Cookbook for Regional Interoperability Detailed Design Paper #028

Non-Functional Requirements for Regional Infrastructure

FOR GENERAL DISTRIBUTION

Version 2.0 – 26th April 2020

Abstract Interoperability Cookbook Anchor Points

Section	Title
3.1	Regional Infrastructure

Table of Contents

Version Control

Version	Release Date	Released By	Reason for Release
0.1	8/03/2020	R Hickingbotham	Preliminary draft (monitoring)
0.2	13/03/2020	R Hickingbotham	Preliminary draft (performance)
1.0	13/04/2020	R Hickingbotham	Preliminary draft (full version)
2.0	26/04/2020	R Hickingbotham	For General Distribution

Reviewers

Initials	Name	Role	Organisation
JM	Julia Millman	Programme Manager	YHCR
GH	George Hope	Lead Technical Architect	YHCR
TD	Tim Davey	Lead Technical Architect	PA Consulting

1 Introduction

1.1 Purpose of this Document

This document is one of a series of design papers which underpin the Abstract of a Cookbook for Regional Interoperability (the Abstract Cookbook). These papers, in their totality, describe the technical components and the standards which form the YHCR System of Systems (SoS). They are intended to inform the development or procurement of software and so are expressed at a level of precision which aims to avoid ambiguity but with the consequence that they are targeted at technical readers.

Design papers are anchored to topics which are discussed in the Abstract Cookbook. They are elaborations of the concepts which were first introduced by the abstract and new content is further detail rather than variations of previously established core principles.

This document (design paper 028 - “Non-Functional Requirements”) establishes requirements and a design for:

- Performance and Scalability;
- High Availability;
- Backing and Recovery;
- Monitoring and Alerting;
- Release Management.

Designs in these areas are tied to the platform on which the software is hosted. In this case the host is the Google Cloud Platform (GCP and the designs make extensive reference to GCP tooling.

1.2 Interaction Patterns

Much of the material in this document is aligned to the interaction patterns that the SoS supports. These are defined elsewhere but for the purpose of containment. These are:

Synchronous Query: A data consumer requests data from the System-of-Systems, which services it in real-time from data providers, and issues the results over same connection on which the request was made. The method is explained further in the design paper for the FHIR Aggregator.

Asynchronous Query: A data consumer requests data from the System-of-Systems which acknowledges the request and drops the connection. The request is deferred to data providers which the System-Of-Systems periodically polls and collects results as they are ready. The data consumer polls the System-of-Systems and ultimately collects an aggregated result set. The method is explained further in the design paper for the FHIR Aggregator.

Subscriptions: A data consumer issues to the Systems-of-Systems a request for data which matches a search criterion. The request is deferred to data providers which send data, as they arise, to the System-of-Systems over a synchronous connection. The System-of-Systems passes on data that it receives to data consumers over a similar synchronous connection.

Subscriptions continue to operate until they are cancelled. The method is explained further in the design paper for the Subscriptions Infrastructure.

Transactional Messaging: A data provider uses the System-of-Systems to deliver a transaction to a data consumer. Messaging is reliable in that the data consumer is required to issue an acknowledgement and the data provider will resend messages for which no acknowledgement is received. The method is explained further in the design paper for the Reliable Messaging Infrastructure.

1.3 Relationship of this Document with Other Standards

This design paper is independent of other standards.

1.4 Intended Users of the This Document

Developers of the System of Systems, data providers and data consumers.

2 Performance and Scalability

It is projected that usage of SoS will grow over several years from fairly modest levels to become a very significant carrier of data operating at large volumes. If this is to be realised, then the SoS must scale with demand and be performant at the highest foreseeable transaction volumes whilst being cost effective at the early stages of use.

This section provides the developers of the SoS with foundations for building software which meets both short term expectations and long-term goals.

It establishes performance objectives which are constant and need to be met regardless of load and it forecasts what the ultimate load might look like. It establishes a model for scaling software in the cloud which enables cost to be closely correlated with demand and identifies the limitations of elastic scalability. It also offers contingency architectures for those components of the SoS which are difficult to scale automatically.

Of the 4 interaction patterns detailed in the introduction, it is the synchronous query pattern that will attract the most attention in terms of performance and this pattern is singled out for early data performance testing. The other three, whilst needing to support a certain level of throughput are asynchronous in nature and some delay in processing can be afforded. Performance testing for these patterns needs to ascertain that capacity can be uplifted as required (through manual intervention if necessary) to ensure that queue sizes, over time, can be managed and long-term transaction output rates exceed transaction input rates.

2.1 Sizing Assumptions and Estimates of Demand

This section is underpinned by assumptions about the use to which the YHCR is to be put, the number of people who are using it, the transactions with which they are engaging, and the data volumes being exchanged. The following are assumed as goals:

Number of clinicians accessing the YHCR during peak periods:	60,000
Average number of care records retrieved from the YHCR by a clinician per hour:	12
Average number of queries placed on the YHCR per care record accessed:	10
Average size of a resource:	5kb
Number of resources returned per query	40
Number of patients/clients registered with the YHCR:	6,000,000
Average number of encounters per patient per year	4
Length of working day	8 hours
Peak activity as a ratio to average activity	4:1
Percentage of population who have a subscription against their care record:	10%
Number of transactional messages generated per encounter:	2
Total number of FHIR resources per care record	4000
Number of patients in cohort for PHM purposes	100,000
Number of cohort requests by PHM for records for a per week	5

The above give rise to the following estimates of long-term demand.

Peak rates of query against the synchronous query pathway:	2,000/s

(Number of clinicians accessing the YHCR during peak periods * Average # of care records retrieved per hour * Average # or queries per care record /3600)

Peak bandwidth required for synchronous query: 3.2Gbit/sec

(Peak rates of query against the synchronous query pathway * Number of resources returned per query * Average size of a resource * 8)

Peak rate of transactional messages: 164/s

(Number of patients registered * Average Number of Encounters per patient / 260 / Length of working day /3600 * Peak activity ratio * Number of transactional messages per encounter)

Peak rate of subscription notification: 8/s

(Number of patients registered * Percentage of patients with a subscription * Average Number of Encounters per patient / 260 / Length of working day /3600 * Peak activity ratio)

Volume of data requested per week by PHM platforms: 10TB/week

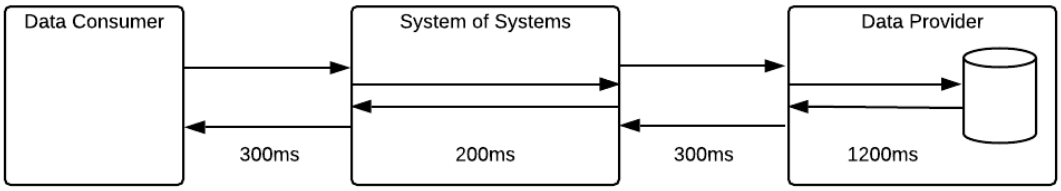
(Number of patients in a cohort * Number of FHIR resources in a care record * Average size of a resource * Cohort requests per week)

2.2 Performance Objectives

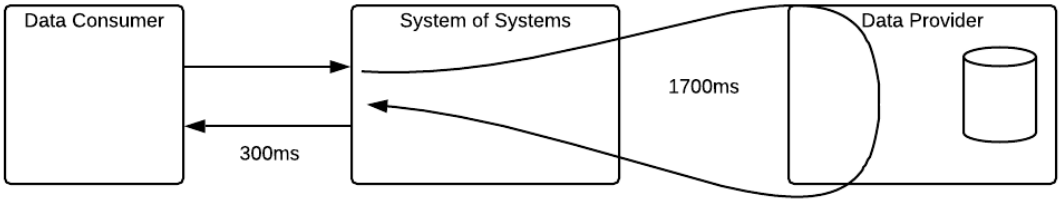
The performance objectives depend on the interaction pattern.

2.2.1 Synchronous Query

Synchronous query is used by user interfaces which are servicing users in real time. An established principle of user interface design is that an action should be responded to with 2 seconds. Ignoring the latency involved in any user interface in responding to a mouse click and re-rendering a screen the 2 second window is occupied by a request being made to the SoS, the request being processed by the SoS, the request being issued to one or more data providers, a response being received from data providers, the response being processed by the SoS, and the response being returned to the data consumer. A possible allocation of the 2 seconds is:

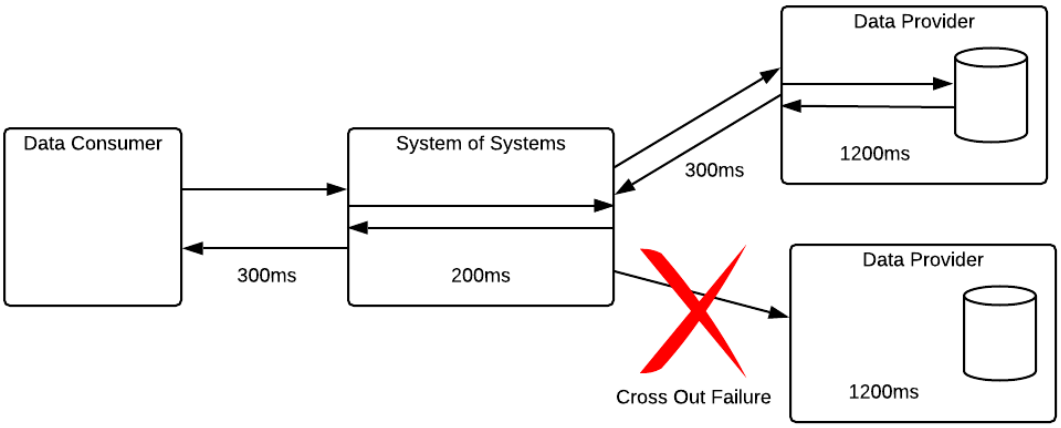


The 300ms allocated to network latency between the data consumer to the System of Systems is outside of the control of the SoS operators. Similarly, outside of their control is both the network latency between the SoS and data providers and the data provider's processing. However, from the perspective of a data consumer the SoS must return in 1700ms for the data consumer to be responsive in the eyes of its users.



If the SoS, is to meet the consumer's expectation, it must be in control of the full round-trip time from its boundary to data be received and processed from data providers. If necessary, it should

terminate connections with data providers that do not yield results within 1500ms. The SoS will insert an appropriate *OperationOutcome* into search results (see Design Paper 017 – Data Impairments for details) and consumers will be expected to inform users of incomplete data.



Performance objectives will be discussed with individual data providers and there may be a higher expectation of maximum response times for individual providers. This will be reflected in the monitoring of these providers and may influence the point in which YHCR operators intervene to remedy a potential degradation in performance. Section 5.5.1 details custom metrics which could be used for monitoring individual data provider response times and for alerting should performance degrade.

The data provider wait time is a configurable parameter and recorded in the Operations Guide.

A data consumer can override default behaviour by adding the HTTP header:

`YHCR-Provider-Wait-time`

with a numeric value representing the time in milliseconds for the SoS to wait for a provider response. The SoS will not wait indefinitely and maximum wait time will be imposed. The maximum wait time will be published in the Operations Guide.

A second performance objective for the synchronous query pathway is that the cost of operating the pathway should be linear with demand up to a maximum transaction threshold. Options for achieving this objective are detailed in section 2.3.

2.2.2 Asynchronous Query

Asynchronous query is used for bulk data requests. Currently the only identified use is Population Health Management (PHM) but other use cases will likely emerge. The asynchronous query pathway places queries with data providers and polls for results which when available are collected. Results are packaged in relatively small packets for collection. There is no particular package structure other than the packages are designed to keep the number of resources relatively constant in each package. Data consumers mirror this process with the SoS.

