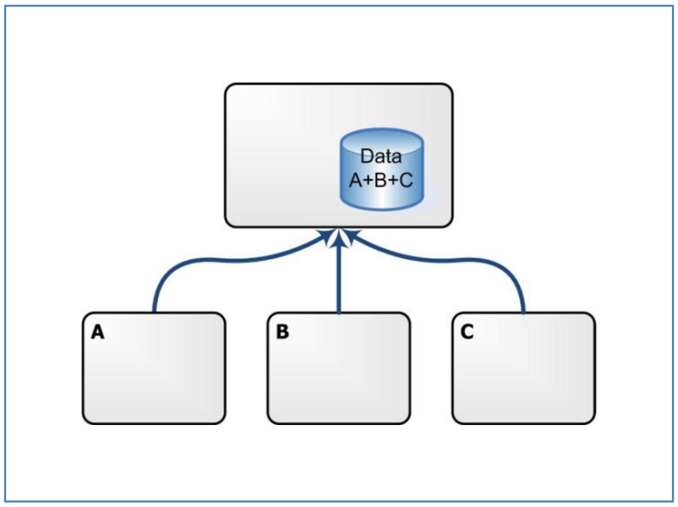

Figure 4. Centralized Data Model

Depending on the scope of the system, this model could be applied at different levels: to a single entity (hospital) – with various departmental or laboratory systems feeding a central repository; to a group or region – with participating nodes (clinics, hospitals, GPs) providing data for a regional repository; to a national system – with regions or other nodes acting as feeders for the national repository; and so on.

The main advantage of this model is its simplicity – the complete data set (for example, the patient's electronic health record) is available from a single, well-known (central) source. The solutions for bringing data into this central repository could be independent from the subsequent access to that data, may use different technology, and could have more relaxed performance and latency requirements (such as asynchronous messaging and periodic file transfer). Providing expected quality of service (availability, resilience), security, and manageability for one central data store may be easier and more cost-effective than managing multiple data stores in participating nodes—especially when these are remote locations with no adequate IT skills or support, like small clinics or GP practices.

Accessing data in the central repository from client nodes depends on the availability, reliability, and performance of all elements in the chain: central data store, communication between nodes, and local systems. The centralized model is a common choice for smaller, local systems. As the scope of the system grows larger and more distributed, the challenge of providing adequate scalability and performance may become a limiting factor. Nevertheless, there are examples of this model being used at a national level (NHS England).

In this model, only the data already saved in the central data store is available. It may originate elsewhere, but unless and until it makes it to the central repository, it does not exist for the consumers. There is no mechanism to request and receive data from subordinate nodes on an "as-needed" basis. Deciding what types of data (and how detailed) should go into the central repository is often a challenge: storing too little can make unavailable important details that may be needed in the future; collecting too many rarely used details would require more storage and can impact performance. The latter approach, however, has proven successful in commercial products like *Microsoft Amalga* (http://www.microsoft.com/amalga) – where collecting *all* data from "feeder" systems and storing it in flexible structures (different from the traditional, highly normalized, relational data models) allows unprecedented flexibility to develop new applications and uses of available data at dramatically lower marginal cost.

Other constraints often limiting the possible scope for using this model are the applicable legal and privacy regulations. In some jurisdictions, personal health data cannot be stored outside the entity that produced or collected the data (such as a hospital or a health provider group). The federated model (discussed in the next section) offers an alternative approach well suited to such cases.

Distributed (Federated) Data Model

In the distributed model, data is not concentrated in a single store, but remains spread across multiple nodes (subsystems). Depending on the scope of the system, these may be regional nodes (in a national system), health providers, and other participants holding and providing data. In order to provide a complete data set (such as a patient health record), relevant subsets must be collected from these different sources and then assembled in a single "virtual" record. This allows presenting the composite record and allowing informed decisions to be made, while complying with legal and privacy regulations not to store data outside of certain boundaries. The model is also well suited for cross-border or cross-agency access to information (for example, between health, social services, and other care providers) – where it is rarely feasible or politically acceptable to construct and maintain a joint data store.

There are various practical implementations of this model: "broadcast", "central index", and "replicated index".

It is possible for the central node (which itself holds no data) to simply "**broadcast**" the request (with the appropriate identifiers and other attributes determining what data is needed) to all nodes, and then process and collate the responses (*Figure 5*). This may be a workable approach for a relatively small number of participating nodes.

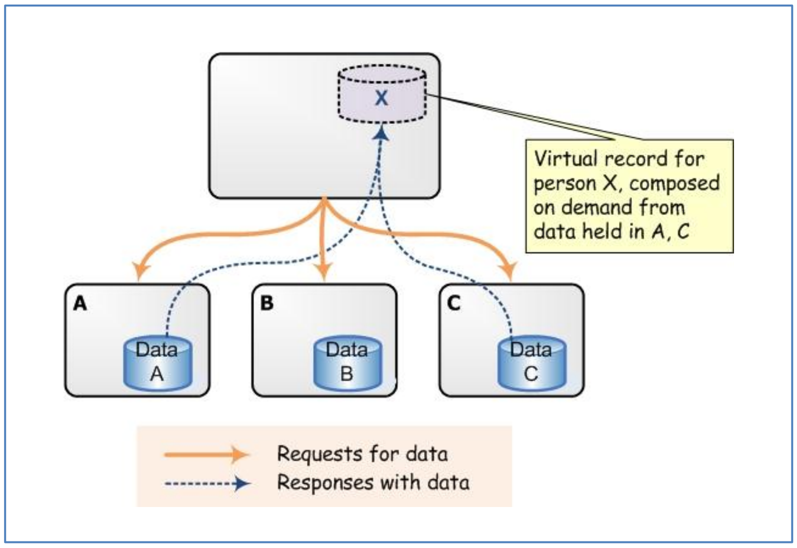
X
Virtual record for person X, composed on demand from data held in A, C
A
Data A
B
Data B
C
Data C
Requests for data
Responses with data

Figure 5. Federated Data Model (Broadcast)

A more common solution (*Figure 6*) is for the central node to hold "**index**" information about which nodes contain information for each particular person (often called a Master Patient Index). In this case, requests are only sent to nodes known to contain relevant data, which is more economical than broadcasting to all, especially for large numbers of nodes. The other advantage is that this central index may also provide a translation between different identifiers that may be used by participating nodes – so that each receives requests in its own "language", independent from the others.

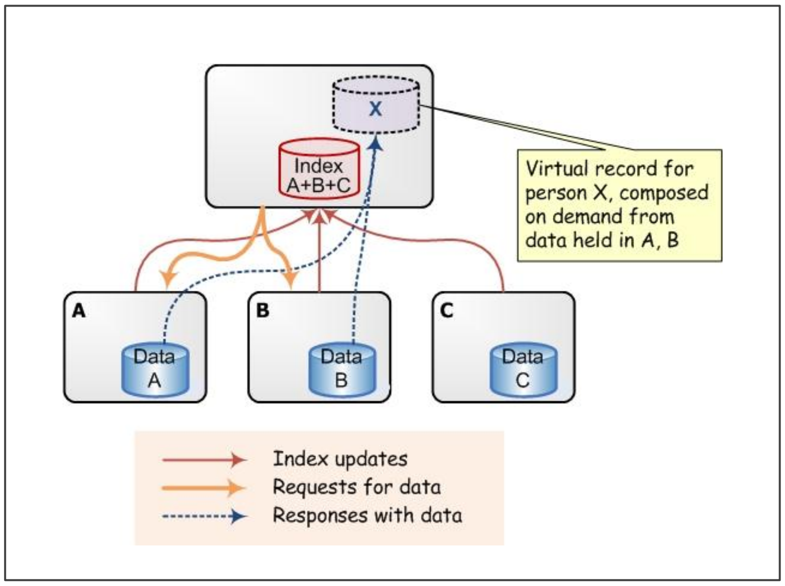

Figure 6. Federated Data Model (Central Index)

The central node may also contain relevant "metadata" associated with individual items stored in the nodes (dates, nature of the episode, type of data, etc.) – allowing search and filtering, and sending requests only for the items needed (such as laboratory results) instead of everything (full medical history). This is exactly the use case supported by the *IHE XDS profile*, with the central node acting as Registry (in IHE terminology) and the nodes as Repositories. The main principle, however, remains: there is no real data stored centrally—only indexing and metadata information. For this model to work, the "index" (or the Registry) must be updated every time a node acquires new data, which is typically done by a message from the node holding the data (Repository) to the central node (Registry).

Another implementation of this model avoids having a central node altogether, by maintaining a "**replicated index**" in each of the participating nodes (*Figure 7*). These are kept in sync by a peer-to-peer replication mechanism – for example, every time a new data item of interest appears in one of the nodes, notifications are sent to all other nodes, and their indexes are updated. Depending on the level of detail (metadata) kept in the index, the granularity of these notifications would vary. In the simplest case, if there is no metadata beyond just the fact that "*node X has information about patient P*", notifications are only needed when a particular node sees a patient for the first time. At any time, in any node, just by checking the local to that node index, it is possible to determine which other nodes contain information of interest, and send requests to them peer-to-peer. Collating responses allows a complete picture (such as a patient history or lab results) to be composed and used to make informed decisions, without storing it permanently (in compliance with privacy regulations). Any new data is stored in the respective node, and others are notified as appropriate.

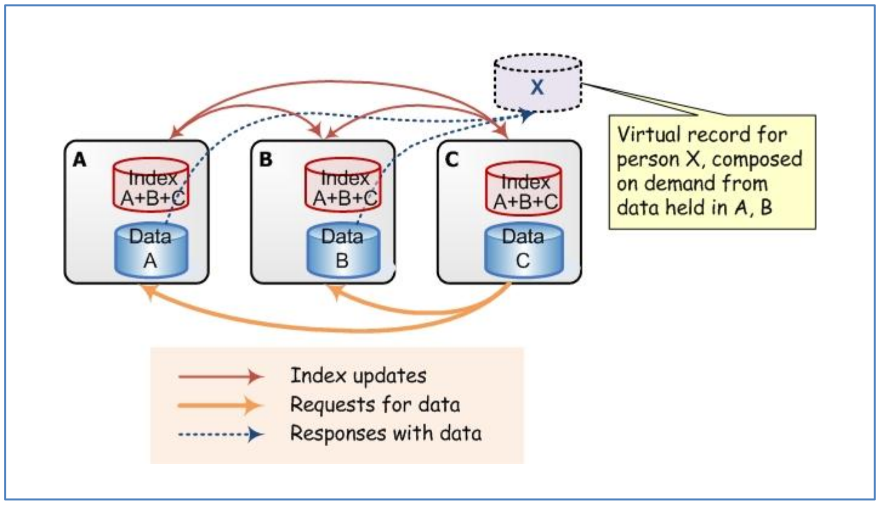

Figure 7. Federated Data Model (Replicated Index)

The commercial product that pioneered the use of the federated model with a replicated among nodes index is the *dbMotion Collaboration Platform* (http://www.dbmotion.com), which can also implement the centralized and hybrid models.

The main advantage of the distributed (federated) model is that data remains stored in the individual nodes where it was originally created or collected. This can be essential for meeting privacy regulations in some countries, where no other option may be compliant. Collecting data from the nodes on an "as-needed" basis is also more economical in terms of storage and communication traffic, rather than accumulating everything proactively in case it may be needed in the future. It also avoids the difficult upfront decisions about what exactly to collect (as in the centralized model) – once the general mechanism to request data from the nodes is in place, it is very easy to extend the range and types of data, or add more nodes.

It is worth noting that, once the mechanism for requesting data from other nodes is in place, more advanced scenarios become possible. Some data may be held in nodes different from the originator—especially when the originating node itself is not capable of providing anytime, on-demand access. If the originator can only provide data periodically and in bulk, not on demand (for example, the results from all lab analyses at the end of the day, or images), these may then be stored in a separate specialized node, which becomes the source of data on demand for all further requests. Nothing else in the models needs to change – only the pointers to the data store now lead not to the originating node, but to the specialized storage facility. The location, size, and other characteristics of such storage nodes can be optimized based on technical, organizational, and other considerations – and change over time with minimal disruption to the rest of the system. This is also fully supported by the *IHE XDS profile*, where the *Originator* of the data (documents) is distinct and separate from the *Repository* (where the data is stored) and the Registry (where the metadata index is maintained) – although some of these may be implemented as a single node.

Implementing a mechanism for requesting and reconciling data from multiple nodes makes the solution more complex, compared to accessing a single central data store. Providing adequate quality of service (availability, performance, latency) for these requests—typically when the data is needed, "real time"—is essential for this model to work. This means that the nodes must be up and running, and data available any time. Communication networks to each node must also be reliable and of sufficient quality. All these factors can make the federated model inappropriate in some cases.

Hybrid Data Model

The hybrid model has both central storage and a mechanism for obtaining data from participating nodes "on demand" – so it is a combination of the centralized and federated models. In fact, both these can be fully supported by the hybrid model as "boundary cases": with 100 percent of the data available from the central store (centralized) or 0 percent centralized (federated). The great advantage of the hybrid model is that the boundary between what data is stored centrally and what remains distributed can be anywhere between 0 and 100 percent, and be easily adjusted over time as requirements, usage patterns, and technology constraints change. In contrast: in the centralized model, data that is not in the central repository can never be available; in the federated model, no data can be obtained without a real-time request/response to the respective node.

The hybrid model (*Figure 8*) allows a flexible approach, where certain categories (and level of detail) of frequently needed data can be stored centrally for quick and reliable access directly from the single store, while further details, rarely used, or large in size data can remain in the originating nodes – and be requested when needed. Depending on where data is stored, the quality of service guarantees (availability, speed of access) can vary, which often drives the decisions of what to store centrally and what to keep distributed.

Typical candidates for patient information to be stored centrally are the basic elements of what is often called "emergency data set"—the essential data usually needed for providing unscheduled or emergency care:

- Basic patient data – identity, contact information, provider affiliation or entitlement
- Core medical information – allergies, current problems/conditions, medications

Beyond that, the next level of detail is the cumulative care history (episodes of care). In the hybrid model, only some summary level information would normally be stored centrally – including date, type, and outcome of the episode – with pointers to the full details available on demand from the respective node. This allows quick "headline" access: list of episodes (chronologically, by type, etc.) from which the care provider can select those of interest and request more details.

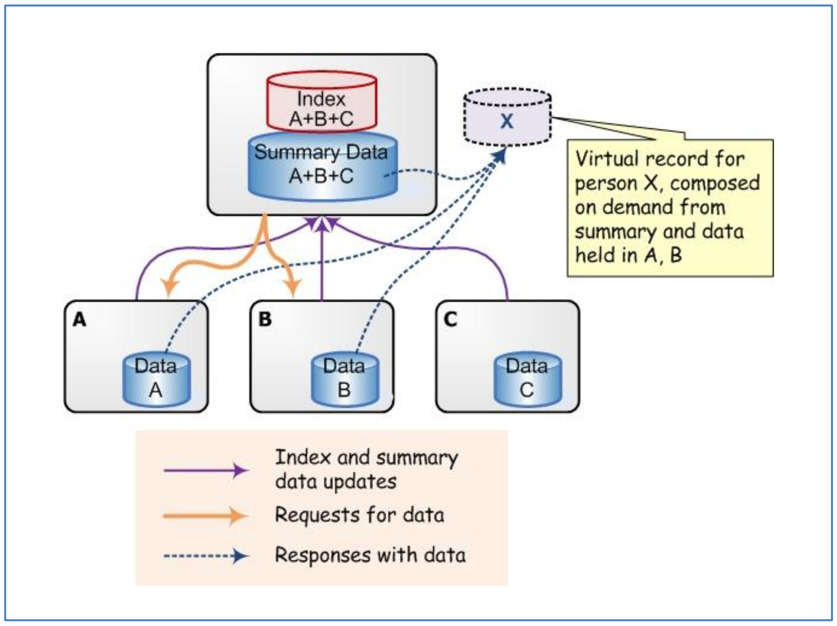
Index
A+B+C
Summary Data
A+B+C
X
Virtual record for person X, composed on demand from summary and data held in A, B
A
Data A
B
Data B
C
Data C
Index and summary data updates
Requests for data
Responses with data

Figure 8. Hybrid Data Model

For the reasons described above, and because the hybrid model still allows implementation of the other two (or anything in between) and is easy to change, we recommend considering adopting this model to ensure future flexibility, even if the initial project goals and constraints lead to one of the other two. Putting in place both a central store and a mechanism for requesting data from the nodes (even of not using one of them from day one) is a much better option than trying to extend at a later date an already deployed central model to allow requesting additional data "when needed", or a federated model to hold some subset of frequently needed data centrally.

Hierarchies and Combinations

All the models discussed above in general architectural terms can be applied in different combinations to complex real-life scenarios, depending on the scope and topology of the system.

In a common three-tier hierarchy with local, regional, and national levels, a *centralized* model for the regions can be combined with a *federated* or *hybrid* model at the national level (*Figure 9*).

Many countries with strong regional focus on care provision are either implementing or considering such an approach (Canada, Sweden, Spain).

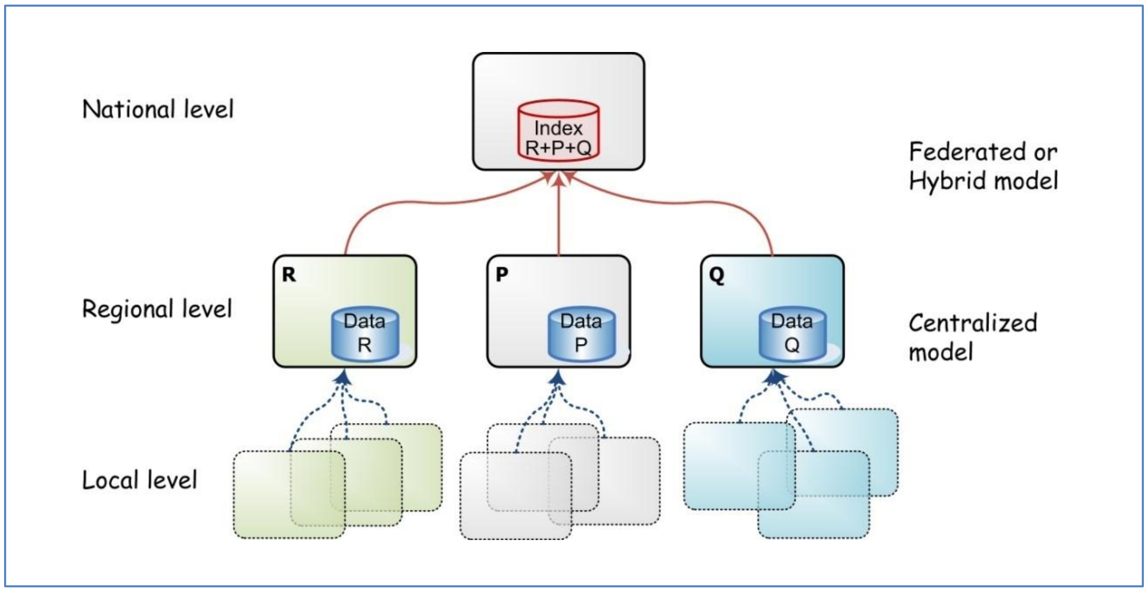

Figure 9. Data Hierarchy

From the point of view of scalability, especially for large systems, it is very important to identify the level of granularity at which data sets are independent, and can be processed and stored separately. For example, all data related to a particular person (like a complete Care Record) is typically independent from the similar data for another person—with the possible exception of mother and child, which often are treated as a unit. This means that even for a centralized model, the actual implementation could be partitioned for scalability, efficiency, and other reasons—for example, multiple physical nodes or data stores can implement a single logical "central" store. For each person, only one of these would be the single central store. This is how large-scale systems typically handle performance, scale, and capacity challenges – allowing seamless redistribution and repartitioning as the system grows.

Another variation of the “central – partitioned” approach described above is the assignment of a “home” node to each individual, and that “home” acting as a central repository for all data related to that person (*Figure 10*). In this case, a set of peer nodes act as the partitioned “central” store – where each is expected to contain the full record for its own “constituency” and can provide it when needed. All relevant information from episodes of care occurring outside of the “home” realm is expected to be transmitted to it (directly or indirectly) by the originator. The “home” nodes could be regions, or countries – each implementing a true “centralized” model (with its own data store), or acting as a proxy and collecting the necessary data from lower-level nodes.

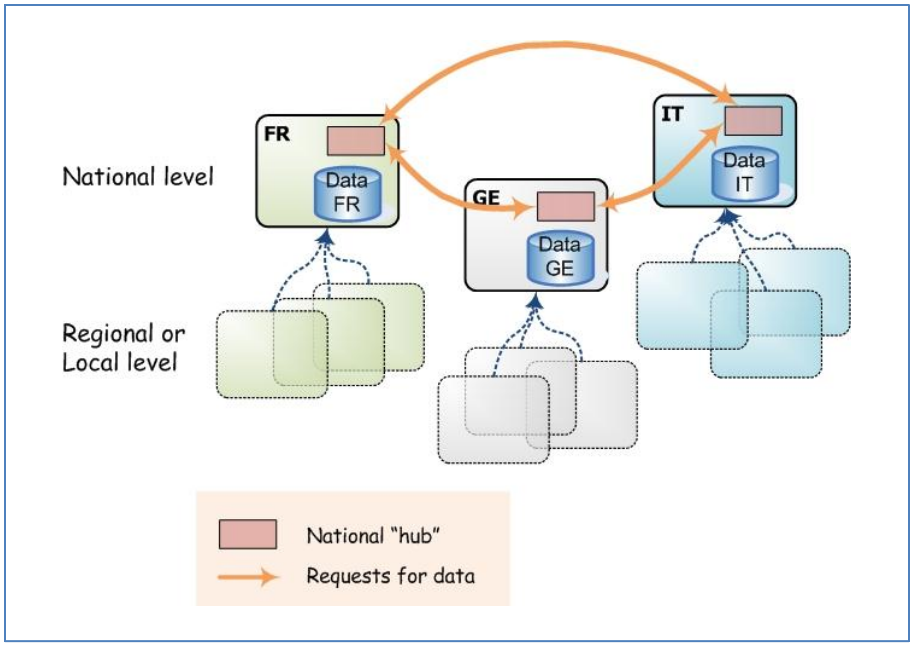

Figure 10. Partitioned Data

In fact, there is an EC-funded project under way (http://www.epsos.eu), aiming to pilot cross-border e-Health interoperability and continuity of care for European patients, which is likely to use this approach.

~~~~~~~~~~

As we have seen in this section, architectural decisions on how to structure, distribute, store, and handle data in complex distributed systems are challenging—and one size certainly does not fit all. It is essential to analyze different types of data and relevant requirements, in order to choose the most appropriate model for each level of the system. Retaining flexibility to change – without causing a major disruption – and tune the models in response to evolving demands and constraints is also essential for long-term success.
~~~~~~~~~~

Identity and Access

Identity and access are important for any large-scale distributed system, and there is plenty of generic guidance and solutions available in this space. However, in addition to the common challenges, there are numerous aspects of identity and access requirements specific to systems in health and social services – which often make traditional approaches and solutions not adequate. This section provides an overview of the most significant of these "additional challenges" and discusses architectural approaches to addressing them.

Multiplicity of…

One of the important challenges is what we call "multiplicity" – which can have many different dimensions.

Systems in health and social services need to support effectively a wide variety of usage scenarios, types of users, authentication methods, identity providers, and technologies. The scale and dynamic nature of such systems require built-in agility to change and extend over time, connecting new participants and broadening supported scenarios, with minimal disruption to existing services and participants.

…User Types

The two major types of users are the *care professionals* (health and care professionals, administrators, etc.), and *subjects of care* (patients, citizens, beneficiaries) – and the approaches to managing their identities and access are inevitably quite different.

Care professionals are typically employed and managed by some organization, and traditional enterprise identity and access solutions are often used. As the scope of the systems expands from the traditional "enterprise" space to regional and national networks, there are some additional challenges (discussed below in the sections Scale and Scope), but still managing this type of organizational user is relatively straightforward.

In contrast, subjects of care are more numerous, and not directly managed – so any identity and access solution must have the scale and characteristics of a consumer service (low-touch, low-cost, self-service) to be viable.

…Usage Scenarios

For care provider type users, traditional systems typically target well-established usage scenarios with workstations and devices within the organization (doctor's surgery, hospital, care worker office). However, the most significant gains in productivity and quality of care can be realized by harnessing the growing capabilities of portable and mobile devices, and extending the access to information and functionality beyond the boundaries of the traditional workplace. While some of these can still be viewed as "extension" and "virtual office" reliant on private networks and infrastructure, an increasing number of scenarios depend on the flexibility to connect and work from anywhere, with minimum dependency on a particular (private) infrastructure – which is a significant challenge for many traditional identity and access solutions. For example, a doctor receiving alerts on a mobile device, then checking a summary of laboratory results, and giving care instructions while on the move is no longer a visionary fiction, but a real capability supported by today's technology. Therefore, the full range of likely usage scenarios for care professionals should be analyzed carefully, and identity solutions chosen accordingly, in order not to constrain the ability to support more ambitious scenarios in the future.

For care subjects, the ability to access their own data, report status, receive reminders, make requests and appointments—remotely, in self-service mode, through a variety of devices and delivery channels—is essential for improving the quality of care while reducing the burden on care professionals. Identity and access solutions should have the flexibility to support such variety of access channels – instead of tying (and constraining) everything to a particular (single) authentication method.

…Authentication Methods

In order to support the wide variety of usage scenarios and access channels discussed above, the identity system must be open to use *multiple authentication methods* – depending on the type of the user, device, communication channel, and usage scenario. Even for the same user (say, a doctor), different authentication methods may be appropriate (or even possible) depending on the circumstances: smart card inserted in a reader of a PC in an office; fingerprint scan on a Tablet PC; proximity smart card in an operating theatre; password or PIN on a mobile device; and so on.

No single authentication method can be expected to work effectively in all situations. Identity systems making hard and restrictive assumptions about relying exclusively on a specific method of authentication (like a smart card) inevitably constrain the ability to support a broader range of scenarios and should be avoided. Ideally, the actual authentication method should be abstracted from the rest of the system – so that the functionality can remain independent from particular (and possibly changing over time) implementation specifics.

Application logic, access to data, and functionality should only be dependent on a generic "strength of authentication" attribute indicating the relative "trust level" of the performed authentication (such as a password, certificate + PIN, physical token/smartcard, or biometric) – not on the specific method and implementation. This will allow easy inclusion of new authentication technologies as they become available and relevant, without disruption of existing systems and functionality.

…Identity Providers

The need to support multiple independent identity providers results from the variety of user types, authentication methods, and substantially different identity management models required.

The typical identity provider for organizationally managed care professionals is some form of an enterprise directory (which might be, as in NHS England, scaled up to a national level). Flexibility to support multiple providers allows a variety of identity systems already in place to be reused through identity federation outside of their original scope (hospital, region), instead of rolling out a completely new national system. This capability also allows extending easily access to solutions and data beyond the boundaries of a particular organization, which is essential for supporting most "continuity of care" scenarios.

Subjects of care are better served by a broader, citizen-centric identity provider. Many countries have implemented (or are planning) national-level, healthcare-specific identity systems for citizens (such as e-Health cards). A better and more flexible solution is to allow other, independent identity providers for citizens (already in place in some cases) to be used in the context of health and social services. This also reduces the overhead and cost of deploying and maintaining a separate identity solution.

…Credentials

Interactions of patients/citizens and care professionals with separate health and social services providers and services often require identifiers that are specific for each service – such as a patient number, insurance plan number, health professional ID, or care provider ID. Enabling access to multiple services electronically, when identification and access controls are implemented independently for each service, usually involves the creation of separate, different credentials for each service – which users have to manage and remember.

This is challenging enough for users, even for frequently used services like online banking. For health services perhaps accessed infrequently when an urgent need arises, remembering passwords and other identifiers for each service is even more cumbersome for users.

An e-Health platform that allows access to multiple services with a single set of credentials is a major advance, contributing to the successful adoption of e-Health services.

It is important to note that use of a single credential for *many* services does not imply that a single credential is suitable for *all* services. While there are advantages in enabling access to several services through one set of credentials, the platform should help provide users with the freedom to choose which services they want to link to a particular set of credentials, and permit the use of several independent set of credentials if this is what they prefer. This aligns well with the principles articulated in the section *Identity Metasystem – Laws of Identity* on page *30*.

Consistent Sign On and Single Sign On

Once users have acquired an e-Health credential, making it possible to access a growing range of services with that single credential is beneficial both for the user, and for the providers of services – minimizing the incremental cost and overhead for enabling access to each subsequent service.

Accessing several services with the same credential can be at different levels, called “consistent sign on” and “single sign on”. The simplest form is *consistent sign on*, where it is possible to provide access to many separate services with a single credential, while still requiring the user to log on explicitly to each individual service. Users benefit from having to maintain and remember only a single credential, but there is an explicit logon step involved for each service (though, in some cases, this explicit logon step may be desirable). Consistent sign on implementation can be through independent services, each with its own user authentication database. These databases help maintain and synchronize the data so that all the separate copies of the sign on credential are the same.

Alternatively, with a *single sign on*, multiple services may rely on a single common authentication service (identity provider), instead of each implementing its own. This helps provide the desired consistency automatically, because there is a single instance of the user’s credentials. A more advanced scenario allows users to access multiple services transparently, after logging on explicitly only once. This helps provide a seamless user experience, and may allow for aggregation of services. Portal sites that offer a view across all services while interacting with individual back-end services are a good example of this.

Identity Mapping

The use of a single credential to access multiple services can certainly be useful, but it helps with only one part of the problem: improving the user experience. The target systems still may need the appropriate service-specific identifiers that help distinguish the user in the context of that particular system—for example, the patient number for electronic health records, the senior citizen ID for home care arrangements, insurance or health plan ID for payments, and so on. These identifiers are essential for helping with the successful mapping and processing of requests, and are typically meaningless outside of the context of the specific service. Most existing back-end systems depend on such distinct service-specific identifiers, and providing the mapping from a single credential to the appropriate service-specific identifiers is essential for the effective delivery of a growing range of services through a common e-Health platform.

During interaction with a service, it is sufficient to authenticate the user (with the single user credential if appropriate), and then pass only the service-specific identifiers relevant to that interaction on to the service, without disclosing the common user identifier that may have been used to correlate the user's identity across multiple services. Where such correlation is needed and beneficial (for example, to provide a better, more seamless user experience and aggregated services), it has to be established explicitly and with the appropriate user consent. Silent enablement is not an option, due to both good practice and legal constraints—for example; some countries such as France, Portugal, and the United Kingdom have explicit regulations prohibiting such correlation.

Helping to provide a reliable and secure one-way mapping of the generic user identity to the service-specific identifiers appropriate for the context of the interaction is one of the fundamental challenges that a common e-Health infrastructure, shared by all participating services, can address.

In many countries, there is no universal national identifier for citizens, and each service uses its own service-specific identifiers. Even in countries where all citizens have unique identifiers (such as Ireland, Belgium, France, and Bulgaria), and healthcare systems can use these to identify the user, there are limitations. For example, where multiple relationships are possible for a single person with a target service— such as dealing with several hospitals, or health insurance providers, or social services— the citizen identifier alone is often not enough to distinguish each relationship, and a service-specific identifier is required.

However, the direct use of citizen identifiers or a national patient number by services, even where technically possible, also raises concerns about privacy, and violates some of the fundamental principles of identity management. This includes *Minimal Disclosure for a Constrained Use* and *Directed Identity*, as discussed later in the section *Identity Metasystem – Laws of Identity* on page *30*.

When the relationships are more complex, the need to map the single credential to the appropriate service-specific identity for each context is even more obvious. There are numerous scenarios common in e-Health services that the simple traditional models for access management cannot adequately handle. While the principle of "a single credential for each user, providing access to multiple services" is a good foundation, there are cases where more complex one-to-many or many-to-many relationships between users and target services have to be handled, for example:

- Several individuals, each with their own credentials, accessing the same target service in the same context (in other words, using the same service-specific identifiers). This may be the case of several care professionals from a single organization, accessing a patient's results in an external lab facility.
- A duly-appointed representative, with a single credential, interacting with a single target service but acting on behalf of many clients with different contexts (different service-specific identifiers) for each client. This could be a relative acting as "health manager" for family members.

A mapping of individual credentials to the target service can help support this type of access authorization, where each access requires specific context identification. It is also important to ensure traceability and non-repudiation of actions by individuals, including auditing of the credentials used—something the common practice of sharing individual credentials effectively circumvents.

Maintaining and keeping track of such mappings (if they do not change too frequently) is possible within a common e-Health infrastructure shared by all services, instead of individually by each service. Apart from the advantage of implementing it only once, existing back-end systems are rarely capable of easily accommodating such new functionality. In cases where mappings are very dynamic (such as those related to shift and team assignments in a hospital), the relevant information may be managed better in lower-level systems.

Scale

The number of users of e-Health and e-Care solutions, whose identities have to be managed, is significant. Deploying and managing a national-level identity system for care professionals may be a challenge beyond the capabilities of typical enterprise identity products (for example, NHS in England employs around 1.2 million people!). Subjects of care are the entire population of the region or country—often tens of millions. In addition to typical technical performance and scale considerations, the requirements and procedures for provisioning of users and providing assistance on such a large scale have to be analyzed very carefully. For example, even with the most optimistic (low) estimates of usage, the cost of staffing and providing a Help Desk to deal with inevitable user issues (password/PIN reset, lost or damaged cards, etc.) for millions of people can easily exceed the entire budget for an e-Health or e-Care solution—yet another reason why tapping into and reusing identity providers already in place is highly recommended.

Scope

The scope of the identity system required to support advanced care collaboration scenarios is typically larger than a single organization. Even when a single national-level identity solution for care professionals (such as health workers) is envisaged, it is important to consider also the existing organizational structures and practices – which are needed to support the provisioning and management of users. These professionals will still need access to local resources (computers, networks, buildings) – and if the “national” identity solution exists independently in parallel, there could be significant overhead and inconvenience (multiple separate logons, several cards, and so on) for the users.

Even if a single national identity system for health professionals is successfully deployed (making all these members of a virtual single “organization”), the next challenge is the authentication and access for users *outside* of this security realm—for example, social and elderly care workers, who belong to other organizations. One approach—adding (and managing) each one of these individually as “external users”—is difficult and costly. A better alternative is to federate the identity systems, leaving each organization to manage its own users, and establishing trust relationships at the organizational, rather than individual, level.

After enabling care provider users from different organizations to collaborate securely and effectively, the next challenge is to allow the subjects of care (the more numerous general population) to get access to data and services. As soon as this access becomes targeted and personalized beyond just publishing generic information (for example, involves personal data), these users have to be authenticated by some identity provider – which is most likely to be separate and independent from the identity provider for care workers. This scenario also leads naturally to federated identity solutions.

Authorization

Requirements and regulations governing access to data and functionality in Health and Social Care are often quite specific, and not easily serviced by mainstream approaches and products. In this section, we are reviewing the most significant such “deviations”, requiring special attention when architecting e-Health and e-Care systems.

Role-Based Access

Role-based access (the mainstream approach for the majority of computer systems) remains applicable to Health and Social Care. It allows access rights to be defined at a "logical" level, in relation to a "role" – and applied automatically to all individuals assigned to the role. However, while in most other industries the mapping of users to roles is relatively static, in our case changes could be much more frequent, if the granularity of roles is too fine or extends to team assignments, working shifts, and so on. Managing such dynamic attributes of users by assigning them to roles and groups through traditional directory and user management tools may prove cumbersome, and alternative approaches are needed.

Legitimate Relationship

Coarse-grained roles are only the initial step in determining the level and scope of access to personal information—for example, a doctor, nurse, or social worker would normally have access to different categories (subsets) or information needed to perform his or her duties. An important concept in many jurisdictions is the explicit "legitimate relationship" link between a specific individual (patient, citizen) and a care provider ("My GP", "My Case Worker") – which restricts access to specifically assigned individuals (rather than all users belonging to a role). This is often linked to privacy regulations, scope of consent given (implicitly or explicitly), and other established practices – which historically matched the physical storage of records. With the proliferation of electronic records and the technical ability to share them between care providers, providing access to information (and improved quality of care) has to be carefully balanced with the privacy constraints.

Sealed Envelopes

In addition to the generic rules for role-based access, uniformly applicable to everybody, many countries also have the provision for further control by the individual of the access to particularly sensitive parts of their health history. The concept of "sealed envelopes" places additional restrictions, allowing access to chosen types of information (such as sexual and psychiatric histories) only in certain circumstances or by separate explicit consent.

Emergency Override

Another commonly seen interesting requirement is to provide a special "emergency override" capability for accessing patient information regardless of the roles, applicable access rules, and "sealed envelopes" – if such access is deemed necessary by the care provider. This is typically invoked in life-threatening situations, when quick and easy access is critical. At the same time, providing such functionality can open up opportunities for misuse (such as using "emergency access" in routine scenarios simply because it is easier than the normal procedures). It is important to find the right balance between servicing genuine emergency needs and preventing abuse—typically through strong auditing, notification, and subsequent review mechanisms.

Identity Metasystem – Laws of Identity

The key principles (Laws of Identity: http://www.identityblog.com) of digital identity systems, formulated by Kim Cameron in 2005, are now widely adopted by the industry:

User Control and Consent – *Technical identity systems must only reveal information identifying a user with the user's consent.* Putting the user in control (what identities are used, what information is disclosed, and for what purpose) is essential for gaining the user's trust.

Minimal Disclosure for a Constrained Use – The solution that discloses the least amount of identifying information and best limits its use is the most stable long-term solution. To minimize the risk and potential damage from a security breach, *identity information should be disclosed on a "need-to-know" basis, and only to the extent needed*. For example, in order to confirm someone's age, it is not necessary to disclose the exact date of birth – which is valuable identifying information in itself. Citizen Identity numbers in use by many countries (which include the date of birth within the number itself) are in clear breach of this principle.

Justifiable Parties – Digital identity systems must be designed so the disclosure of identifying information is *limited to parties having a necessary and justifiable place in a given identity relationship*. While it is usually beneficial to consolidate identity providers for different government services, using the same identity by the individual to access more private services may not be acceptable.

Directed Identity – *The identity system must support "unidirectional" identifiers to prevent unnecessary release of correlation handles.* Directed identity allows identification of the person to the particular service, but the identifiers used cannot be linked with any other service used by the same person. Some of the technologies currently in use (RFID, Bluetooth) are not compliant with this principle – they can emit identifying information indiscriminately – and propositions for their use to identify individuals (in patient bracelets, passports, and ID cards) should be scrutinized very carefully to eliminate possible abuse.

Pluralism of Operators and Technologies – A universal identity system must channel and enable interworking of multiple identity technologies run by multiple identity providers. It is not realistic to expect that a single (centralized monolithic) identity system can support all possible scenarios and categories of users – this is why a metasystem (a collection of independent systems) is needed. It is not only a matter of having identity providers run by different parties, but of having identity systems that offer different features.

Human Integration – The universal identity metasystem must *define the human user to be a component* of the distributed system integrated through unambiguous human–machine communication mechanisms offering protection against identity attacks.

Consistent Experience Across Contexts – The unifying identity metasystem must guarantee its users a *simple consistent experience* while enabling separation of contexts through multiple operators and technologies. Users must be able to see their multiple identities and parties involved, and choose the one appropriate for the context through consistent, unambiguous user interface.

The main categories of "players" in the Identity Metasystem are as follows:

- Relying Parties – systems and applications that require identities
- Subjects – individuals and other entities about whom claims are made
- Identity Providers – which issue identities

Aligning the identity and access solutions with these principles, widely supported by the industry and policymakers, addresses most of the identity and access challenges discussed in the previous sections, and is highly recommended as a general approach.

Interoperability

At any level—from national and regional e-Health systems to a single organization—interoperability between a growing numbers of independent nodes is one of the major architectural challenges for e-Health initiatives. While even complex commercial systems need to integrate with a finite and typically well known from the start set of systems, e-Health solutions usually face a bigger challenge. They must provide a platform for integrating a larger, and to some degree unknown, set of current and future services. Making the integration easier and avoiding disruption to existing services is a major factor for the successful adoption of an e-Health solution.

Various options for integrating separate systems (which could be running on different platforms and software stacks) are possible, and the following sections discuss these. Choosing the most appropriate option depends on the specific constraints, type of interoperability required, and other factors.

Note: Detailed generic Microsoft guidance on this topic is available in the document *Integration Patterns* available at http://msdn.microsoft.com/practices/patterns/default.aspx?pull=/library/en-us/dnpag/html/intpatt.asp. Here we focus primarily on the aspects most relevant to e-Health.

Interoperability Layers

As discussed in the section *Integration and Interoperability in Health and Social Care* in Part 1 of this guide, there are several aspects of interoperability that have to be catered for. To paraphrase, we need to provide interoperability at the following levels (starting from the most basic and moving up):

- *Technical* interoperability – providing the basic means for connecting participating nodes
- *Syntactic* interoperability – general rules and protocols governing the initiation and termination of connections; types, format, and sequence of messages, and so on.
- *Semantic* interoperability – allowing communicating parties to have a consistent, mutual understanding of the content of the conversation

These can be viewed as a "stack", where each layer depends on the ones below – but are also independent, and each level can be realized by a variety of providers. Such separation and flexibility, common in other industries, allows significant economies by reusing existing experience, established standards, and infrastructure.

To make an analogy with the telephone system: technical interoperability is the taken-for-granted these days ability to use a mobile phone pretty much anywhere in the world; syntactic interoperability is the ability to successfully place a call (or send a text message) to anyone (possibly in a different country), and all necessary in the process exchanges of messages; semantic interoperability is the ability to actually understand each other (using common language).

The actual routing of the call (mobile network base stations, land lines, undersea cables, satellite links), the process of cross-billing between participating providers, and other mechanics remain hidden within the infrastructure – and may vary widely depending on the circumstances. Naturally, just being able to call someone is of little use if you cannot understand each other – and this remains something the parties have to agree on and handle between themselves, the telephone infrastructure does not offer any help (yet). However, you still would not expect the phone (or network) you use when speaking French to be different from the one for Spanish.

In many cases today, the e-Health systems are still where the telephony (or banking, or many other industries) were some decades ago—islands of incompatible systems, protocols, terminology, and coding.

The discussions that follow, and the different interoperability options presented, can include any subset of these levels: for example, interoperability adapters (in the central hub or in the remote spokes) may support not only technical and syntactic, but also semantic interoperability—conversions of language, terminology, and so on.

Role of Standards and Interoperability Profiles

Industry standards can help achieve interoperability by narrowing down the wide variety of options available at any of the levels discussed above in *Interoperability Layers* to a more manageable subset.

However, agreeing on a standard does not automatically guarantee interoperability. Without widespread adoption and support in commercial products and tools, a standard (even when officially endorsed by an authoritative body) may remain outside of the mainstream development, an exotic aberration few can understand, and wane unused. Sometimes, this is a result of sudden changes in the direction of technology development—for example, the use of the Internet universal communication medium—and often hard to predict. Technical superiority or other positive qualities of a standard are not a guarantee either, and there are plenty of such examples in IT and other areas.

The other potential problem is that standards are usually too broad, with many options to choose from – and flawless interoperability resulting purely from adherence to a standard is a more of a lucky coincidence when both parties made the same choices (or both sides are implemented by the same developer). In addition to that, there could be multiple standards available at different *Interoperability Layers*. – increasing the number of possible permutations of individual choices made. The recommended way to handle the latter is to promote isolation and independence between the layers -- so that, for example, a decision on how to achieve interoperability at the transport or messaging infrastructure layer does not impact the choices and standards available at other levels.

An approach gaining popularity in the industry, and promoted by policy bodies (such as the European Commission mandate on health interoperability M/403[1]), is the creation of *interoperability profiles*. Targeting a specific usage scenario, these profiles define the applicable set of existing standards and a particular way of using them to achieve interoperability. Reference implementations from different vendors are tested to confirm compliance with the profile and provide practical demonstrations of cross-platform interoperability. As a result, compliance with established interoperability profiles provides a good foundation for achieving interoperability in e-Health systems. The most relevant to our area are the interoperability profiles for *Web Services Interoperability* (WS-I[2]) and those developed by *Integrating the Healthcare Enterprise* (IHE[3]).

For a more detailed discussion, see the section *Utilizing Standards for Greater Interoperability* on page *100*.

Part 5 – References of this guide contains a detailed list of pointers to standards, related bodies, and resources.

Integrating Natively

Native integration between two systems is possible when they are compatible to the required level. In other words, there is compatibility between the infrastructure and networking systems, the data access features, the service and component model, the integration of processes, security, and identity implementation, and systems management. If two systems can interact successfully, then successful integration may require only minor extra effort.

This was generally the case when both systems were running on the same platform, using compatible software, object model, and development tools. However, a service-oriented architecture (SOA) helps such native interoperability to exist across a much broader range of platforms, as long as these are compliant with the industry standards.

[1] http://www.ehealth-interop.nen.nl/

[2] http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[3] http://www.ihe.net

Native integration is the "ideal case" that e-Health systems should aim for whenever possible because it minimizes the effort required for successful integration. However, the architecture should also cater for cases where the constraints on existing or future systems that do not or cannot support standard SOA interfaces, such as Web Services, could otherwise prevent integration.

Adapters in the Hub

One common approach for integrating a broad range of different systems, possibly residing on diverse platforms, is the addition of a set of adapters to the central system or hub, each adapter reaching into a different type of remote system. *Figure 11* illustrates this overall approach, where the hub contains four adapters that provide access to four services potentially using different integration technologies.

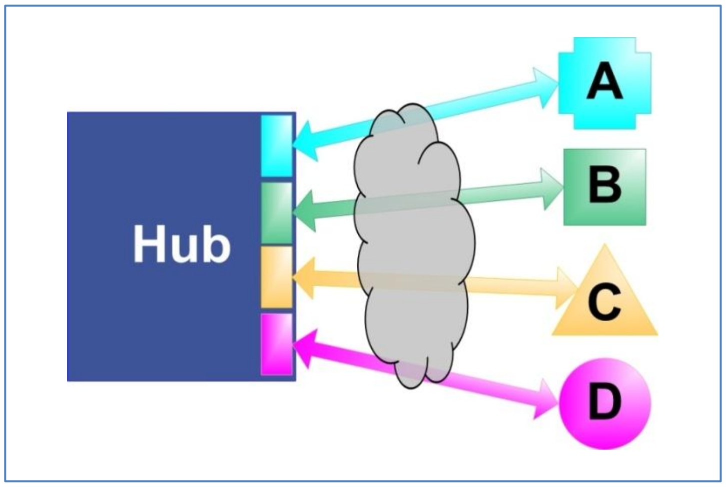

Figure 11. Using Adapters in the Hub to Integrate Disparate Systems

The advantage is that the remote systems can remain unchanged, and the hub helps take care of the extra work required by, for example, bridging network protocols, transforming data, matching semantics, and managing the flow of messages exchanged.

The disadvantages of this approach include:

- Various native protocols are propagated beyond remote systems all the way to the hub where the adapters reside, which may be cumbersome or impossible over a wide-area network or the Internet.
- Integration of new types of systems requires changes to the hub in the form of a new adapter, which may result in a disruption to other services.
- Scalability is limited, especially when the number and variety of systems to integrate with grows over time.

For these reasons, the "adapters in the central hub" approach is not usually feasible beyond the concept of a few adapters supporting widely used and well-established standards.

Adapters in the Remote Spokes

Another variation of the adapter approach described above is the use of “remote adapters”. In this case, matching of the specifics of a given system, such as network connectivity and protocols, data formats, and semantics, occurs within the remote system. Each system is responsible for its own integration up to the level of some common standard.

The widely adopted and recommended current standard for this is Web Services. Fronted by the appropriate adapter, such remote systems can integrate natively with the hub. The amount of custom integration work depends on how different the remote system is from the common standard but, generally, it is comparable with the effort required in custom integration through the "adapter in the hub" approach, and only the location is different.

The adapters may become part of the original remote system, or implemented separately on some intermediary platform. This second approach may be useful when changes to the existing systems are not possible. *Figure 12* shows the high-level architecture where the adapters are located at, but are not part of, the remote system.

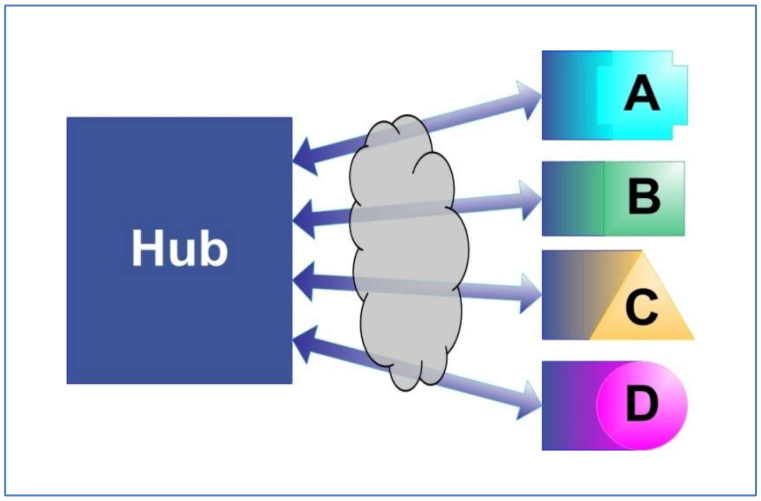

Figure 12. Using Remote Service Adaptors to Integrate Disparate Systems

The main characteristics of this approach are:

- Each system integrates to a common standard and the integration is implemented locally to the remote system, potentially making it easier.
- Communications to all remote systems over a wide area network (WAN), or other networks such as the Internet, can be standardized to use the same protocol and interfaces.
- New services can easily be added because the central hub is not affected. There is no need for new adapters within the hub, and all remote services can integrate in the same standardized way.
- The integration effort is still multiplied by the number of remote systems, and potentially repeated, though some commonality may exist between the adapters for individual services.

Isolating Common Functionality

Isolation of the generic common functionality required for the integration of remote systems with the central hub—such as security, reliable delivery, and message store-and-forward—from the integration that is specific, customized, and unique to each target system provides certain advantages. It is possible to reuse a common implementation of such a generic integration base, which eliminates repeated and redundant effort and leaves only the requirement for specific integration implementation against each target service.

Figure 13 shows how specific parts of the integration implementation, marked "**X**" in the figure, can be reused in more than one adapter, reducing development effort and integration timescales.

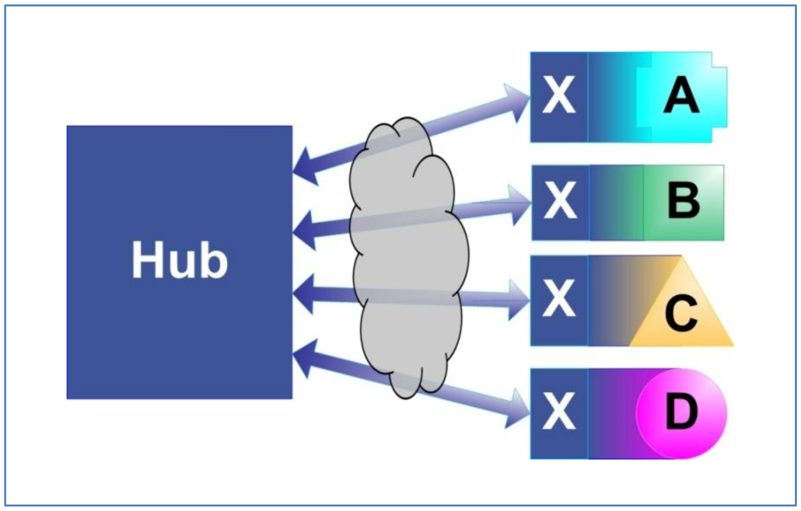

Figure 13. Reusing Generic Integration Features in Multiple Remote Adapters

The development, testing, and maintenance of such a "base" or "starter" integration solution can be done once, and then reused by a growing number of participating nodes – significantly simplifying the integration.

Investing once in the design, development, and testing of such a common integration solution, and offering it to affiliated services as a base for their specific integration, can provide significant savings while still assuring the quality of the base solution. The common integration solution can deal effectively with most of the difficult issues around reliable remote communication, security, and other requirements, most of which may require expertise not readily available at each healthcare agency—especially as the spread of e-Health services reaches smaller entities such as local authorities. Service-specific integration, when performed locally to the target system, can be much simpler and easier than full integration all the way to the hub.

Microsoft has published for free reuse the *Health Connection Engine*, which is an example of such a "starter" integration solution, implementing most of the commonly needed functionality. (http://www.microsoft.com/industry/healthcare/technology/solutions.mspx#HealthConnectionEngine)

Combining this architectural approach with the appropriate governance and commercial arrangements—for example, offering a complete base integration solution that includes hardware, software, installation, and support—can be a significant positive driver for the successful adoption of online delivery for a growing range of e-Health services. The positive experience in several countries demonstrates the advantages of such an approach. It is a much better proposition than providing prospective participants with just a specification for common integration with the hub, and leaving them to do the rest on their own.

Securing the Solution

Of all the issues that face the architect and developer of an e-Health solution, security is the one that can potentially cause the most problems if not implemented in sufficient breadth and depth throughout the entire system. Often, security is treated as an "add-on" feature that is considered only after design and implementation of the solution is complete. This approach invites disaster, and instead it is imperative to build security into the architecture of the application right from the start.

Why It Is Important

All Web sites, services, and applications must protect themselves from intrusion and damage caused by malicious persons, automated code such as viruses and worms, and service interruptions such as denial of service (DoS) attacks. However, e-Health projects carry specific risks – generally beyond those even of financial institutions such as online banking sites. This is because they present a high-profile target for:

- Politically-inspired attackers determined to infiltrate the system and cause as much havoc to services as possible.
- Attacks on individual users for any number of non-political reasons—for example, attempting to destroy a user's confidence, intimidate, or prevent access to services for which they are entitled.
- Attempts to gain access to sensitive personal information about prominent individuals (politicians, celebrities).
- Persons attempting to perpetrate identity theft. For example, obtaining details of a user's identifying numbers, address, health records, and other detailed information that can be used to impersonate the user and aid identity theft.
- Fraud or attempted fraud if users can make unauthorized changes to the data—for example, by avoiding or reducing payment for services.

Some organizations and institutions can "hide" the results of security breaches; however, e-Health applications are usually highly visible, and so any successful attacks are likely to be equally visible to the community. Such events will prove embarrassing both politically and in terms of public relations with suppliers and users. The result of even minor security failures can add unplanned cost and inconvenience to users and healthcare organizations by destroying confidence in the system, and thereby making acceptance harder to achieve in the long term.

Architecting Secure Solutions

The section *Securing the System* (page *110*) of the *Reference Architecture* later in this guide contains specific advice and guidance for helping design and build secure solutions. This includes techniques for using multiple layers of security for achieving the three main goals of confidentiality, data integrity, and reliable authentication – as well as understanding the issues of threat modeling and the suggested countermeasures available.

However, security goes beyond the technological implementation, and any design process must also address other considerations for architecting a solution that can adapt to an ever-changing security landscape. These include the following:

- Designing for the expansion of services and features over time
- Implementing secure storage of sensitive project data, and devices such as digital keys and certificates
- Managing the development team and the implementation methodology to ensure that best practices are followed
- Testing the application is all scenarios to ensure that the architecture design is robust and implements the planned security requirements

- Hardening the environment during and post deployment, and ensuring that it meets configuration and installation requirements
- Monitoring the performance of the application and services, auditing processes, and creating suitable action plans to manage intrusion or data corruption
- Implementing an ongoing process to respond to any newly found security risks, and regularly updating software and hardware with the latest service packs

Microsoft provides a wealth of guidelines, patterns and practices, systematic development documentation, and best practice white papers to help design, build, and deploy secure solutions. See the Microsoft Security Developer Center at http://msdn.microsoft.com/security for more information.

Scalability and Performance

E-Health solutions typically have to meet stringent requirements for performance and scalability. By their very nature, many of the interactions with healthcare services are infrequent, perhaps annually or quarterly, and with significant seasonal peaks. Providing adequate throughput capacity to handle such peak loads within acceptable performance targets is critical for the successful adoption of e-Health services. Any failures such as delays or unavailability of the service during these busy periods are highly embarrassing, undermining public confidence in the services and affecting negatively one of the key drivers of using electronic services—the ability to conduct electronic interactions faster and easier than through traditional channels.

Starting small, e-Health solutions tend to grow over time as the popularity of the online services and the number of active users increases. Where delivery of these services is through a shared common infrastructure, there is also growth of the number of services offered, in addition to the increased usage of each service. Therefore, it is important to scrutinize and validate any “estimated” and “projected” growth numbers – as these often tend to be overly optimistic.

Measuring such requirements against available known metrics (current volumes of interactions with services through traditional channels, penetration of online services in other similar areas) and constraints (network capacity, back-end systems capabilities) is a useful initial filter. There have been cases where a simple calculation of the volume of interactions multiplied by the average estimated size resulted in volumes of data traffic far exceeding the actual network capacity, providing a clear indication that the projected volumes were unrealistic and unachievable due to other constraints. Blindly accepting such “projected” performance requirements, and trying to meet them, can unnecessarily complicate the design and implementation of an e-Health platform.

The topic *Performance and Scalability* (page *116*) in the *Reference Architecture* section of this guide contains advice on designing e-Health solutions for high performance and scalability.

Availability, Resilience, and Disaster Recovery

As part of the design brief for implementing a scalable system that meets performance requirements, there are other related issues. Lack of performance obviously affects availability of the solution by causing intermittent or even extended failures when users attempt to connect to and use the system. Providing high availability means ensuring that all parts of the infrastructure, such as the hardware, software, and network, are resilient. In other words, a single point of failure should not cause a catastrophic effect.

In fact, availability and resilience cover a multitude of issues connected with the architecture of the solution. The topic *Performance and Scalability* in the *Reference Architecture* section of this guide discusses the way that hardware scaling can help to protect against individual hardware and software failures by employing multiple and in some cases redundant backup components that take the processing if required.

Of course, no one can guarantee that failures will never occur. It could be a simple software fault, a failed hardware component, or even a street contractor digging through your network feed. More dire circumstances, such a serious fire or structural accident at the location where the servers reside, will almost guarantee that the application will fail. Other issues to consider are malicious operators, innocent mistakes by staff, or data errors that destroy whole disk arrays.

To cope with such issues requires a well-defined, thorough, and fully tested emergency backup plan to be in place pending such a disaster. This plan will attempt to cover all eventualities ranging from simply reloading data from backup disks or tapes, to moving the entire operation to a different location with equipment and software installed ready for immediate use. Enterprise suppliers provide a range of such services, which can help to greatly reduce the downtime for vitally important applications such as e-Health solutions that must be available at all times.

Enabling Advanced e-Health Services

E-Health is not just about converting paper records to electronic ones, but also about making these electronic records available, when required, to healthcare providers across the chain, from general practitioners to hospitals and pharmacies. Beyond this, it applies ICT to telemedicine, enabling, for example, patients to have a consultation with hospital-based specialists over a video link from their GP’s surgery. It permits analysis results and images to be transmitted for assessment by specialists. It allows patients to monitor chronic conditions themselves from home.

These services may be provided to different types of users: care professionals, accessing a variety of systems; administrators, for example, who wish to manage patients’ admission records electronically; or the general public via information portals established by governments and care providers to enhance public awareness of health issues and promote improvements in public health.

When developed independently, each service may implement the identification and communication processes in a different way, and users may have to maintain separate credentials (such as user names and passwords) for every service with which they interact, as illustrated in *Figure 14*. As the range of the available services online grows, so does the complexity of managing user identification, and communicating securely and reliably with that user.

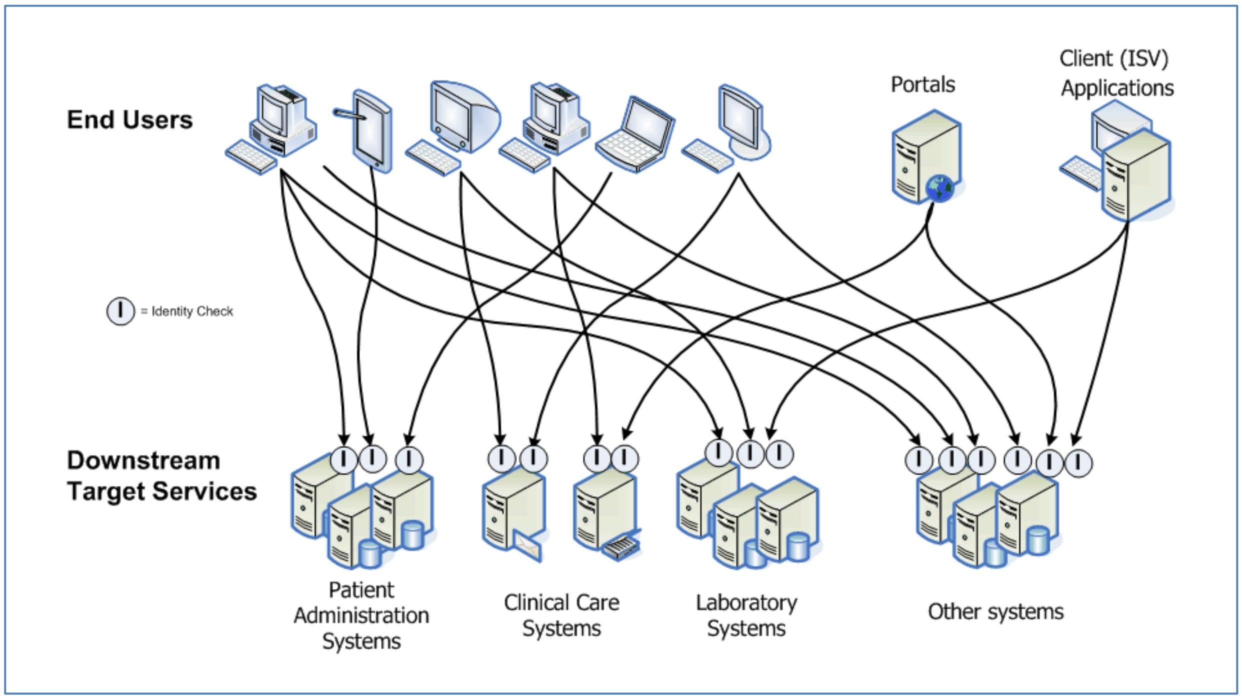

Figure 14. Each client deals independently with each back-end service

Obviously, this approach leads to problems with duplication of development effort, multiple identities for each user, and makes it difficult to achieve any joint orchestration of processes. For example, users have to log on to each service in turn to register changes such as a new address or updating of a contact phone number.

The concept of a common gateway or hub may provide a solution to some of these issues. *Figure 15* shows how a common hub or gateway might act as a broker or integrator for multiple back-end systems. Now each user connects to a single endpoint, and a single set of credentials provides identification. Development effort can be concentrated on providing a secure and reliable connection point for all kinds of users. The gateway or hub takes over all responsibility for helping the user connect to the appropriate back-end service.

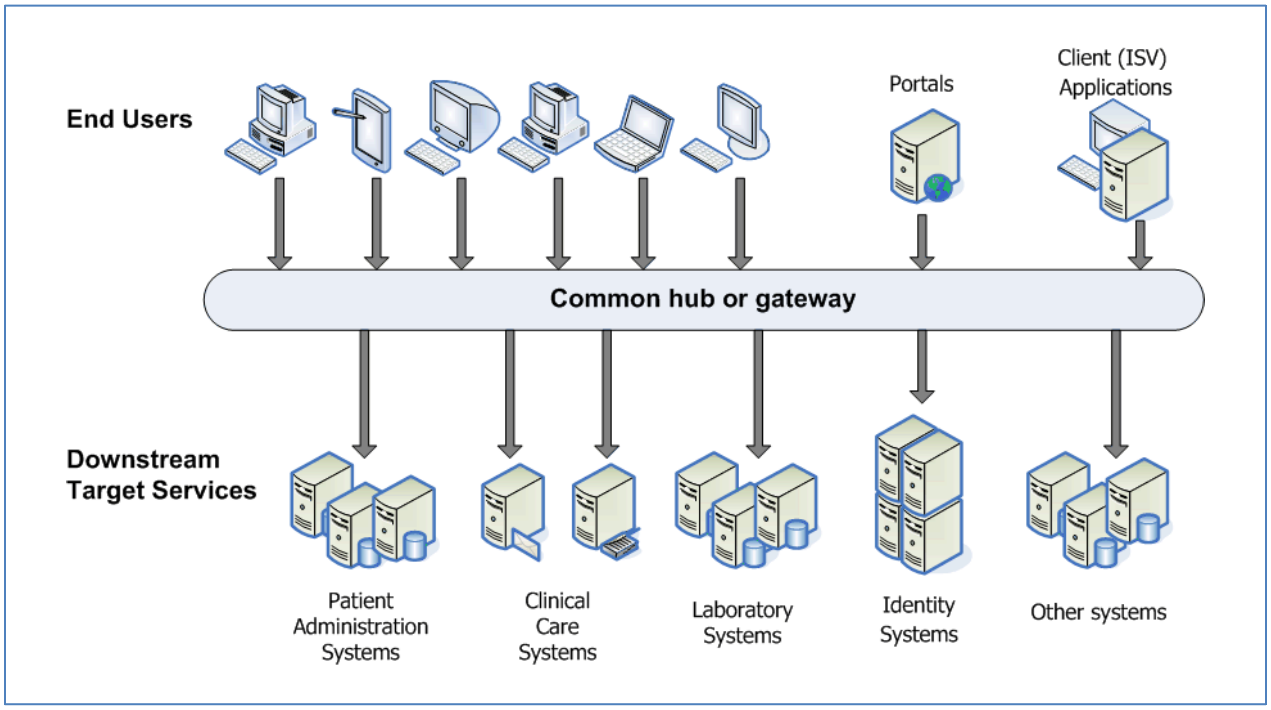

Figure 15. A common hub or gateway rationalizes communication between clients and back-end services

This still does not address all the issues, however. For example, how does the hub know which back-end service the user has permission to access, and how does each back-end service know that the user has been correctly identified? Extending the gateway or hub to perform two separate sets of tasks can solve this problem, and allow users to perform multiple operations in one go (for example, updating their address).

The first set of tasks is not only to identify (authenticate) the users, but also to have knowledge of the services that they may use and their status within that service. For example, which users have registered for, and have permission to access, patient records online? Alternatively, which users can act as an agent for other users—perhaps by being able to retrieve patient records on behalf of their senior clinician? For this to be possible, the hub must be able to route messages to the back-end service and access those using credentials provided by the user; or with some equivalent "service ticket" that indicates the correct authentication of the user.

The second set of tasks is to orchestrate the processes that the user performs. This involves extending the capabilities of the gateway or hub to expose functionality to the user. For example, it may provide the interface necessary for changing an address or phone number, and then update all the other back-end services that store and use this information. It also allows one service to pass messages and updates on to other services, perhaps prompting the patient administration service to send out "change of circumstances" notifications and the appropriate information when a birth is recorded (see *Figure 16*).

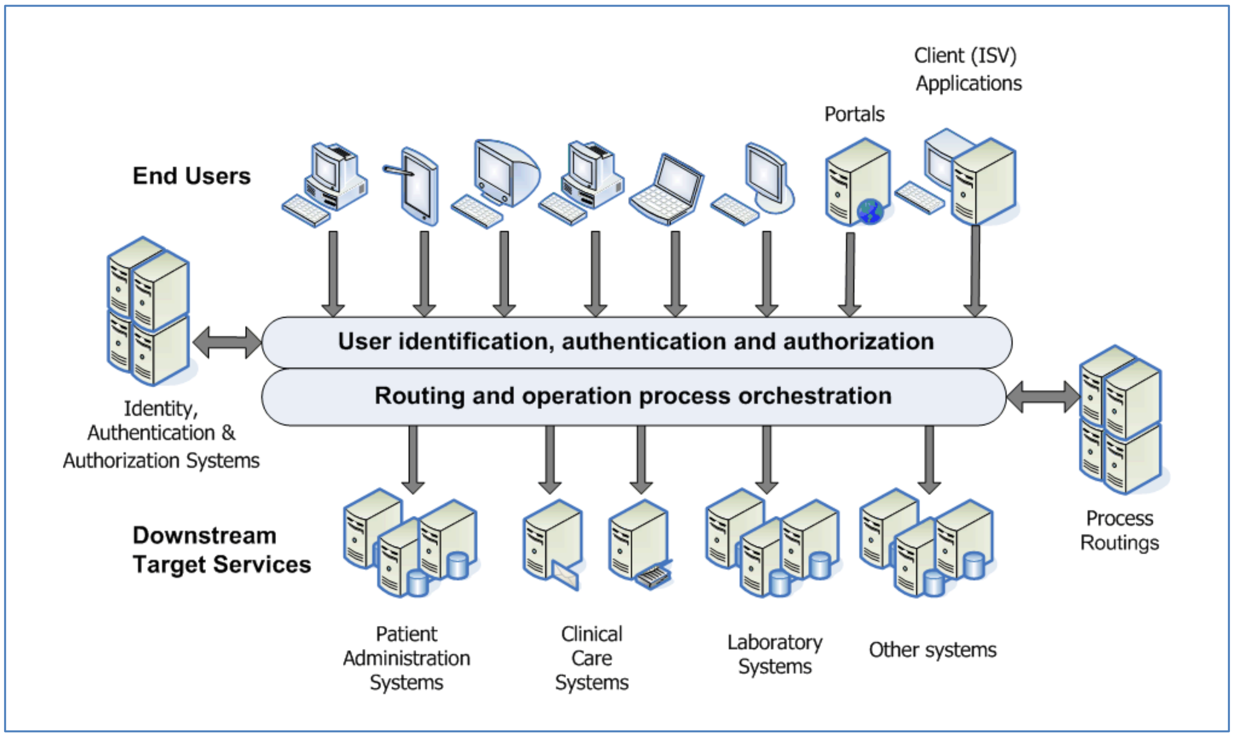
Client (ISV) Applications
Portals
End Users
User identification, authentication and authorization
Routing and operation process orchestration
Identity, Authentication & Authorization Systems
Process Routings
Downstream Target Services
Patient Administration Systems
Clinical Care Systems
Laboratory Systems
Other systems

Figure 16. The hub can also take over responsibilities for message routing and process orchestration

These diagrams show only the most basic types of infrastructure, indicating why the gateway or hub approach may be an appropriate approach for integrating e-Health systems. In the later sections of this guide, the topology and architecture is explored in more depth.