From the perspective of a consumer performance of the asynchronous query pathway begins with the placement of a query and ends when that result packet has been collected and so is measured as:

- time to place queries with data providers + time to collect results and serve them

The first part is much smaller than the second and performance objectives therefore relate to the collection of results rather than the placement of queries.

It is expected that data providers' asynchronous queries will be processed overnight or against an offline database that may be 24 hours out of date. A data consumer should expect results to be available from the SoS within about 48 hours. These expectations will be reviews as use cases for asynchronous query evolve.

Based on the sizing estimates above, at peak up to 2TB data will be exchanged over a 24-hour period. This will be packaged into units of approximately 1000 resources per page or 5MB. These will be retrieved at a rate of 5 transactions per second. Average network bandwidth required be approximately 200Mbit/s. These performance objectives are an order of magnitude lower than the requirements for the synchronous query pattern.

The performance objective for asynchronous query is for these rates to be sustainable over a 24-hour period. The performance of the asynchronous query pathway will be soak tested over several hours at anticipated peak volumes and cost correlated.

It should be noted that the PHM platform is also hosted on GCP and uses an internal network connection to avoid ingress/egress charges.

2.2.3 Subscriptions

According to the assumptions above, subscriptions will ultimately be placed for approximately 600,000 people. Subscriptions will be registered over several months and the comparative load on the SoS will be minimal. Subscription placement can be complex, and if a subscription is made against a patient cohort they can result in many 1000s of subscriptions being placed with data providers. Subscription placement will not be instantaneous or by any interpretation real-time. A data consumer might reasonably expect a subscription to reach existing data providers within a few hours.

Subscriptions will yield a single resource, potentially, each time a patient record is updated. Notifications will mainly be issued over a Rest Hook. Peak subscription notification rates will be in the order of 8 transaction per second at a bandwidth of 320kbit/s.

Use cases are developing around subscriptions which require notification of an event to be near real time. As an example: a subscription may be made to calls to 999 with a notification set to a care team to prompt local intervention. Near real time in this case is measured in a few minutes. Subscription notifications are asynchronous and a delay between an event and notification being received by a subscriber is inevitable. The SoS should set the expectation of the lag but it should nbe noted that the greatest dependency is with the data provider and use cases which rely on near real time delivery should ensure that the relevant providers are able to meet this expectation.

Performance objectives for subscriptions are:

- subscription notifications should be processed by the SoS and a delivery attempt is made within 30 seconds;
- the subscription placement message pathway should create subscriptions with providers within 3 hours.

2.2.4 Transactional Messaging

Transactional messaging is used to deliver a packet of data (or references to data) from a data provider to a pre-meditated data consumer in response to an event occurring. Use cases might include:

- delivery of a discharge summary of a letter from a care provider to a GP;
- placement of a referral from a district hospital to a tertiary care centre;
- a transfer of care from one care setting to another.

Many of these use cases are not particularly time critical (although most would reasonably expect delivery of a message within a few hours). However, some transactions such as the handover of care from an ambulance to an emergency department, happen in real-time and there is an expectation here that the transactional messaging pathway operates accordingly and delivers messages within no more than, say, 1 minute.

Transactional messaging is defined by design paper 006 – "Reliable Messaging Infrastructure". This paper treats all messages as equally important and does not offer a mechanism for varying the quality for different use cases. This may be a possible enhancement to the SoS but for now the transactional message pathway will need to treat all messages as time important.

Message size is likely to vary substantially. A reasonable estimate for the transfer of care from an ambulance is 15kb.

The sizing basis above implies a performance objective that the transactional messaging pathway should support sustained peak message rates of 164 messages/s with delivery time within 1 minute.

2.3 Scalability in the Cloud

Traditionally, scalability has been thought about using two models:

- vertical scalability where an applications capacity is tied to the size of machine which hosts it and increasing capacity requires an increase in CPU, RAM and disk space;
- horizontal scalability where application capacity can be increased by provisioning new instances of the application.

Horizontal scalability tends to be preferred because new, relatively cheap, host infrastructure can be provisioned quickly compared to incrementally upgrading host infrastructure with increasingly expensive components.

The cloud, to some extent, blurs the distinction between the two approaches. CPUs, Memory and disk space dedicated to resources can be scaled up and down in near real-time. The total capacity of a single resource can be increased to very high levels at a cost which is equivalent or less than the cost of provisioning multiple resources to achieve the same capacity.

In the cloud the key scalability strategies are:

- elastic whereby software determines the computational resources which it needs to accommodate point demand as demand is placed on the software;
- static whereby a configuration change is made in advance of a forecast change in demand and which stays in place until a revised forecast is made.

From a cost perspective elastic scalability is attractive, costs are entirely variable, and so a predetermined cost can be attributed to a transaction. Elastic scalability also enables consistent performance. As demand increases additional computational resources can be allocated to allow a predetermined performance objective to be maintained. Elastic scalability enables cost to be calculated as a function of performance objective: the cost per transaction increases in line with the stringency of performance objectives.

Not all GCP cloud resources are elastically scalable. A notable instance of a statically scalable resource is Cloud SQL.

A Cloud SQL resource must be allocated CPUs, RAM and disk space. These allocations cannot be changed programmatically although they can be reallocated by an operator without downtime.

An application whose performance is constrained by SQL input / output must be pre-scaled to accommodate anticipated demand and there is a fixed cost of operating the Cloud SQL components of the solution regardless of demand.

The System of Systems uses Cloud SQL as a backend for FHIR Stores which are used for:

- Master Patient Index data;
- Audit data;
- Subscription notifications;
- Ad-hoc central persistence requirements.

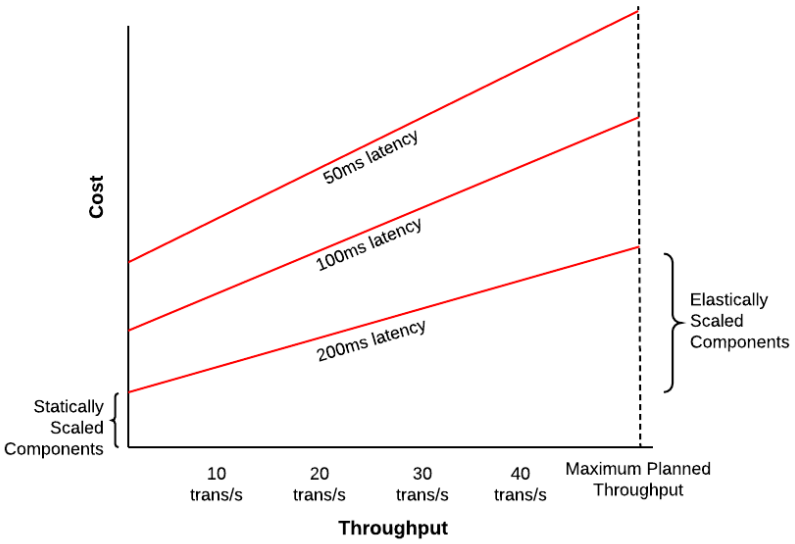
The last of these categories covers data which the YHCR chooses to persist centrally for functional reasons. There is no data of this type currently.

Outside of SQL, most of the SoS components are hosted in Kubernetes. Kubernetes facilitates elastic capabilities by:

- automatically spawning new Pods (which contain a collection of micro services), to allow a solution dynamically balance its workload internally with in the constraints of CPU and RAM allocated to the Kubernetes cluster;
- automatically allocate and deallocate CPU and RAM to the Kubernetes cluster.

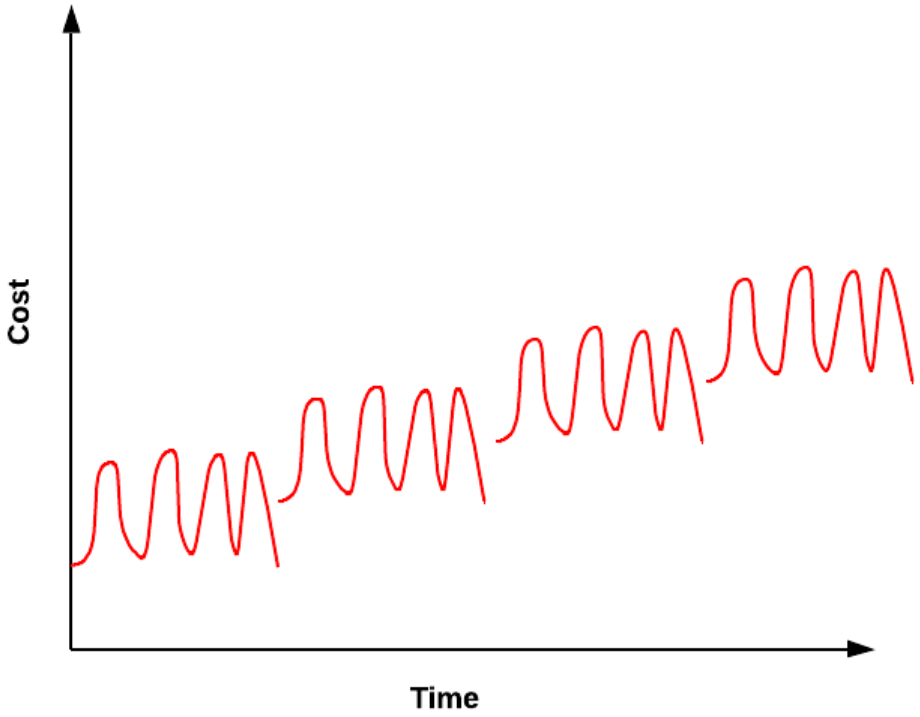
The above behaviour is achieved through configuration. The configuration effectively allows the resources which are allocated to processing a transaction to be pre-determined. Transaction trip times or latency therefore becomes a cost benefit decision.

The cost behaviour of the System of Systems within the bounds of a pre-sized maximum throughput will be as follows.



The latency figures above are for illustration purposes only and are not commitments.

Over time, as new participants are onboarded, maximum throughput will be raised through manual resource reallocation and the lon term cost behaviour of the SoS is expected to follow the following pattern.



These assumptions will be tested through performance testing.

2.4 Performance Testing

2.4.1 Synchronous Query Testing

This section describes performance tests which are being conducted as part of the development in order to inform configuration option for Kubernetes Dynamic scalability and to assist in the development of a costing model. The test harness, tests, data and tooling will be available for periodic performance testing after the development has completed.

Performance testing is targeted at the SoS platform and not public networks nor systems managed by participants to the YHCR.

The tests are executed by a test harness which comprises:

- load simulators;
- data source simulators.

Load simulators execute transactions on the YHCR at predefined rates. Several load simulators can operate in parallel to generate load at high transaction rates. Load simulators implement the protocols used by data consumers to interact with the YHCR. As the SoS is data content agnostic the simulators repeat standard queries rather than attempting to be representative of the mix of queries that might be issued by data consumers. However, the queries are designed to return approximately 40 messages per bundle which is representative of real data content sizes.

Data source simulators act as data providers and are cloud hosted instances of FHIR stores. The FHIR stores are loaded with resources which allow query results to be surfaced. The tests will be executed with multiple data source simulators to ensure that the aggregation pathway is fully exercised.

The SoS will be seeded with data that mirrors expected volume when running at full capacity:

- 6m patient records in the MPI with Linkages, on average to 3 data providers;
- consent records covering 10% of the population.

Tests will be executed at a range of transaction rates as follows:

- 10 transactions/s;
- 50 transactions/s;
- 100 transactions/s;
- 500 transactions/s;
- 1000 transactions/s;
- 2000 transactions/s.

At each rate the Kubernetes configuration and distribution of services between Kubernetes Pods will be manually flexed in order to locate the configuration that:

a) minimises System of System latency;
b) minimises RAM and CPU utilisation and therefore cost.

Note that the manual flexing is intended to discover the optimum topology of micro-services. In a live environment Pods implement the optimum topology and will be allocated resources dynamically so that scalability is fully elastic.