Realizing the Value of Common Infrastructure

A common e-Health integration solution, shared between many services, can very effectively address many of the architectural challenges discussed so far. For more details, see the *Reference Architecture* section of this guide on page *44*. While such an approach has definite advantages, implementing it may be challenging due to political, commercial, and other constraints. The following sections discuss the most common issues and challenges.

The Need for an Owner and Sponsor

In many countries, healthcare agencies implement segmented e-Health initiatives with little or no coordination, often as an extension of their current systems and projects. Even if some central healthcare body in charge of e-Health initiatives exists, it may have limited power to facilitate and coordinate implementation, and no funding to support the actual design, development, and deployment of a shared infrastructure. As a result, even though most agree with the benefits of having a shared integration platform and see the resulting savings, it may be difficult in practice to deliver it in the absence of a clear owner and sponsor.

The best results occur where there is a central body with the necessary authority and funding to promote the adoption of e-Health services, and capable of driving delivery of a common integration platform. Such a body also acts as a facilitator, hosting discussions and driving for a consensus among stakeholders, and as a custodian of the overall architecture.

Initial Investment with Delayed Future Benefits

Investing in the architecture, design, development, and deployment of a shared integration platform builds a foundation for the successful and efficient delivery of e-Health services. However, the initial phases deliver primarily infrastructure, which is not visible to the users and stakeholders, and does not directly generate any benefits. It only enables the success of e-Health services built on this foundation, as they become available.

To be able to demonstrate the possible benefits from the platform, quick deployment of a carefully selected initial set of services is essential. This makes the benefits to the service providers and their constituency more tangible. In addition, if the platform is architected and implemented, the cumulative net benefit will grow with each new service connected to it, while the incremental cost of new additions will be lower.

Common Conflicts with Individual Projects

When designing and implementing the shared e-Health platform alongside the initial set of services that will use it, there is often a tension between the generic nature of the common platform and the specific requirements of each service. While such friction can be creative and positive in providing early validation of the actual fit between the platform providers and the service providers, it can also influence negatively the design of the common platform to adopt and implement functionality that does not belong there.

There is a natural tendency, especially under the pressure of tight timescales and budgets, for each owner of a part of the whole to push the boundary between services and get someone else to do particular work. For example, as discussed in the section on

Interoperability, the integration work necessary to connect a node to the common infrastructure can be performed either at that node or at the hub. Combined with the fact that e-Health projects are often driven primarily by one, or few, powerful agencies (and the scope and budget for the common infrastructure may be carved out from their own), the pressure to implement as common functionality—which is not genuinely generic and reusable, but saves a particular participant time and cost—can be very hard to resist. However, this will inevitably compromise the integrity of the architecture, and will affect subsequent services – which may all have to adapt to changes made to suit only a few.

It is essential to safeguard the integrity of the architecture from attempts to insert into the common shared infrastructure services and functionality that are not genuinely generic and reusable by the majority of participants.

Reference Architecture

This section presents typical reference architecture for implementing an e-Health solution at any level—from local enterprise to regional to national and cross-agency systems. It consists of the following sections:

- ***Principles Guiding the Architecture*** (page *45*) lists the overall principles guiding the architecture, as discussed in more detail in other parts of this guide.
- ***The Connected Health Services Node*** (page *46*) introduces the concept of a generic e-Health node providing a common infrastructure that can be shared by multiple providers of e-Health services.
- ***Services Provided in the e-Health Reference Architecture*** (page *48*) discusses the various services that may be provided through the e-Health Services Hub including:
    - *Identity Management Services* (page *49*)
    - *Authentication and Authorization Services* (page *59*)
    - *Service Publication and Discovery Services* (page *71*)
    - *e-Health Business Services* (page *75*)
    - *Electronic Health Record Services* (page *76*)
    - *Health Domain Services* (page *78*)
    - *Health Registry Services* (page *78*)
    - *Integration Services* (page *79*)
    - *Data Services* (page *98*)
    - *Communication Services* (page *99*)
- ***Deployment Options*** (page *102*) looks at various deployment models to be considered when implementing the e-Health reference architecture against varying jurisdictional requirements and constraints.
- ***Securing the System*** (page *110*) provides pointers to available general guidance on building secure systems, and security architecture aspects specific to e-Health systems.
- ***Performance and Scalability*** (page *116*) looks at the approaches for creating a solution that meets the criteria for availability, robustness, and performance, capable of growing over time to support increased usage and a broader range of services.

Principles Guiding the Architecture

In order to tackle successfully the challenges discussed in *Addressing Common Architectural Challenges* earlier in this guide, the Connected Health Framework reference architecture described in this section follows the key principles listed below.

Flexible and Agile

As discussed in the section *Flexibility and Agility* (page *9*), being able to cope effectively with changes and adapt to accommodate new requirements is essential for e-Health solutions aiming to provide an agile platform for a growing range of participants and services.

Flexibility provided by the architecture and built in from the start is the key for the success of e-Health solutions. The requirement for extending and adapting the core infrastructure in response to the changing scope and requirements (such as adding new services, participating nodes, authentication methods) is a mainstream use case, properly supported by the framework, processes, and tools, and not an afterthought or periodic activity between major releases.

The architecture should also have the flexibility to support e-Health solutions at all levels: enterprise, regional, national, international – as well as cross-agency systems.

Service Oriented

Adopting a Service Oriented Architecture (SOA) approach to the implementation of large, complex software systems like e-Health integration solutions allows effective addressing of the fundamental challenge of ensuring that these systems are adaptable, flexible, and reliable in the face of change.

The functionality of the e-Health integration platform should be accessible as a set of generic services, available for use independently and as required. Service Oriented Architecture (SOA) is a framework and a set of policies and practices that help enable implemented application functionality to be published as services at a granularity relevant to the service requestor. They abstract the implementation by using platform-agnostic, standards-based interfaces.

Interfaces and Standards

In order to achieve the required independence for different participating nodes from the implementation and platform, and in compliance with principles of Service Oriented Architecture, interfaces between autonomous subsystems and nodes should be defined using applicable industry standards. This should be the preferred approach for all *Interoperability Layers* (see page *32*)—technical, syntactic, and semantic.

At the technical infrastructure level, Web Services are becoming the dominant way of exposing services in an industry-standard, implementation-independent, and platform-independent way. All major vendors are adopting Web Services standards, and there is a growing range of software products supporting these standards. They will gradually reduce the share of custom application development in favor of the use of commercial products where appropriate. Relying on the products and tools to implement fundamental features natively, such as security and reliable messaging, will reduce the cost and effort required to implement the required functionality.

For more details and pointers to the full set of Web Services specifications, see *Part 5 – References* of this guide.

Service Discovery

The reference architecture should include an infrastructure for registration and discovery (at design-time and/or run-time) of the available services, and a repository for the relevant metadata such as schemas, interfaces, and policies – using applicable industry standards.

Federated Security

As discussed in the section *Identity and Access* (page *24*), the architecture must support a variety of authentication methods, credential types, identity providers, authorization methods, and trust models—some not yet known. This architecture should provide a generic framework, with the ability to add new providers and methods over time, and construct systems with various topologies to satisfy specific requirements and constraints. It should not assume that central authentication and management of all potential users is possible by a single provider.

Adoption of the principles discussed in *Identity Metasystem – Laws of Identity* (page *30*) and Web Services standards in this space will allow interoperability between different implementations and effective use of commercial products with these capabilities as they become available.

Secure

Multiple layers of protection and different aspects of security should be part of the architecture from the start to enable the construction of solutions to the appropriate levels of security for e-Health services. The security model itself should be flexible, and permit adoption of various current and emerging security technologies without major redesign.

Scalable and Performant

The architecture should provide adequate performance, and support growth to meet increasing demand and a widening range of services offered.

The Connected Health Services Node

A common infrastructure implementing a set of generic reusable services, shared by multiple providers of e-Health services, overcomes many of the challenges outlined in the section *Addressing Common Architectural Challenges*. In the context of the Connected Health Framework we use the term Connected Health Services Node.

While the benefits of sharing common services in such a node generally increase with the number of services using it, instead of each one implementing similar functionality separately, there is no assumption or requirement to have only a single node for all e-Health services. Multiple nodes may exist, implementing to varying degrees a subset of the full range of services, and cooperating with each other in a variety of topologies—hierarchies, peer-to-peer networks, and so on. The aim is to provide flexibility capable of accommodating different requirements and topologies, and achieve adaptability to the required scale of implementation – while following the same generic blueprint with a core set of common services.

The e-Health Services Node provides a common set of services available to different types of clients. The node itself acts as a client in turn, calling other external systems (services)—see *Figure 17*.

Client Interactions

Architecturally, everything accessing the services provided by the e-Health Services Node is considered a client. These can be Web sites and portals servicing their own users but relying on the services in the node, applications running on client systems, or any other systems that request services from the node.

Typical categories of clients:

- **Portals** – which may use the identity and security services provided by the node to authenticate their users, perform maintenance functions, submit documents on behalf of the user through the node messaging services, or access other services.
- **Client Applications** – such as clinical applications running on client workstations or servers, may communicate with the node to submit requests and access data and other functionality.
- **Other Systems** (or other nodes) – which act as clients and access the node functionality. These can be third-party, back-end (healthcare agency) systems, or other nodes.

All client interaction is through published interfaces, based on industry standards. This provides the necessary openness, compatibility with a wide range of commercial products, and cross-platform interoperability. Client interactions with the e-Health Services Node are independent from the actual implementation and platform used by the clients.

It is important to think ahead, and expose the full functionality of the hub services via programmatic interfaces. Some implementations of such systems may include a Web user interface as part of the hub, as the only way to access some basic functionality. Over time, however, many providers prefer to use the hub functionality from their own applications and portals – providing better, more seamless user experience. Adding programmatic interfaces to a running system is far more difficult than including them in the design from the start.

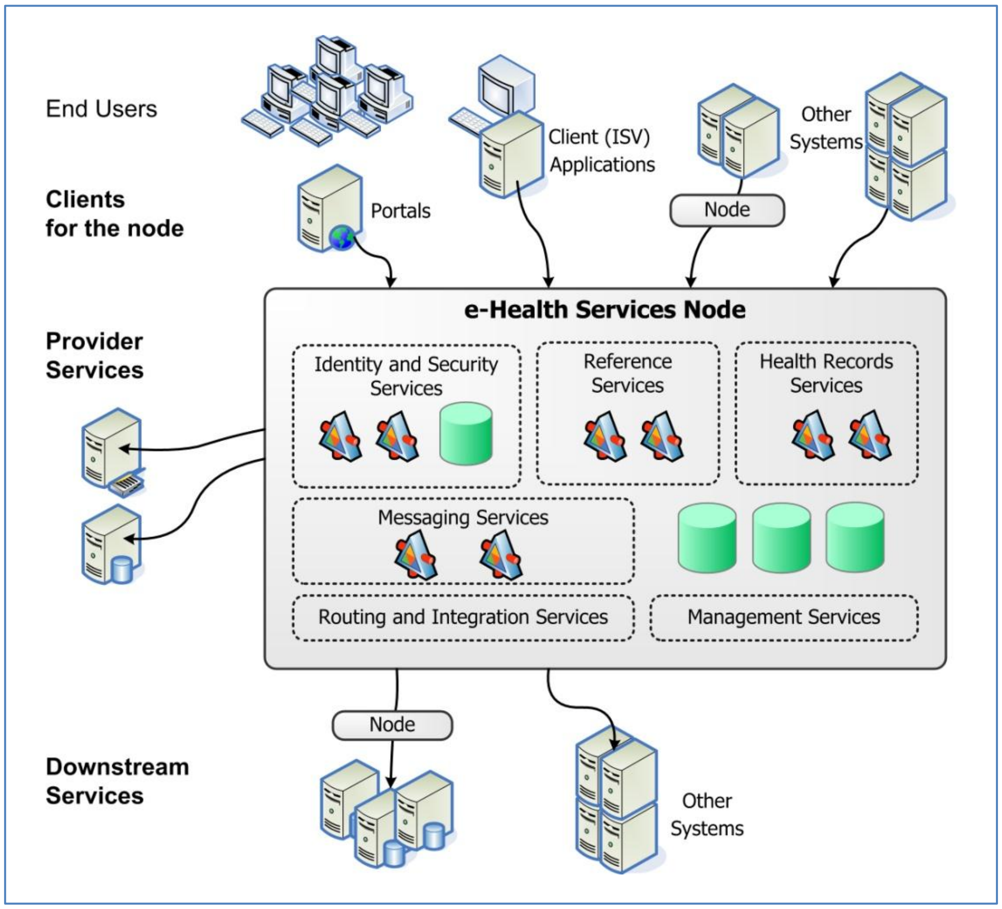

Figure 17. Context and External Interactions of an e-Health Services Node

Interactions with External Services and Nodes

The e-Health Services Node acts in turn as a client when accessing other (external to the node) services. Architecturally, these interactions are the same – but the major types are:

- **Provider Services** the node relies on for its core functionality – like external authentication providers called by the node to verify credentials.
- **Downstream target services** that receive requests from clients sent through the node. The hub acts as an intermediary and may add some value within the routing path, such as validation. The target services could be healthcare agencies or other systems, or integration hubs fronting such systems.

Services Provided in the e-Health Reference Architecture

The e-Health Reference Architecture provides a comprehensive set of generic services, different subsets of which are likely to be present in any instance of an e-Health node – depending on its role and position in the chosen topology.

The following sections describe major design considerations and aspects specific to e-Health for these services.

Identity Management Services

From the Identity Metasystem perspective (see *Identity Metasystem – Laws of Identity* on page 30), an e-Health node can play the role of *identity provider*, *relying party*, or both – depending on the positioning of the node in the overall topology of the system, and distribution of functionality between nodes. Respectively, some of the Identity Management services discussed in the following sections may be only partially (or not at all) present in a particular node.

Core Identity Model and Principles

This section defines the core entities within the identity model, their relationships, and the principles driving the identity management aspects of the architecture. The generic model for identity management should be open and flexible enough to accommodate different (current and future) requirements for different types of authentication, mapping of users to services, delegation of authority, and so on.

Core Entities within the Identity Model

The following terms describe entities that represent the core identity:

- **User** (claimant, constituent) – the individual to be identified, authenticated, and authorized to use certain services. Credentials and identity represent users within the model.
- **Identity** – represents and identifies a user within the identity management system. A user can choose to maintain several separate identities, typically representing different roles or groupings of services that remain segregated.
- **Credential** – information or other items that are verifiable to assert an identity. Multiple credentials may be associated with the same identity. Typical examples are a user identifier and password, a digital certificate, and so on.
- **Service** – a logical grouping of business functionality offered by a service provider, with consistent rules governing authorization and access for users. A healthcare agency can offer several separate services, and the access rules and levels of authentication required may be different for each one. Alternatively, grouping the business functionality from several agencies as a single service is possible as long as these are consistent and there is a clear ownership and responsibility assigned for the aggregated service.
- **Enrollment** – the linkage of identity to a particular service, with the relevant service-specific context attributes (identifiers). A valid and active enrollment entitles the owner (or a duly authorized representative) to access the service in the context defined by identifiers.
- **Identifiers** – information attributes attached to an enrollment, which uniquely identify and provide context for the relationship of an identity with the respective service. Examples of such identifiers are a citizen tax reference, a national insurance number, a social security number, a company registration identifier, a value-added tax registration number, a property tax identifier, a utility provider identifier, or an account number.
- **Group** – represents a collection of users sharing the same enrollments, typically as representatives of an organization.
- **Role** – a broad category of identity used to define or constrain access to the appropriate services for all users within that identity. The typical set of roles includes:
    - Individual (citizen, patient, clinician, consumer, client, and so on) – represents the clients for healthcare services.

- Organization (group) – represents organizations (businesses) where multiple individuals in a group can share the same enrollments and act on behalf of the organization.
- Intermediary (agent, delegate) – the permanently or temporarily appointed representative of an individual or organization, authorized to act on its behalf in the context of specified service.
- Healthcare – individuals and systems representing healthcare agencies and authorized to access specific services.

The Identity Model

Figure 18 illustrates the relationships between the entities listed above within the identity model. Note that:

- A User can have multiple Credentials.
- Each Credential maps to an Identity, and the same Identity may be asserted by several Credentials.
- Each Identity links to a single Role, which determines the subset of Services available to that Identity.
- Multiple Identities may link to a Group.
- A Group can own multiple Enrollments for Services.
- Each Enrollment maps to a single Service.
- Each Enrollment for a Service has associated Identifiers, uniquely identifying the context of the relationship between the owner of the Enrollment and the Service.

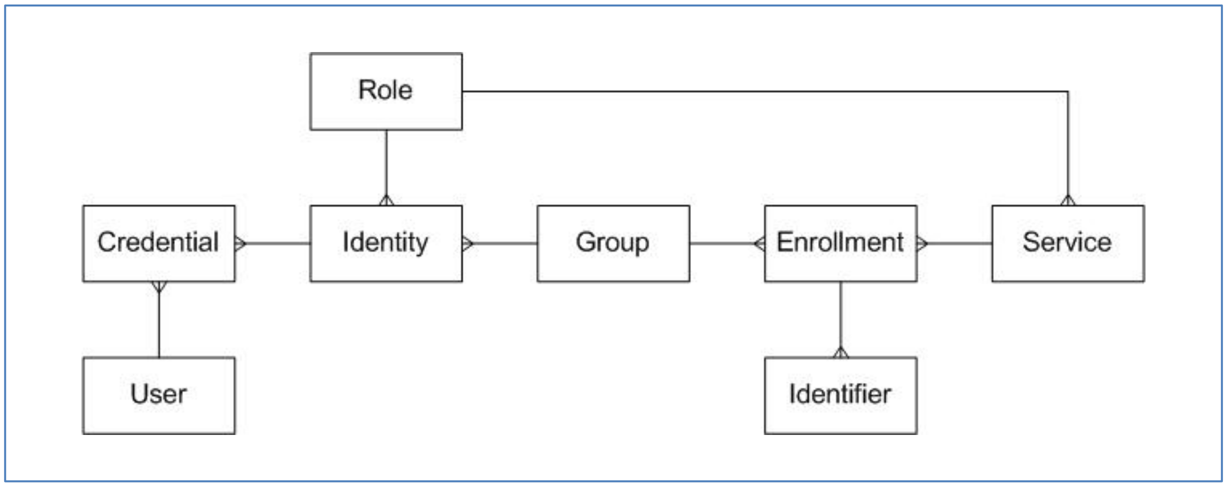

Figure 18. The Identity Model

Separation of Identity from Authentication Credentials

An important principle that ensures the flexibility of the model is the separation of identity from authentication credentials. The use and independent verification of multiple credentials may assert a single identity. This can be useful for authentication through different delivery channels, where the technology and usage scenarios restrict the ability to present some types of credentials (for example, a lack of smart card readers, or only numeric data entry is available).

This multiplicity of credentials can evolve and extend an existing set of relationships (enrollments) by upgrading to higher levels of authentication – enabling access to a broader range of services with minimum overhead and disruption. Such separation of credentials from identity also allows external credential authentication, outside of the e-Health Services Node. This is the foundation for federated authentication and other advanced scenarios.

Different authentication providers may be involved, depending on the type of credential and its issuer, yet the rest of the authorization and mapping to services remains generic. The relationship of several credentials to a single identity supports this.

Support for Federated Authentication

As a forward-looking solution, expected to be in use without major redesign for many years, it is imperative to consider support for a federated authentication model from the start. In other words, assume there will be multiple authentication and identity providers. To achieve this, it is important to segregate pure authentication (implemented natively by existing and future products and technologies) from the specific functionality like the mapping of users to services (which is likely to be specific and require custom-designed processes).

Federated authentication should be based on the current industry standards such as WS-Trust, WS-MetadataExchange, WS-SecurityPolicy, and WS-Federation (see *Part 5 - References* of this guide for more details)—to ensure interoperability across different platforms and providers, and to maximize the use of commercial products implementing those standards.

For more details on the Microsoft vision for an interoperable architecture for digital identity that assumes people will have several digital identities based on multiple underlying technologies, implementations, and providers, see Kim Cameron's Identity Blog (http://www.identityblog.com) and specifically *Identity Software + Services Roadmap* (http://www.identityblog.com/wp-content/images/2009/01/PDC-2008.pdf).

Generic Pattern – Pluggable Providers

The implementation of identity services follows the same generic pattern, as illustrated in *Figure 19*. It consists of a common unified service interface front-end with a variable number of "pluggable providers", which may be of the following core types:

- An **internal provider** of the service, with the relevant reference data being held locally – this is the traditional "in house" approach to identity and authentication, with data in a directory or other storage
- Some **internal processing**, where the logic is in the code so there is no dependency on reference data – typically used to perform fast and efficient validation of a security token
- As an **interface** to an external provider of the relevant service, which may include remote reference data – when we rely on "outsourced" identity and authentication services, for example, sharing a common identity service for citizens

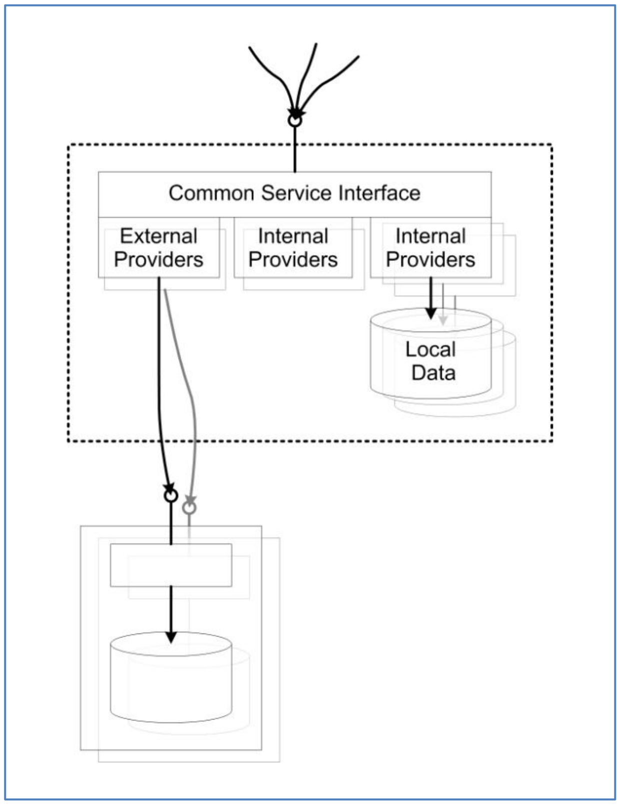

Figure 19. Generic Pattern for Pluggable Providers

Multiple instances of the same type of provider may be present, for example, performing different types of validation or accessing different external providers. The goal is to have all external interactions based on Web Services standards, but where this is not possible or feasible the interface component can perform the necessary transformation to the appropriate (legacy or otherwise non-standard) interface of the provider.

Initial User Provisioning – Knowledge-Based Authentication

The process for initial provisioning (identification and registration) of users for a large-scale e-Health solution has to be efficient, secure, and require minimum human interaction from the service provider side. A “self-service” feature driven by the users themselves is the ideal. Flexibility to define the specific rules and process for each service separately is essential for the successful accommodation of evolving and diverging requirements.

Given the challenges specific to e-Health solutions, such as a large number of prospective users, no prior *online* relationship, and infrequent interactions, an approach named Knowledge-Based Authentication (KBA) is gaining popularity. Although these methods are especially useful for large constituencies of unmanaged users (like citizens), they can also be used to improve the efficiency of provisioning managed users (like care practitioners).

Briefly, KBA uses information provided by the “claimant” to verify the claimed identity – which could be related to the user in general (identifying the user), or to the context of a specific service (for example, date of the visit with a physician or medical record number).

Figure 20 shows the generic architecture of a knowledge-based authentication system. The flow of data, indicated by the numbers in the figure, is as follows:

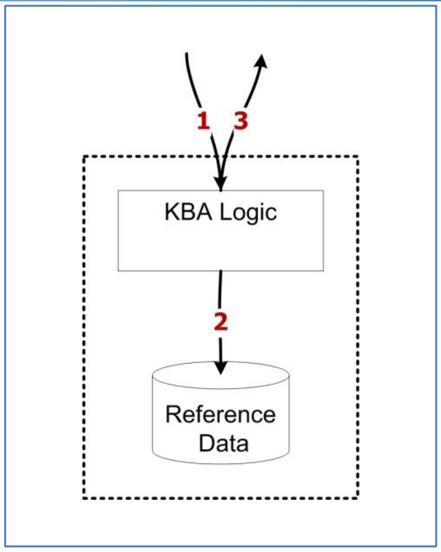

Figure 20. Generic Architecture of Knowledge-Based Authentication Service

1. The claimant presents a set of information items (perhaps preceded by challenging the claimant with a set of questions).
2. The KBA logic matches the submitted items (and any answers) with the reference data, which may be local or remote.
3. The response indicates authentication success or failure, and can contain additional information—for example, the claimant's unique identifier that results from the matching process, which may be different from the information items provided.

Note: For more details of Knowledge-Based Authentication, see *KBA Applicability to e-Gov* by Mindy Rudell, Dick Stewart, Robin Medlock, Angel Rivera: http://csrc.nist.gov/archive/kba/Presentations/Day%202/Rudell%20-%20KBA%20Applicability%20to%20e-Gov.pdf

---

Choosing Appropriate Information Items for KBA

The reliability of KBA authentication depends on the nature of the information items chosen and the validation process. Typical requirements driving these choices include the following:

- The information is clearly bound to and *uniquely identifies* the claimant—a zip/postal code by itself is clearly not enough, but a citizen ID or social security number might be.
- The information is legitimately known to the claimant, but *not readily available* to others. A company registration number printed on letterhead and visible to all clients is not suitable, but some specific data from the last health services bill, like a medical record number or payment amount (which is normally known only by the legitimate recipient), could be.
- The *scope for guessing* the correct answer or value *is limited*—the range is big enough to make trying all possible values unfeasible, and there is no easy way of deriving a correct value from another known legitimate one. (For example, incremental numbers should be avoided.)

- *Values that change* regularly over time (for example, the last claim or payment amount) can further reduce the likelihood of a successful guess, hence providing a better choice than constant data items.

Note that these requirements apply to the set as a whole. In other words, several information items, each one of which may not itself be sufficiently secure, can be combined to achieve the appropriate level of assurance that only the legitimate claimant would know them all.

Verification Logic and Reference Data

After the claimant has supplied the set of information, verification takes place, typically against some reference data, by applying appropriate matching rules. Certain transformations are possible for approximate matching—for example, making the comparison case-insensitive, ignoring spaces, partial matching of items like post codes and addresses, or tolerating spelling variations and abbreviations. In general, however, it is difficult to define sufficiently robust matching rules for items that have a wide variety of possible formats (like addresses), and so the use of items with more predictable formats is generally preferable.

The matching rules can also include cross-item dependencies. For example, a successful match of three out of four items may be deemed sufficient. Since the matching logic can be custom-developed and, if necessary, be different for each type of authentication and service, it can accommodate a very wide variety of requirements—both for current and future requirements. This is essential for the flexibility of the solution and is one of the key architectural principles (as described in *Flexible and Agile* in the section *Principles Guiding the Architecture* earlier in this guide).

The reference data used to perform validation can be local (provided by the respective owner and hosted in the node), or remote (where the node makes calls to the owner to perform the verification, and the data remains with the owner). It is also possible to implement verification logic that uses only the information items provided by the claimant and does not require additional reference data. In this case, the consistency of the whole set must be assured. As an example, one of the data items could be a result of some secret transformation performed on the others, similar to the check digits on credit cards, but more secure. The claimant’s unique identifier must also be provided, or be derivable from the data items supplied.

Figure 21 illustrates the three different types of validation that are possible using a pluggable provider such as a KBA service:

- A – the reference data is remote and the KBA logic calls the provider to perform the validation
- B – the validation logic is not dependent on any reference data
- C – the node hosts the validation logic and reference data, and accesses it locally

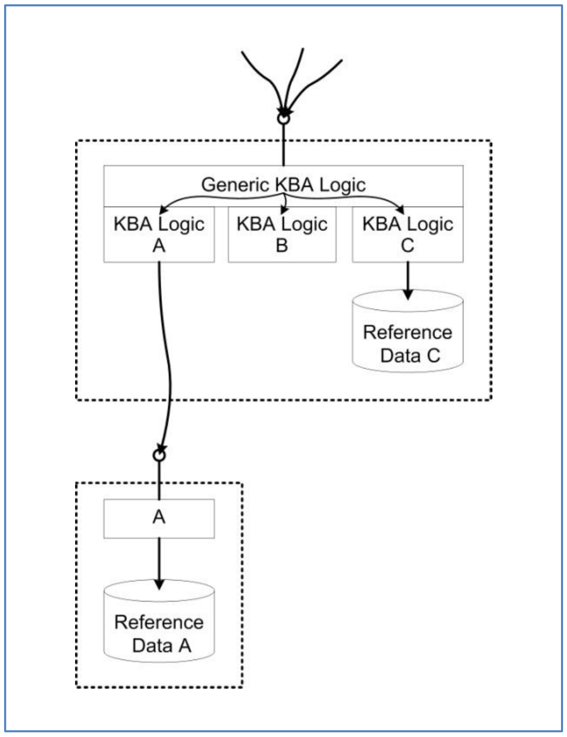

Figure 21. Different Types of Validation with Local and Remote Reference Data

The reference architecture supports "pluggable" validation modules of all types, and the decision of which to use in each case depends on numerous factors. The most important considerations include the following:

- **Response time for validation** – reference data hosted locally in the hub can help support faster validation. The latency of calls from the hub to a remote data store could be significant, impacting performance and negatively affecting the user experience.
- **Availability** – hosting reference data in the node helps make it self-contained and independent from other systems, so authentication availability is that of the node itself. Relying on remote reference data for the validation makes the availability of the solution dependent on the remote system.
- **Data volume** – the amount of reference data could be significant, and capacity planning should consider this. In cases where only a small proportion of the potential user population is likely to use the service, keeping a copy of the reference data for the whole constituency on the node could be expensive and cumbersome.
- **Updates to the reference data** – the reference data stored in the node must keep up to date with any changes originating at its source. Depending on the volatility of the data, the volume of these changes (for the whole constituency that might use the service) can be significant. Relying on reference data that remains at the source eliminates the need to update copies in the hub.
- **Privacy constraints** – in some countries, regulations prohibit hosting of service-specific data outside of the relevant agency, so uploading such reference data to a central node is not an option – regardless of other factors and technical feasibility.

The choice of the most appropriate mechanism for validation and location of the reference data may vary across different service providers.

Obtaining Information Items for Knowledge-Based Authentication

The information items provided by the claimants for knowledge-based authentication can be something they know or have received in the course of normal interactions with the service provider (such as correspondence or bills), or can be items specifically communicated for the purpose of initial online authentication. It is also possible to link together the process of obtaining the necessary data items to some procedure, accreditation, or verification of documents (presented remotely via mail or face-to-face). Knowledge-based authentication is a generic mechanism, independent from the exact mechanics of obtaining the necessary data. These processes remain out-of-band, may vary between different services, and support linking to an appropriate external process.

The Service-Centric Approach to User Provisioning

Uniform and reliable initial identification of users to a level that is acceptable for all services is often difficult to achieve. To provide the necessary flexibility, and accommodate the potentially diverging requirements that current and future service providers might have, service-specific initial user identification is preferred. This allows the definition of data items, validation rules, and reference data for each service, completely independent from other services.

Such service-specific initial authentication may be in addition to some prior validation of the claimant as individual in a generic sense (irrespective of the services concerned), or there could be no reliance on the generic user identification at all – with each service acting completely independently and not trusting any other. In the latter case, the registration of the user is conditional upon successful initial identification for at least one service. This approach is suitable for countries where no generic citizen identity is established, and initial identification must rely on the service providers for reference data and validation procedures.

Note: While it is possible to register users without initial validation, and later bind that identity to services, avoid this approach. In the absence of a reliable initial authentication, it is difficult to protect the node from denial-of-service attacks through malicious registration of a large number of bogus users (not validated for any services), because there is no mechanism to distinguish such registrations from the legitimate ones. This can tie up data storage and computing capacity, potentially affecting the availability of the service, and increase running costs. The service-centric approach ensures application of some validation before new users can register, and it is therefore safer. It also allows implementation of additional activation procedures, dependent on data provided by services (like a mailing address) that may not be available otherwise. For a more detailed discussion of the activation options, see *Activation of Registrations and Enrollments* later in this section of the guide.

Service Enrollment and Identity Mapping

After a successful initial identification, in the context of a particular service, the appropriate set of unique service-specific identifiers assists creation of an enrollment to that service. This process binds an identity with that service. The generic identity itself remains transparent and not visible to the service provider – instead it translates to the identifiers meaningful to that service. For example, enrollment may link Joe Smith with the Health Service using a Health Service ID as an attribute of the enrollment, and with the Elderly Home Help Service using a HHS ID number and other items as attributes. By storing the appropriate set of service-specific unique identifiers as attributes of the enrollment, all subsequent authentications and authorizations can map the single generic identity to the multiple service-specific identities.

This ”identity mapping” is one of the unique value-added features of the e-Health Services Node. Even if authentication is completely externalized (in other words, there is no identity store within the hub and it relies only on external authentication providers through federated trust), the mapping of the verified generic identity to the appropriate service-specific identifiers ensures that the target service receives the usual identifiers it can recognize. This complies with the principles of *Minimal Disclosure for a Constrained Use* and *Directed Identity*, as discussed in *Identity Metasystem – Laws of Identity* within the main section *Addressing Common Architectural Challenges* earlier in this guide.

Activation of Registrations and Enrollments

Knowledge-based initial identification can be supplemented with additional procedures to improve the security beyond just relying on the claimants to supply the correct information items (and on the low probability of others being able to do so successfully and pose as the legitimate claimant). These procedures can be invoked after a successful knowledge-based initial validation, and may include sending additional information to the claimant that they must then enter to complete the verification process.

A typical implementation (as used in the United Kingdom and other countries) involves obtaining the registered mailing address for the claimant from the service provider, and then sending a one-time “activation code” to that address. Only after presenting this activation code to the hub does the enrollment for the service (initially created in a “pending activation” state) become active and the functionality become available. This procedure improves security by making the successful receipt of the activation code an additional check before enabling the service enrollment.