The latency and costs of operating in these configurations for a period of 10 minutes will be recorded. Note that testing will be conducted within the GCP cloud to avoid ingress and egress charges.

2.4.2 Soak Testing

The above tests demonstrate that the SoS can handle burst rates at the anticipated demand. It is equally important to demonstrate that transaction throughput can be sustained for extended periods. The above performance tests will be executed for a 24-hour period at a transaction rate of 500 transactions/s.

2.4.3 Testing of Asynchronous Patterns

Similar tooling will be developed over time for asynchronous patterns. At this time volumes on these patterns are anticipated to be low and there is less of a requirement for dynamic scalability – queue sizes can be monitored, and operators can intervene to resize resources as necessary.

As demand for these patterns grows, appropriate performance testing will be conducted.

2.4.4 Reuse of Performance Test Tooling

Responsibility for performance of software provisioned by data providers lies with the organisation providing the data. However, the YHCR can assist by making available the tooling which has been developed to test SoS infrastructure for application in testing the performance of data providers.

2.5 Options for Improving the Elasticity of Input / Output Bound Components

The SoS FHIR Stores have been architected to be performant up to high transaction volumes and performance tests will ensure that they function cost effectively at peak expected volumes.

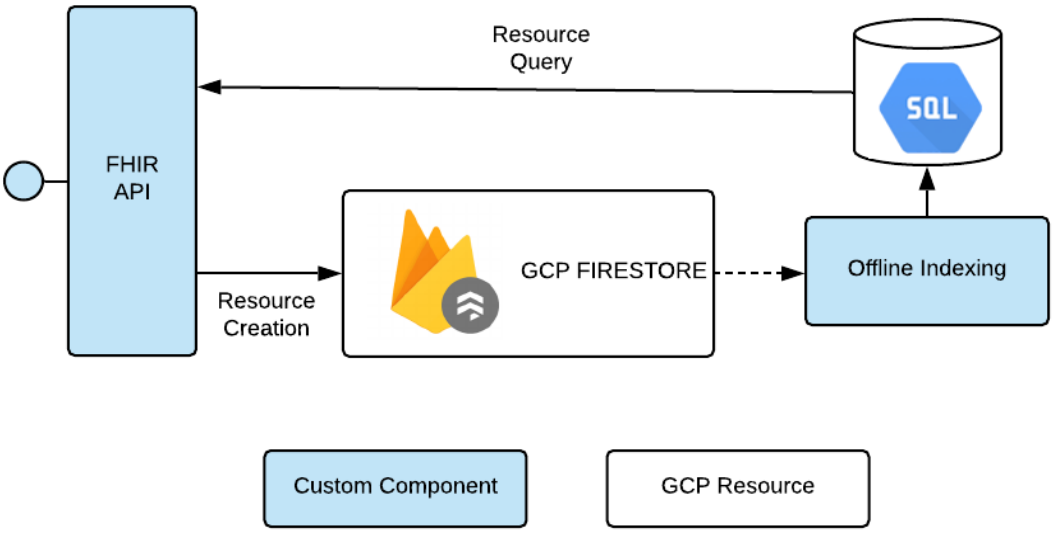
However, once a full cost profile is available after period of live operation (or should performance trials dictate earlier intervention) then consideration may be given to re-architecting certain of the cloud FHIR stores to introduce options for elastic scalability with a trade-off on architectural simplicity.

This section describes in outline two possible architectures which will increase the elasticity of FHIR Stores. The suitability of each depends on the nature of data being hosted.

Ideally scalability will be achieve natively using cloud resources. Google continue to innovate and wherever possible and cost effective then use should be made of new product sets in preference to developing proprietary solutions. These options will only be pursued if there is not a feasible alternative available from GCP.

2.5.1 Use of an Object Store with Offline Indexing

This option is well suited to data which is created in volume but queried infrequently. The data needs to be searchable, but searches will typically take place some time after the data is created. There is no business impact if a search were to exclude recently created data. An example of this type of data is audit records.

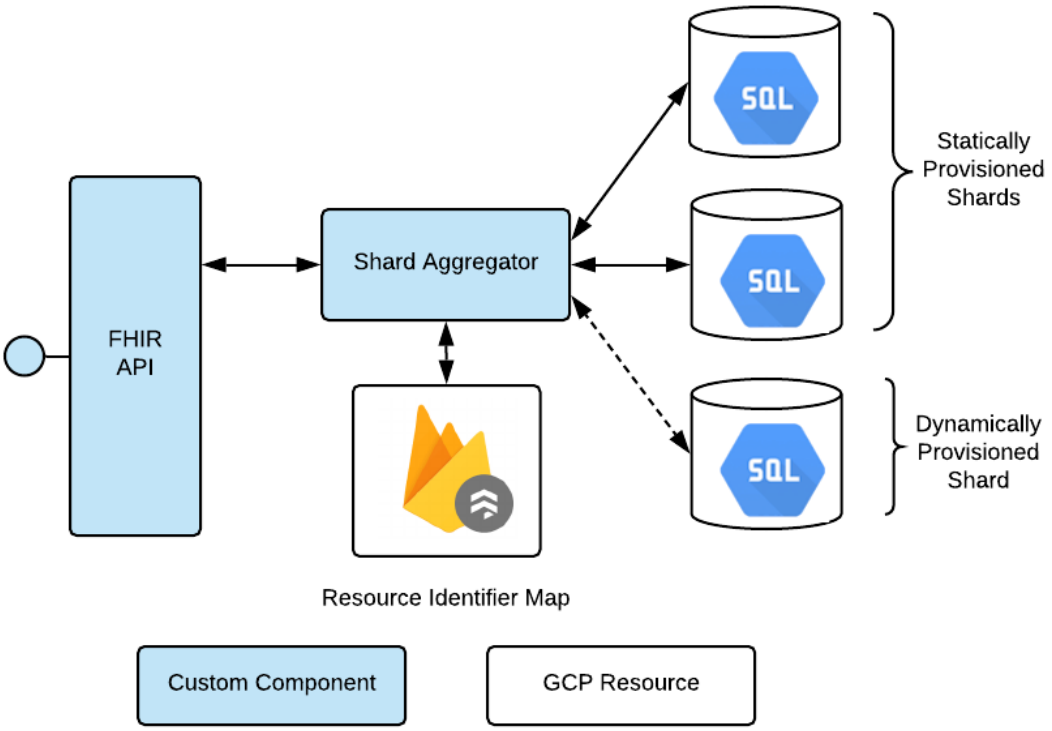


Resources are created in the GCP Firestore – an object database that is elastically scalable but with basic index capabilities. The REST interface which was involved to create data returns immediately. Asynchronously, a process creates indices over search terms and persists them in Cloud SQL. Queries run over the Cloud SQL database.

The architecture moves index creation to an offline process and therefore allows CPU and RAM utilisation to be flattened.

2.5.2 Dynamic Sharding

This option is well suited to situations where a lot of data is created, data is queried in real time and where there are large variations in demand with the system being quiescent for long periods.



Resources are spread across shards. A resource identifier map locates the shard to which resource has been allocated for easy retrieval. The resource identifier map uses an elastically scalable object store.

A Shard Aggregator mediates between the FHIR API and shards, aggregating data reurned by many shards to present it as a single bundle.

On resource creation resources are allocated by the Shard Aggregator to a particular shard and persisted in a Cloud SQL database. As demand increases, a new shard can be dynamically provisioned, and resources will begin to be created within it. As demand falls the shard is decommissioned by moving data to statically provisioned shards. Dynamically provisioned shards are reclassified as static at certain data volume thresholds.

By dynamically provisioning Cloud SQL resources the architecture ensures that costs are only incurred for excess capacity during periods of high demand. However, it should be noted that executing this architecture is complex.

3 High Availability

When building traditional on-premise hosted systems high availability is an architectural design decision. A good architecture enables high availability by allowing redundancy to be built in at every level of the solution stack:

- domain name servers are geographically distributed;
- the service is available on multiple IP addresses are distributed across backbone providers;
- load balancers distribute requests to multiple application servers;
- application servers interact with clustered databases;
- a synchronously replicated SAN distributes data across multiple data centres;
- etc.

No one component is a single point of failure and high availability is a balance of probabilities and cost. A greater number of redundant components lowers the probability of simultaneous failure but raises the operational cost.

Hosting in the cloud doesn't change these basic principles although it does offset responsibility for provisioning redundant components to the cloud provider. It should be acknowledged that the cloud provider is now a potential single point of failure. A solution which is truly architected for high availability in the cloud should be hosted by multiple cloud providers. Ideally the solution should run in parallel between the providers so that failure of any one is nearly undetectable. At the very least the service should be able to be hot swapped between providers. However, very few solutions take these measures. The ethos of hosting in the cloud is to offset responsibility for managing redundancy to the cloud provider and the YHCR, at this point in time and in common with most cloud solutions, is dependent on a single provider. Its planned availability is to an extent defined by the service levels offered by its host.

This does not imply that there are no design decisions to be made but the nature of design decisions is different and if a solution is hosted in a single region they are mainly related to the configuration of cloud resources rather than the solution topology.

3.1 GCP Regions and Zones

Google operates data centres in various regions across the world. A region is a location in which one can choose to host their data and compute resources. The SoS is hosted in europe-west2 which is located in London, UK. It is the only GCP region in the UK.

A region comprises a number of zones. A zone is a physical datacentre. Europe-west2 has 3 zones meaning that any resource in the SoS can be physically located in one of three data centres.

By default, compute and storage resources are located in a zone. Whilst the Google data centre will have high levels of redundancy built-in, this makes compute and storage resources susceptible to a zonal failure.

3.2 Objectives for High Availability in the Cloud

The following sections detail how high availability will be achieved for mission critical SoS environments (live, system test and staging). The objectives of a high availability configuration are:

- to ensure that the environment is resilient to failure of a single GCP zone;
- to provide for 99.8% service availability in the environment;
- to allow the environment to be maintained with zero downtime.

Note that resilience to a regional failure is not an objective. This would entail hosting the system outside of the UK and would be significantly more costly. Restoring a service from the unlikely event of a regional failure will be the role of Disaster Recovery process which is considered in section 4. Section 5 details release management procedures will address the risk of non-availability because of human error.

3.3 GCP Resource High Availability Configuration

The SoS uses the following GCP Resources:

- Compute Engine;
- Kubernetes Engine;
- Cloud SQL;
- Cloud Storage;
- Fire Store;
- Load Balancer;
- DNS;
- Cloud Tasks.

The following sections details the high availability configurations possible for each GCP component used by the SoS. The configurations which are proposed for a highly available environment are listed in section 3.5

3.3.1 Compute Engine

Managed instance groups allow a set of virtual machines (VMs) to be managed as a single instance.

VMs in the group can, optionally, be located regionally and if so then they are physically distributed across zones. Virtual machines which are in trouble are automatically restarted. Traffic is load balanced between machines.

Independently of instance groups, VMs can use regional storage which replicates data across the two zones in a region.

The SoS uses Compute Engines to host:

1) the Onboarding Suite.
2) a bastion service to allow administrator access to an environment independently of the GCP console.
3) the Release Console (section 6).

3.3.2 Kubernetes Engine

Compute engines can be clustered across the zones in a region. GCP offers multi-zonal clusters and regional clusters. For both, nodes of the cluster are distributed across zones and will continue to operate in the event of a zonal failure. Multi-zonal clusters do not cluster the Kubernetes control plane (master node) which continues to be hosted in a single zone. Loss of the control plane means loss of API access to the cluster, loss of scheduling and loss of auto scalability.

A regional cluster distributes the control plane across zones and has the following advantages:

- If one or more (but not all) zones in a region experience an outage, the cluster's control plane remains accessible as long as one replica of the control plane remains available;
- During cluster maintenance such as a cluster upgrade, only one replica of the control plane is unavailable at a time, and the cluster is still operational.