A successful attack by someone pretending to be the legitimate claimant would require not only obtaining and presenting the correct information items for knowledge-based identification, but also intercepting and tampering with postal mail. Depending on the reliability of the mail system and the respective legal deterrents (in the United States, for example, tampering with mail is a federal offence carrying serious penalties), this can significantly improve the overall security of the solution for initial identification.

Other variations of this approach may use different channels for transmitting the activation code, such as e-mail, mobile phone, or text message. The challenge with these is the reliability of the “address” used—while postal addresses are held by most service providers, other types of addresses may not be readily available and reliable. Consideration of potential threats is necessary when choosing the method of communication to minimize the opportunities for malicious misrepresentation. One possible way of improving the reliability of an e-mail address is to use one bound to the identity of the claimant through previous verification, as is the case with e-mail addresses included in some digital certificates.

The use of a separate activation loop can improve the security, but comes at a cost:

- The necessary reliable addressing information must be available from the service provider, and transmitted (in bulk or individually, as enrollments occur) to the node to initiate the secondary loop.
- Facilities for secure printing and mailing of the activation codes (similar to the mailing of bank and credit card PINs) must be procured and deployed locally or outsourced to external providers. Not all agencies that intend to use e-Health services may have such facilities themselves, and these are typically provided though the central shared e-Health infrastructure.
- Secure communication of the activation codes and addresses to the printing facility – this is very sensitive information that should be adequately protected by technical means and operational procedures.

- Delay can be introduced into the process; printing and postal delivery extends the process from initial identification to active service enrollment by several days. This may cause significant inconvenience to prospective users, especially when they need to use e-Health services immediately only to discover that the process will take too long to actually complete.

It is possible to eliminate the delay factor, while preserving other aspects and advantages of the additional activation loop. The UK Inland Revenue (tax agency) and other services successfully implemented a modified approach called “immediate activation.” This involves generating activation codes and communicating these in advance to the prospective users of a service.

The initial knowledge-based identification of users for the service can optionally extend to request the activation code along with other information items. The reference data includes the activation codes sent to the users, and the validation procedure tries to match these as well. If the claimant provides the correct code, activation of service enrollment takes place at creation and the respective services are available to the user immediately. By changing the sequence of the separate verification loops (mailing of activation codes still occurs, but in advance), such modified processes can provide the same level of security as delayed activation but without the delay. The advance mailing of activation codes also acts as an unsolicited invitation, raising the awareness of the availability of the online service among its constituency and increasing the take-up.

A Generic Credential and Authentication Provider

The identity management functionality of the node relies on one or more authentication providers, some of which may be external. In such cases, the node makes calls to these external providers to verify presented credentials. It is possible for a node to rely only on such external providers, and not to have a credential store of its own. However, it is also possible for the node to have an internal credential and authentication provider. This allows users who have not established prior relationship with one of the accredited external credential providers to register successfully, and the node to then issue the appropriate credentials.

To cater for a variety of evolving requirements for different types of credentials, it is important for the generic credential provider in the node to be flexible in supporting and issuing different types of credentials. Specific rules are configurable on installation of the provider, or dynamically at run time. Some examples of different types of credentials are as follows:

- **User ID and Password** – the credential provider generates User IDs that are globally unique within the scope of the node, with a preconfigured format and length, and users can either choose or are issued an initial password (again, in the appropriate format and with the appropriate complexity rules applied). For successful authentication, a full and valid User ID and Password combination is required.
- **Memorable Words** (with or without prompts) – these can be combined with User ID and Password for added security, and required either selectively (for some sensitive operations such as a password reset) or at all times. It is common to request only parts of the memorable word, for example several randomly chosen letters (such as the third and fifth, changing for every interaction), or to have several pairs of question/answer combinations stores and ask for one of them.
- **Certificates** – the generic credential provider for the node may issue digital certificates to users. While technically feasible, operational logistics and management require consideration. Owning such a facility without the relevant experience and infrastructure can be a challenge and a distraction. In practice, e-Health implementations to date in many countries have opted for relying on external accredited and trusted parties (commercial or government agencies) to issue certificates and handle the relevant procedures.

The choice of type of credentials, and the relevant rules, should take into account the following points:

- **Prior constraints and requirements** – such as a desire to use already established unique identifiers (like a citizen ID) as the User ID, instead of randomly generated ones.
- **Scale** – the range and type of identifiers should be large enough to offer a sufficient number of unique identifiers for the potential number of users.
- **Security** – the size and complexity of the identifier and password should cover a large enough range to make brute-force attacks (guessing IDs and passwords by trying all possible combinations) unfeasible.
- **Convenience** – users should be able to comfortably handle, and ideally remember, their IDs, passwords, and other information. The longer and more complex these are, the higher the probability that users will have to write them down somewhere, undermining the increased security that the complexity was intended to provide. This decision requires a careful balance between convenience and security.
- **Delivery channels** – the choice of identifiers and passwords can constrain the use of certain delivery channels. For example, interactive voice response (IVR) telephony systems, mobile phones, and IPTV may be limited to numbers only. Again, the desire for higher security is a balance against the intended use and convenience. It may be appropriate to consider using separate sets of credentials for different channels, distinguishing the level of security and services offered.

Authentication and Authorization Services

The authentication and authorization functionality provided by the e-Health Services Node is generic, and applies to all types of identities, such as those that represent individual users (directly or through intermediaries), organizations, systems, and other entities. The functionality can be used directly—for example, by portals to verify the identity of interactive users, in which case the node acts as an identity provider and secure token service. Indirect use is also possible—for example, by the messaging services to validate the identity used to sign a request and the authorization for the target service.

Authentication

Authentication validates the credentials presented by the user and maps them to a specific identity.

Levels of Authentication

Each authentication can be associated with a level, which indicates the reliability of the authentication type and related procedures.

One such classification used in the United Kingdom is “tScheme” – where levels 0, 1, 2, 3 represent increasing reliability of the authentication, which may be appropriate for different types of online activity:

- Level 0 – ***no authentication***: the real-world identity of the registrant is not verified (for example, anonymous access, or users volunteer some information like name or e-mail address – which may be stored and used, but not verified)
- Level 1 – on the ***balance of probabilities***, the registrant’s real-world identity is verified (for example, online ordering of a publication using a credit card, with the goods delivered to the account holder’s address)
- Level 2 – there is ***substantial assurance*** that the registrant’s real-world identity is verified (for example, submission of a Value-Added Tax, Sales Tax, form, which is legally binding)

- Level 3 – verification of the registrant's real-world identity is ***beyond reasonable doubt*** (for example, an online application for a passport)

The higher the level, the greater the assurance required for the verification of the identity of the registrant. A combination of technical methods and procedures achieves the appropriate level of assurance, as prescribed and accredited by tScheme.

The advantage of using tScheme or a similar system is the uniformity of requirements and the universal acceptance (and compatibility) of providers who are accredited under tScheme by all government agencies. Establishment of something similar, to stimulate the adoption of e-Health services, is possible in countries where such a framework does not already exist.

The association of a level with each authentication, and the definition of a rule for the minimum level required for each type of operation, results in basic access control checks at various stages of processing becoming easy and very effective. For example, "Which services can be offered to a certain user for enrollment?" The answer is "those with a minimum required level less than or equal to the authentication level". Can the user submit a particular type of request? The answer is "yes, if the required minimum level for that type of request is equal to or less than the authentication level, plus any other authorization rules." This can also work when the same individual is using different methods of authentication depending on the circumstances. For example, a doctor using a mobile device to receive important notifications may get access to a subset of the normal functionality available when logged on with a smart card on a PC.

When the tScheme granularity is not sufficient to meet some specific requirements, its further extension is possible, in two ways:

- By adding more interim levels, such as 1.5, 2.1, 2.3, while preserving the simplicity of the basic rule (authentication level must be >= the minimum level required). Different types of authentication are still compared purely on their levels, so that 2.2 is always higher than 2 regardless of the authentication provider being used), and that ranking is universally accepted by all participants and providers of target services.
- By distinguishing between authentication providers, even if they are at the same level, and applying more complex rules—for example, matching both the authentication provider and the level. While this flexible approach accommodates a wide variety of specific requirements (such as "require level-3 authentication with smart card embedded in a driver's license, but do not accept level-3 authentication from another provider, like a credit card"), it limits convergence and reuse of common authentication providers across the full range of e-Health services. In the extreme, an "each service with its own authentication provider" model is not desirable, and applicable only to meet specific requirements.

In line with the principle of flexibility (see the earlier section *Principles Guiding the Architecture*), the authentication and authorization service of an e-Health Services Node should enable the full flexibility described above (smaller granularity levels and rules that are more complex) to meet requirements that may emerge over time.

Presenting Credentials and Claims

Authentication depends on the validation of credentials or claims presented by the requestor. Upon successful verification, these can be associated with the appropriate identity, which permits creation of new claims linked to that identity. The generic "provider" model enables the use of multiple providers of different types for validating credentials.

Some possible authentication flows that use different authentication providers are shown in *Figure 22*. The numbered steps within the figure demonstrate how, when the authentication request contains credentials and depending on their type and origin, validation is available:

- Directly by an internal (local) provider (**1**), which compares them against the local credential store (**2**).
- Through a call to a trusted external provider (**3**), which performs its own validation of the credentials provided.
- When the authentication request contains a security token, it is usually validated locally (**4**). This can be a security token issued by an STS trusted by the hub, or token issued by the hub itself – as a result of a previous authentication performed by the hub through any of the possible routes described earlier.

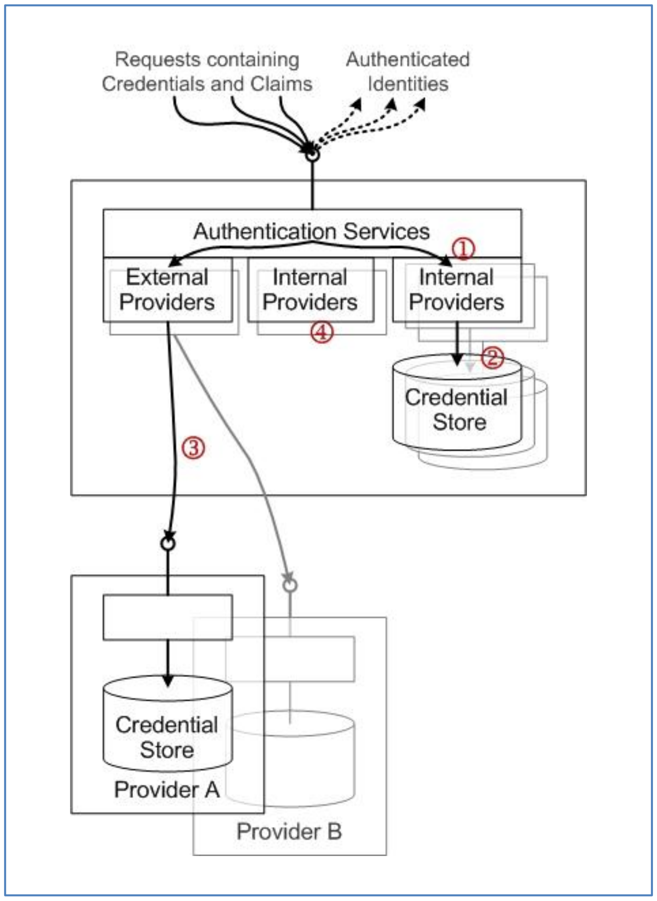

Figure 22. Invoking Different Authentication Providers

Regardless of the particular types of validation and provider invoked in the process (internal or external), a successful authentication maps the presented credentials or claims to an identity. This is the basis for further mapping to service-specific identifiers (see the later section *Mapping Identity to Service-Specific Identifiers* on page 64) and the issuing of security tokens (see the later section *Security Token Service* on page 65).

Federated Authentication

Traditional authentication models, where a single identity provider performs verification of credentials, limit the use of a wider range of pre-existing or future identity providers, and inhibit access by distinct groups of users to a common set of e-Health services.

Federated authentication provides more flexibility, which enables users from different (independent and separate) trust domains to authenticate with their credentials in the home domain, but gain access to resources in other domains – based on established trust relationships between domains. Local identities are not required for target services and thus identity information and other attributes can remain hidden as appropriate. This is in compliance with the principles of *Minimal Disclosure for a Constrained Use* and *Directed Identity*, as discussed in *Identity Metasystem – Laws of Identity* (page *30*) within the main section *Addressing Common Architectural Challenges* earlier in this guide.

A set of Web Services specifications and standards define a consistent and extensible security model for such federation that is independent from specific platforms and implementations. This permits effective integration and the use of commercial software products. For more details of the standards such as WS-Security, WS-Trust, and WS-Federation, see *Part 5 – References* of this guide.

An e-Health Services Node can perform several distinct roles and participate in different types of federated trust topologies. These topologies are often composed of multiple nodes, as discussed in the section *Deployment Options*(page *102*) later in this guide. The hub may act as:

- A ***requestor Identity Provider*** – authentication performed by the hub produces a security token for presentation to the target service to gain authorized access.
- A ***resource (Relying Party) identity provider*** – which verifies security tokens presented by requestors, and issues tokens granting access to the target.
- A ***broker of trust*** between two or more parties (as illustrated in *Peer-to-Peer on* page *105* in *Deployment Options*), in which case the trust relationship of each party with the broker establishes brokered trust between the parties themselves.

Figure 23 shows an example of the typical flows between the parties involved. The numbered steps in the figure are as follows:

4. Requestor A presents the necessary credentials to its identity provider in Node 1.
5. The Requestor Identity Provider in Node 1 validates the identity and returns an ***identity security token*** to Requestor A.
6. Requestor A presents the identity security token to the Resource Identity Provider in Node 2.
7. The Resource Identity Provider in Node 2 checks the validity and origin of the identity token and returns an ***access token*** for the target resource. The Resource Identity Provider does not need to recognize Requestor A or its identity—it trusts the Requestor Identity Provider in Node 1 and the identity token issued by it.
8. Requestor A presents the access token to Target Resource C (for which Node 2 manages security) to gain access to this resource.

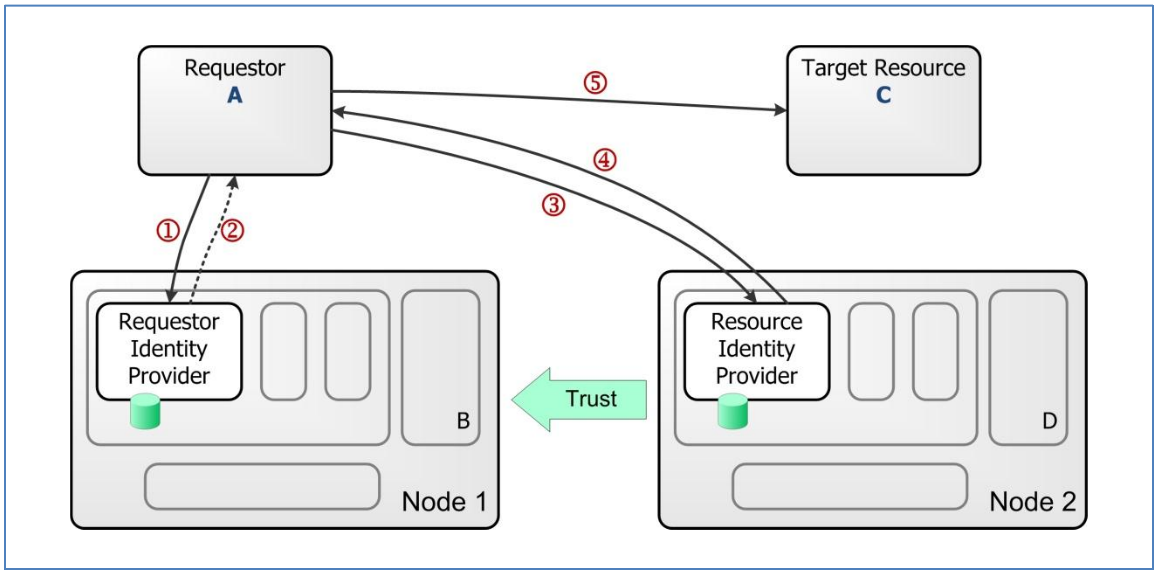

Figure 23. Federated Authentication

This example deliberately separates all parties to illustrate the interactions more clearly. Requestor A could be a portal or other system, Node 1 could be a central e-Health Services Node, and Target Resource C could be some e-Health service provider that relies on Node 2 for security and integration. However, many other variations are also possible while implementing the same general flow. For example, the messaging subsystem (B) hosted in Node 1 can also play the role of a requestor, obtaining the necessary identity and access tokens in order to send a message to the messaging subsystem (D) hosted in Node 2.

Authorization

When authentication is complete and the identity established, authorization typically follows – at the appropriate granularity for the context level. At a minimum, authentication produces some unique identifier representing the authenticated entity, but it may also provide other attributes, which can be the basis for authorization decisions (such as role or rank). In some cases, combining authentication and authorization can provide efficiency in a single call to the Authentication and Authorization subsystem.

There are two main options for authorization requests:

- **Get All** – enumerate everything for which a given identity is authorized (and which the requestor is authorized to access – see comments below)
- **Explicit** – the authorization request explicitly specifies the targets sought for authorization

While the *Get All* approach may seem simpler and easier, there are some serious considerations to take into account before deciding to use it – especially in the context of e-Health services. It may lead to unnecessary disclosure of information to the requestor, which it may or may not be entitled to see. In general, such approach could violate some of the key principles of the *Identity Metasystem – Laws of Identity* (page 30).

For example, consider a user who holds a single credential that provides access to a range of services. When accessing an e-Health Web portal to book an appointment with a doctor, that user may not wish the portal to be able to obtain identifiers related to a support group for recovering alcoholics (or even discover that the user is enrolled for such services), if these facts are not relevant to the requested service. Allowing the portal to perform

Get All authorization operations could provide information about all the relationships between that user and other services, which may not be appropriate.

The recommended approach in such cases is to use *explicit* authorization. The requestor specifies the target authorization sought, and the authorization provider responds accordingly. This is not limited to a single target at a time; the request can contain a list of target services. Additional checks may be performed based on the identity of the requestor (in case of intermediaries, checking what targets the requestor is allowed to check authorizations for), or based on an explicit consent from the user, contained in the request itself.

There are some exceptions to this general recommendation when there is a legitimate need (and the user has provided consent) to obtain authorization for all services that user has permission to access—for example, to provide a complete list of all available services for maintenance purposes.

Checking Service Enrollments

In the generic identity model discussed earlier in *Core Identity Model and Principles* (page *49*) in the main section *Identity Management Services*, the granularity of authorizations is at the level of a target service – for example, "*can or cannot access service X".* This dictates the definition of the target services – each has to be a grouping of actions (requests or document submissions) with uniform access rules.

Authorization requests must map the previously authenticated identity against the valid existing and activated enrollments for services. For *Get All* requests, authorization returns the full list of all valid enrollments mapping to the identity. *Explicit* authorization requests compare the list of services requested with valid enrollments for that identity, and returns only a subset of that list (the services for which a valid enrollment exists).

Mapping Identity to Service-Specific Identifiers

An important aspect of authorization is the mapping of the generic (and potentially common for many services) identity to the service-specific identifiers associated with each of the active service enrollments found. This allows the generic identity to remain hidden, and provides the appropriate context for the relationship with the particular service. Target services can independently select the identifiers to use, and do not have to change their back-end systems to accommodate some new identifier (unless they choose to do so). Once the generic identity has been bound to a service through the enrollment process within the context of a service-specific set of identifiers, subsequent authorization calls for that identity and target will return the identifiers. For more details, see *Service Enrollment and Identity Mapping* (page *56*) earlier in this guide.

Delegation of Trust

In the case of delegated authority, which is the assignment of other users to act on behalf of the principal (such as the mother acting on behalf of family members), the mapping process is more complicated. It involves searching for a valid chain of mappings from the identity to the target service. However, the result is fundamentally the same: "Identity A can access Service S in the context of the service-specific identifiers X, Y, and Z" (in the example above, A is the mother; X, Y, Z are the family members). This type of delegation is suitable for relatively stable, long-term delegation that remains valid until specifically revoked.

For more transient delegations (such as delegations valid only in the course of a specific interaction like submitting a request or accessing test results), the authority to delegate can be included within the authorization request itself. In this case, the process of mapping the identity to service enrollments takes into account delegation (producing the same output as a permanent delegation, but with an explicit validity time window and other limitations). However, this does not result in permanent storage of delegation relationships, and previous transient delegation does not affect subsequent requests – unless they also contain transient delegation instructions.

Granularity of Authorization

Authorization checks performed by the node are limited to the level of a target service, based on the active enrollments and any transfer of authority known to that node. Further, finer-grained authorization decisions can be made elsewhere (including in another node), based on the service-specific identifiers provided by the authenticating node. The appropriate granularity of services and the level of detail provided in the identifiers is an important design decision. This must balance the convenience of receiving detailed identifiers that may include role, rank, and other attributes from the authorization process, with the need to maintain up-to-date copies of these attributes within the hub.

An example of such "cascading" authorization could be a doctor in a hospital seeking access to some patient data. The doctor is authenticated by a central e-Health Services Node with the appropriate credentials (for example, a smart card), issued by that node or some other trusted authority, and confirmed as having active enrollment for "National e-Health" – with the appropriate unique identifier (such as a Clinician ID). Additional attributes resulting from the authentication may include more specific role or rank.

As far as the central node is concerned, the authenticated identity of the doctor allows access to "National e-Health" services, and the context is the respective service-specific IDs. Based on these IDs, the hospital portal (or any other system) can make further authorization decisions – using locally available information. For example, they can check for appropriate links to the specific patient, team or shift assignments and role, or the doctor, and permit access to information. In this scenario, the target service maintains fine-grained authorization information about the specific assignments and so on, and does not expose this information to the central node – which simply authenticates the user and authorizes to the level of target service as a whole. This allows for rational distribution of information across the system, taking into account the nature and volatility of the data – as discussed earlier in *Types of Data* (page *15*) in the main section *Addressing Common Architectural Challenges*.

Security Token Service

The e-Health Services Node can help perform the role of a Security Token Service (STS), as defined in WS-Trust, WS-Federation, and WS-Security standards (see *Part 5 - References* for links to the WS standards). The node can therefore participate in a federated network of Web Services providing authentication, authorization, and other functionality that facilitates effective integration based on industry standards and specifications.

Security Token Services issue, validate, and exchange security tokens. A requestor sends a request, and if the policy permits and the recipient's requirements allow, the requestor receives a security token response.

Security Tokens

Security tokens contain sets of claims, and the issuing authority asserts the validity of these claims. The issuer can cryptographically sign security tokens to guarantee their integrity (no alterations made since signing) and enable verification of their origin. Successful authentication and authorization by an e-Health Services Hub typically results in the return of security tokens to the caller.

Issuing Security Tokens

A security token represents the result of the authentication and authorization performed by a hub. This token will contain one or more of the following types of claims:

- Identity information resulting from the authentication, with related attributes such as the level and type of authentication
- "Authorized for X" information – for example, a list of target services

- Service-specific identifiers and attributes for each of the authorized services

Depending on the particular role of the node and its location within the overall topology, the type of claims contained in the security token will vary. The issue of multiple tokens for different target services is also possible. The next section of this guide provides some examples of possible scenarios.

Possible Scenarios

The interactions between several hubs and the roles their respective Security Token Services play can vary, depending on the chosen topology and model. *Figure 24* illustrates some of the possible variations. The numbered steps in the figure are as follows:

1. A user interacts with a Portal and presents his or her e-Health Credentials

2. The Portal sends a request to Node A (which may be a central e-Health node), presenting the user's credentials and specifying the target Service X

3. Upon successful authentication of the user and authorization for Service X, Node A returns two tokens issued by STS A:

   a. An *authentication token* **Au** that confirms successful authentication of the user, and which can be used for obtaining additional authorizations later without requiring full user authentication through credentials

   b. An *authorization token* **AzX** for the user for Service X, which contains the appropriate service-specific identifiers defining the context of the relationship of the user with Service X (Health ID or Social Services ID)

4. The Portal, on behalf of the user, makes the initial call to Service X and presents the authorization token **AzX** issued by STS A

5. STS X checks the validity of the token AzX, and issues a new token X (a token specific to this STS), containing the claims appropriate for accessing Service X. For example, it may translate or map the service-specific identifiers contained as attributes in token AzX to more specific role and other information within the context of Service X

6,7,8 … Subsequent calls from the user and Portal present token X , which can be validated very efficiently—far more efficiently than the initial validation of AzX and translation or mapping of attributes. Transfer of the appropriate service-specific identifiers for Service X, which are contained in Token **X**, into the call context also permits authorization decisions within the service

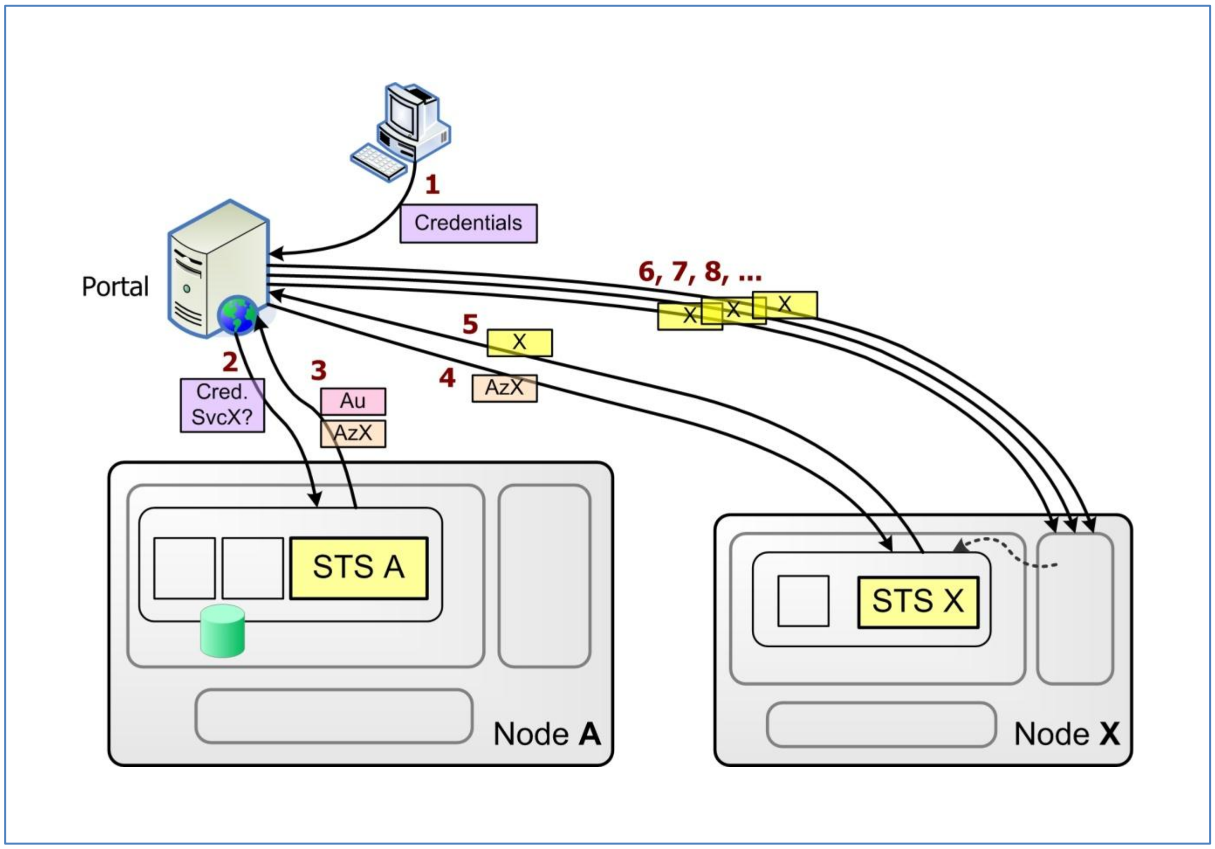

Figure 24. STS Roles and Token Exchange

Continuing from the previous example, after interacting with Service X, the user (or the Portal on behalf of the user) needs to access another Service Y. *Figure 25* illustrates this, but there are subtle differences from the previous example.

Assuming that the authentication token (**Au**) previously issued by STS A is still valid, and has been preserved reliably and securely by the intermediary or by the user, obtaining a token for accessing another Service Y could be simpler and not require the original credentials from the user. This is the concept of a single sign on across several services. The flow is as follows:

1. The user needs access to another Service **Y**
2. The Portal sends a request to Node A, presenting the authentication token **Au** previously issued by STS A, and specifying the target Service Y
3. Upon successful verification of the validity of the authentication token **Au**, and authorization of the user for Service Y, Hub A returns two tokens issued by STS A:
   a. Optionally, a refreshed or renewed *authentication token* **Au**, perhaps with an extended validity window to allow continuation of the original authentication and “single sign on across services” capability (subject to compliance with the appropriate policies that balance security and convenience)

b. An *authorization token* **AzY** for the user for Service **Y**, which contains the appropriate service-specific identifiers defining the context of the relationship of the user with Service **Y**

4. The Portal makes, on behalf of the user, the initial call to Service Y and presents the authorization token **AzY** that was issued by STS A

5. STS Y checks the validity of the token **AzY**, and issues a new token **Y** (specific to this STS), containing the claims appropriate for accessing Service Y.

6,7,8 … Subsequent calls from the user or the Portal present token **Y** , which can be validated very efficiently—far more efficiently than the initial validation of AzX and translation or mapping of attributes. Transfer of the appropriate service-specific identifiers for Service Y, which are contained in Token **Y**, into the call context also permits authorization decisions within the service

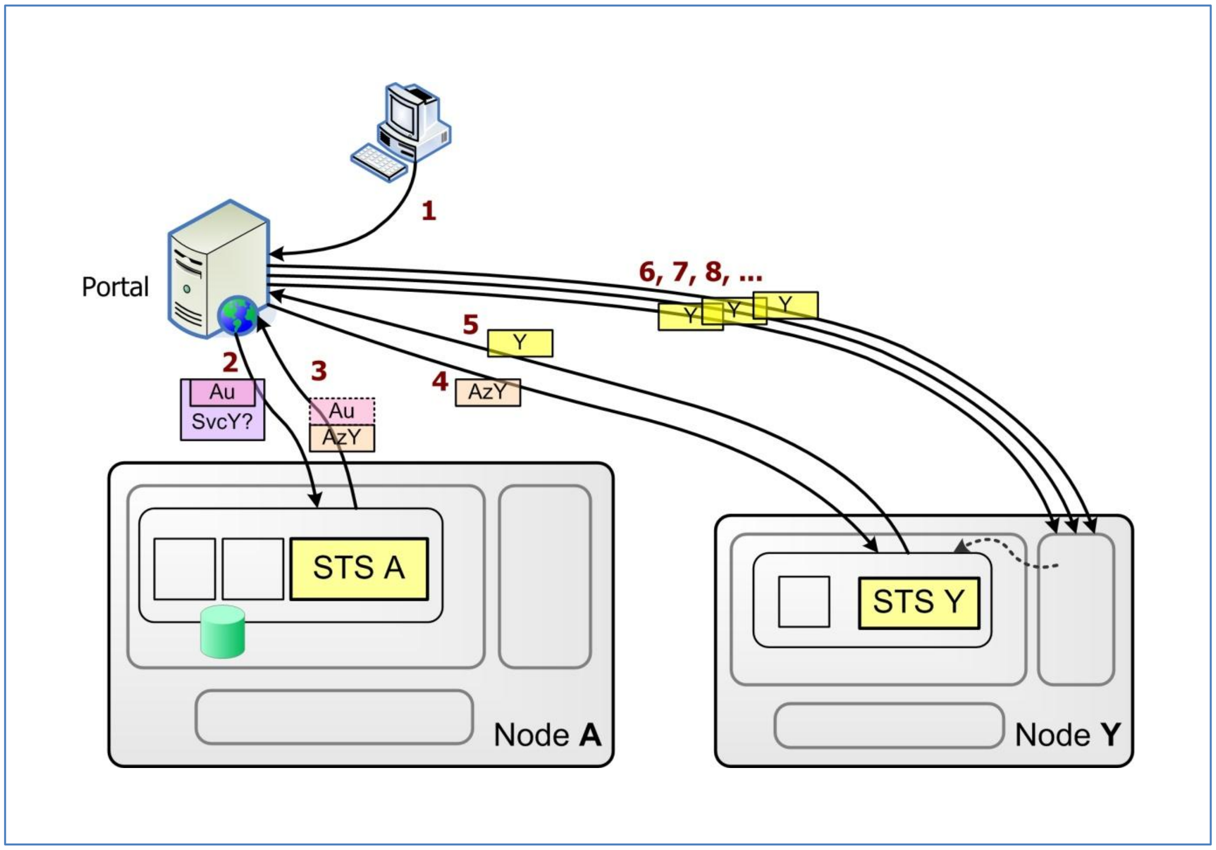

Figure 25. STS Roles and Token Exchange, Next Service

The STS functionality of the generic e-Health Services Node supports numerous other scenarios, enabling the deployment of different topologies matching the specific requirements and constraints.

Privacy Services

Data Protection and Privacy

Depending on the implementation and the chosen topology, an e-Health Services Node may host personally identifiable information related to very large groups of users—potentially the entire population of a country. Legal and regulatory constraints may affect architectural decisions, narrowing the choice of available technical options.

Identifiers and Reference Data

Even if a node is not storing any health records data, there are still data protection and privacy considerations to be taken into account. For example, if the local regulations prohibit any kind of sharing of information (even just identifiers) between healthcare agencies, or hosting agency data on infrastructure it does not control, then the initial identification of users when enrolling for services will have to be performed remotely. The central node will have to make a call to the service provider to validate the information provided by the user, instead of performing such validation against reference data previously uploaded to the central hub. (For a more detailed description of these options, see *Verification Logic and Reference Data* (page *54*) in the Identity *Management Services* section earlier in this guide.)

When reference data for individual services resides on one central node, specific measures are required to protect the access to service-specific data. For example, keeping the reference data for each service in its own separate database enables segregation of access control for individual administrators.

Indexes and Metadata

In some of the possible data topologies (as discussed in *Data Topologies* starting on page *16*), a node may contain a master index and pointers to actual data storage, potentially also performing the necessary mapping of identifiers from different domains related to the same individual. This data could be abstract enough (and free from any personally identifiable information) not to be subject to the usual personal data protection regulations, potentially making the design and compliance of such a node easier. However, in some jurisdictions (such as the United Kingdom), even such cross-service "linking" information is considered sensitive – and has to be protected accordingly.

If the "master index" facility also stores additional attributes to facilitate matching (such as a person's name or date of birth) across domain and organizational boundaries, then it will be subject to all regulations related to personally identifiable data.

In the models where one node contains only summary information (metadata—date, type of encounter, code of the condition or analysis) to facilitate searching, and the actual data (documents, test results) is stored in another node ("Registry" and "Repository" in the IHE XDS profile terminology – see *Distributed (Federated) Data Model* on page *18*), the metadata may still be considered sensitive information—for example, when the sole existence of data implies the presence of a disease/condition deemed sensitive. In such cases, the metadata store ("Registry") has to comply with the regulations governing protection of personal data, or *pseudonymization* techniques should be used (see section *Anonymization and Pseudonymization of Data* on page *70*).

Health Records Data

Actual health records data is most sensitive, and requires full protection in compliance with the applicable in the jurisdiction data privacy regulations. As discussed in *Data Topologies* (page *16*), in many cases the particular constraints of these regulations (like "*data cannot be stored outside of the legal entity which collected it*") drive the choice of topology for the e-Health system, rather than architectural or technical considerations.

Anonymization and Pseudonymization of Data

Collecting and storing medical history data is considered beneficial for supporting future clinical decisions related to the individual with the right information. For this to work, the data must remain linked to the individual and be properly protected.

Other beneficial uses of collected data are for reporting, identifying trends, generating statistics, facilitating medical research, and public health monitoring. These do not typically need to identify particular individuals, but rather rely on more general and derived attributes (e.g. "45-50-year-old male", "living in area X", etc.).

Anonymization services remove identifiable personal elements from the data, making it less sensitive and potentially not subject to stringent regulations governing privacy of personal data, while retaining its value for legitimate secondary uses like research and reporting. Replacing specific personal information with broader categories (see the examples above) does not automatically guarantee complete anonymity – care must be taken to analyze the possibility that a combination of attributes may yield a sufficiently small set of individuals to still allow identification.

Anonymization is an irreversible, one-way process, making it impossible to revert back and identify the individual from anonymized data. In some scenarios, the processing and analysis (possibly combined with other information) may identify a particular group of subjects who have to be contacted—for example, those at increased risk, or suitable candidates for new treatment.

Pseudonymization allows holding data in non-identifiable form to protect personal privacy, while providing a mechanism for reverse identification with sufficient protection against misuse. To achieve this, identifiable personal elements are removed, and replaced with some abstract unique "keys" representing the individuals without disclosing their identities – allowing legitimate secondary use of the data. The information and procedures needed to perform the reverse mapping to specific individuals are protected and controlled to prevent misuse.

Consent Management Service

The Consent Management Service is responsible for helping to manage patient consent directives in the context of e-Health solutions and services. Privacy and corresponding consent policies vary significantly across country, jurisdiction, and legislative boundaries, resulting in the need for a highly flexible and configurable consent management service.

Patient consent directives can be:

- Implied or explicit
- Stored in a local, central, or multiple locations
- Applied to data and services from a summary view down to specific health domain data elements
- Overridden as required in emergency situations

The Consent Management Service records, manages, and validates patient consent information and ensures that data access requests are compliant with the recorded consent directives during a request for patient information.

The patient consent directives may be augmented by other privacy restrictions including legislation, jurisdictional policy, and domain-specific business rules.

The Consent Management Service is exposed in the form of a set of Web Services that provide the following functions:

- Record and revoke patient consent directives
- Validate patient consent directives
- Override patient consent directives
- Provide access control to patient data based upon recorded consent directives (and optionally other managed business rules)
- Record and audit all patient-consent-related activity

Service Publication and Discovery Services

Service Directory Service

One of the core principles of a service-oriented architecture is the location transparency of the services it may provide to consumers. The physical location of a service may change for various reasons, such as more than one geographically dispersed service offering the same service contract, or a disaster recovery failover that makes a service in a different location available.

A consuming application may fix the physical address of the service out-of-band at design time through manual discovery, and store that address in some service configuration. However, movement or deprecation of the service will mean that the consuming application no longer works. In addition, the consumer should not be concerned about where the service lives. Instead, it should be concerned with what the service offers to the consumer.

As a result, it is necessary for the consumer to fix the physical address at runtime to avoid the hard lock-in at design time. This is the purpose of a service directory. It enables consumers to discover services, given a set of searchable criteria, at runtime and when required.

Due to the lack of physical location awareness of the consuming application prior to using the service, it is necessary for the provider of the service to offer a reasonably coarse-grained interface—in other words, an interface that does not provide a granular or “chatty” interface that may cause performance issues when the service is geographically located far away. Services should accept messages that contain enough information to perform a business operation in its entirety, without resorting to expensive chat backwards and forwards between the consumer and the provider – as is typically the case with distributed object RPC-style protocols.

The Service Directory Service should contain registrations for all of the e-Health Hub services (EHR, health registry, security, integration, etc.) to allow consumers to discover the correct service—for example, a Submission Service in a particular region or a local Security Service that offers authentication services. *Figure 26* illustrates the interaction between a consumer and a provider while utilizing a Service Directory.

The service provider registers itself with the service directory out-of-band, so that it is can be found by service consumers. After registration of the service, the service consumer can search for a service based on searchable metadata associated with the service. After finding the appropriate service, the consumer binds to its physical address and uses the service.

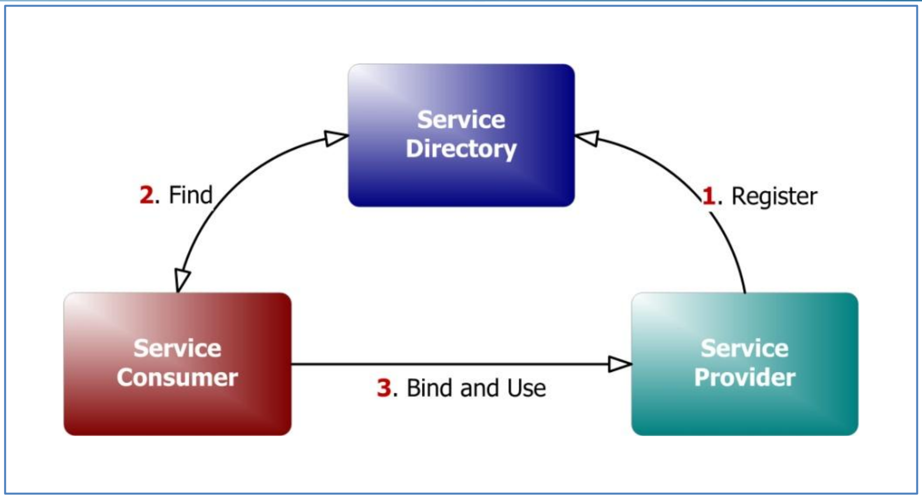

Figure 26. Interaction Between a Service Consumer, Service Provider, and a Service Directory

UDDI

A standard that has gained some acceptance in the industry, and which provides the necessary protocols and structure for a service directory, is UDDI.

Note: UDDI stands for Universal Description, Discovery, and Integration, and version 3 is an OASIS Standard, although the WS-I Basic Profile 1.1 (for providing a basis for interoperability between Web Services) currently specifies version 2.

OASIS manages the UDDI specification, available at http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3.

UDDI helps enable the registration of Web Services by providers, and publication of Web Service definitions to consumers. In effect, it is an advertising and discovery service for Web Services.

UDDI services generally offer three categories of information: white, yellow, and green pages. White pages contain contact information, addresses, and so on for the businesses that provide Web Services. Yellow pages contain the categorization for the Web Services through standard taxonomies. Green pages provide the technical specifications of the Web Services, such as binding information and consumer implementation details.

Note: Due to the potentially complex relationships that can be created in a UDDI registry, the possibility of extending the data model with new categorization schemes, and ensuring that consistent naming standards are adopted, it is necessary to plan the structure of a UDDI registry prior to implementation. It is useful to consider planning of the design and implementation in a similar fashion to the implementation of a technology such as Active Directory.

Service Metadata

In UDDI, each service can have associated metadata that consumers can use to search for a service supporting a particular operation or feature. This metadata is the tModel that is used to allow the exchange of information such as a service description (which may be implemented using WSDL), or pointers to other metadata associated with the service. A tModel is not specific to a particular service entry in UDDI, but can provide a reference in its own right to search for compatible Web Services. Therefore, multiple services can all point to the same tModel. In addition, each service can reference multiple tModels where that service may implement multiple interfaces.

Publishing a Service

UDDI offers both a publisher and an inquiry SOAP API for interacting with the UDDI service registry. Service providers use the publisher API to enter information such as business details, new services, tModels, and classifications into the UDDI registry. All of the publisher API operations transmit the requests to the UDDI operator though HTTPS to ensure the privacy of information during registration.

To use the publishing API, a service provider must first register with the UDDI operator (the owner of the service directory) to gain credentials for use in interaction with the registry. After receiving the credentials, the service provider must then get an authentication token every time they wish to use the publisher API to enter information about their services, which takes advantage of the `get_authToken` operation.

The UDDI data model includes the following entities, which the publisher API manipulates when changing information about the provider or about the services that the provider offers:

- `businessEntity` – describes a business or service provider. Associated with this structure are all of the other structures below, and contact information such as the name and address of the provider. Providers may represent organizations, departments, and so on, and a publisherAssertion entity represents the relationships between business entities.
- `businessService` – describes a logical group of one or more services offered by the provider and can be categorized using a categorization scheme that may indicate its geographical location, service type, quality of service (QoS) parameters, and so on.
- `bindingTemplate` – describes the binding for an instance of a Web Service that is associated with the business service logical grouping. This binding specifies, in the case of SOAP-based Web Services accessed over HTTP, the URL location that the service consumer can physically find the Web Service.
- `tModel` – provides a fingerprint or collection of information about the specification of a Web Service. Typically, this represents the location of the WSDL interface specification for the service, but it may represent any other descriptive data.

Figure 27 illustrates the relationships between these top-level entities.

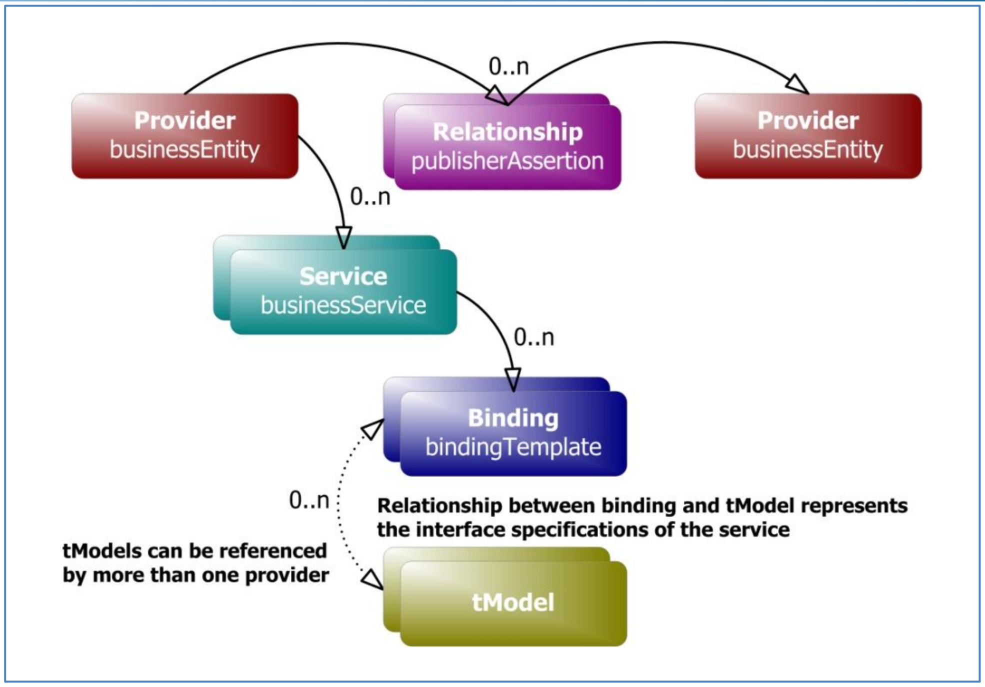
Provider
businessEntity
0..n
Relationship
publisherAssertion
Provider
businessEntity
0..n
Service
businessService
0..n
Binding
bindingTemplate
0..n
Relationship between binding and tModel represents the interface specifications of the service
tModels can be referenced by more than one provider
tModel

Figure 27. UDDI Entity Relationships

Finding a Service

The inquiry API allows a service consumer to find and bind to a Web Service at runtime. Searching is available against businesses, services, and tModels (service interfaces) that return a set of results. The consumer can then iterate through the response, selecting the most appropriate entry.

It is not typical to bind to the contract at runtime, as this is a task for the development phase. The developer has to understand the interface of the service in order to consume it within the application, and this is not possible dynamically. The dynamic binding at runtime relates only to the physical location (the URL or address) of the service that offers the expected interface.

The API offers a set of core operations that are central to searching for a specific set of related services:

- `find_business` – this operation finds information about one or more businesses that are associated with certain categories or identifications. A search is also possible through a name or tModel reference.
- `find_service` – this operation returns a list of services that meet search criteria, such as the service type.
- `find_tModel` – this operation returns tModel structures that permit a consumer to search the registry for matching services.

It is possible to implement a simple form of load balancing by iterating through a set of services (that all implement the same service interface) in a round-robin fashion for each service invocation. In addition, if a Web Service invocation fails, the design of the consumer can provide features to either retry this service or choose the next service in the list. This provides an element of failover protection.

Replication

The UDDI registry has support for replicating its data with other registries in a similar way to the Domain Name System (DNS). A specific replication API supports this, and registry operators can use this to ensure synchronization of the UDDI registries. While updates do replicate changes to entities (such as service addresses), the service provider publishing the new information should return to the same operator they have registered with to update the existing information.

Replicating the registries ensures that a consumer can query a local Service Directory to find a service where the service provider for that service has a publisher's registration with a different Service Directory. The only out-of-band configuration required by the consumer is the location of a suitable Service Directory. Without replication, the consumer would always have to know—prior to searching for a service—the registration location of a particular service in order for the consumer to locate the physical address.

e-Health Business Services

This section of the document describes the e-Health business services that directly support the specific health domain requirements of any e-Health solution. The e-Health services and supporting business components are wholly dependent upon the specific nature of the healthcare requirements being addressed.

The e-Health Business Framework discussed in *Part 2* of this set of documents describes the concepts of Patient Care Records, Care Pathways, Patient Journeys, and Health Care Providers. Together these concepts help to describe and provide a frame of reference for the specific healthcare solution under consideration.

As an example, *Figure 28* shows a portion of a generic Care Pathway (somewhat simplified) for Colorectal Cancer. The diagram for the full Care Pathway is large and is included as an appendix in *Part 5*. We have also included subsidiary diagrams showing the individual phases of the Care Pathway. This Care Pathway has been constructed to illustrate the care process for colorectal cancer and is based on professional advice. We have structured the Care Pathway into a longitudinal, step-by-step process and grouped these steps into meaningful “blocks”. We have analyzed the process into four phases (in pink): Examination, Treatment, Post-Operative Treatment, and Follow-Up. Each of these phases comprises a number of activities (in green), each activity involves the carrying out of a clinical process, and each clinical process (in white) comprises a number of actions.

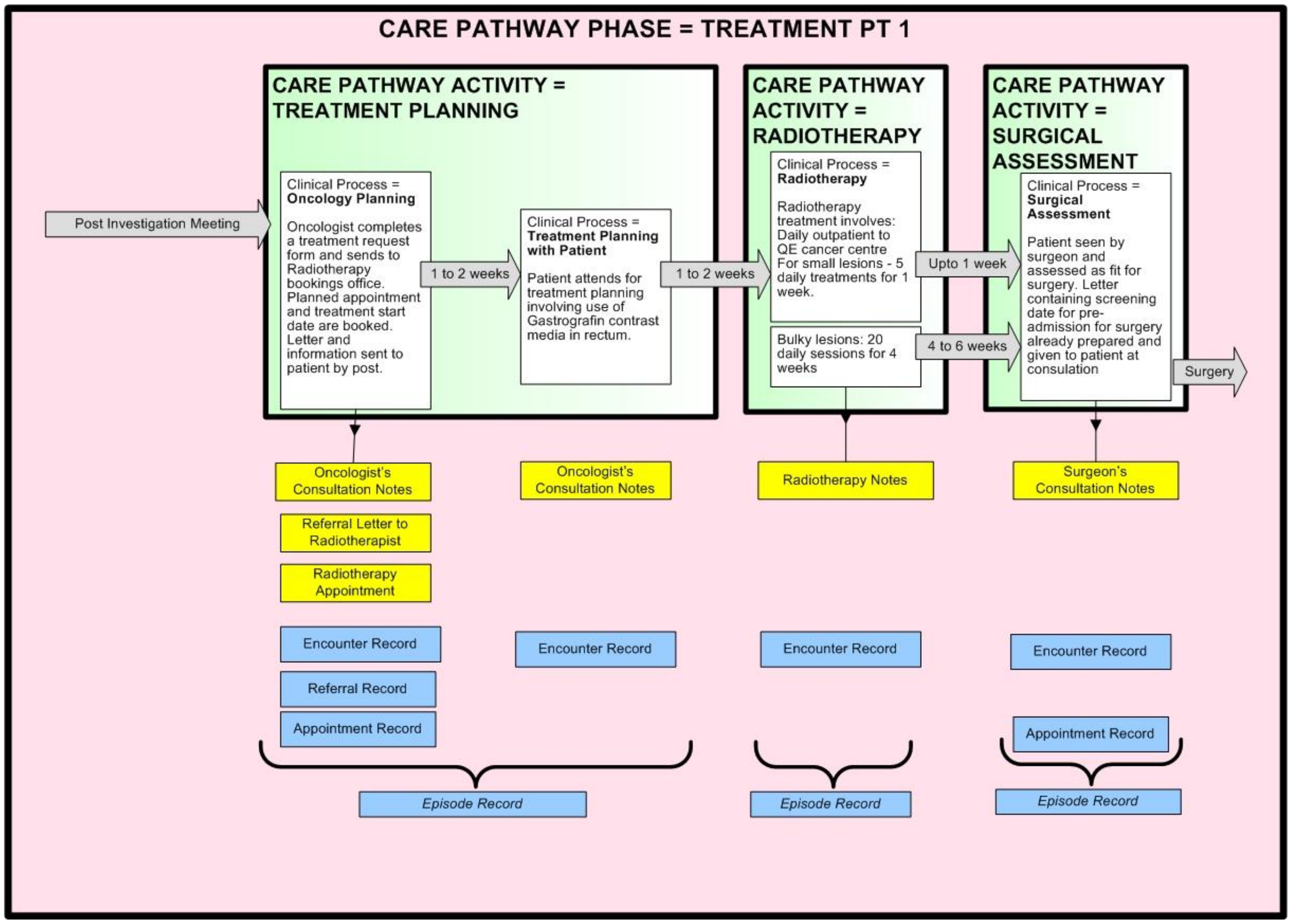
CARE PATHWAY PHASE = TREATMENT PT 1
CARE PATHWAY ACTIVITY = TREATMENT PLANNING
CARE PATHWAY ACTIVITY = RADIOTHERAPY
CARE PATHWAY ACTIVITY = SURGICAL ASSESSMENT
Post Investigation Meeting
Clinical Process = Oncology Planning
Oncologist completes a treatment request form and sends to Radiotherapy bookings office. Planned appointment and treatment start date are booked. Letter and information sent to patient by post.
1 to 2 weeks
Clinical Process = Treatment Planning with Patient
Patient attends for treatment planning involving use of Gastrografin contrast media in rectum.
1 to 2 weeks
Clinical Process = Radiotherapy
Radiotherapy treatment involves: Daily outpatient to QE cancer centre For small lesions - 5 daily treatments for 1 week.
Bulky lesions: 20 daily sessions for 4 weeks
Upto 1 week
4 to 6 weeks
Clinical Process = Surgical Assessment
Patient seen by surgeon and assessed as fit for surgery. Letter containing screening date for pre-admission for surgery already prepared and given to patient at consulation
Surgery
Oncologist's Consultation Notes
Referral Letter to Radiotherapist
Radiotherapy Appointment
Oncologist's Consultation Notes
Radiotherapy Notes
Surgeon's Consultation Notes
Encounter Record
Referral Record
Appointment Record
Encounter Record
Encounter Record
Encounter Record
Appointment Record
Episode Record
Episode Record
Episode Record

Figure 28. Care Pathway Fragment

The e-Health services described in this section realize the required functionality to support the Care Pathways and Patient Journeys necessary to fulfill against any specific e-Health solution. We have grouped the e-Health services into three categories, each responsible for a specific set of the broad e-Health business requirements which, when combined, fulfill against the required e-Health solution requirements. These groupings are Electronic Health Record (EHR) Services, Health Domain Services, and Health Registry Services. It should be apparent from this discussion that the e-Health services, much more than the other services described as part of this reference architecture, will be specific to and dependent upon the specific e-Health scenarios that a specific solution is aiming to address.

Electronic Health Record Services

The Electronic Health Record (EHR) Services are a set of e-Health services responsible for capturing, persisting, summarizing, and providing access to longitudinal patient health information in both a summary and detailed format. These services are central to all e-Health solutions and represent a centerpiece of the e-Health Service Hub.

All domain-specific e-Health solutions (such as lab and drug information systems) are dependent upon the EHR services to provide the required summary patient information as well as a mechanism to locate the source health system containing the complete health records. The EHR services are responsible for accepting and marshalling requests for the creation, update, and queries of patient and patient encounter data; acting as an intermediary to the various domain services and repositories; and acting as a data source itself in the form of a summary patient health record.

The EHR Services are supported by a Summary Clinical Data Repository (SCDR), which contains the minimum set of patient and patient encounter data required to support a high percentage of all patient health information requests at a summary level. The SCDR also contains pointers for each of the patient and encounter records to the source system from which the summary information was derived.

It is recommended that the information model supporting the SCDR be aligned with the HL7 v3 Reference Information Model (RIM). This alignment will help to ensure that the exposed EHR services fully support an HL7 v3 message infrastructure and will provide a high level of fidelity between the available services and SCDR. The Record Locator Service (RLS) of the Connecting for Health/MA-SHARE RHIO in the United States is very similar to the Connected Health Framework EHR Services where no summary data is stored in the SCDR. The SCDR is still used for storing pointers to the detailed clinical encounter records.

Note: Although we expect new systems to be built using newer standards and formats such as HL7 Messaging v3, there might be a requirement that legacy formats still be accepted, especially where the burden of translating to different standards might impose too high of a barrier for participation in the e-Health system. As an example, the vast majority of the hospitals in the United States implement HL7 v2 for their laboratory results (HL7 v2.x ORU messages), given the inherent complexity in transforming from HL7 v2 to HL7 v3, a system that aims to provide access the majority of hospitals might want to enable them to publish lab results in a canonical HL7 v2 format detailed in the implementation guides. In other circumstances, it might be required to support different standards because of their applicability and scope, such as DICOM for imaging, NCPDP for U.S. pharmacies, and so on.

The following EHR Services are published through the e-Health Service Hub. The expectation is that these services are based on HL7 v3 messages exposed through a set of Web Services:

- ***EHR Access*** Services
  The EHR Access Services provide for the querying and retrieving of summary patient and patient encounter information from the SCDR. The query parameters are based on rationalized patient and provider identifiers. The summarized and aggregated patient and patient encounter information returned will depend upon the query parameters supplied to this service. The returned summary patient and patient encounter information will also provide source service location (service address) for each summary patient encounter.

- ***EHR Update*** Services
  The EHR Update Services allow participating clinical systems and services to update the SCDR with patient and patient encounter information. The Health Domain Services (described below) will be the primary clients of the EHR Update Services. Whenever activity occurs within one of the Health Domains, the associated patient encounter will be updated in the EHR through the EHR Update Services.

- ***EHR Process Orchestration*** Services
  The EHR Process Orchestration Services maintain and execute the process workflows or “orchestrations” that control the interaction of supporting e-Health Service Bus services required to fulfill the EHR Services. This can include orchestration of Integration Services, Security Services, and Health Domain Services.

- ***EHR Business Rule*** Services
  The EHR Business Rule Services maintain and execute business rules associated with the execution of all EHR Services. Business rules can range from simple validations to complex business logic.

Health Domain Services

The Health Domain Services are a set of e-Health services responsible for capturing, persisting, and providing access to specific health domain information (such as lab information systems, drug information systems, and diagnostic imaging systems) in both summary and detailed format. These Health Domain Services are central to all e-Health solutions and govern access to detailed patient clinical encounter activity. The Health Domain Services use the EHR Services described previously, to keep the summary patient EHR information current. Similarly, the Health Domain Services are used to retrieve the detailed clinical encounter records that are summarized by the EHR Services. The Health Domain Services that are implemented and deployed will vary by e-Health solution and will depend upon the specific e-Health solution requirements. Health Domain Services may be used for a single health domain or to aggregate information from across multiple health domains.

The following Health Domain Services are required for each health domain (such as a lab or pharmacy) that is supported as part of the e-Health solution. The services are published through the e-Health Service Hub with an expectation that the services are based on HL7 v3 messages.

- ***Health Domain Access*** Services
  The Health Domain Access Services provide for the querying and retrieving of health domain information from the source domain system (such as a lab system or repository). This set of services is typically used to get detailed domain information based upon summary information provided through the EHR Services.
- ***Health Domain Update*** Services
  The Health Domain Update Services provide for participating clinical systems and services to update the source domain SCDR with patient and patient encounter information. The Health Domain Services (described below) will be the primary clients of the EHR Update Services. Whenever activity occurs within one of the Health Domains, the associated patient encounter will be updated in the EHR through the EHR Update Services.
- ***Health Domain Process Orchestration*** Services
  The Health Domain Process Orchestration Services maintain and execute the process workflows or “orchestrations” that control the interaction of supporting e-Health Service Bus services required to fulfill the Health Domain Services. This can include orchestration of Integration Services, Security Services, and EHR Services.
- ***Health Domain Business Rule*** Services
  The Health Domain Business Rule Services maintain and execute business rules associated with the execution of all Health Domain Services. Business rules can include simple validations through complex business logic.

Health Registry Services

The Health Registry Services are a set of e-Health services responsible for maintaining central health registry indexes. The health registries governed by this set of services include patient, healthcare provider, and location. These services provide indexing, access, update, and matching/linking functionality for each of the health registries.

The following Health Registry Services are published through the e-Health Service Hub. The expectation is that these services are based on HL7 v3 messages exposed through a set of Web Services:

- ***Health Registry Access*** Services
  The Health Registry Access Services provide for the searching, selecting, and retrieving of health registry (patient, healthcare provider, location) data. The registry services use both exact matching and probabilistic linking algorithms to return the candidate set of information. The Health Registry Access Services will typically employ EMPI (Enterprise Master Patient Index) solutions to create and maintain the registry index.
- ***Health Registry Update*** Services
  The Health Registry Update Services allow participating clinical systems and services to update the health registries (patient, healthcare provider, location) with new or changed registry information. The update services are typically called in conjunction with or in response to a related domain or EHR activity (such as an "*admitting a patient*" event in a connected clinical system).
- ***Health Registry Process Orchestration*** Services
  The Health Registry Process Orchestration Services maintain and execute the process workflows or "orchestrations" that control the interaction of supporting e-Health Service Bus services required to fulfill the Health Registry Services. This can include orchestration of Integration Services, Security Services, and Health Domain Services.
- ***Health Registry Business Rule*** Services
  The Health Registry Business Rule Services maintain and execute business rules associated with the execution of all EHR Services. Business rules can include simple validations through to complex business logic.

Integration Services

Underlying and directly supporting the e-Health Business services described in the previous section is a rich set of integration services which are responsible for providing a fabric of connectivity among all participating healthcare systems and stakeholders. The integration services ensure the interoperability between connected healthcare systems and services, providing the required network and application protocol bridging, syntactic and semantic message transformation, message routing and process orchestration, and transaction management. These services are offered in a secure, reliable, and highly available architecture.

Submission Service

The reference architecture described in this guide enables multiple channels to interact with the e-Health Services Node electronically. A key part of the interaction with any Healthcare agency is document submission. Ordering lab tests, viewing a patient's summary health record, or updating a patient's address—all of these operations require submission of some form of document (and sometimes attachments such as images).

The Role of a Submission Service

In an e-Health Services Node, the Message Submission Service is responsible for the receipt of documents and the processing of these documents on behalf of the channel or other peer Message Submission Services that initiated the delivery.

The document delivery envelope is a message that contains not only the document itself, but can also contain metadata about the document such as the identity of the sender, the originating application or service, the date of submission, and the type of document contained within the message. Message metadata is useful to any service that needs to understand some general properties about the payload (the document itself) in order to make decisions on how to process or route the message. The document type and the destination agency service are examples of metadata that, at a minimum, are useful when routing the message to the correct agency.

The Submission Service is also responsible for checking the security of the message and preventing tampering of the content. In addition to checking the validity of the message, the Submission Service may have to decrypt the payload in order for it to route the message to an agency service that does not have the technology in place to decrypt the message.

After checking the validity of the message, the Submission Service should verify the authenticity of the sender by ensuring that the embedded security token is valid. It must then authorize the submission based on the requirements of the target service, by checking the authentication level against the minimum authentication level specified for that service. In other words, a target service may require verification of the sender's identity through a username and a strong password. If the verification was through a username and a weak password, the service may not accept the message. The authentication level must be a property of the message (digitally signed so that it is tamper-proof) for the Submission Service to perform the authorization.

After verification of the sender and confirmation of the appropriate authentication level for the target service, authorization of the operation can take place at a coarse-grained level. The operation is the submission of a document of a certain type to a particular service, or it could be routing of the document on to another peer Submission Service. However, preauthorization can take place before routing. Authorization involves checking a set of claims against the Authorization Service. These claims may range from the sender having the permissions to submit a document of type X, or belonging to group Y, or any combination thereof.

Document routing to the destination service takes place after successfully passing the security checks. This may be on to a peer Submission Service, in which case the security checks may or may not have occurred depending on the requirements of the sender. The requirement may be that the message is checked only when it reaches its ultimate destination, or it may be that message checks are carried out by an intermediary Submission Service.

Basic Architecture of the Message Submission Service

The basic architecture of the Submission Service divides into two main areas:

- A black box public side used by consumers that are submitting documents
- A white box private side used by the service provider (central Healthcare, an authority, or local health care body, or a single agency offering Submission Services) that enables integration with the agency systems

Figure 29 highlights these two areas for a basic Submission Service architecture.

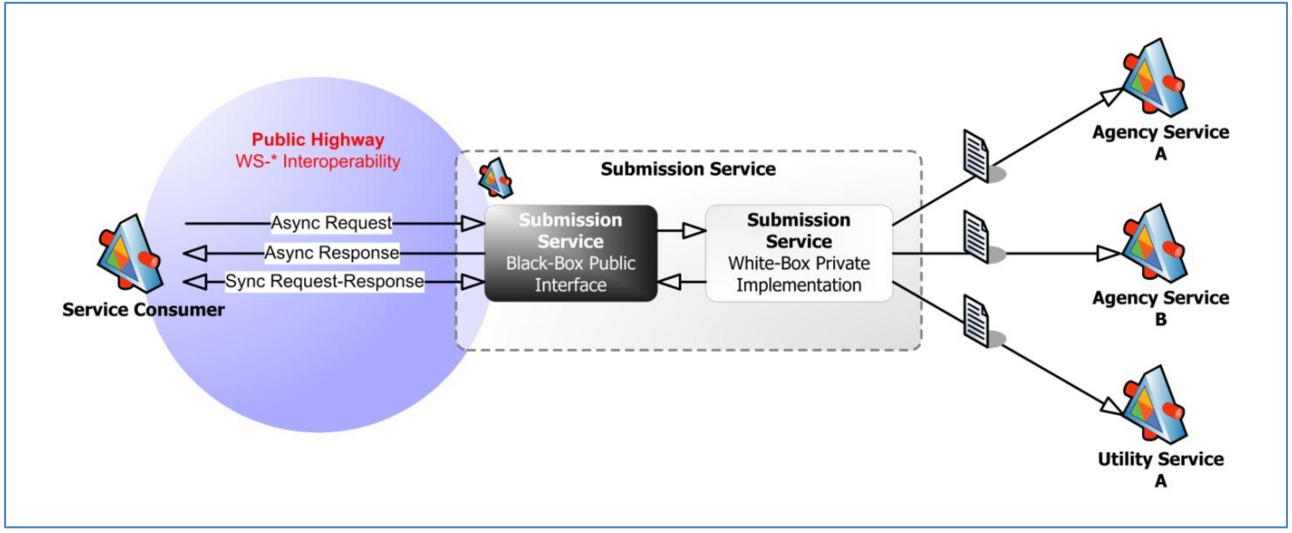

Figure 29. Basic Architecture of Submission Service

Public Interface

The public interface of the Submission Service defines how the consumer interacts with the service; what operations are supported; the necessary protocols and message formats that the service supports; and the non-functional requirements placed on the consumer by the service, such as what level of encryption and signing is mandatory. It is how the consumer sees the service.

An external service consumer—which can be a portal, an ISV application, another service, or a partner—interacts with the Submission Service through its public interface. The provider of the service offers an interface that, at its most basic level, enables the consumer to submit documents both synchronously and asynchronously. Supporting this interface will be ancillary operations that allow the consumer to check submission status, and retrieve and clean up responses if submission is asynchronous.

In most cases the public interface of the Submission Service will be wrapped by a set of healthcare-specific interfaces that are published through the Service Directory Services.

Figure 30 illustrates a high-level view of the public interface of the Submission Service.

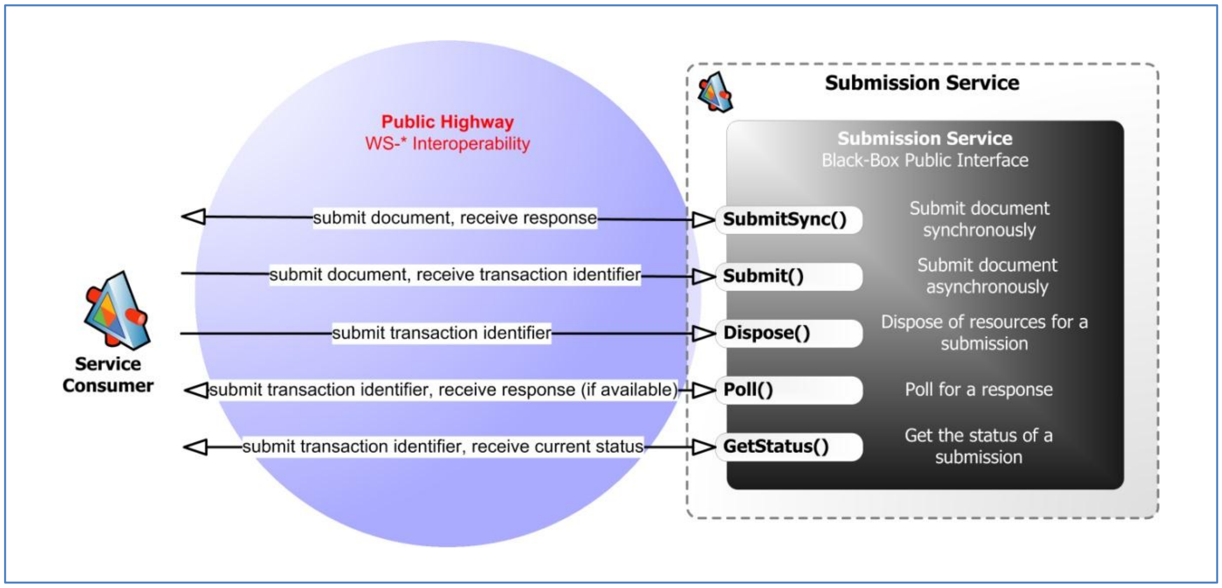
Public Highway
WS-* Interoperability
Service Consumer
Submission Service
Submission Service
Black-Box Public Interface
submit document, receive response
SubmitSync()
Submit document synchronously
submit document, receive transaction identifier
Submit()
Submit document asynchronously
submit transaction identifier
Dispose()
Dispose of resources for a submission
submit transaction identifier, receive response (if available)
Poll()
Poll for a response
submit transaction identifier, receive current status
GetStatus()
Get the status of a submission

Figure 30. High Level Services Provided by the Public Interface of Submission Service

Asynchronous Communication

The Submission Service implements the basic communication patterns (synchronous and asynchronous) to ensure support for as wide a client base as possible. However, if consumers are blocking through a synchronous interface, the service has to process and return a response on the same connection. This means holding onto resources longer than would be the case in an asynchronous scenario, thus limiting the scalability of the service.