3.3.3 Cloud SQL

Cloud SQL can be configured for high availability in which case it is located in a primary and secondary zone within a given region. Under this configuration it is made up of a primary instance (master) and a standby instance. Through synchronous replication to each zone's persistent disk, all writes made to the primary instance are also made to the standby instance. In the event of an instance or zone failure data continues to be available to client applications.

3.3.4 Cloud Storage

GCP offers different classes of storage which are appropriate for different access profiles. Standard storage is used for immediate access requirements for which high availability is a consideration.

Standard storage can be zonal, regional, dual-region and multi-region.

3.3.5 Fire Store

The Fire Store is only available in high availability configuration having a regional location by default. Multi-regional location is an option for extreme availability.

3.3.6 Load Balancer

Load balancers are global resources. GCP using any-casting to advertise a static IP address from different points in its network. Failure in any one of its one part of its network will not affect service availability.

3.3.7 DNS

DNS is inherently highly available. On GCP all zones are hoisted on 4 separate domain name servers.

3.3.8 Cloud Tasks

Cloud is an asynchronous queuing mechanism used in transactional messaging. Cloud tasks is inherently highly available being located in a region and persisting data across multiple zones.

3.3.9 Memorystore

For use cases requiring high availability, Memorystores are replicated across two zones.

3.3.10 Key Management System

Cloud KMS is available in several global locations and across multi-regions, for low latency and high availability.

3.4 Core and Subsidiary SoS Services

Requirements for high availability differ for different SoS Services. The interoperability services for which performance objectives are defined in section 2.2 are core services for which any downtime has significant business impact. The SoS provides other services for which some downtime is tolerable. These include:

- The Onboarding Suite (design papers 020 and 021);

- Data Quality Analysis Tooling (section 5.4);
- Forensic Analysis Tooling (section 5.5).

These are service that this document describes as subsidiary. Significant availability of these services is desirable during working hours but availability targets are significantly lower than core services. Investment in high availability configuration of GCP components is not warranted for these services.

3.5 High Availability Environments

The following environments will be configured for high availability:

- Live;
- System Test;
- Staging.

Live will be allocated resources to enable it to meet the performance objectives set out in section 2. System Test and Staging, while configured for high availability, will be allocated lower levels of resources.

Development and Sandpit will be tethered to a single zone and will not be highly available.

	High Availability	**Standard Configuration**
Compute Engine - Onboarding Suite	Single zone	Single zone
Compute Engine - Bastion Server	Regional Instance Group Standard Storage	Single zone
Compute Engine - Release Console	Single zone	Single zone
Kubernetes Cluster	Regional Cluster	Zonal Cluster
Cloud SQL	Synchronous replication over two zones	Single zone
Cloud Storage	Regional	Single Zone
Fire Store	Regional	Regional
Load Balancer	N/a	N/a
DNS	N/a	N/a
Cloud Tasks	N/a	N/a
Memorystore	Replicated over two zones	Single zone
Key Management System	N/a	N/a

3.6 Availability Targets

The benchmark for availability targets is SLAs offered by Google. It should be noted that SLAs are provided for individual resources rather than a zone or region. Unavailability can therefore compound from multiple resources being unavailable at different times which could lead to the SoS experiencing availability below the lowest Google SLA without GCP itself being in breach. It also should be noted the SLAs provided by Google are expectations which if not me result in a service credit. Credits at the published SLA level are relatively minor (typically 10%) and only become significant (50%) at much lower levels of performance (typically <95% availability).

SLAs operate on a monthly basis. The following are monthly downtime allowances at various levels discussed in this document.

Availability Target	Allowed Monthly Downtime
95%	36 hours
99.5%	3.6 hours
99.9%	43 minutes
99.95%	22 minutes
99.99%	5 minutes

SLAs for key GCP resources are published as:

	High Availability Configuration	Standard Configuration
Compute engines	>=99.99%	>=99.5%
Cloud SQL	>=99.95%	none
Stackdriver	>=99.95%	
Cloud NAT	>=99.9%	
Cloud DNS	100%	

Because of the potential to compound faults, an expectation for availability of the System of Systems will be slightly below the lowest of these: >=99.8% or downtime of about 80 minutes per month.

Software faults and unplanned maintenance have the potential to decrease service availability further. However, these factors are under the control of the operators of the YHCR and should be controlled through contractual SLAs.

3.7 Planned Maintenance

The technology stack which is behind the SoS should allow for zero downtime maintenance:

- microservices running under Kubernetes can be upgraded in flight without detectable service interruption;
- all databases are schema-less and data structures can be changes without database management activities;
- backups can be taken whilst services are operating.

There also techniques available to the operators of the SoS which have been facilitated by the component-based architecture which allows a component to be isolated, replicated, upgraded and traffic re-routed without service being impacted.

The YHCR may plan maintenance windows in the early stages of operation to provide some room for error but should strive for zero downtime maintenance as experience grows.

If the SoS is unavailable due to planned maintenance, then the HTTPS RESTful services will be routed to a server which responds with an HTTP 503 Service Unavailable response.

4 Backup and Recovery

4.1 Introduction

The objective of this section is to define processes and procedures which will allow the System of Systems to be recovered in the event that high availability measures detailed in section 3 fail to maintain continuity of service. This might be because of a regional GCP failure or, possibly more likely, because of human error or systematic data corruption.

The recovery mechanisms are designed to allow the SoS to be reinstated within a few hours, ideally with no loss of data, but with priority that a service can be restored rapidly, even if sub-optimally.

CloudSQL backup does not currently provide for point in time recovery which means that some data loss might be inevitable, or recovery processes may take time to reinstate data.

There are also limitations to CloudSQL backups which must either be accepted, or off-cloud solutions developed. Considerations include:

- there is only one GCP region in the UK. If data is to be held only in the UK, then a risk must be accepted that data may be lost because of a regional GCP failure.
- 7 historical backups are held. Backups are not intended as data archives.
- backup files ae not accessible and cannot be copied off-cloud.
- backup periodicity cannot be less than 4 hours. Without measures to replay transactions then up to 4 hours of data might be lost.

Key features of this design include:

- native GCP backup processes will be used with backups being held in the UK;
- an export of data (rather than backup) will be periodically be taken off-cloud for classes of data which cannot be recreated;
- the regional FHIR Stores will serialize transactions to Cloud Storage which enable transactions to be replayed and backups restored without data loss.

4.2 Data Content of the SoS

Design paper 018 – "Regional FHIR Store" among other topics, documents the regional data model. In summary the following data items are maintained centrally by the SoS. RAG coding is used to identify the seriousness of data loss.

Data Item	DB	Source	Fluidity	Impact of Loss
Regional patient resources and Linkages	FHIR	Patients are registered by data providers when an encounter occurs. Demographics are supplemented from PDS.	Rapid	PIX is the determinant of potential sources of data about a person. Loss of linkages results in queries not being issued to certain data providers and gaps in the medical record for data providers with clinical safety implications.
Regional organisations and practitioners	FHIR	ODS	Slow	The data is used to unify records from different data providers. Loss will lead to duplicate data being presented to data consumers but with low clinical impact.

Consent Policies	FHIR	Manually created by SoS Operators	Slow	Consent cannot be enforced. Data may be released in contravention of patients' wishes.
Consent Resources	FHIR	Created by patients and their representatives operating through data consumers	Medium	Consent cannot be uniformly enforced. Data may be released in contravention of some patients' wishes.
Audit resources	FHIR	Created by SoS components	Rapid	The YHCR will be in contravention of governance responsibilities. Data access requests may not be serviced.
Participant registry	FHIR	Created and managed by the Onboarding Suite	Slow	Participants may not be able to interact with the YHCR. Data consumers are aware of non-participation by data providers.
Patient cohorts	FHIR	Created and managed by data consumers (currently only PHM)	Slow	Subscriptions and queries will be disrupted. Currently this will be localised to PHM.
Subscription Cache	SQL	Created and managed by data consumers.	Rapid	Subscriptions will not be propagated out to data providers making new contact with subscription subjects resulting in gaps in data returned to data consumers with consequential clinical safety implications.
Subscriptions (regional FHIR Stores)	FHIR	Created and managed by the SoS	Medium	Subscription notifications will not be made for changes to regional data. Currently regional data is non-clinical in nature and the risk for clinical safety is negligible.
Asynchronous Query Control Data	SQL	Created and managed by the SoS	Medium	Asynchronous query results will not be collected from data providers and will not be available to data consumers. Data consumers will receive errors when attempting to collect data from lost queries and will have the opportunity to re-issue the query. Currently this will only impact PHM
In flight transactional messages	Queue	Creates by a data provider	Medium	The reliable messaging pattern will cause messages to be resent and re-acknowledged. No data will be lost.
Active JTIs	Mem	Created by IAM	Rapid	Data consumers would be transiently rejected. Consumers would issue a new assertion and regain access.

Database storage technologies are:

- FHIR: PostgreSQL as the persistence layer for a regional FHIR Store.
- SQL: PostgreSQL
- Queue: Cloud Tasks
- Mem: Cloud Memorystore

The SoS also manages configuration data and security certificates. All configuration data is under source code control (see section 6). Certificates are managed by the GCP Certificate Keystore. Both

are secure sources of backup for these data items and the potential for data loss (assuming rigorous operational procedures are followed) is negligible.

4.3 Approach to Backup

All PostgresSQL databases are backed up using native GCP backup capabilities. Backups are stored by GCP in the UK region.

The periodicity for running backups will be set separately for each Cloud SQL resource. Backups will initially be made daily with the frequency increased as participation in the YHCR grows and emphasis is placed on shortening time to recovery.

The FHIR Stores managing patient resources, linkages, consent policies and consent resource will be backed up most frequently given the critical nature of the data and high impact of data loss.

Cloud Tasks and Cloud Memorystore will not be backed up as there is negligible impact of data loss.

4.4 Data Recovery Procedures and Contingency Plans

Standard PostgreSQL backup procedures lead to a recovery point as the time of the last backup. Data which was created or modified after the backup was taken is lost. PostgreSQL has a method for snapshotting the file system and replaying write-ahead logs which allows for point in time recovery. But this approach requires downtime to take the backup and the method is not natively supported by GCP.

The FHIR Store (design paper 018) mitigates the risk by serialising FHIR resources created or modified to GCP Storage. This enables transactions to be replayed against a restored database to recover the current position. Recovery procedures will be documented in detail in the YHCR Operations Manual but in outline recovery of a FHIR Store will entail:

- Suspending the SoS Kubernetes cluster;
- Applying the relevant PostgreSQL backup;
- Replaying persisted FHIR resource captured on SQL Storage from some time before the backup was taken;
- Reinstating the Kubernetes cluster.

The approach to high availability and rigorous backup processes minimise risk to the participants of the YHCR. However, contingency measures should still be put in place to maintain service continuity in the event of data loss. These measures will depend on the nature of data loss but in outline these will be:

Data Item	Contingency Plan
Regional patient resources and Linkages	Switch the FHIR Aggregator to disregard PIX and issue queries to all potential data providers. The data availability service is switched to respond positively for all requests. Data providers replay PIX registrations from the point of data loss.
Regional organisations and practitioners	The last ODS download files are reprocessed.
Consent Policies	Consent policies will be set up infrequently. The YHCR operators service desk software will provide details of past policies created and policies can be

	manually re-established.
Consent Resources	Access for consent sensitive reasons for access are disabled. Data consumers are approached to determine whether transactions can be replayed. Potentially patients may need to reinstate consent wishes.
Audit resources	No mitigation is possible. See note on off-cloud export below.
Participant registry	The onboarding process is re-executed (rapidly) for affected participants.
Patient cohorts	Cohorts are recreated by data consumers.
Subscription Cache	Subscriptions created at data providers are queried and reconciled with the SoS subscription cache. Cache entries are recreated based on subscriptions placed with data providers. Details of subscriptions held are reconciled to data consumers records.
Subscriptions (regional FHIR Stores)	The SoS subscription cache is used to recreate subscriptions at regional FHIR stores.
Asynchronous Query Control Data	No mitigation is necessary.
In flight transactional messages	No mitigation is necessary.
Active JTIs	No mitigation is necessary.

The above mitigations should be documented as detailed operational procedures by the YHCR operators.

4.4.1 Exporting Data Off-cloud

Whilst GCP Cloud doesn't permit access to backup files, there is an established technique for exporting data from a backup restored to another non-operational database to allow the export to be copied off-cloud with no no-downtime or performance impact on the original service. This technique will be used to periodically save a copy of audit resources, data which cannot be recreated and where total loss is unacceptable.

4.5 Validation of Backups and Recovery Rehearsal

The integrity of backups should be periodically tested by the YHCR operators. The periodicity of backup validation will be published in the Operations Guide.

Recovery from backup and contingency plans for recovering from data loss should be periodically rehearsed and recovery time measured. Decreasing recovery time is an opportunity for continued service improvement.

4.6 Disaster Recovery

In the event of a significant GCP outage the SoS may need to be recreated.

Building of all environments is fully scripted and can be recreated with minimal operator involvement.

Data recovery will be a mix of recovery from backup and actioning recovery contingency plans.

Operational procedures will be put in place for informing YHCR participants of a major outage and business continuity plans will be executed locally.

5 Monitoring and Alerting

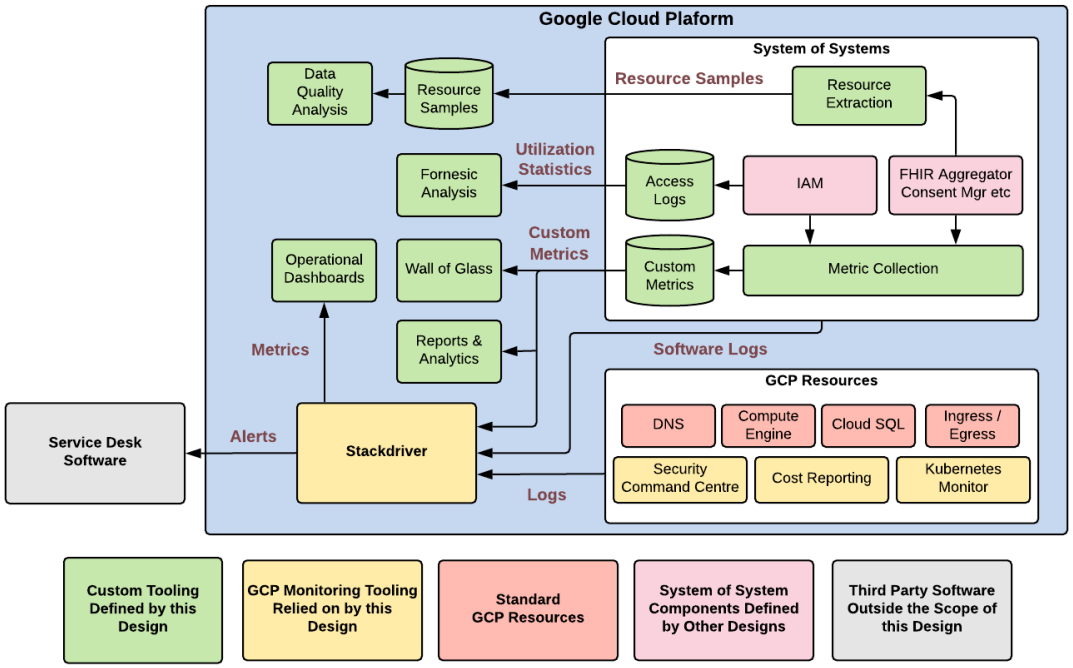
5.1 Monitoring Defined

Monitoring is the act of measuring characteristics of a system and its environment in order to optimise important aspects of its behaviour. This paper considers different ways that measurements can be used to influence change:

- Alerts being notifications which are sent to operators to prompt for intervention in real-time.
- Dashboards being visualisations of measurements which help operators predict potential pending issues and intervene appropriately.
- Metrics being statistics which help operators understand current behaviours and determine corrective actions.
- Analytics being retrospective interpretation of statistics which support future optimisations.

5.2 A Monitoring Architecture

The overarching architecture for monitoring which is presented by this paper is as follows:



Custom metrics and logs are extracted from SoS components. These are interpreted by monitoring tooling alongside logs and metrics which are produced by GCP resources. Where possible out-the box GCP products are used and, in particular, the design places reliance on Stackdriver, Google's operations suite for log analysis, telemetry and alerting.

Standard products are supplemented by custom analysis and visualisation software. These focus on providing insight in the particular characteristics of the SoS's role as an interoperability platform.

Of note are custom two tool which aim to highlight issues with the quality of data passing through the SoS and highlight instance of misuse of the platform.

5.3 Scope and Requirements for Monitoring

One of the primary reasons for monitoring a system is to identify issues which might impair the quality or security of the service. Ideally the design of the monitoring solution should allow these issues be identified prior to them impacting end users. But there are other reasons for monitoring which should also be considered including:

- learning about service usage in order to inform future improvements;
- facilitating support of the service by providing information which reveals the potential source of unexpected behaviour;
- supporting contract enforcement through the measurement and reporting of key performance indicators.

There are also several aspects of a service which might be places under the scrutiny of a monitoring solution. These include:

- performance;
- availability;
- data quality;
- security (attacks and breaches);
- profile of usage;
- integrity of usage;
- infrastructure utilisation;
- cost.

The following table uses these dimensions to construct requirements for a monitoring solution. Note that some of the requirements are not met by this design and others, whilst enabled by the design, require procedures to be put in place or reports written to be fully realised.

The cells of the table are colour coded accordingly.

Requirement is fully supported by this design	Requirement is enabled by this design but requires supplementary processes or tools.	The requirement is not me by this design

	Quality of Service	Learnings	Facilitation of Support	Contract Enforcement
Performance	1. Identify	2. Capture	3. Segregate	4. Correlate

	sources of degrading performance and initiate response	history of performance correlated with usage.	performance issues at participants from those with the SoS.	performance measurements with KPIs and track periods of breach.
Availability	5. Rapidly identify service non-availability and initiate response	6. Report impact of non-availability on service users	7. Pinpoint cause of non-availability.	8. Track periods of non-availability.
Data Quality	9. Locate sources of poor data quality and initiate correction.	10. Profile data coding being used and support initiatives to standardize data.	11. Highlight in real-time impact of poor data quality on service operation.	12. Associate instances of poor data quality with achievability of KPIS.
Security	13. Rapidly report breach/DOS attempts, automatically contain and initiate a response	14. Capture impact of security incidents on service users	15. Ensure visibility of in-flight security incidents and response	16. Classify detected security incidents according to controllability.
Profile of Usage	17. Measure service usage in terms that can be related to service quality expectations.	18. Create a history of service usage by participating organisation.	19. Categorise incidents and indicators of degraded service by usage type.	20. Associate service performance measurements with type of use.
Integrity of Usage	21. Pinpoint localised abuse of service and initiate response.	22. Develop history of patterns of usage that enable definition of abuse to be refined.	23. Highlight in real-time detected instances of possible service abuse.	24. Classify detected incidents of service abuse according to controllability.
Infrastructure	25. Measure infrastructure resource utilisation metrics and initiate response to overloaded resources.	26. Capture history of infrastructure resource utilisation and correlate with quality of service measurements.	27. Highlight in real-time resource overloading and correlate to a service feature.	28. For KPIs phrased in terms of resource utilisation track period of actual utilisation statistics in breach
Cost	29.Report actions taken to dynamically scale	30. Correlate cost of operating service with	31. Facilitate investigations in cost anomalies	32. For KPIs phrased in terms of cost control

	infrastructure that have cost implications.	usage.	through granular reporting of cost	track periods of non-adherence.

Certain of these requirements are supported by capabilities which are native to the Google Cloud Platform and for which there is no design required. In particular:

- the Security Control Centre monitors breach and denial of service attacks and whilst a design for the SoS can add to these capabilities.
- cost analysis tooling can allow costs to be extracted which can be used in conjunction with metrics described her to support cost related requirements.

Requirements will be met through a mix of out-of-the box GCP tooling, custom monitoring and software, data quality analysis tooling, and forensic analysis tooling. A brief response to each of the above requirements is followed by more detailed description of each of these tool sets.

Note that service abuse considered in these requirements is defined as intentional access by users at data consumers to records which they have no legitimate reason to view.

5.3.1 Performance Monitoring

1. Identify sources of degrading performance and initiate response

Custom metrics which captured by SoS components measure aspects of performance and if thresholds are passed then alerts are generated soliciting action.

2. Capture history of performance correlated with usage

Custom usage and performance metrics which are captured by SoS components are persisted and available for retrospective analysis.

3. Segregate performance issues at participants from those with the SoS

Custom metrics separately capture internal transaction processing times and the round-trip times to invoke services at participant organisations and for national services.

4. Correlate performance measurements with KPIs and track periods of breach

Automatic analysis of the impact of a performance measurement on breach of a KPI is not supported by this design. However custom performance metrics which are captured by SoS components are persisted and available to use for analysis of contractual compliance.

5.3.2 Availability Monitoring

5. Rapidly identify service non-availability and initiate response

Custom metrics captured by SoS components measure participants' connection to SoS services. Metrics also analyse terminated connections and error responses. Thresholds can be set which cause operators to be alerted about global or local availability problems.

6. Report impact of non-availability on service users

Service availability is established by custom metrics and are persisted. Profiles of service usage are also captured by custom metrics and these can be used to analyse impact of a service outage.

7. Pinpoint cause of non-availability

Software components raise 'fatal' exceptions when an error condition indicates the loss of a service. Standard GCP tooling causes alerts to be sent to operators and to report the condition on a SoS wall of glass. Alerts and error reports identify the troubled component.

8. Track periods of non-availability

Service availability is established by custom metrics and are persisted. Reports can establish periods of non-availability.

5.3.3 Data Quality Monitoring

9. Locate sources of poor data quality and initiate correction

Data quality analysis tooling sample checks data flowing through the SoS for operator defined definitions of data quality impairment. The analysis can be used by operator to identify the sources of poor data and procedural mechanisms can be invoked to initiate correctional action.

10. Profile data coding being used and support initiatives to standardize data

Data quality analysis tooling can be used to extract attributes of data passing through the SoS and develop a history of the coding employed by individual participants. Statistical analysis tools (outside the scope of this paper) could be used to identify significant deviations in coding practises.

11. Highlight in real-time the impact of poor data quality on service operation

Data quality will be measured but retrospectively. Generally, this data will not be available to operators in real time and so this requirement will not be met in the strict sense that it is meant. However, software components that detect data quality issues that impact their operation will throw exceptions and standard GCP tooling will alert operators.

12. Associate instances of poor data quality with achievability of KPIS

Data quality measurements will require manual interpretation to assess the impact on the achievability of KPIs.

5.3.4 Security Monitoring

13. Rapidly report breach/DOS attempts, automatically contain and initiate a response

GCPs Security Command Centre detects security incidents, takes corrective measures and alerts operators. Custom metrics recorded by the SoS Identity and Access Management (IAM) component detect fallacious OAuth claims and possible replay attacks. Custom alerting software informs operators.

14. Capture impact of security incidents on service users

This requirement is not directly supported by this design although measurements recorded by components covered in this design could support the analysis.