Using the asynchronous interface for document submission is preferable, as this affords better scalability because the consumer is not required to block waiting for a response. In addition, the underlying technology used to implement the Submission Service does not have to ensure it returns a response in as short a time as possible.

The consumer provides the service with a location where that service can deliver its response, independent of and after processing of the request. If the consumer is unable to provide a response location, it has to resort to polling for a response from the Submission Service. This implies that the service has to maintain the state for any responses received asynchronously from agency services, utility services, or other peer services, and raises questions such as "*at what point do I start to clean up responses that have not been fetched by the consumer?*"

The simplest approach is to offer a "dispose-like" operation on the Submission Service that permits the consumer to dispose of their own resources once they have finished with the response. In addition, the service itself cleans up resources based on a period of decay for forgotten responses, or after the consumer has collected them. In essence, this is a garbage collector for response messages.

Synchronous Communication

In some cases, the client may not support an asynchronous pattern that requires a location for the receipt of asynchronous responses. The client can still poll for a response asynchronously, or may prefer to call the Submission Service synchronously for other reasons. An example is to provide immediate responses to the end user.

Protocols such as HTTP are request/response protocols, where the Web browser posts a request to the Web server and the Web server generates an HTML page, returning it to the client on the same connection. In a synchronous scenario, the response would be the actual response from the target service(s), rather than merely an acknowledgment response that would be produced in an asynchronous scenario.

However, at the Submission Service level (rather than at the target service) a synchronous pattern may be extremely difficult to achieve due to dependencies on downstream services (other services or utilities) that provide the response. Generating a synchronous response is entirely dependent on the implementation and support of the synchronous pattern by these downstream services as well. This means that the consumer must be able to discover the capabilities of the health or utility service at design time to see if it supports a synchronous or asynchronous interface.

Although the asynchronous pattern is preferable, it does mean that the public interface becomes a little more complex to support polling and cleanup operations. However, exclusive use of the synchronous pattern by consumers (where downstream health services support this pattern) can introduce additional scalability problems that otherwise may not exist using the asynchronous pattern for document submission. Therefore, it is sensible to only promote the synchronous pattern where it is necessary—for example, with a patient health summary service that retrieves summary data from a local store and provides an immediate response.

The communication services and protocols supporting both the synchronous and asynchronous message exchange are described more completely in the *Communication Services* section later in this document (page *99*).

Private Implementation

The public interface of a document Submission Service is concerned with interaction with consumers in a standardized way. However, the private implementation is concerned with how the service will actually process requests, integrate with agency or utility services, return responses, and ensure that—operationally—it provides the correct information to other services.

The Submission Service has many responsibilities that require implementation, including the following:

- Receive and process requests
- Return responses either synchronously or asynchronously, depending on the requirements of the agency or utility service
- Store requests where delivery to the target agency or utility service is not currently possible
- Store responses where they cannot be delivered asynchronously, or if the consumer supports only polling for a response
- Authorize the requests
- Process polling requests
- Process cleanup requests
- Audit requests, responses, and any security-related operations such as authorization
- Provide instrumentation in the form of information on errors, warnings, and basic operational information; performance counters for determining service health; and runtime tracing information that can be switched on or off to aid diagnosis
- Integrate with agency and utility services
- Route documents on to agency, utility, or peer services

- Support higher-level application protocols through the orchestration of business processes

This and the following sections examine these responsibilities. Meanwhile, *Figure 31* provides a high-level view of the implementation of a Submission Service.

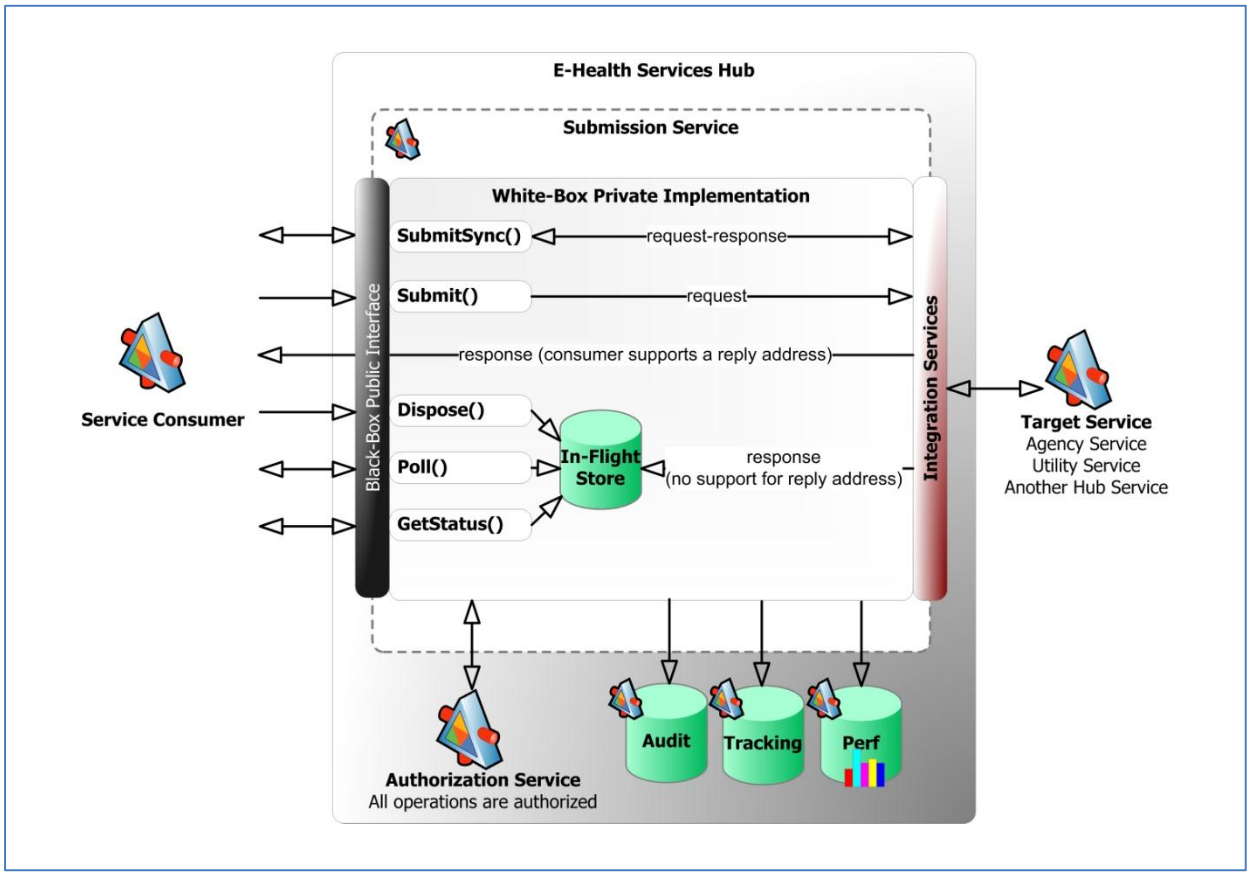

Figure 31. High-Level Services Provided by a Private Implementation of a Submission Service

Messaging Services

The message format is XML, which is fast becoming the de-facto format for the communication between applications (service consumers) and Web Services (service providers). XML provides a defined structure that gives meaning to data.

Note: The World Wide Web Consortium (W3C) is the organization that governs the XML specification, currently a W3C Recommendation at version 1.1. The W3C are at http://www.w3.org/. The XML specification is at http://www.w3.org/XML/.

The rise of XML is due in part to its ease of use, and the fact that it is text-based, which ensures that even the simplest of systems has the ability to understand a message formatted as XML. Support for XML is widespread across multiple operating systems, with even some databases now supporting XML as a native data type.

As such, using XML to provide the basic message structure makes sense when building a framework designed to ensure interoperability between systems and applications in a heterogeneous health care world.

Envelopes, Headers, and Payloads

The "envelope" format for messages enables message metadata to be stored alongside the payload. The format includes a header that contains the metadata, and a body that contains the payload. An example of an envelope format in use today is SOAP, the XML-based protocol (adopted by the majority of Web Service software manufacturers) that facilitates application communication and wraps requests and responses with associated metadata. The Submission Service uses the SOAP format for encapsulating requests and responses to and from other services. This ensures the Submission Service is as interoperable as possible with any healthcare service that runs on any platform and uses any operating system technology.

Note: The W3C governs the SOAP specification, available at http://www.w3.org/2000/xp/Group/ and is currently a W3C Recommendation at version 1.2.

Figure 32 highlights the structure of an envelope-formatted document, where the envelope provides the structure to contain both the header and body.

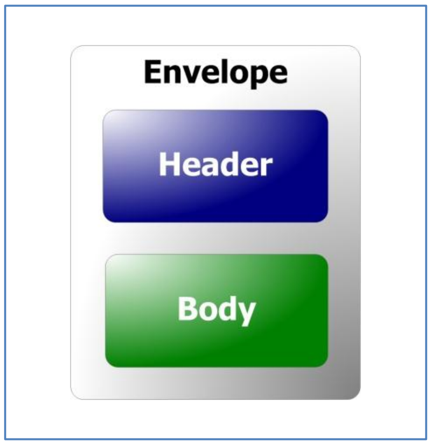

Figure 32. An Example of an Envelope-Formatted Document

This format also supports nesting—for example, a custom envelope format used by a healthcare service may permit a SOAP payload nested inside a SOAP message. This implies several levels of message metadata: the metadata (business metadata) associated with the document separated from the metadata associated with the SOAP message (technical metadata that may be concerned more with message security and reliability).

Figure 33 shows a nested-enveloped-formatted message where the body of the root-level document contains yet another envelope. This nesting can continue, with nested envelopes containing other envelopes if required.

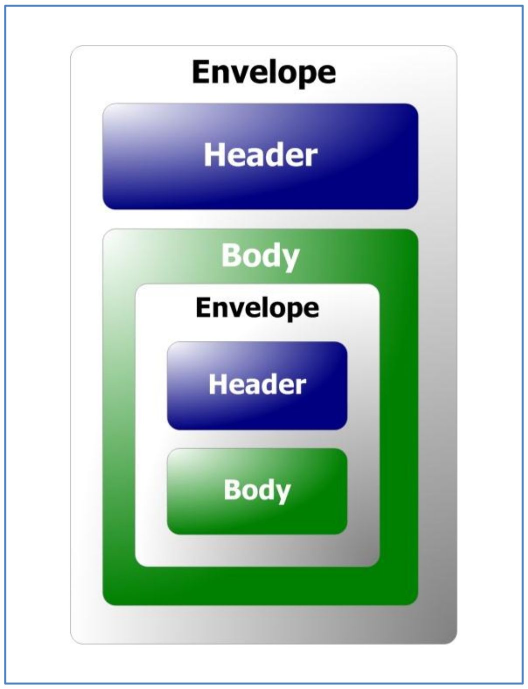

Figure 33. An Example of a Document Containing Nested Envelopes

Message Attachments

Sometimes it is necessary for some extra collateral to accompany an XML document. This might be scanned paper documents, X-ray or scan images, or any other digital representations.

Associating attachments with an XML document has been technically possible for some time, through various mechanisms and following various specifications such as SOAP with Attachments. The latest specification, which deprecates all others, is based on two specifications by the W3C: XML-binary Optimized Packaging (XOP) and the SOAP Message Transmission Optimization Mechanism (MTOM).

Note: The XOP specification is a W3C Recommendation found at http://www.w3.org/TR/xop10/. The MTOM specification is also a W3C Recommendation, and found at http://www.w3.org/TR/soap12-mtom/.

The purpose of XOP is to enable Base64 encoding for repackaging of binary data into a MIME-formatted message. The schema for the document will define a Base64 type as an `xs:base64Binary` data type, and this is a supported lexical canonical form contained in the XML InfoSet specification (an abstract data model of a serialized XML document).

Note: The Base64 characters must be in a canonical form—that is, no extraneous white space preceding, inline, or following the encoding. If this is not the case, it will be impossible for the receiving Web Service to reconstitute the original encoding when including the white space from the binary optimized format.

A binary format is more likely to provide optimizations for applications than a Base64-encoded version that has to be decoded. Binary format is also likely to be smaller. In place of the Base64-encoded data in the XML InfoSet, an XOP specific element provides a link to the XOP optimized binary content in the MIME message. The resulting package (after serialization of the InfoSet) is known as an XOP package, and contains the XML document (XOP) and the extracted content in a MIME Multipart/Related package.

MTOM is a specification that describes how to optimize transmission of the SOAP message (in particular the envelope), and relates to the XOP specification where it describes how the optimization is implemented. The Web Service that receives the SOAP message is responsible for reconstructing the original message by decoding the optimized binary content back into the Base64 representation, although this is not mandatory.

Securing the Messages

The basic communication with Web Services involves sending and receiving messages. Typically, SOAP is the format used for these messages. The problem with transmitting documents as payloads of SOAP messages is that there are no inherent security features built into each message. SOAP as a specification does not define any security support, only the basic structure of a SOAP message (envelope, header, and body). Consequentially, there are two main approaches to securing the messages: transport-level security and message-level security. Their respective advantages and disadvantages are discussed in the following sections.

Transport-Level Security

This lack of standards or specifications early in the development of Web Services, and in particular SOAP, meant reliance on other available mechanisms to provide security. These mechanisms were environment-based, such as transport-level security using protocols such as SSL, TLS, or IPSec. They rely on the establishment of a *secure channel*, following an initial "handshake" to confirm the identities of the endpoints and exchange the necessary encryption keys. After that, the channel protects the integrity and privacy of the traffic between two endpoints for the duration of the session (see Figure 34).

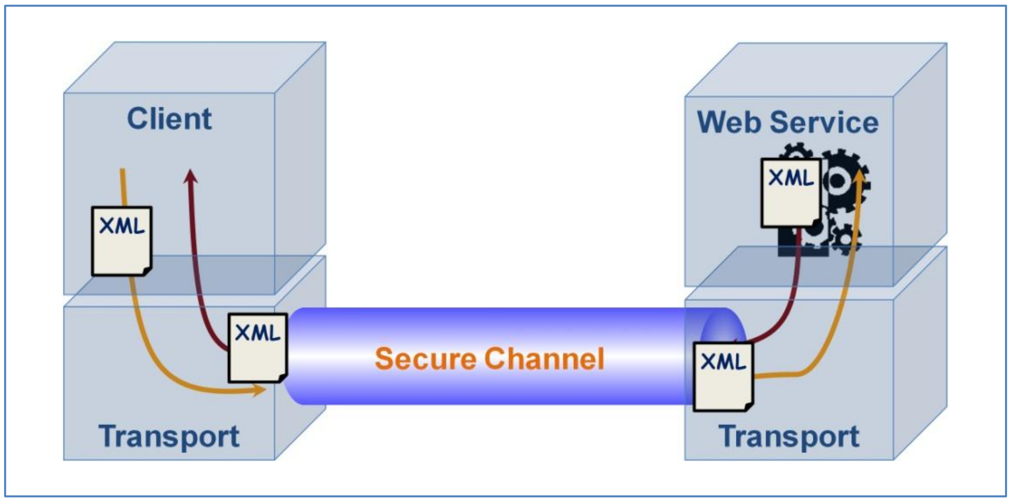

Figure 34. Transport-Level Security

The main advantage of transport security is that it is mature, well-established technology, widely supported in commercial products – even incorporated in the network infrastructure. Originating and terminating the secure channels at the edge of a node, using specialized equipment and hardware accelerators, can offload the additional processing from the main servers, and could be completely transparent to the applications.

However, transport-level security also has some notable shortcomings:

Security applied at the transport level protects message privacy and integrity point-to-point only between the sender and the receiver. Outside of the secure channel between the sender and receiver, the message is potentially unsecured and viewable as plain text. At this point, the security of the message is dependent on the underlying platform to ensure the privacy and integrity of the message.

For this reason, there are no guarantees that a message has remained private and has avoided tampering along the entire route between the sending and receiving applications. Even worse, if the message path routes it through one or more intermediary nodes, the message is potentially unsecured as it moves from one network to another.

Even within the boundaries of a single system, transport-level security is often provided by specialized hardware and software (for performance and efficiency reasons), located at the edge of the system. Secure sessions terminate at that point. Even when the appropriate mutual authentication has taken place to establish a connection, information about the origin of the message may be lost between its arrival at the edge of the system and actual processing.

Message-Level Security

Security at the message level aims to overcome the issues with transport-level security, and guarantee true end-to-end privacy and integrity of the messages, independent from the transport used and any intermediaries along the route. Messages themselves are self-contained, can be encrypted, and carry all the necessary security information (see *Figure 35*).

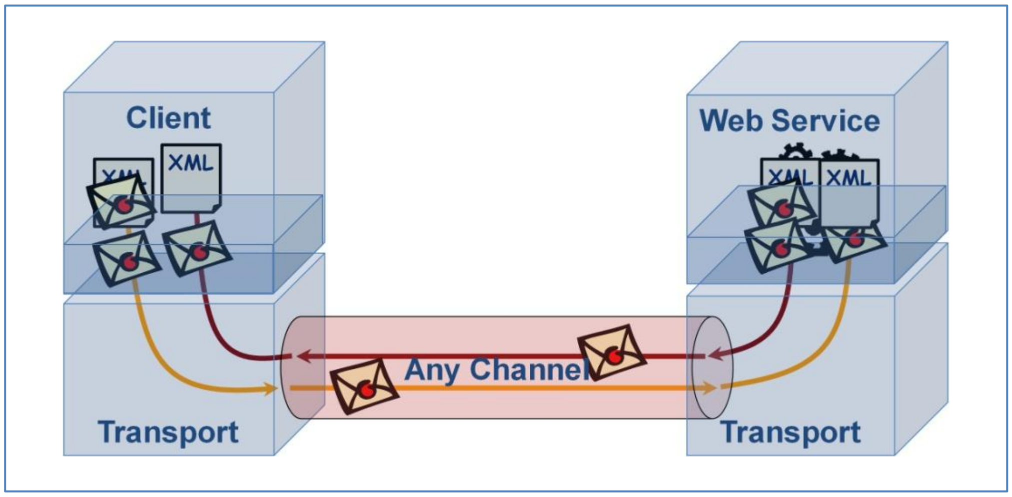

Figure 35. Message-Level Security

To achieve this, various specifications introduce message security information into the SOAP header. The most prominent of these is WS-Security.

WS-Security defines a set of SOAP headers used to ensure the preservation of privacy and integrity for messages sent from sender to recipient, and regardless of the networks traversed, the number of intermediary nodes, and the platforms that the message may reside on before re-routing.

Note: OASIS manages and coordinates the WS-Security standard; see: http://www.oasis-open.org/specs/#wssv1.1

Privacy

Encryption ensures the privacy of the message, and involves scrambling the plain text into indecipherable binary data using a standard encryption algorithm such as Triple DES, AES, or RSA. There are two types of encryption—symmetric and asymmetric—and both use mathematical algorithms that involve encryption keys for both encryption and decryption.

Symmetric encryption requires both the sender and receiver to know the same key (called a shared secret), and relies on some key exchange mechanism to ensure both parties are in possession of an identical key before communications can be secured.

Asymmetric encryption uses two different keys: public and private. The public key, which the key owner gives to anyone wishing to protect the privacy of messages sent to and from the key owner, enables encryption of the plain text. The private key (which only the key owner knows) enables decryption of the message.

Only the private key can decrypt what the public key encrypted, and only the public key can decrypt what the private key encrypted. As long as the key owner keeps the private key safe and private, it is a convenient method of protecting the privacy of messages because there are no restrictions when sending out a public key.

Why would the use of asymmetric encryption not be the obvious choice all the time when it is easier to distribute keys? This is because the mathematical algorithm and size of keys involved (in order to resist brute force attacks) means that it is computationally more expensive and consumes far more processing power than symmetric encryption. As a result, it is more efficient to use symmetric encryption where possible, particularly during a message exchange between Web Services. Asymmetric encryption is useful for establishing a shared secret or session key, which then secures the rest of the conversation using symmetric encryption. Secure Sockets Layer (SSL) over HTTP is an example of a protocol that works in this way.

Encryption at any level is possible in a SOAP message when using WS-Security. Encryption is possible for the entire payload, or selected for parts of it. Even the SOAP headers can be encrypted where appropriate. WS-Security takes advantage of the XML Encryption specification that describes the conventions used to include encryption in a SOAP message.

Integrity

While a message content may not be sensitive or confidential to require encryption, there may be a need to ensure that tampering has not taken place on the message while in transit and the integrity of the message is intact. This is particularly useful for transmission of a letter or request, where verification of the sender that produced the document is possible. This is known as signing the document, and results in a digital signature.

Signing a document involves running a hash algorithm such as SHA-1 against it to produce a small, fixed-length binary value called a hash or message digest that is unique, and which differs for even the smallest change to the source document. The private key then encrypts the message digest to produce a digital signature. The receiver uses the public key to decrypt the signature and retrieve the message digest, and then runs the same hash algorithm against the message to produce another message digest. Comparing the two digests reveals whether or not the message has suffered tampering. It so, rejection takes place. A match guarantees that the message is identical to that signed at source.

The problem with running a hash function against XML is that two XML documents may be syntactically identical, but still different due to extraneous white space, carriage returns, and so on. These affect the formatting of the XML document, but not the syntax. Therefore, it is necessary to run a canonicalization algorithm against the XML document before running the hash function to strip out white space according to a predefined set of rules.

Note: The W3C have released a specification called Canonical XML (sometimes described as C14N canonicalization) that describes the rules for the canonicalization of documents for use in signing. It is a W3C Recommendation at version 1.0 and is available at http://www.w3.org/TR/xml-c14n.

In a SOAP message that uses WS-Security, application of digital signatures over a combination of arbitrary parts of the message is possible, in the same way as encryption. WS-Security specifies the use of the XML Signature specification that describes the conventions used to sign and include digital signatures in a SOAP message.

Note: The XML Signature specification is a W3C Recommendation and is available at http://www.w3.org/TR/xmldsig-core/.

Authorization

The Submission Service should expect to receive a security token in the SOAP header that identifies the sender upon receipt of document. The sender should have authenticated with their Identity Provider prior to submitting the document, and—provided there is a trust relationship between the domain that issued the token and the domain (Relying Party) that the Submission Service runs in—the service accepts the sender's authentication status. A digital signature applied to the token ensures that tampering cannot take place, and integrity checks are possible on the message.

After verification of the security token, the Submission Service is responsible for checking that the sender has the authorization to submit a document to the agency or utility service by checking the enrollments against the Authorization Service. In addition, checks take place to ensure that the level of authentication that the sender used to verify their identity with their trust domain matches the minimum level accepted by the agency or utility service. Any negative response from the Authorization Service causes the Submission Service to return a SOAP fault message that indicates the sender failed authorization in terms of submitting the document to the service.

Message Validation

Assuming that verification of the sender and authorization to submit the document and the message are successful and that checks on the message headers succeed, the payload may require validation. Validating the payload ensures that the structure and content is in an acceptable format before delivery of the message to the target agency or utility service. Validation of the payload on behalf of a peer service can take place before it is re-routed. This could be because the peer service may not have the necessary resources to accomplish this.

The Submission Service should return a SOAP fault message if validation of the document fails. The advantage in the Submission Service performing basic validation is that the target agency service receives only documents that are correct according to their schema.

Normally, a schema defined using XML Schema (XSD) performs validation. XSD allows the structure of the XML document to be strictly defined, including restrictions on repeating elements, either/or existence of elements, and so on. XSD also allows type checking against content data types, as well as checking against allowable sets of data (enumerations) and complex pattern matching of content data.

In cases where higher-level semantic validation (such as extended vocabulary validation or business rules validation) is required, it might be appropriate for the recipient to perform this step and—if the original message is not correct—eventually return an application-level error message to the sender.

Message Store

The Submission Service should offer the sender an indication of safe receipt, and a guarantee that the document they submitted will not be lost or ignored. To implement this guarantee, the document is stored temporarily as it flows through the system. If the Submission Service is unable to deliver a document to an agency service, utility service, or peer service, it holds it in the message store until delivery is possible. The Submission Service may implement a retry algorithm that ensures periodic delivery attempts.

The message store also acts as a temporary store for responses not yet delivered to the sender (assuming the sender provided a response location), or not yet retrieved by the sender when it polls for responses.

The message store is critical to the operation of the Submission Service, and so suitable protection from software failures and environmental disasters is required.

Routing Service

One of the core purposes of the e-Health Integration Services is to route documents to the correct destination. This may involve routing the document to:

- A peer service located locally at a health agency, if coming from a domain-level Submission Service.
- A peer service located centrally at domain level, if coming from a local Submission Service.
- A peer service located in another trust domain, if coming from either a domain-level or local-level Submission Service.
- A utility service.
- A target service.

Figure 36 shows the various routing paths that a document may take. The key to the colored lines is as follows:

- Red – peer-level service physical route
- Dotted Red – peer-level service logical route—the physical routing to the node may be by way of other nodes
- Blue – agency service route
- Green – utility service route

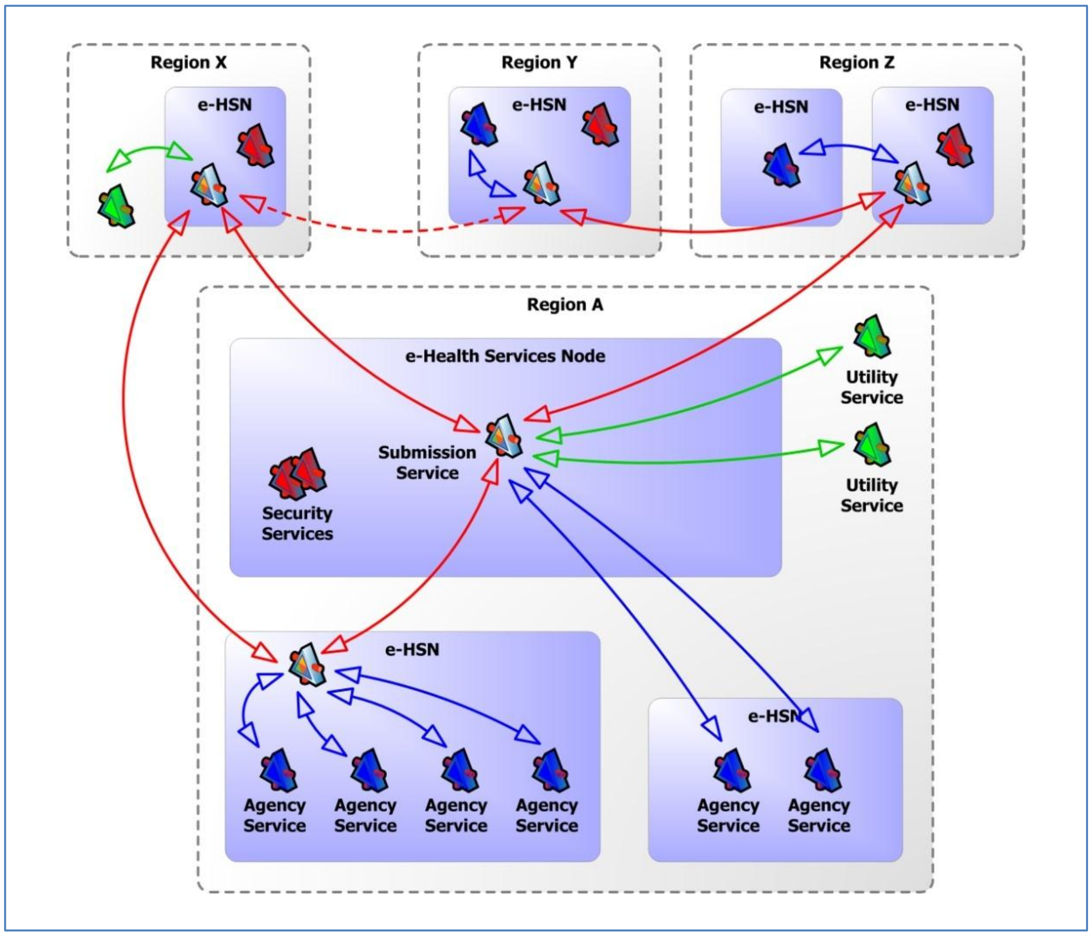

Figure 36. Document Routing by the Submission Service

Addressing Web Services

The location of a Web Service can be determined at runtime by querying the UDDI repository. This preserves one of the core tenets of a service-oriented architecture: location transparency. The address returned from the query represents the physical location of the Web Service. This may take the form of an HTTP URL if communication with the Web Service is over the HTTP protocol, but it could equally represent another type of address if a different protocol is used—for example, basic TCP/IP.

A SOAP message that contains a business document should include SOAP headers containing the address of the endpoint that is to receive and process the business document. This endpoint is likely to be the address of a Submission Service that is responsible for forwarding the message to the target health or utility service. In addition to the endpoint address, a set of properties can be attached that describe the health or utility service required, and any further technical or application-specific properties dependent on the requirements of the target or utility service. This allows routing of the message to a Submission Service that uses the associated properties to re-route the message to the correct service.

The WS-Addressing specification allows Web Service addresses to assume a standardized format within the SOAP header. The recipient of the message (the endpoint address) is identified by the `To` element, and the operation to be performed on the Web Service is identified by the Action element. The `From` element identifies the source of the message, and the `ReplyTo` element specifies where to send replies. Finally, the `FaultTo` element identifies where to send errors. Associated with each of the endpoint address elements are optional reference properties represented by the `ReferenceProperties` element.

Message Paths and Intermediaries

It is possible to submit a message to a Submission Service that is acting as an intermediary, but is not the service identified by the endpoint address in the `To` element of the SOAP header. In this instance, the Submission Service re-routes the message based on business rules that map the `To` address to the next node in the path. This node may or may not be the ultimate endpoint, thus the SOAP message follows a dynamical message path through a set of intermediary nodes.

While it is possible to define the path taken by a SOAP message in the header (and early specifications such as WS-Routing attempted to do so), this is not always desirable. If the header defines the route, then, effectively, changes are not possible unless an intermediary can update the header. In addition, the intermediary must remove its entry from the path, or set a flag to indicate processing has taken place.

This introduces a security issue. A digital signature is likely to be in use to protect the addressing header and avoid delivery of the message to a malicious node. If the intermediary is required to update the header, it must be unsigned – and this could leave an attacker free to update the addressing details in the header to include a malicious node. As it is obvious that signing of the header is required, dynamic routing decisions are not possible at the intermediary node and the header should define a fixed route.

Peer Routing

Routing between peers involves forwarding messages on to other Submission Services, whether they exist in the same trust domain or are located in other trust domains. This may be because the address of the ultimate endpoint is not available for one reason or another.

The target service for the message is determined by mapping the endpoint address (the ultimate receiver) to a node that is the next hop or the ultimate endpoint. The mapping rules exist in a rules engine or routing cache (similar to a network router) that network administrators can use to optimize message delivery.

Metadata-Based Routing

Sometimes, mapping the address in the header is not sufficient to determine the message route. Additional properties are often required for this. These properties are associated with the message, but generally not considered part of the payload. This metadata may take the form of the network protocol in use (HTTP, TCP, UDP, IPX, etc.), the health service expected to process the document, the sender of the message, and so on. These properties can supplement the decision-making process when determining the next hop.

Content-Based Routing

Taking the concept of metadata-based routing further is Content-Based Routing (CBR). This is routing based on the contents of the payload. CBR is a powerful mechanism for providing fine-grained control over the rules for routing messages. The node performing the routing has to understand the payload in order to extract the relevant properties used for routing.

Transaction Service

It may be necessary for message delivery to be a part of some higher-level transaction that encompasses multiple operations. Transactions can be "short lived" and "long running."

Short-lived transactions are analogous to database transactions, and are atomic in nature. All of the operations within a transaction are successful or none succeeds—it is an "all-or-nothing" approach. In addition to the atomicity of the transaction, the effects of the operations are isolated from other transactions until committed, otherwise inconsistent results may occur. Due to the requirement for isolation of a short-lived transaction from other operations, holding onto resources is a possibility, until all operations within a transaction are in a consistent and ready-to-commit state. This forces the transaction to be short-lived in order to avoid resource contention and concurrency issues. The effects of the transaction on all of the resource managers involved in the transaction are durable—that is, once committed, the effects are permanent and available to other operations. These are the principles of ACID (atomicity, consistency, isolation, durability) based transactions.

Long-running transactions, on the other hand, may span days, weeks, or months – something that a short-lived transaction simply cannot afford to do. These types of transactions typically occur in processes where some form of business-to-business message coordination is required. A long-running transaction may spawn several smaller atomic transactions for individual operations, but the effect of the entire long-running transaction does not conform to the ACID rules that short-lived transactions generally support.

If a failure occurs during an operation in a long-running transaction, automatic rollback of the effects of any previous operations (as occurs in an atomic transaction) is not possible. This can leave external resources in an inconsistent state. With long-running transactions, it is necessary to implement transaction compensation logic to cater for failures. Rollback is through specific compensation logic that understands how to roll back the results of each previously successful individual operation.

Currently, Web Service operations are unable to take advantage of transactional behavior because the SOAP specification does not provide the necessary protocols for enlistment in a transaction. Therefore, each SOAP message is a separate operation in the absence of some higher-level application protocol providing transactional-like behavior (a SOAP processor would not understand this anyway, as it would be application specific).

The WS-Coordination specification, along with the WS-AtomicTransaction and WS-BusinessActivity specifications, aims to provide the necessary protocols for Web Services to take part in different types of transaction. These specifications define a set of standard SOAP headers that allow Web Service operations to be part of a multiparty distributed transaction.