15. Ensure visibility of in-flight security incidents and response

Operators have access to the GCP Security Command Centre dashboard and visualisations of security related metrics captured by custom SoS components.

16. Classify detected security incidents according to controllability

The requirement is not supported by this design.

5.3.5 Profile of Usage Monitoring

17. Measure service usage in terms that can be related to service quality expectations

Custom metrics captured by SoS identify participants' types of user and reasons for accessing the YHCR. Service expectations are formally defined in terms of response times for which measures are defined in section 2 of this paper to ensure adherence.

18. Create a history of service usage by participating organisation

All access by users at a participating organisation, and their reason for access, are captured by custom metrics, are persisted, and are available for retrospective analysis.

19. Categorise incidents and indicators of degraded service by usage type

Metrics captured the System of Systems can be associated with interaction patterns which is a good indicator of service use. Theoretically, information about the context of access could be extracted from access claims made by participants and could be associated with service metrics and exceptions recorded but this is not considered by this design.

20. Associate service performance measurements with type of use

Custom metrics describing performance are captured for each interaction pattern which is an indicator of type of use. Greater resolution of usage could be captured but is not considered by this design.

5.3.6 Integrity of Usage Monitoring

21. Pinpoint localised abuse of service and initiate response

Forensic analysis tooling will provide evidence of potential service abuse but will run retrospectively and will not be helpful in initiating immediate response.

Custom metrics developed for IAM will provide some real-time statistics for unusual authentication requests, but these are targeted at identifying security incidents rather than abuse by correctly authenticated users.

22. Develop history of patterns of usage that enables definition of abuse to be refined

The forensic analysis tooling described in this document develops a history of data accessed by end users of systems using the YHCR. The design of the tooling is targeted to premeditated situations which could be indicators of abuse. These are focussed on location of access, time of access, frequency of access and number of patient records accessed. Additional data could be captured such as type of data accessed which could lead to more sophisticated analysis, but this is outside of the scope of this paper.

23. Highlight in real-time detected instances of possible service abuse

The forensic analysis proposed by this design operates retrospectively and will not be available for real-time use.

24. Classify detected incidents of service abuse according to controllability

The requirement is not supported by this design.

5.3.7 Infrastructure Monitoring

25. Measure infrastructure resource utilisation metrics and initiate response to overloaded resources

The GCP Stackdriver tool captures metrics produced by GCP resources such as compute engines and storage. Key resource utilisation metrics are defined in this document and thresholds will be set in the YHCR Operations Guide that result in operators to be alerted by Stackdriver of overload conditions.

26. Capture history of infrastructure resource utilisation and correlate with quality of service measurements

A set of statistics will be extracted from GCP Stackdriver and persisted for later analysis.

27. Highlight in real-time, resource overloading and correlate to a service feature

Operators have access to Kubernetes and GLC Cloud SQL Dashboards that help visualise key resource utilisation statistics. Resources are tagged to identify the role that they perform in the SoS solution.

28. For KPIs phrased in terms of resource utilisation track period of actual utilisation statistics in breach

KPIs have not yet been defined but should a KPI relate to resource utilisation then statistics could be extracted from GCP Stackdriver and persisted for latter analysis.

5.3.8 Cost Monitoring

29. Report actions taken to dynamically scale infrastructure that have cost implications

Section 2 identifies mechanisms for dynamically scaling the solution. All actions resulting in automatic allocation of resources which have cost implications are logged and can be extracted from Stackdriver.

30. Correlate cost of operating service with usage

Detailed cost reports are available from the GCP cost analysis tool. Custom metrics which analyse usage can be associated with cost.

31. Facilitate investigations in cost anomalies through granular reporting of cost

GCP records costs by day by resource. Standard cost analysis tooling allows trends to be investigated and correlation of cost reports with custom activity metrics will facilitate causal analysis of costs incurred.

32. For KPIs phrased in terms of cost control track periods of non-adherence

Detailed cost reports are available from the GCP cost analysis tool.

5.4 Out-of-the-box GCP Tooling

5.4.1 Stackdriver

Stackdriver is a tool for:

- real-time management and analysis of logs;
- dashboarding of log and performance data;

- profiling, analysing and debugging application performance;
- alerting.

For core GCP components Stackdriver is configured to log administrator activity, data access and system events. In addition to standard logging Stackdriver agents may be installed in Compute Engines to enhance visibility of administrator activity. Stack driver will also be used for custom monitoring of SoS components.

In the SoS Stackdriver is also used for:

- analysing logs emitted from SoS components and identifying issues which may impact the quality of service;
- capturing metrics (both out-of-the box and custom);
- generating alerts for measurements which exceed safe thresholds;
- visualising metrics on dashboards.

SoS components generate logs in the following conditions:

- debug data which is only produced if software is run in a specified mode and is used mainly in development;
- informational data which helps operators to trace an issue with a live service;
- errors where a unexpected situation is encountered with a single transaction;
- fatal errors where an unexpected situation is encountered which may affect service for multiple transactions.

Stackdriver is configured to alert operators of all fatal errors and to render other errors on an operator console.

Stackdriver captures all standard metrics and the custom metrics defined in section 5.3.1. Any metric can be graphed by the standard Stackdriver visualisation tooling. Stackdriver is used to alert operators of key metrics which fall outside of pre-defined thresholds. Alert-able custom metrics are identified in section 5.3.1. Standard metrics for which thresholds will be defined include:

- `database/disk/utilization`
- `database/cpu/utilization`
- `database/memory/utilization`
- `container/accelerator/memory_used`
- `container/cpu/utilization`
- `tcp_ssl_proxy/egress_bytes_count`
- `tcp_ssl_proxy/ingress_bytes_count`

Thresholds will be documented in the YHCR Operations Guide.

Stackdriver logs will be archived after 14 days to Cloud Archive Storage.

5.4.2 Kubernetes Monitoring Engine Dashboard

The Kubernetes monitoring engine dashboard provides detailed information about a GKE cluster and its nodes and is used by operators for more in depth monitoring of the SoS compute capability.

5.5 Custom Monitoring

Standard GCP metrics focus on measuring utilisation of GCP resources. This section described custom measurements which are aligned to the functionality performed by the System of Systems and the experience of its users.

Custom metrics are statistics which are captured during a measurement window. Measurement windows are defined appropriately for the metric being captured and define a consistent time period over which metrics can be analysed historically.

Metrics have a unit of measurement. Statistics whose interpretation is dependent on the size of the measurement window are reported as a rate or percentage rather than absolute number so "Connection failures or timeouts by data provider" is reported as a number per second rather than an absolute figure and " Incomplete message pathways" is reported as percentage of all message pathways.

Metrics are either informational or alert-able. Informational metrics are captured and persisted and can be used to support analysis or be used in reports. Alert-able metrics should be monitored in real-time and if they fall outside acceptable thresholds.

Certain metrics are key indicators of health of the SoS and these are graphed and made visible to operators on a dashboard or a wall of glass. There are a number of options for visualising metrics in GCP, Prometheus being a commonly used option.

The custom metrics which are implemented are detailed in Appendix 1. Key metrics which are alert-able or will be displayed on a wall-of-glass are listed below.

5.5.1 Metrics

The metric numbers used in the following table are those used by Appendix 1 which provides a full description.

	Metric	Alerts	Dashboard
Synchronous Query			
1	Average internal round-trip time	Y	Y
2	Maximum internal round-trip time	Y	Y
3	Incomplete message pathways	Y	
4	Average external round-trip time by data provider and in total	Y	Y
5	Maximum external round-trip time by data provider and in total	Y	Y
6	Connection failures or timeouts by data provider	Y	
9	Query rate by data consumer and in total		Y
Asynchronous Query			
16	Query backlog by data provider and in total	Y	Y
Subscriptions			
30	Average internal subscription processing latency	Y	
31	Maximum internal subscription processing latency	Y	
36	Subscription placement queue sizes by data provider		Y
48	Notification delivery queue sizes by data consumer		Y
Reliable Messaging			
50	Message throughput by sender and in total		Y

51	Average message internal latency	Y	
53	Message/acknowledgement delivery queue size by recipient		Y
Identity and Access Management			
57	Claim rate by participant and in total		Y
58	Claim rejection rate by participant and in total	Y	
60	Rate of duplicate JTIs detected by participant	Y	
Regional FHIR Store			
66	Resource read rates by store		Y
67	Data read rates by store		Y
71	Request rejection rates	Y	
Consent Management			
75	Resources withheld due to a policy		Y
PIX and the Availability Service			
78	Patient/client registrations by data provider and in total		Y
79	Failed patient/client registrations by data provider and in total	Y	
82	Latency for PDS tracing		Y
83	Permanent tracing error rates by data provider and in total	Y	
84	Transient tracing error rates		Y
85	Rate of data availability requests by data consumer and in total		Y

5.6 Data Quality Analysis Tooling

Data quality analysis tooling tests individual resources passing through the YHCR for adherence with data quality standards. The data quality analysis tooling described here operate on a sample basis. It should be noted that these tools supplement standard data schema validation that that, optionally, can be performed on all resources supplied by a data provider.

Resources are extracted from synchronous query, asynchronous query and subscription notification pathways on a sample basis. Sampling intervals are random, but over time comply with configurable sample rates defined by:

- data provider;
- resource type.

Samples are extracted from the message pathway and are processed asynchronously in order to avoid placing load on the operational system.

Data quality is defined by:

- the correct FHIR profile is being used for the maturity level declared by the data provider;
- the resource structure and content comply with the profile definition;
- codes used for codable concepts exist in the coding system being used and that the coding system is permitted by the profile (see note below);
- custom rules.

The testing for existence of codes in coding systems will be performed for any coding system loaded into the SoS. SNOMED-CT will be loaded. Coding systems will be refreshed periodically according to a periodicity defined in the YHCR Operations Guide.

5.6.1 Custom Rules

Custom rules are a set of pairings of conditional statements and assertions. A conditional expression is a FHIR search path which applies to search terms entirely enclosed in the resource being tested. If a conditional expression is true when applied to a resource being tested, then the assertion is executed. An assertion is a FHIR search path which applies to search terms entirely enclosed in the resource being tested. If an assertion is true, then the resource passes the data quality check. Any assertion that yields a false result means that the resource fails.

There are limitations to the tests that are possible with custom rules defined in this way. for instance, a custom rule could be used to verify that:

An observation which is a blood test for sodium levels yields a result measured in moles per millilitre.

Whereas it is not possible to conduct the following test:

If the patient is a child of less than 10 years, then an observation for weight must be less than 140kg.

The reason being is that the condition used the Patient resource whereas the assertion is based on the Observation resource.

Whilst it would be possible to construct software that collected related resources and enabled a mix of resource content to be used in rules, this considered a candidate for future development rather than a core capability if the initial product set.

5.6.2 Recording of Data Quality Test Results

Timestamp, organisation, References to any resource failing data quality checks are persisted alongside a reason for failure. If the reason is a custom rule, then a copy of the rule is persisted.

Failure data will be available for reporting and analysis but use of the data and mechanism for improving data quality at source is outside the scope of this paper.

5.6.3 Capture of data for comparative analytics

The data quality analysis tooling allows data items to be extracted from resources and persisted for later analysis. Such analysis may compare data from different sources to identify different clinical or coding practices in organisations. For instance, extraction of diagnoses codes may reveal concentrated clusters of codes used in one care setting which do not appear in another. This may be explained through organisations offering differing specialties or maybe to different coding practices by different conditions.

Data extraction is performed in a similar manner to the application of custom rules. Pairings of conditional statements and extraction paths are recorded for resource types. A conditional expression is a FHIR search path which applies to search terms entirely enclosed in the resource being tested. If a conditional expression is true when applied to a resource being tested, then the extraction path is used to locate a single attribute of the resource. An extraction path is a FHIR Path which applies to properties entirely enclosed in the resource being tested.

Extracted data is persisted and available for analysis.

5.7 Forensic Analysis Tooling

Forensic analysis attempts to detect service abuse by monitoring the following aspects of user behaviour:

- the volume of data accessed by a user;
- the times of access by a user;
- the number of patients/clients whose data is accessed;
- the use of interaction patterns;
- reasons for accessing the YHCR;
- the data consumers employed by a user to gain access.

A user is identified in the claim made by a data consumer to the YHCR when gaining authorisation. Different schemes may be used by different consumers to identify a user. Possibilities are identified in design paper 005 – "Identity and Access Management". The ability of the forensic analysis tooling to consolidate data across consumers is entirely dependent on the consumers adopting the same identification scheme.

Forensic analysis tooling operates on audit logs written by the synchronous, asynchronous and subscription interaction patters for resources released from the boundary of the YHCR.

The structure and content of the audit record is described in design paper 009 – "Auditing".

The forensic analysis tool runs daily and from audit records produces a summary of activity for each unique user identity in claims. The data corresponds to the aspects of user behaviour identified above.

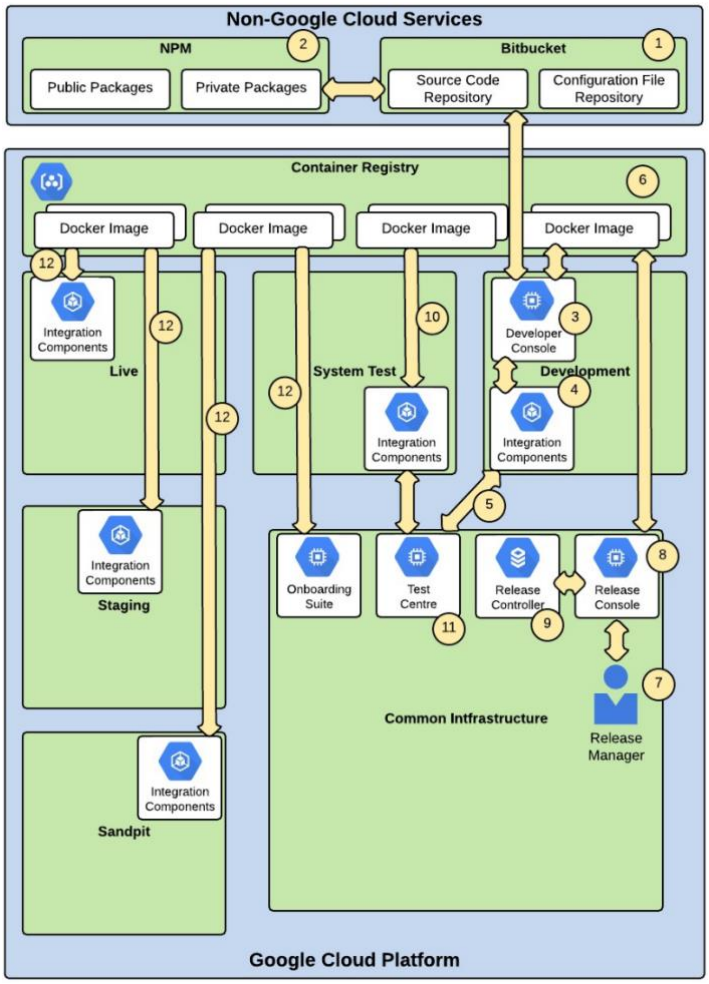
6 Release Management

Th is section considers the procedures and technologies used in managing the release of code and configuration to operational environments. The purpose of release management is to minimise the risk of unintended consequences when deploying new software or configurations. It encompasses:

- controlling access and updates to source code;
- ensuring that there is audit trail of changes to all software and configurations;
- segregating duties in the development and operations team to control the risk of malicious damage;
- ensuring the integrity of testing requirements;
- enforcing change management controls and procedures;
- orchestrating the application of code changes and ensuring consistency between environments;
- complying with planned maintenance downtime allowances and promoting zero downtime maintenance;
- providing contingency process for backing-out unsuccessful changes.

This design builds on the concept of environments which were introduced by design paper 020 – "Onboarding for Data Providers" and the reader is encouraged to review section 2 of this paper.

An overarching architecture for release management is shown below:



1. Bitbucket is used as a source code management system. The SoS uses private repositories meaning that only YHCR developers and operators have access to material placed in these. All source code, environment build scripts and configuration change scripts are under source control management.
2. The Node Package Manager (NPM) is used to distribute library code or packages. Public packages are developed by third parties for general use. Private packages are developed specifically for the YHCR. All source code for private packages is under source code management in Bitbucket.
3. Developers work on local machines initially but as code matures, it is published to the GCP development environment. Developers use a GCP compute engine, the Developer Console, for code management and to develop and apply configuration change scripts to the development environment.
4. Developers use the Developer Console to create container images. All SoS software including integration components, the onboarding suite, monitoring tooling and release management software are containerised. Integration components such as the FHIR Aggregator, IAM etc. are deployed in Kubernetes. Other software such as the Onboarding Suite are deployed as Docker containers in Compute Engines. In the development environment images are manually deployed.
5. The test-centre can be used to test components in the development environment.
6. When an image or configuration change script has been tested it becomes a candidate for release to other environments. The developer pushes tested images to the GCP Container Registry.
7. An operator with the role of Release Manager deploys software to non-development environments from a Compute Engine, the Release Management Console. Access is restricted the Release Manager Role
8. The Release Management Console runs a custom software component, the Release Controller which automates deployment in non-development environments.
9. The Release Controller groups changes within a build and persists the build definition and history of components deployed in a Cloud SQL database. A build definition is made up of a set of image versions, the target components to which the images apply, and a set configuration change script.
10. The Release Controller is initially used to deploy a build to the System Test Environment. The deployment pulls image versions from the Container Registry and executes configuration change scripts.
11. The build is tested in the System Test Environment using the test-centre.
12. Once tested the build is promoted simultaneously to the sandpit, staging, and live environments, The Release Controller interacts with the Onboarding Suite to ensure that rules are respected which restrict changes to environments which are occupied by participants who are in the process of onboarding (see Design Paper 020).

6.1 Source Code Management

Separate Bitbucket repositories are maintained for:

- each containerised image;
- private NPM packages;
- environment build scripts and standard configuration files;
- configuration change scripts.

A Bitbucket image repository contains:

- the source code which runs in the image

- the dockerfile image build script;
- docker compose file and Kubernetes YAML scripts.

Branching of Bitbucket repositories is controlled by the Release Manager.

A branch represents a release candidate. All repositories have a 'latest' branch. All code changes are applied to the latest branch and are selectively merged into release candidate branches which have yet to be deployed as an image to a non-development GCP environment (or have been backed out of the GCP System Test environment because of problems). Once a branch has been deployed to the GCP System Test environment then it is locked in Bitbucket and further changes cannot be made.

Branches are given release numbers which correspond to image versions.

6.2 Use of the GCP Container Registry

The GCP Container registry is a secure private docker image repository which features access control, vulnerability scanning, automatic deployment to Kubernetes and App engines and image versioning.

Docker images are built from the Developer Console and are pushed to the Container Registry.

Each image has a Project-ID which corresponds to the Bitbucket repository name, and Image Digest which uniquely identifies the version and an optional tag.

By convention the tag will contain the repository branch version identifier.

Role based controls restrict access to images:

- Developers can both push and pull images from the development environment but no rights in other environments.
- The Release Manager has no rights in the Container Registry other than through use of the Release Controller.
- The Release Controller, through its service account, has pull rights in the system test, sandpit, staging and live environments.

6.3 The Release Controller

The Release Controller is a custom software tool which the manages the deployment of images and configuration changes to environments. If performs the following functions:

- Enables a build to be constructed of image versions and configuration change scripts. Scripts will be applied in a defined order.
- Tracks images installed in each environment and protects against an image being inadvertently re-used.
- Enforces the requirement that builds must enter System Test before being applied elsewhere.
- Provides a 'one-click' upgrade for the System Test environment.
- Provides a 'one-click' simultaneous upgrade for Common Infrastructure, Sandpit, Staging and Live environments.
- Orchestrates upgrades by:
    - pulling images from the Container Registry;
    - suspending containerization technology where necessary;
    - executing scripts;

    - creating and running containers.
- Automates backout of changes made to an environment by restoring replaced image versions and reversing configuration changes.
- Tracks authorisation to make changes as awarded the Change Authorisation Board.
- Records a full audit trail of changes made.

6.4 Maintenance of Environment Build Scripts

Environment build scripts are used to create new environments from scratch. They operate on configuration files which environment specific parameters and create a fully functional environment with minimal operator intervention.

Configuration change scripts are developed to accompany new software releases and make incremental changes to an environment's configuration.

Environment build scripts will periodically be uplifted to incorporate the changes made by configuration change scripts. The optimal periodicity will be determined in light of operational experience but at this stage, six-monthly is thought reasonable.

In the interim new environments will be created by running the latest environment build script followed by running in turn each post-dated configuration change script.

6.5 Integration of Release Management and Change Management Processes

All changes to environment will be under the control of the YHCR Change Authorisation Board (CAB).

The CAB will authorise:

- the development and changes to software and configurations;
- the inclusion of changes in a build for deployment to the System Test environment;
- deployment of a build to an operational environments.

Th facilitate its functioning the operators of the SoS will support the CAB with impact assessments, statements of testiing performed, and completed system test reports.They will dovetail their release management cycle with CAB meetings to ensure changes can efficiency progress from development, testing and into live operation.

Appendix 1 - Metrics

	Metric	Definition and Interpretation	Units	Measurement Window	Alerting Threshold	Dashboard
	Synchronous Query					
1	Average internal round-trip time	The time from receiving a request from a data consumer to the latest request connection being placed with a data provider plus the time from receiving the last response from a data provider to the start of the aggregated response being returned to the data provider. This is a key measurement of controllable internal performance.	ms	10 mins	200ms	Y
2	Maximum internal round-trip time	The worst case internal round-trip time encountered in the measurement window. Average measurements can mask isolated issues. This metric draws attention to outlying problems.	ms	10 mins	500ms	Y
3	Incomplete message pathways	Percentage of the number of queries received which did not result in a query response being returned. Perpetual occurrences may indicate a functional problem with the system of systems.	#	10 mins	1%	
4	Average external round-trip time by data provider and in total	Measured from a point that a socket connection is requested to the endpoint to a result set being received. This is a key measurement of network latency and performance of individual data providers.	ms	10 mins	800ms	Y