Depending on the consumer requirements, document submission to a Submission Service could take part in a short-lived or long-running transaction, depending on the capabilities of the agency or utility service. The maturity of these specifications needs to occur with general support from the underlying platform before this becomes a reality, however.

Mapping Service

The message mapping services provided by the e-Health Service Node as part of the integration services supply the core functionality to transform source message formats into target message formats. This may be an incoming message format transformed to an internal canonical format used by the service hub, or an internally formatted message to an outgoing message format required by a peer health service.

The following message mapping services are provided by the e-Health Service Hub:

- Support for transforming messages to and from HL7 2.x, HL7 Version 3, XML, and custom flat file formats.

- Support for various inbound and outbound messaging protocols such as MLLP, Web Services, HTTP, HTTP(S), FTP, MSMQ, and MQSeries.
- Standardized Schema Library—a common best practice for integration and brokering systems is to abstract the schema of the incoming message from the schema of the outgoing message. This is accomplished by transforming the incoming message to an internal canonical schema and then to the target schema on the outbound message transfer. This isolates changes to either inbound or outbound message formats, preventing any required changes to correlated subscribing or publishing parties.

Transformation maps are used to process and convert the content and structure of any source information (based on its schema representation) into any target message format (also based on schema). Information is mapped from one or more attributes in a source schema to one or more attributes in the destination schema based on the semantic relationships between the attributes.

In most integration mapping scenarios, a canonical schema is used to represent a standard version of the message schema, independent of the mapping endpoints.

Maps can be reused and modified as needed to implement any number of transformation requirements. These maps are typically based on XSLT, an open standards protocol for transforming XML information.

Orchestration Service

The Message Submission Service routes documents according to rules based on the service address mapping, metadata, or content. In addition, the Submission Service takes advantage of industry standards and specifications such as those from W3C and OASIS, in conjunction with the WS-* specifications, to provide basic messaging requirements such as privacy, integrity, guaranteed delivery, transactional behavior, and attachments.

At a messaging level, these requirements are critical in some cases and desirable in others. What is not covered, however, is the ability to layer an application protocol on top of the simple message exchange that coordinates message flow and delivery/receipt of messages to and from endpoints. This application protocol, or “business process”, generally arises from both static and dynamic business rules. Sometimes the term “system workflow” applies—workflow between local or remote systems that does not require human interaction.

The execution of a business process by an orchestration engine ensures that a message flows through the process and may be delivered to any number of endpoints (partners, services, applications, etc.) based on the rules set out in that process. For example, a message exchanging protocol that manages message flow between a consumer and a payment service provider (PSP) involves the following steps:

- Send a preauthorization request to the PSP
- Receive the acknowledgement receipt
- Check the acknowledgement receipt for confirmed authorization
- Then, if authorized:
- Send the payment request to PSP
- Receive the acknowledgement receipt for the request
- Check the receipt for errors

This process could be further refined to select the most cost-effective PSP based on payment value and the cost per transaction as charged by the PSP.

In this instance, the static part of the business process defines the steps, decisions, and branching required as part of the business logic. It is static because the basic flow remains the same until a change in business requirements may result in some extra steps, such as involving a line of business application in the process.

The dynamic part of the business process defines the parameters or business policy that affects the outcome of any decisions. Changes in business policy require a consequent reflection in runtime behavior to maintain the agility of the business. Following on with the same example, the cost-per-transaction rates may be a part of the dynamic business policy that can be changed immediately to reflect current rates offered by payment service providers, or the payment value threshold by which one PSP is preferred over another may be changed according to the policy in effect at the time.

Figure 37 is a simplified view of the above interaction between two participants. The colored triangle symbols represent the public interfaces, and therefore interaction points in the process, while the document symbols represent data flow.

The WS-BPEL specification (formerly called BPEL4WS) represents the interactions between these participants, and an orchestration engine uses this to execute the process logic. Therefore, BPEL offers a standard way for processes to be represented that is technology-agnostic and can span system and organizational boundaries.

Note: OASIS manages the WS-BPEL specification, which is at Draft stage for version 2.0, and is available from http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel.

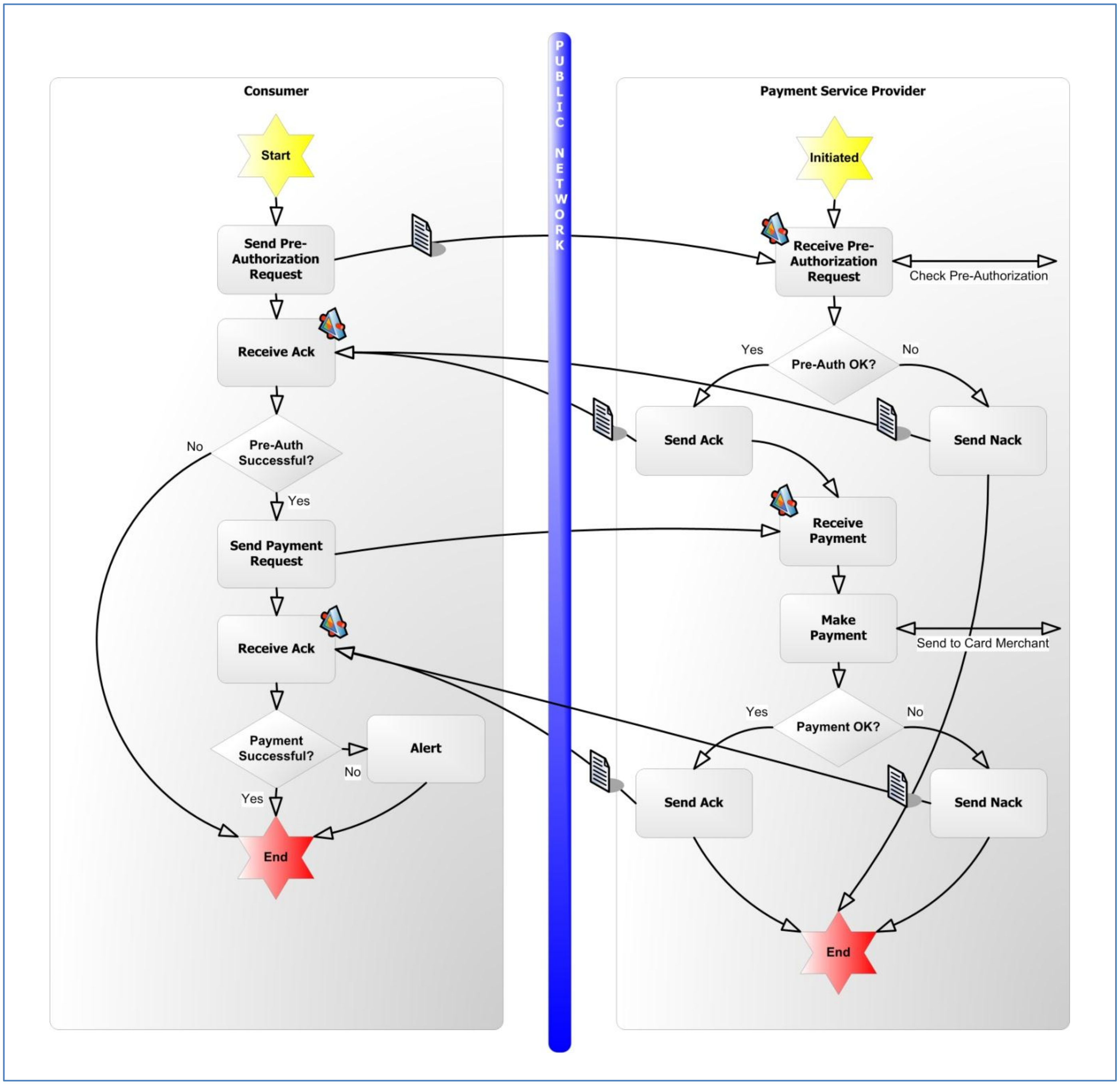

Figure 37. Example of Business Processes Implementing a Business Protocol Between Two Parties

Business Process in the e-Health Services Node

While the node exists to facilitate e-Health communication, some of the communication may be more complex than the simple delivery of a document from one endpoint to another. Where higher-level business protocols and processes exist, the node must represent and execute these processes in a standardized way supported across multiple platforms and technologies. Examples of these higher-level business protocols are those that are published through the *Electronic Health Record Services*, *Health Domain Services* and Health Registry Services (*see these sections earlier in the document for a complete description of the e-Health business processes)*.

Other than the public interface (the business protocol that is used to interact between multiple parties), the inner workings of a business process are generally private to an organization. In the e-Health Services Hub, they exist behind the public service layer and in the Routing, Integration, and Business Process Services private layer, which is available for customization by the owner—whether that is a central host, a jurisdictional body, or a partnership sharing the cost of running a hub.

Data Services

The Data Services provide a set of data access, update, and management services for data that is managed by the e-Health solutions. In all cases, these services are consumed by e-Health business services, and not exposed directly to external systems and services. The following types of data services are provided:

- **Data Access and Persistence**
  These services provide standard data access and persistence services to the e-Health business services and components. These services are provided in a manner that is independent of the underlying data storage technology and database management system.
- **Data Aggregation Services**
  These services provide a means to aggregate data from a number of data stores in accordance with a specific schema definition.
- **Data Caching Services**
  These services store temporary copies of data that have previously retrieved from participating data sources. The caching services provide for faster responses to user queries. The location and use of data caching services are dependent upon data store location, user query patterns, and system performance criteria.
- **Data Loading and Replication Services**
  These services provide for initial data store loading as well as replication to and from other associated data stores.
- **Data Code Set Services**
  These services provide for the storage and translation of multiple code and terminology sets used by the various participating health services and solutions. Alignment to an agreed-upon code set is supported through this set of code set services.

The specific Data Services that are deployed as part of any e-Health solution will depend upon a number of factors including the type, location, and storage mechanism of the data; the data transfer objects used to communicate data between services; and the consumption patterns of the stored data.

Data Access Patterns and Practices

The concepts of patterns and practices are in the index page of the Microsoft Patterns and Practices Center on MSDN. The main index of available Patterns and Practices is at http://msdn.microsoft.com/practices/.
A number of guides specific to best practices for data access and management can be found within this set of Patterns and Practices.

The guide entitled *"Designing Data Tier Components and Passing Data Through Tiers"* outlines a series of best practices for designing and developing a set of Data Services on the .NET platform. This pattern and practice document is available at http://msdn.microsoft.com/practices/compcat/default.aspx?pull=/library/en-us/dnbda/html/boagag.asp

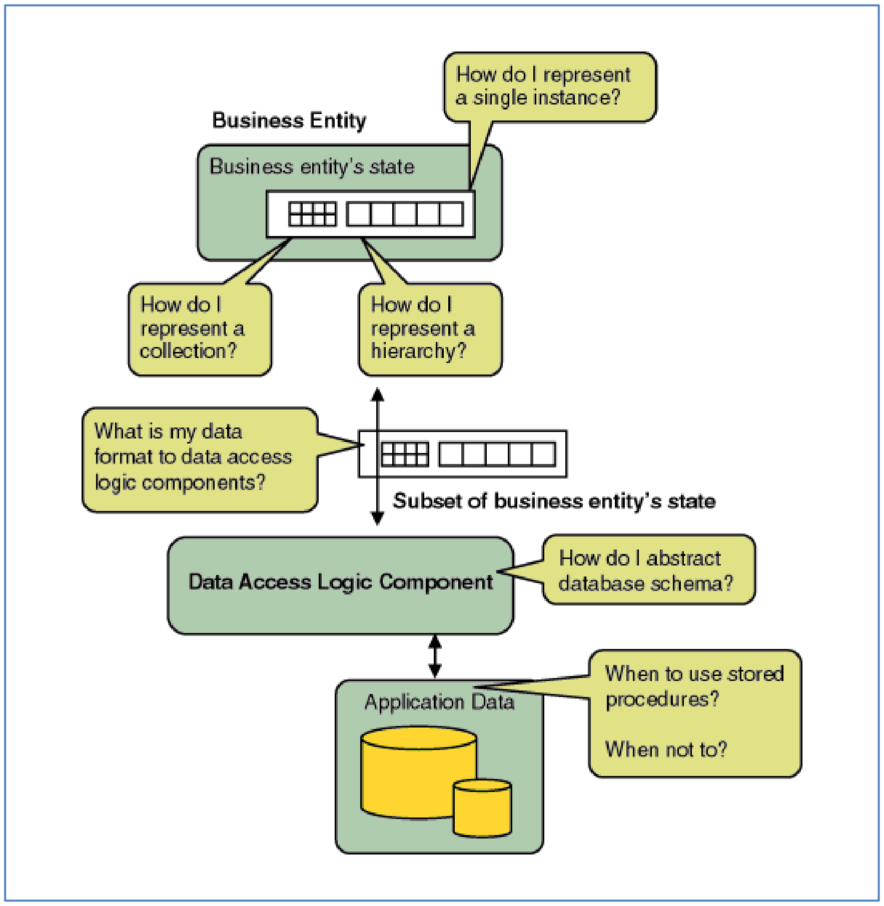
How do I represent a single instance?
Business Entity
Business entity's state
How do I represent a collection?
How do I represent a hierarchy?
What is my data format to data access logic components?
Subset of business entity's state
Data Access Logic Component
How do I abstract database schema?
Application Data
When to use stored procedures?
When not to?

Figure 38. Technical considerations that influence the design of data access logic components and business entities

Communication Services

The Communication Services provide the required foundational communication infrastructure upon which the rest of the e-Health services architecture depends. These Communication Services are supplied through infrastructure which supplies core network and application protocols support, which in most cases is already fully or partially deployed in the target solution environment. It is important that the Communication Services adhere to and support relevant industry standards relative to both network and application communication protocols.

Utilizing Standards for Greater Interoperability

The network protocol services provided by the Communication Services are based almost exclusively on named and de-facto network standards including TCP/IP, (S)HTTP, SMTP, and FTP. Exclusive to healthcare and HL7-based systems is the use of MLLP (Minimum Lower Layer Protocol) as another network-level protocol supported by the Communication Services layer. The basic network protocols can further be extended to include vendor-specific message queuing services such as the Microsoft MSMQ and IBM's MQSeries.

Beyond the network-level communication layer, various industry specifications exist to support application-level communication services. These include WS-Policy for describing the requirements and capabilities of a Web Service, WSDL for describing the contract of a Web Service, XML Schema (XSD) for describing the format of messages, SOAP for providing the basic protocol used to wrap a Web Service request and response, XML for providing the basic structure of a Web Service request and response, and HL7 for providing the accepted standard for health message payloads. The Reference Section at the end of this guide provides links to more information on these specifications.

Typically, HTTP is a high-level protocol (it sits higher in the stack than a network protocol such as TCP) that contains the SOAP request when making a submission to a Web Service. In reality, however, any network protocol such as TCP or UDP is suitable. As long as a "listener" that can receive and process the SOAP requests is running, the network protocol is almost irrelevant. HTTP, however, is a convenient protocol due to its monopoly of browser-based Internet traffic. The convenience of using a Web server that already accepts HTTP or HTTPS on IP ports 80 and 443 means that the network infrastructure (including firewalls, routers, and so on) is already in place to cope with traffic of this type. As such, Web Service SOAP requests naturally fit into this environment.

Due to the proliferation of differing platforms and technologies in healthcare, it is essential to ensure that Web Services are interoperable, regardless of the technology used to implement them. Using the *Web Services Interoperability WS-I* Basic Profile as a basis for interoperable Web Services (http://www.ws-i.org/Profiles/BasicProfile-1.1.html) ensures a baseline of standards that all implementations of a Web Service should adhere to. The basic profile specifies a minimum set of specifications that Web Services should support to ensure interoperability across diverse platforms. These specifications include SOAP 1.1 (although SOAP 1.2 is now ratified as a W3C Recommendation), XML 1.0 and HTTP 1.1 for messaging and message formats, WSDL 1.1 and XML Schema 1.0 for service description, UDDI v2 for service publication and discovery, HTTP over TLS and SSL, and X.509 Public Key Infrastructure Certificate and CRL Profile for security. The HL7 Profile for Web Services provides a standard means of communicating HL7 messages across a Web Services transport.

Integrating the Healthcare Enterprise (IHE) (http://www.ihe.net) is an initiative by healthcare professionals and the industry to improve the way computer systems in healthcare share information. IHE promotes the coordinated use of established standards such as DICOM and HL7 to address specific clinical need in support of optimal patient care. In particular, IHE offers an extensive set of *interoperability profiles* for a variety of clinical and IT infrastructure scenarios, including Cross-Enterprise Document Sharing (XDS). The IHE IT Infrastructure integration profiles are available at http://www.ihe.net/Technical_Framework/index.cfm#IT .

Message Security

After agreement of the core platform of specifications, the next level is to ensure that there is support for message security. As discussed earlier (page *88*), *Message-Level Security* has advantages over transport-based security in that it provides end-to-end protection, independent from the transport and any intermediaries along the route, because the security is applied directly to and carried in the message. Using the WS-I Basic Security Profile for message-level security (http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html) ensures that there is interoperability at the security level and that all parties understand how the messages security is applied. The basic security profile specifies the acceptable security mechanisms for Web Service communication. This includes transport-layer security (SSL and TLS) as well as SOAP message security through the WS-Security specification (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss), and including the security token types supported by this specification. There are additional profiles that specify further token types such as REL (Rights Expression Language) and SAML (Security Assertion Markup Language). In addition, other WS-* specifications support other aspects of message security such as WS-Trust for issuing security tokens, WS-SecureConversation for enabling session keys to be established for session-based security, and WS-SecurityPolicy that complements WS-Security by providing assertions that relate at the message level to the types of tokens supported, the algorithms, and so on.

Service Requirements

The service itself may have mandatory requirements for any consumer that interacts with the service. These may include the supported security tokens or the algorithm for signing the messages. This information must be available in a standardized form across all consumer services, and is the purpose of the WS-Policy specification that enables the specification of service requirements and capabilities through a policy definition discoverable at design time. As the specification does not make any assumptions about policy discoverability, and has no attachment to a Web Service, it is necessary to consider other technology-specific specifications such as WS-PolicyAttachment – which defines the mechanisms for associating policy with Web Services.

How to Address Web Services

To enter into a conversation with a Web Service, it is vital that each party understands the message exchange protocol and the delivery location. Using the WS-Addressing specification (currently in Working Draft status) for message transmission through networks in a transport-neutral manner ensures that both parties can understand how each one addressed the message, and the delivery location for the messages. The specification provides a common means to address Web Service endpoints using an endpoint reference, and to convey information about the routing of a message and invocation of a service using message information headers. This means that representation of both synchronous and asynchronous message flow is possible within a SOAP header in a standardized fashion that is transport-neutral, as opposed to relying on certain transport-specific headers to convey information about the Web Service request. An example of this is the SOAP-ACTION header entry in HTTP that describes the SOAP operation at a transport level, which is the case today with the majority of HTTP-based Web Services. Moving this information into the SOAP header, and therefore into a part of the message, ensures that transmission of the SOAP message is possible over any network protocol, and that the correct operation on the right service is executed.

Reliable Delivery

In any distributed architecture that supports message delivery, there are likely to be some messages deemed critical, and for which reliable delivery is a requirement. Use of the WS-ReliableMessaging specification ensures that messages are transmitted with a guarantee of reliability (http://schemas.xmlsoap.org/ws/2005/02/rm/). This involves a handshake of acknowledgement messages between the two parties (the sender and the receiver) to ensure delivery of the message. In addition to reliable delivery, WS-ReliableMessaging also supports preservation of message ordering so that the receiving service accepts the messages in the order they were sent.

Transactional Support

Sometimes there is a requirement that a Web Service request participates within a transaction that is part of a larger operation. These transactions can range from simple database transactions (known as atomic transactions that typically follow the classic ACID rules) to long-running business transactions that may span days, weeks, or months rather than seconds. Using the WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity specifications ensures support within the SOAP message for the representation of transactions such as these, permitting the recipient of the request to participate in the associated transaction.

Message Attachments

Attachments to messages are quite common in e-Health. There may be a need to attach an image to a message for transmission to a department service. In this instance, the protocol used to communicate with the Web Service must provide support for embedding attachments. Both the SOAP Message Transmission Optimization Method (MTOM) and XML-binary Optimized Packaging (XOP) specifications support embedding binary data with XML into a MIME-formatted message, and then binding the result to SOAP.

Web Service Metadata

With the proliferation of Web Services, it will be necessary to provide a standard mechanism for consumers to discover the associated metadata of a Web Service at design time, in a transport-neutral way. This enables the consumer to understand the requirements and capabilities of the Web Service. The metadata includes policy information (WS-Policy), the contract (WSDL), and the schema (XSD). The specification WS-MetadataExchange provides a standard mechanism for consumers to query a Web Service for its associated metadata.

In summary, the Communication Service must be based on non-proprietary standards and specifications to ensure that supported e-Health services are interoperable across differing platforms and technologies. The creation of a standardized “public highway” that all can understand and interoperate with is essential for the successful adoption of e-Health services.

Deployment Options

The generic nature of the reference architecture and functionality of an e-Health Service Node enables numerous different topologies. Depending on the specific requirements and location of each node, different subsets of the generic functionality may be used – giving flexibility of choice in where and how to perform certain operations, store data, authenticate users, provide reference information, and so on. Possible topologies range from a single central hub to a federated peer-to-peer network of cooperating federated hubs, or multilevel hierarchies that are even more complex.

Single Central Node

Using a single central e-Health Services Hub is the most obvious configuration (see *Figure 39*). The central e-Health Services Hub provides a common platform implementing essential functionality such as identity and security, processing and routing of messages, and reference services, and may hold health records or indexes – depending on the chosen data topology. Concentrating key functions in a single hub shared by all providers of e-Health services offers convenience, efficiency, and reliability, and allows fast deployment and rapid growth both in the number of services offered and their usage. This model is commonly at lower levels (such as a hospital or group), but there are also examples of the central model being used at a regional and national level.

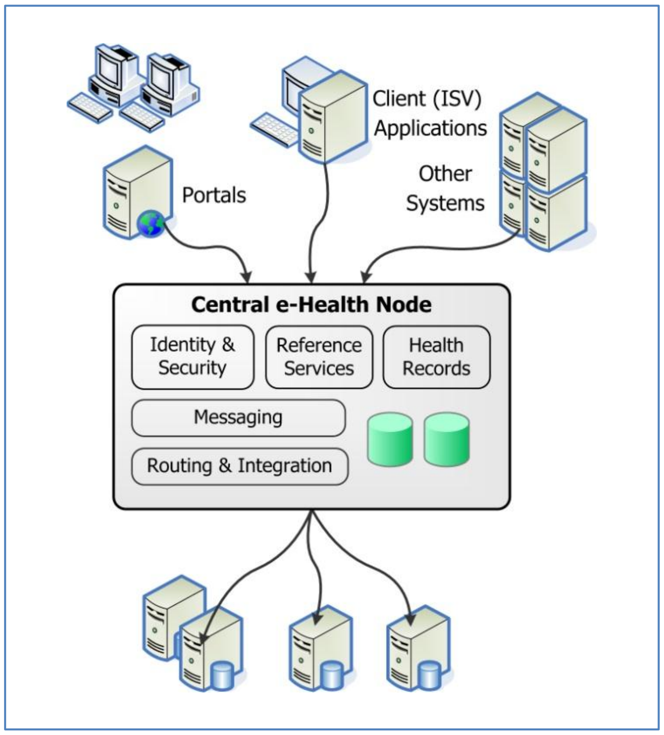

Figure 39. Single Central Node

Node as a Platform for Integration

A cut-down version of the generic node with only the functionality relevant to the role of an integration system can be a very effective platform for connecting to the back-end systems of the service provider (see *Figure 40*).

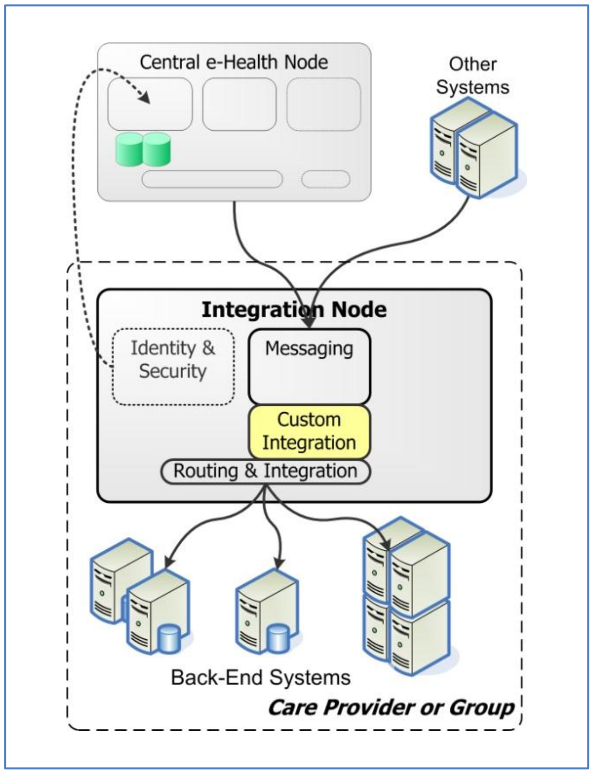

Figure 40. Integration Node

The integration node can use generic messaging, routing, and integration functionality for receiving messages from a central node or other systems, performing the necessary validation and passing the messages on to the target systems. In addition to the generic messaging and integration capabilities, the node can host custom integration and transformation functionality that is specific for the particular health agency or back-end systems. For more details on how the capabilities of the hub can facilitate integration with back-end systems, see the topic *Integration Services* (page *79*) earlier in this document. Also see the section *Isolating Common Functionality* (page *35*) in Addressing Common Architectural Challenges.

The integration node can trust and rely on the central node to validate and authenticate incoming messages, or it can perform additional checks of its own to authenticate and authorize the request. This may involve calling external services, including the central node—especially in cases when the messages are not coming from a trusted central node.

Peer-to-Peer Nodes

In many cases, it could be inefficient or impractical to channel all traffic through a central node, especially if there is not much value added by the central node during the process (for example, validation, authorization, or tracking). Peer-to-peer communication can be used instead, exclusively or in addition to communications via the central node, or for specific types of messages—for example, returning a response directly to the client system, even if the request came through the central node, or at certain phases of conversations comprising multiple messages, such as after the initial interaction.

Instances of the generic node can facilitate effective peer-to-peer communications between services, performing some of the standard tasks that otherwise would require implementation by the services themselves. Depending on the type of interactions and the features required, deployment of different subsets of the generic node functionality is an option.

Figure 41 illustrates one possible flow of messages between peer hubs, where the central hub performs the role of authentication provider and Secure Token Service (STS).

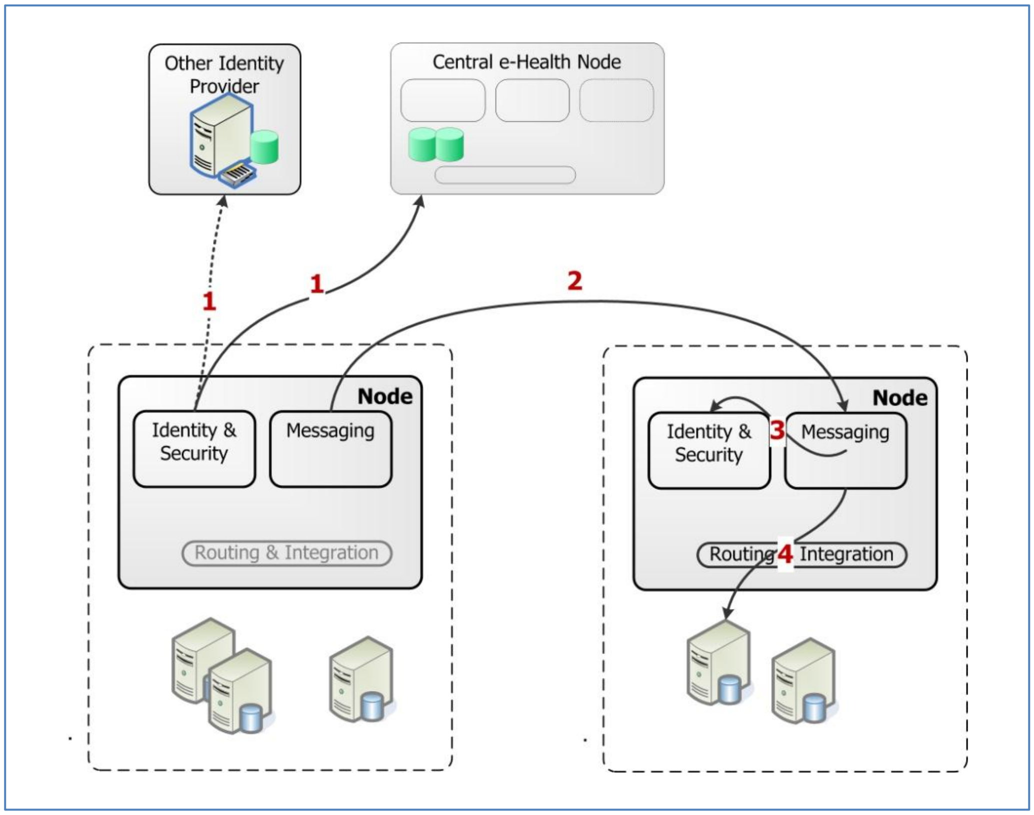

Figure 41. Peer-to-Peer Nodes

The steps within the process shown in the figure are as follows:

1. The originator calls the central node or another identity provider to obtain a security token appropriate for the target, which should be issued by a Security Token Service (STS) trusted by the target service.
2. The message, including the STS token, passes to the receiving node.

3. The validity of the STS token is checked, and appropriate authorization decisions are made by the receiving node's Identity and Security services – with or without using internal to the node data or calls to other parties.
4. The message passes to the target service for processing.

In this scenario, contact with the central node is only during the initial step, in order to obtain a valid STS token. After that, numerous direct interactions between the peer node can take place (within the time window of the validity of the token), without involving the central node – which in this case only plays the role of authentication provider (STS).

Various other scenarios are also possible. For example, after the initial exchange described above, the recipient node may issue its own service-specific "session" token and return it to the originating node. Subsequent calls can present this token, and the recipient can validate them very effectively.

Hierarchy of Nodes

More complex topologies are possible, where node at several levels cooperate and execute within a distributed and federated model. Some of the generic node functionality (such as routing, orchestration, or authorization decisions at different levels of granularity) is distributable between node, based on considerations such as performance, availability, and ownership of data and updating and maintenance procedures. Instead of the "one size fits all" approach, solutions adapt to fit a much broader variety of requirements and constraints. *Figure 42* shows an example of such hierarchy.

The left-hand side of *Figure 42* shows one of the possible flows and distribution of functionality (with emphasis on message routing by a Regional or Group Node) through the steps:

1. The Central Node accepts and processes messages as usual, including performing validation and checking authorization for the requested service.
2. The message then flows to a lower-level Regional or Group Node, which may service several agencies or service providers.
3. The Regional or Group Node may perform further processing and advanced message routing, perhaps to cater for dynamic circumstances such as load redistribution or system failures, or to implement orchestration of business processes between several back-end services. The ownership and responsibility for these remain at the group level, and the Central Node is not involved.
4. Messages pass to the appropriate service providers, which may themselves be fronted by Integration Nodes.

The right-hand side of *Figure 42* shows another flow and distribution of functionality, with emphasis on authorization at different levels of granularity by an Agency Hub (as discussed in *Granularity of Authorization*, page *65*):

1. The Central Node accepts and processes messages as usual, including performing message validation and authentication and authorization checking. This time, however, the granularity of the authorization is at the service level rather than Regional or Group Node level.
2. The message then flows to the Agency Node.
3. The Agency Node may perform additional authorization checks based on the claims received with the message. For example, the identity can map to more specific roles and assignments (including delegation of authority) that exist at this level.

4. Messages pass to the appropriate service providers, which may be fronted by Integration Nodes.

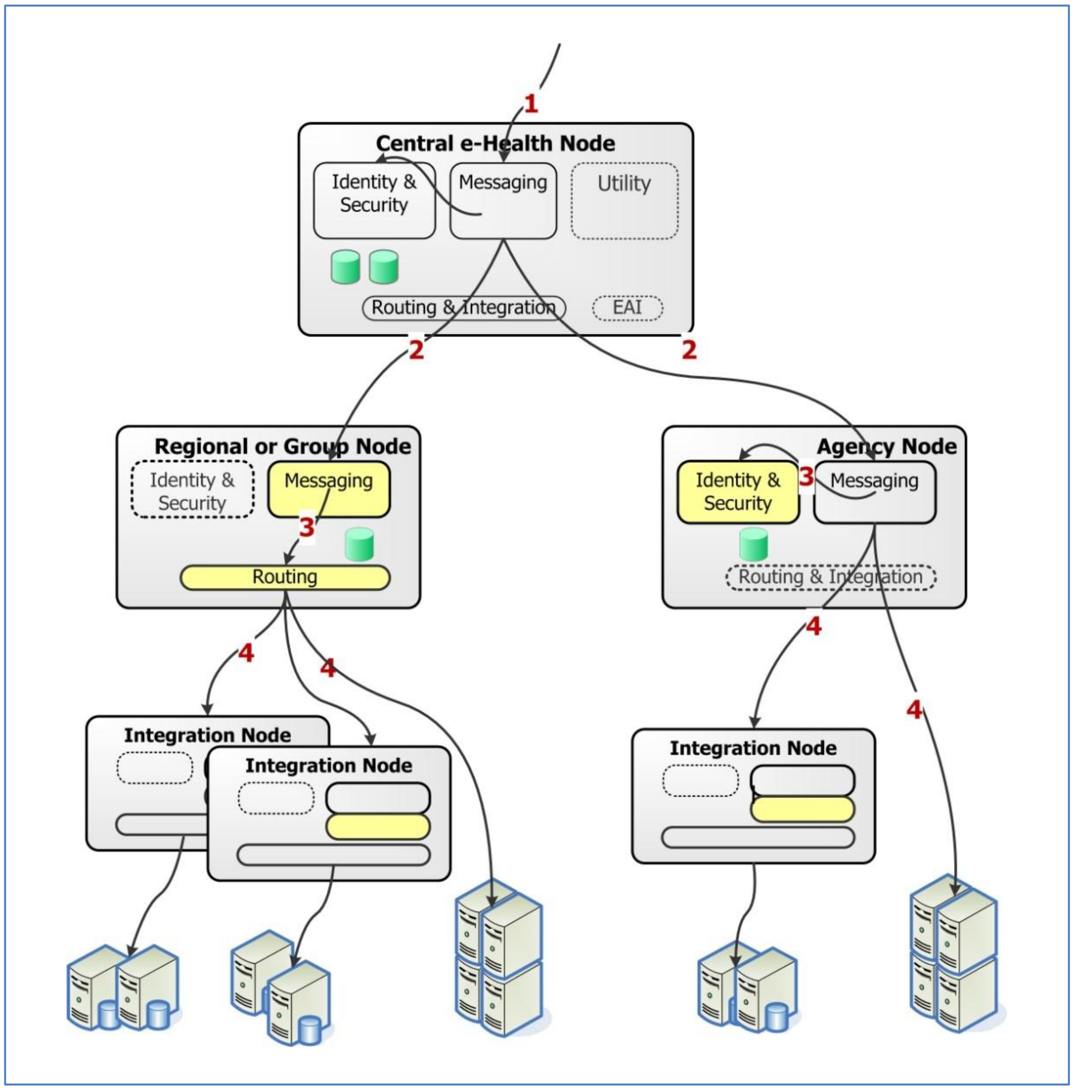

Figure 42. Hierarchy of Nodes

The terms “Group”, “Regional”, and “Agency” Node, and the selected functionality in the examples above, are for illustration only. Various other combinations are, of course, possible. The generic node functionality is distributable between several instances of the node, allowing solutions tailored to cater for the specific requirements and constraints.

Cloud Services, SaaS, and Software + Services

In the previous sections, we emphasized the important flexibility of the architecture to support a variety of topologies and distribution of particular services between nodes, and provided some examples.

Deciding where particular functionality or data should reside in the system, how it will be accessed by other nodes, and ensuring the agility to change such distribution over time is, in our view, one of the key architectural considerations for a successful e-Health system.

All these discussions also apply to the case where some of the functionality or data may be provided externally, by a hosted or “cloud” service. Architecturally, there isn’t a significant difference between the case where a node “consumes” services from another node (within or outside the e-Health domain), or from a “cloud” service. The requirements for openness and stability of interfaces, robustness, and scalability are much higher for a “cloud” service – but from the consuming node’s perspective, it is still very similar—such as consuming identity and authentication services for care professionals from a national node (within e-Health) vs. authenticating citizens through a shared e-Government facility (a node outside the e-Health domain) vs. authentication of citizens with a “cloud” service. The differences are mostly in the deployment, hosting arrangements, business model, and so on – largely transparent to the consumers of those services. Wherever possible, we should aim to define interfaces and interactions with services in a way that can be consistent, supports all these models, and allows switching between them with little disruption.

With the growing prominence and widespread availability of “cloud” services primarily in the consumer space, interesting new opportunities emerge for e-Health services solutions. Offerings like *Microsoft HealthVault*[4] and *Google Health*[5] allow consumers to store, organize, and share their health-related information in a “cloud” service. It is also possible to upload health information from one provider and then share it with another – effectively using the cloud service and data store as a bridge between two systems that otherwise may not be able to exchange information directly. Since the patient/consumer is in complete control and decides what information to upload and who to share it with, many of the tricky issues related to privacy of personal data and user consent (typical obstacles when connecting two systems directly) are substantially simplified.

The diagram in *Figure 43* illustrates one possible usage scenario of a Cloud e-Health Node to make available health information originating from one care provider to another.

The numbered sequence of steps is as follows:

1. The user asks the Care Provider A to upload particular health information to his or her personal data store in the Cloud e-Health Node.
2. Appropriately formatted information is transmitted to the Cloud e-Health Node via the Provider A integration facility (Node A), using the published protocols for interaction with the Cloud e-Health Node.
3. The user accesses the Health Portal provided by the Cloud e-Health Node, and authorizes Care Provider B to access selected portions of his or her health information
4. Care Provider B requests and receives the selected health information for the user from the Cloud e-Health Node. The information may then be stored (if permitted) in local systems for Provider B.

[4] http://healthvault.com

[5] http://www.google.co.uk/intl/en-US/health/about/index.html

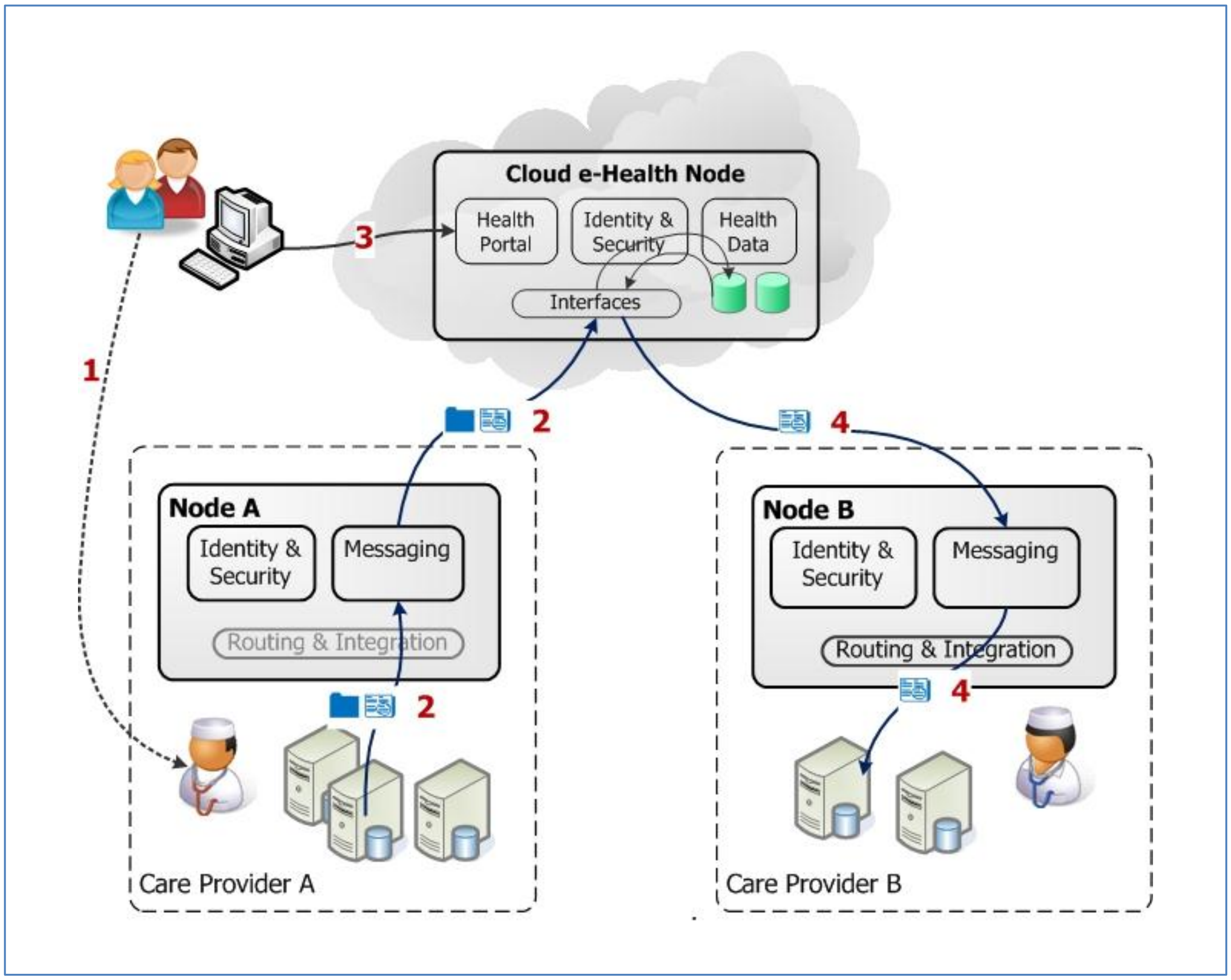

Figure 43. Using Cloud Services

The currently fashionable "*Software As A Service*" (*SaaS*) trend emphasizes the role of externally ("cloud") hosted services, often to the exclusion of any local functionality and data. The assumption is that with these provided by a hosted service, systems can rely exclusively on such SaaS, and deliver all the necessary functionality through a slim client platform consisting of low capability (i.e. cheap) hardware, operating systems, and browser. While this approach may be adequate in some scenarios, it is not likely to be universally applicable, to provide the rich user interaction experience, to provide seamless integration with common desktop productivity tools, and to offer the ability to work offline (disconnected). For these reasons, we believe that a broader, more inclusive approach called "*Software + Services*" (*S+S*) is a better choice. Like the Hybrid Data Model (page 21) discussed earlier in this guide, S+S combines the benefits of locally deployed functionality and data with externally provided shared services—allowing the choice of the appropriate (for the context and usage scenario) mix of both, and enabling easier moving of the boundary between "local" and "cloud" as needs, requirements, and technology evolve over time.

Securing the System

One of the most important aspects of any computer system is securing it in such a way that protects it against malicious users, automated code attacks, denial of service incidents, and exposure of confidential information. This is particularly important in e-Health projects because the highly sensitive nature of the data it contains makes it a prime target for attacks aimed at identity theft, as well as damage to the data and/or service interruption. The architecture of a secure solution requires both general security topics, as well as those specific to e-Health integration solutions, be addressed.

Generic Approaches to a Secure Solution Architecture

The three main security goals when building any system are as follows:

- **Confidentiality**: The system must ensure that access to data, and to all operations that affect data, are only available to the appropriate users. No other user should be able to view, delete, or change any data.
- **Integrity**: All actions taken with data must protect that data from corruption. Operations must succeed and leave the data in the correct new state, or all fail and leave the data unchanged—and notify the user and/or appropriate authority of any failure. All changes to data must also be visible, generally through audit tracking, and allow reconstruction of the original data and monitoring of changes.
- **Authentication**: Each user requires accurate identification at the point of entry, and again as required, before permitting any actions within the application. This includes viewing, deleting, and changing data.

In general, these three goals require an architecture that uses logical tiers, and implements security across the tiers. As each tier of the application makes a call to another tier or a remote service, it provides authentication information for that call, based on the authentication of the user at the initiation of the current service call. If the receiving tier or service requires additional information, it will prompt for this before allowing the process to continue.

Security Patterns and Practices

The concepts of patterns and practices are explained on the home page of the Microsoft Patterns and Practices Center at http://msdn.microsoft.com/en-gb/practices/bb190357.aspx . The main index of available guidance related to security is at http://msdn.microsoft.com/en-gb/practices/bb978782.aspx#security .

The Microsoft Security Developer Center provides guidance on the architecture and implementation of secure solutions using the .NET Framework and Web Services. The page http://msdn.microsoft.com/security/default.aspx contains general advice, plus a link to the Microsoft Patterns and Practices Center that provides scenario-specific recommendations.

The first step in implementing a secure architecture is to understand the threats faced by the application, and the techniques available to counter these threats. *Figure 44*, taken from the document "*Improving Web Application Security: Threats and Countermeasures"* (http://msdn.microsoft.com/en-us/library/ms994921.aspx), illustrates the common threats. This pattern and practice document is available at http://msdn.microsoft.com/library/en-us/dnnetsec/html/ThreatCounter.asp .

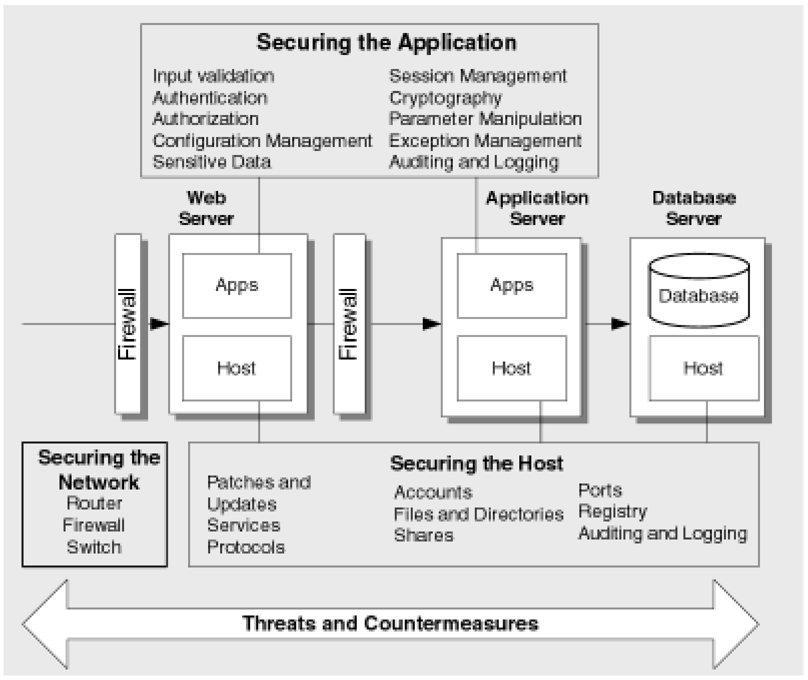

Figure 44. Scope of Improvements for Web Application Security: Threats and Countermeasures[6]

This pattern and practice document describes the concept of threat modeling, securing the individual layers of the application, securing the host, and securing the application itself. It also contains checklists that help to turn the information into actions, and a series of "How To" articles that describe individual tasks in a systematic fashion.

Security Best Practices

More information on the design of secure solution architectures is available from the Security Best Practices site at http://msdn.microsoft.com/en-gb/practices/bb978782.aspx#security. This site provides a series of articles that describe best practices in specific scenarios. These include:

- Secure Coding Guidelines for the .NET Framework
- Code Access Security Policy Best Practices
- Minimizing the Code Exposed to Untrusted Users
- Designing Application-Managed Authorization
- Building and Configuring Secure Web Sites

[6] From "*Improving Web Application Security: Threats and Countermeasures*" http://msdn.microsoft.com/en-us/library/ms994921.aspx

Security Architecture Specifics for e-Health Systems

E-Health integration projects share many of the characteristics of other similar large-scale online services (such as online banking and e-commerce retail sites). All the usual security threats, and the general guidance summarized in the previous section, still apply. In addition, there are some unique security aspects specific to the architecture of e-Health systems discussed in this section.

Infrastructure and Networking

In some countries, dedicated infrastructure, including networks, may exist specifically for use by healthcare agencies and providers. It may be considered to be more secure (at varying levels) than public networks or the Internet, with specific rules and accreditation procedures that control who can connect to it, and how. After connection, the infrastructure may provide sufficiently secure communication at the network connectivity level between parties, which simplifies the integration process.

In cases where such infrastructure is available and already connects the providers of e-Health services, exposure of their services to the public electronically in a secure manner is easier through a common e-Health Services Node. This can act as a bridge between the secure healthcare infrastructure and the public networks. Instead of each service owner having to meet the security required within their systems before connection to the secure and public networks, this work (and the relevant accreditation) can be performed only once for the central node. The service providers then need only to connect to the shared node.

However, even when such secure infrastructure is available, and the service providers can benefit from it, oversimplification of the architecture is still undesirable. There should be no assumption that all service providers, now and in the future, will connect uniformly through such a secure network.

The architecture should support features adequate for secure communication over public networks, preferably as a simple configuration option on a per-service or per-node basis. This enables the common e-Health Services platform to accommodate future requirements such as extending the range of providers of e-Health services beyond just healthcare agencies that have access to the secure network.

The accumulated experience from numerous e-Health integration projects in many countries shows that the requirements for such flexibility usually emerge much sooner than originally anticipated, and a constrained architecture designed under such assumptions about the specific types of connectivity required may lead to expensive rework later.

Securing Hosts

E-Health systems are often a complex web of different nodes, on a variety of platforms, with distributed ownership and maintenance procedures, and sometimes lack central authority enforcing consistently and effectively good security practices. While in some countries all elements of the healthcare infrastructure must be compliant with the appropriate security standards (like ISO 17799) and accredited before allowed into production use, this may not the case everywhere. Adequate security of the individual hosts in such environments is a far more complex task than in commercial environments. Therefore, it is very important to assume that some hosts may suffer successful attacks, and need to have sufficient additional layers of defense to cope effectively with such compromise should it occur.

Assumptions about, and over-reliance on, the physical security of systems participating in an exchange of messages may be a reasonable compromise for systems hosted in secure data centers, such as a hospital. An example of this might be deciding that TLS/SSL mutual authentication at the endpoints is a sufficient guarantee that the requests must be coming from a trusted source, without a requirement for message-level security.

However, when the same service becomes available to a wider client base, the risk of a compromised host is much higher. For example, opening the service to remote pharmacies, where the hosts are not physically secure and the staff are not IT professionals, requires additional layers of security—in particular over-the-wire message encryption.

Architecting appropriate (multiple) layers of security into the system from the start, with the ability to switch them on and off as needed, is the recommended approach to achieve flexibility and to gracefully handle a wide variety of current and future, known and unknown, requirements.

Securing hosts remains an important element of the overall security agenda, and the recommendation is for implementation to as high a level as possible. Depending on the platforms and versions of system software used, a range of effective options are available—from those designed for managed environments (SMS, SUS), to consumer-oriented services (like automatic Windows Update)—and should be used to keep systems adequately patched with all the latest updates and security enhancements.

Denial of Service Attacks

Denial of Service (DoS) style attacks are common against commercial and public systems, and are likely—with an even greater potential impact—against e-Health systems. In addition to being an attractive and high-profile target, the scale and the number of users who may be affected is greater. With this in mind, special care is required to minimize the opportunities for such attacks against e-Health hubs. Some typical areas for such attacks include:

- Registration of large numbers of *bogus users* – when the legitimacy checks on user registration requests cannot be effectively accomplished, an attacker can create a very large number of "made-up" users. This can expend significant computational resources and permanently consume storage for such registrations. Both of these factors can cause disruption of the service for legitimate users.
- Repeated *invalid logon attempts* – in the absence of effective "volume throttling" control, such as a count of failed attempts followed by account closure, an attacker can execute a large number of authentication requests. Again, this consumes server resources, and may trigger account lockout through repeated failures for legitimate users.
- Repeated *computationally intensive requests* – any other type of request executed repeatedly, which consumes server resources. This includes submitting valid but large documents through a document submission system.

Protecting against such attacks is an important consideration that affects the architecture of the system. Following are some examples of techniques to minimize the possibility of a successful attack.

Countering the Registration of Bogus Users

If there is no reliable method to confirm whether user registration attempts originate from legitimate users, make the initial user registration dependent on some other process that can be validated—for example, successful enrollment for a service. Avoid allowing users to register simply by choosing a name or ID and a password, because this provides no effective way of distinguishing real from malicious registrations.

Binding of the identity to a target service requires successful matching of the information provided by the user with some reference data, and so new users can only register after a successful enrollment for at least one service. This effectively eliminates the risk of malicious registration of a large number of bogus users.

For details of binding users to services, see *Initial User Provisioning – Knowledge-Based Authentication* (page 52) in the *Identity Management Services* section earlier in this guide.

Countering Invalid Logon Attempts

Restricting repeated invalid logon attempts can be difficult, especially when random User IDs are used. Keeping count of invalid attempts needs some unique ID to be associated with each attempt, and this is typically the User ID. A random spread of these malicious attempts across the entire range of possible IDs makes such counting at the system or application level impossible.

One distinguishing characteristic of such attacks is a high frequency of requests coming from one or a few sources, and appropriate defense may be possible at the network infrastructure level based on the location address (for example, the IP address). However, in cases where large numbers of legitimate users may be accessing the services from an environment that presents itself as a single source (for example, through a proxy server), this may incorrectly be identified as an attack. Therefore, any such defenses require very careful tuning to make this distinction.

To minimize any disruption caused by attacks locking out legitimate users, it is important to incorporate a self-healing mechanism—for example, lockout for a configurable period only, followed by an automatic unlock. This way, if an attack causes the lockout of a large number of users, there will be an automatic recovery after some period of unavailability of the service to these users.

In the absence of a self-healing mechanism, a successful attack will not only disrupt the service when it occurs, but may also affect the helpdesk, which must reset all these users manually.

Countering Computationally Intensive Requests

An example where designing to minimize the impact of malicious computationally intensive requests is essential is the scenario of digital certificate renewal. Digital certificates may bind an identity with the appropriate e-Health service enrollments. Logons with a certificate normally involve checking the origin of the certificate (the provider) and the validity (not expired and not explicitly revoked), and then mapping the certificate unique identifiers to the identity. This enables lookup of other mappings that link the identity and service enrollments.

When external certificate providers are used, the periodic renewal of certificates may occur, as far as the e-Health node is concerned, out-of-band. For example, when a certificate is renewed (some providers do this transparently to the user), they issue a new certificate with the same details, except that it contains a new serial number and expiration date. Therefore, the first authentication attempt against the hub with the renewed certificate would fail the normal matching, because the node is unaware of the renewal and still holds the old serial number. To handle this gracefully, a more complex matching of other details from the certificate is required and, if successful, a silent renewal takes place within the hub. It updates the stored serial number, while preserving all other relationships and linkages.

To support this renewal logic, the node must store more details beyond just the serial number (perhaps the distinguished name and other attributes) to recognize and match the old and the renewed certificate. The search for such matches on long text strings, especially when the user population is large, could be expensive in terms of resources used. This search will be invoked every time a certificate serial number is not found in the local reference store, including the case where authentication is attempted with an arbitrary certificate from the trusted provider. Some of the techniques commonly used to minimize the impact of this type of attack are as follows:

- Precede the search and matching of the long identifying string (distinguished name) with matching on a hash of that string – which is a much shorter value. This makes the search operation more efficient. Compute and store the hash along with the original distinguished name, appropriately indexed, then compute the hash of the distinguished name from the certificate and search for this. Although the probability of collisions (where two different strings produce the same hash value) is extremely low, comparison of the original distinguished names is possible after successful matching on the hash.
- Employ tracking by certificate ID of authentication attempts. The possession of a valid certificate from a trusted provider is a prerequisite to initiate such authentication (otherwise it would fail the validation checks prior to the searching for a matching "old" certificate), so it is likely that the same certificate(s) will be used repeatedly. Keeping track of suspicious certificate IDs that cause repeated failures, and checking this "black list" before more expensive operations are attempted, can minimize the impact of such attack.

These are just a few selected examples illustrating the additional challenges and considerations in the area of security that influence the architecture of e-Health solutions.

Performance and Scalability

Implementing a solution that is secure, reliable, and robust is a necessity, but many solutions fail to provide the required performance, and hence do not achieve user acceptability, after deployment into the “real world.” Testing can highlight many issues and locate sections of the application that are causing bottlenecks or affecting overall throughput. However, a thorough understanding of the requirements and the actual or projected usage patterns for the application is necessary to be able to build in the required performance levels.

Microsoft provides a complete online book as part of the Patterns and Practices group documentation. The publication “*Improving .NET Application Performance and Scalability*” is available from http://msdn.microsoft.com/en-us/library/ms998530.aspx. It discusses the various roles involved in the life cycle of implementing a solution, including architects, designers, developers, testers, and administrators. *Figure 45* (taken from that document) shows the overall scope of the guide.

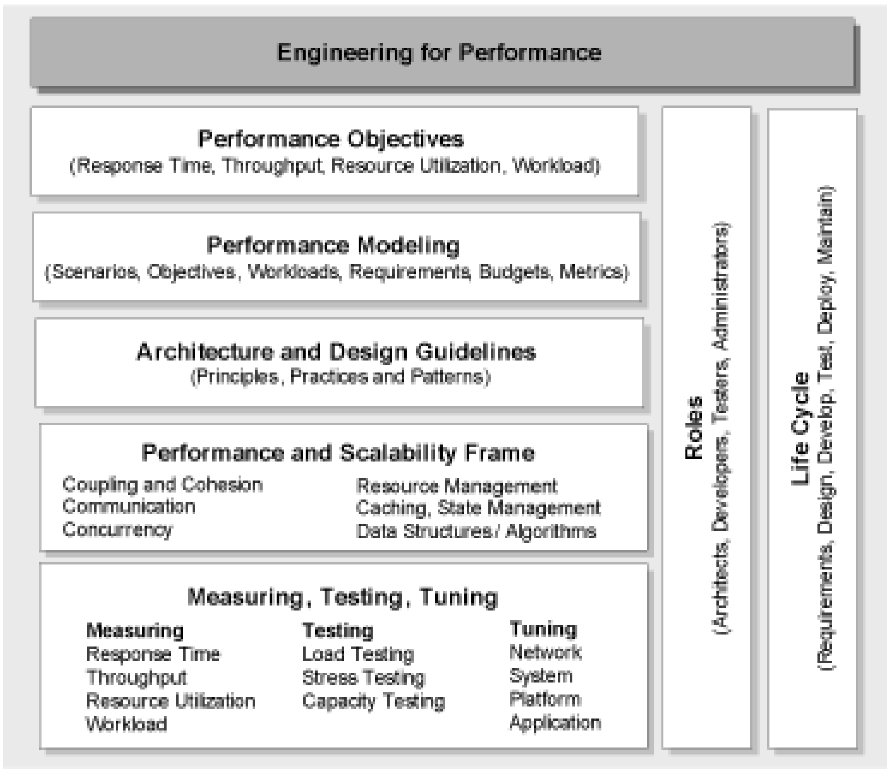
Engineering for Performance
Performance Objectives
(Response Time, Throughput, Resource Utilization, Workload)
Performance Modeling
(Scenarios, Objectives, Workloads, Requirements, Budgets, Metrics)
Architecture and Design Guidelines
(Principles, Practices and Patterns)
Performance and Scalability Frame
Coupling and Cohesion
Communication
Concurrency
Resource Management
Caching, State Management
Data Structures/ Algorithms
Measuring, Testing, Tuning
Measuring
Response Time
Throughput
Resource Utilization
Workload
Testing
Load Testing
Stress Testing
Capacity Testing
Tuning
Network
System
Platform
Application
Roles
(Architects, Developers, Testers, Administrators)
Life Cycle
(Requirements, Design, Develop, Test, Deploy, Maintain)

Figure 45. Scope of the Microsoft Patterns and Practices Guide on Performance and Scalability[7]

[7] Taken from “*Improving .NET Application Performance and Scalability*” at http://msdn.microsoft.com/en-us/library/ms998523.aspx

Capacity Planning

It is generally impossible to design a solution that will provide acceptable performance unless you can project the requirements during the design phase. This is especially true of a distributed system, where simply adding extra memory, CPU power, or disk space to a server may not benefit performance if there are bottlenecks in other stages of the distributed processes. Capacity planning allows the designer and architect to determine a realistic set of requirements for the application as a whole, and from this calculate and test the requirements for individual parts of the application.

The article "*Tools for Capacity Planning*" published by Microsoft at http://msdn.microsoft.com/en-us/library/ms978394.aspx explores the general principles of capacity planning and system requirements analysis, and focuses on the wide variety of tools that are available for testing the different components of an enterprise system. It discusses:

- Gathering the requirements information: the number of users expected at peak and on average, the usage pattern profile, the required response time for each process or activity, and other factors such as the maximum percentage CPU load you will accept on the servers.
- Using a range of sizing tools to estimate the hardware required, including validation of the capacity of a specified configuration, and the use of load simulators such as Microsoft Application Center Test (ACT) and LoadSim to test the configurations.

Understanding System Performance Factors

A more detailed discussion of performance involves some of the factors that directly affect the network and hardware implementation you choose. It is vital to understand the implications of percentage-availability requirements, and the way that scaling a solution can meet requirements. For distributed enterprise applications, locating any single points of failure that could prevent the complete system from performing correctly is obviously a major issue, and can be mitigated by adopting a load balancing and/or clustering technique where several servers are connected together to share the load. If one server fails, the remaining servers take over and share the load (a process called failover).

Load balancing and clustering techniques include Microsoft Cluster Service (MSCS) and Network Load Balancing (NLB), plus a range of hardware solutions available from other manufacturers. Most enterprise-level application components such as Microsoft Exchange Server, Microsoft BizTalk® Server, Microsoft Internet Information Server (IIS), Microsoft SQL Server®, and Microsoft Commerce Server can benefit from these products. It is also important to ensure adequate protection for hard disks and other storage components against individual failures. Fault-tolerant RAID disk configurations provide a solution, perhaps as a shared disk array connected via a high-speed channel using fiber or SCSI.

Scale Up or Scale Out?

The term "scale out" refers to the process of clustering and load balancing a solution, because it spreads the processing load outwards over multiple machines and multiple hardware installations to provide failover protection for applications. However, the alternative approach, named "scale up," is suitable for processes that must offer high availability, and which require servers with large memory and processing power capabilities.

Fault-tolerant servers are available, which use hot-swap components that allow the server to survive hardware failures without interruption to the application. Combined with new processor types and the ability of operating systems to address large quantities of memory, the scale-up solution is a useful alternative approach. The document

Overview of System Performance at http://msdn.microsoft.com/en-us/library/ms978391.aspx discusses the issues of availability, scalability, and performance in detail, and provides links to other useful resources.

Performance Tuning of Windows Services

As well as implementing a robust and reliable infrastructure that can cope with individual hardware failures, the configuration of each service in use affects overall throughput of applications. For example, careful configuration of IIS and a judicious choice of session state maintenance techniques will help to tune the performance of ASP.NET and Web Services. Meanwhile, the use of correct indexing of the data can greatly enhance the performance of SQL Server.

The guide "*Tuning .NET Application Performance*" (http://msdn.microsoft.com/en-us/library/ms998583.aspx) discusses performance tuning in ASP.NET, and contains links to other documents concerned with performance tuning of IIS, and SQL Server indexing considerations. In particular, it discusses the technique of using stress testing and Windows performance counters to detect bottlenecks in the application.

Performance Considerations for Web Services

The use of Web Services offers numerous benefits, but also presents some additional challenges in terms of performance. In general, Web Services involve more processing work and network bandwidth than traditional integration methods like direct binary-format data access against a relational database. Serialization of data into XML at the boundary of a service introduces additional processing overhead. Using message-level security to Web Service calls (which is often required – as discussed in the section *Message-Level Security* earlier in this guide) also introduces additional overhead for calculating digital signatures, adding authentication information to outgoing messages, and validating these by the recipient Web Service. In order to minimize the performance impact while still benefiting from Web Services, the granularity of functionality exposed by services should consider the invocation models used and other best practices.

The chapter Improving Web Services Performance from the guide referenced at the start of this section at focuses on providing design guidelines and demonstrating techniques such as state management, asynchronous invocation, serialization, and threading. A thorough understanding of these topics is necessary to develop efficient Web Services.

Architecting e-Health Solutions for Performance and Scalability

Good understanding of the general principles and techniques for the architecture and design of scalable, high-performance systems (outlined in the previous section) is the essential foundation for a successful implementation of any large-scale system. In addition to these factors, there are some performance and scalability aspects specific to e-Health systems that may further influence their architecture.

Higher Expectations and Requirements

Typically, the performance and availability requirements and expectations for e-Health solutions are higher than for comparable commercial systems. E-Health solutions are often considered part of the critical national infrastructure, expected to be always available, responsive, and able to gracefully handle significant peaks of unexpected workload. Any failures such as delays or unavailability of services are highly embarrassing, and can affect very large numbers of users, which undermines public confidence and weakens one of the key drivers for using online services – the ability to conduct electronic interactions faster and easier than through traditional channels.

Size and Growth

E-Health solutions often start on a small scale and then grow over time as the popularity of the online services and the number of active users increase. The three main dimensions of growth, the product of which determines the required capacity, are as follows:

- Number of users (total and concurrently active)
- Number of services available
- Level of usage of each service, and complexity of the functionality offered

With all these dimensions typically growing over time, the demand for handling increased workload requires careful planning, and architecture capable of scaling smoothly.

It is rarely an option to design, procure, and deploy from the start a system sized to handle the expected workload many years ahead. It is more common to design in the capability for such growth, deploy a small starting configuration, and then add capacity as required. This ability to add more processing power with minimal disruption to the active production system is critical for e-Health services projects.

It is important to have reliable estimates for the absolute maximum expectations in various dimensions: users (is it 1 million, 100 million, 1 billion?), services (10, 100, 1,000?), transactions/requests per second (100, 1,000, 10,000?). The architecture of the solution and the technical options available can change significantly depending on these factors—for example, using a single directory for a billion, or even 100 million, users might not be the most appropriate option, and a more distributed or federated model should be considered. The benefits from concentrating certain services in a single central node, and their shared use by many service providers, may require evaluation against the resulting size and capacity necessary from that node, and a more distributed model adopted.

Distributed Nature of e-Health Systems

In addition to purely technical options and considerations for scaling up, scaling out, and geographic distribution, other factors often influence decisions on the distributed architecture that is appropriate. E-Health systems are often composed of a diverse web of departmental/agency, central, regional, and local systems, with their respective owners, funding, suppliers and partners, and project timelines. Synchronizing and optimizing all these into a coherent integrated architecture from a purely architectural and technology perspective is rarely possible—consideration of various political, commercial, and legal factors often results in a technically suboptimal architecture.

The flexibility (or lack of) for rearrangement and rationalization of the existing distribution and topology of the participating systems is very important. Identification of possible constraints and limitations that may influence the shape of the solution is a prime necessity.

Key principles driving the reference architecture for an e-Health integration solution are to provide maximum flexibility and to be able to accommodate a variety of requirements – including the diverse and distributed nature of e-Health systems.

Some of the features of the architecture that provide such flexibility and help achieve performance and scalability are as follows:

- A *federated approach* to initial identification of users and subsequent authentication, allowing distribution of the reference data and credential stores (see the section *Identity Management Services*, and *Federated Authentication* in the *Authentication and Authorization Services* section earlier in this guide). Among other advantages, this also allows handling of a very large numbers of users, managed in a distributed fashion.
- Support for a *variety of topologies* composed of many instances of the generic e-Health Services Node, allowing the distribution of functionality and workload to match the specific requirements and constraints (see the *Deployment Options* section earlier in this guide).

Usage patterns

Given the flexibility to implement some of the functionality as a cascading sequence of calls between autonomous, distributed services, it is very important to consider the performance impacts of such distribution. An authentication call to a node may require a call to an external authentication provider. The latency of the original call depends on not only the performance of the hub itself, but also on the availability and performance of the external to the hub service it relies on. While there are situations where such distribution is desirable or even inevitable, consider the impact on performance before making a decision.

Another important aspect affecting performance is the choice between synchronous and asynchronous processing. While Web Services support both, the general recommendation is that the use of synchronous (RPC-style) calls should be restricted to cases where it is necessary, and the responsiveness of the target service is adequate. Making a synchronous call to a remote Web Service blocks the processing flow, and ties up resources throughout the whole chain. If the latency of that call is significant, a relatively small number of concurrent "calls in progress" can exhaust resources such as threads and memory on the server making the calls—in other words, the hub. It also affects overall performance, including making the whole service unavailable.

In contrast, asynchronous calls complete the "request" part as soon as the call goes out, and release most of the resources used. Processing of responses takes place independently as they arrive, with only a little extra work required to reconcile them with the original requests. The time lag between the request and the response, while still important because it affects the overall latency for the original caller, has little impact on the performance of the hub. As a result, asynchronous processing can generally achieve higher throughput than synchronous processing, and should be preferred.

The asynchronous model of communication also allows more graceful handling of bursts of activity by buffering the requests and processing them as quickly as the throughput capacity allows. This may occur implicitly (provided transparently by the platform) or explicitly (through implementation of queuing or store-and-forward mechanisms). As long as the total backlog of requests can be stored, and then processed within an acceptable timeframe, the usage model remains the same and provides the expected level of service. In contrast, synchronous calls have far less tolerance for peak loads. On reaching the processing capacity limit, further requests begin to fail and require resubmission by the caller.