5	Maximum external round-trip time by data provider and in total	Measured from a point that a socket connection is requested to the endpoint to a result set being received. Average measurements can mask isolated issues. This metric draws attention to outlying problems.	ms	10 mins	2s	Y
6	Connection failures or timeouts by data provider	Percentage of queries that did not complete. Perpetual occurrences may point to a network issue or functional problem with a data provider.	#	10 mins	1%	
7	Data volume by data provider and in total	The total size of query results received in the measurement window from a data provider. This understanding is important for cost attribution and network sizing. When balanced against metric 7 then relatively high values can point to inefficient data representation by a provider (ie: reliance on unstructured data)	MB/s	10 mins		
8	Number of resources served by data provider and in total	A count of the resources in the query result sets being returned in the measurement window by each data provider. This is an indication of the contribution being made by a data provider and low volumes relative to the providers size may indicate poor data coverage.	#/s	10 mins		
9	Query rate by data consumer and in total	The number of queries issued in the measurement by data consumers. A key measurement of the usage and availability of the SoS. Low values relative to consumer size could	#/s	10 mins		Y

		point to a narrow implementation.				
10	Average page size requested by data consumer	For paginated result sets the average page size requested by a data consumer in the measurement window. Pagination is an important mechanism for a data consumer to control performance. Very large page sizes (or not using pagination) can reduce data acquisition rates from the perspective of an end-user. A change in average page size indicates a probable change to consuming software and may require re-assurance from both the YHCR and NHS Digital (if GP Connect is being used).	#	10 mins		
11	Maximum page size by data consumer	For paginated result sets the largest page size requested by a data consumer in the measurement window. Average measurements can mask isolated issues. This metric draws attention to outlying problems.	#	10 mins		
12	Average number of resources returned per request by data consumer	For non-paginated queries the average of the number of resources return in query responses within the measurement window. A large number of resources per query may result in poor performance for end users. Where large numbers are required then the data consumer should use the asynchronous query pattern and this metric may be used to inform conversations with data consumers.	#	10 mins		
13	Maximum number of resources	For non-paginated queries the maximum number of	#	10 mins		

	returned by data consumer	resources return in query responses within the measurement window. Average measurements can mask isolated issues. This metric draws attention to outlying problems.				
14	FHIR Aggregation Multiplier	Measures the number of data providers receiving a query from a data consumer averaged over the measurement window. This is an informational statistic that highlights the efficiency of the SoS.		1 day		
15	Analysis of the reasons for access by data consumer	The reason for access is presented in a consumer's claim when requesting authorisation to retrieve data from the SoS. This metric collates the number of requests being made by each consumer for each reason for access. Use of synchronous query may be more appropriate for some reasons for access than other. This metric helps the YHCR understand the use to which it is being put and facilitates dialog with data consumers on improving their usage.	#	1 day		
Asynchronous Query						
16	Query backlog by data provider and in total	The number of queries which have been placed with data providers but for which results have not yet been collected. Partially collected results are included in the count. Calculated as the maximum in a measurement window. High or rising values can reveal an inefficiency with a	#	1 hr	50	Y

		data provider or a problem with the SoS results collection mechanism.				
17	Query backlog by data consumer and in total	The number of queries which have been placed by data consumers and for which results are now available from the SoS but which have not yet been collected. Partially collected results are included in the count. Calculated as the maximum in a measurement window. High or rising values can reveal an inefficiency with a data consumer or a problem with the SoS results collection service.	#	1 hr		
18	Average collection time by data provider and in total	Measurement of the time between an asynchronous query being placed with a data provider and the time that results are completely collected. The latency of query result collection is a performance indicator for the System of Systems and a continuous improvement objective should be to reduce this figure over time. Long collection times can be an indicator of inefficient offline processing at a data provider.	min	1 hr		
19	Abandoned result collection rate by data provider	The ratio of asynchronous queries placed with a data provider for which collection was abandoned due to timeout[(x)] against the total number of queries placed. The ratio is calculated as: the number of abandoned collections in the measurement window again the number of queries placed in the measurement window.	#	1 hr		

		Data consumers are informed of abandoned query collection using the data impairment mechanisms detailed in Design Paper 017 and can take corrective action on a case by case basis. However, perpetual occurrences can indicate a problem with individual data providers or the SoS collection mechanism.				
20	Discarded result rate by data consumer	The ratio of asynchronous queries placed with the SoS by a data consumer for which results were never collected[(x)] despite being available from the SoS. The ratio is calculated as: the number of result sets discarded in the measurement window again queries placed in the measurement window. Perpetual occurrences may indicate problem at a data provider or with the SoS results collection service.		1 hr		
21	Query rate by data consumer and in total	The number of queries issued in the measurement window by data consumers. A key measurement of the usage and availability of the SoS. Low values relative to consumer size could point to a narrow implementation.	#/s	1 hr		
22	Result collection poll rate by data consumer and in total	The number of polls for results issued in the measurement window by data consumers. A key measurement of the usage and availability of the SoS.	#/s	1 day		
23	Average poll rate productivity by data consumer	In accordance with the FHIR standard data consumers poll the SoS to determine the status of result availability. The metric measures the number	#	1 day		

		of polls made in the measurement window before results are available. A high value indicates that a data consumer is consuming unnecessary resources by polling too frequently. Low values indicate that that result collection latency could be reduced by more frequent polling.				
24	Average poll rate productivity by data provider	In accordance with the FHIR standard data the SoS polls data providers to determine the status of result availability. The metric measures the number of polls made in the measurement window before results are available. A high value indicates that the SoS is consuming unnecessary resources by polling too frequently. Low values indicate that that result collection latency could be reduced by more frequent polling.	#	1 day		
25	Asynchronous query data volume by data provider and in total	The total size of query results received in the measurement window from a data provider. This understanding is important for cost attribution and network sizing.	MB	1 day		
26	Asynchronous query resource counts by data provider and in total	The total number of resources included in query results received in the measurement window from a data provider. This metric is a measurement of relative contribution by data providers.	#	1 day		
27	Asynchronous query data volume by	The total size of query results returned to data	MB	1 day		

	data provider and in total	consumers in the measurement window. This understanding is important for cost attribution and network sizing.				
28	Asynchronous query resource counts by data consumer and in total	The total number of resources included in query results returned to data consumers in the measurement window. This metric is a measurement of usage of the SoS by data consumers.	#	1 day		
29	Analysis of the reasons for access by data consumer	The reason for access is presented in a consumer's claim when requesting authorisation to retrieve data from the SoS. This metric collates the number of requests being made by each consumer for each reason for access. Use of asynchronous query may be more appropriate for some reasons for access than other. This metric helps the YHCR understand the use to which it is being put and facilitates dialog with data consumers on improving their usage.	#	1 day		
Subscriptions						
30	Average internal subscription processing latency	A measurement the time between the SoS receiving a subscription and it being placed in queues for delivery to data providers. This is a key measurement of the performance of subscription processing.	ms	1 hr	1s	
31	Maximum internal subscription processing latency	The worst-case internal subscription processing time encountered in the measurement window.	ms	1 hr	5s	

		Average measurements can mask isolated issues. This metric draws attention to outlying problems.				
32	Rate of subscriptions placement by data provider and in total	The number of subscriptions delivered to each data provider in the measurement window. This is an indication of the health of the subscription processing. A fall in transaction throughput could indicate a problem with a data provider or an issue with the SoS.	#/s	1 hr		
33	Rate of subscription receipt by data consumer and in total	The number of subscriptions received by each data consumer in the measurement window. This is an indicator of service availability. A rapid increase may overload the SoS or cause an increase in resource utilisation with cost consequences.	#/s	1 hr		
34	Average subscription delivery attempts by data consumer	The average number of attempts made to deliver a subscription to a data provider in a measurement window. Large or rising values indicate incompatibility between a data provider and the SOS or may point to a software fault.	#	1 hr		
35	Placement latency by data provider and in total	A measurement of the average delay between a subscription being extracted from a queue for delivery and acceptance of the subscription by a data provider. Large are rising values could indicate a network problem or inefficient processing by a data provider.	ms	1 hr		

36	Subscription placement queue sizes by data provider	The number of subscriptions queued for delivery to a data provider. The measurement is taken at the end of each measurement window. Large and rising queue sizes might be a result of demand exceeding capacity to scale or may indicate a network problem or inefficient processing by a data provider.	ms	10 mins		Y
37	Subscription placement multiplier	The number of subscriptions placed with data providers exceeds the number placed by data consumers with the system of systems due to subscriptions being non-provider specific and expansion required for subscriptions made to patient cohorts. The measurement is taken at the end of the measurement window. This is an informational statistic that explains how the SoS is being used and may inform software design decisions.	#	1 day		
38	Number of active subscriptions by data provider and in total	A count of the number of active subscriptions which have been placed with data providers as measured at the end of the measurement window. This is an informational statistic that explains how the SoS is being used.	#	1 day		
39	Number of active subscriptions by data consumer and in total	A count of the number of active subscriptions which have been placed by data consumers with the SoS as measured at the end of the measurement window. This is an informational statistic that explains how the	#	1 day		

		SoS is being used.				
40	Average notification latency by data provider	A measurement of the average time between a resource being created or modified and it being delivered to the SoS in a subscription notification. The resource modification date is obtained from the resource meta data. This is a key indicator of the currency of data available through subscription. A continuous improvement objective for the YHCR should be to reduce this value over time.	#	1 day		
41	Notification volume by data provider and in total	The number of notifications received from a data provider in the measurement window. A key sizing statistic for the YHCR. Rapid fluctuations may cause scalability problems.	#/s	1 day		
42	Notification delivery error rates by data provider	The number of attempts in the measurement window to send a notification to the SoS which were dropped. Failed attempts should be retried; however, perpetually high rates might indicate a compatibility problem between the SoS and a data provider or software fault.	#/s	1 hr		
43	Consumer aggregation efficiency rate	The number of notifications dispatched to data consumers in a measurement window should exceed the number of notifications received from data providers. This is because the SoS consolidates identical subscriptions from different consumers into a single subscription to data providers.	#	1 day		

		This is an informational statistic that illustrates efficiency of processing of the SoS.				
44	Average internal notification processing latency	Measures the time between the SoS receiving a notification and it being placed in queues for delivery to data consumers. This is a key measurement of the performance of subscription processing.	ms	1 hr		
45	Maximum internal notification processing latency	The worst-case internal notification processing time encountered in the measurement window. Average measurements can mask isolated issues. This metric draws attention to outlying problems.	ms	1 hr		
46	Average notification delivery attempts by data consumer	The average number of attempts made to deliver a notification to a data consumer in a measurement window. Large or rising values indicate incompatibility between a data consumer and the SOS or may point to a software fault.	#	1 hr		
47	Notification delivery latency by data consumer and in total	A measurement of the average delay between a notification being extracted from a queue for delivery and acceptance of the notification by a data consumer. Large are rising values could indicate a network problem or inefficient processing by a data consumer.	ms	1 hr		
48	Notification delivery queue sizes by data consumer	The number of notifications queued for delivery to a data consumer. The measurement is taken at the end	#	10 mins		Y

		of each measurement window. Large and rising queue sizes might be a result of demand exceeding capacity to scale or may indicate a network problem or inefficient processing by a data consumer.				
49	Analysis of the reasons for access by data consumer	The reason for access is presented in a consumer's claim when requesting authorisation to register subscriptions with the SoS. This metric collates the number of requests being made by each consumer for each reason for access. Use of subscriptions may be more appropriate for some reasons for access than others. This metric helps the YHCR understand the use to which it is being put and facilitates dialog with data consumers on improving their usage.	#	1 day		
Reliable Messaging						
50	Message throughput by sender and in total	The number of messages and acknowledgments received during the measurement window. A drop in throughput could indicate a problem with a sender, network por internal problem with the system of systems.	#/s	10 mins		Y
51	Average message internal latency	A measurement the time between the SoS receiving a message or acknowledgement and it being placed in queues for delivery to recipient. This is a key measurement of the performance of message processing.	ms	10 mins	1s	

52	Message/acknowledgement delivery latency by message recipient and in total	A measurement of the average delay between a message being extracted from a queue for delivery and acceptance of the message by its recipient. Large or rising values could indicate a network problem or inefficient processing by a recipient.	ms	10 mins		
53	Message/acknowledgement delivery queue size by recipient	The number of messages/acknowledgements queued for delivery to a recipient. The measurement is taken at the end of each measurement window. Large and rising queue sizes might be a result of demand exceeding capacity to scale or may indicate a network problem or inefficient processing by a recipient.	#	10 mins		Y
54	Average message delivery attempts by data consumer and in total	The average number of attempts made to deliver a message/acknowledgement to a recipient in a measurement window. Large or rising values indicate incompatibility between a recipient and the SOS or may point to a software fault.	#	1 hr		
55	Average message size by sender and in total	The size of messages received in the measurement window from a sender. This measurement is important for cost attribution and network sizing.	MB	1 hr		
56	Reliable messaging efficiency by message/acknowledgement recipient and in total	The reliable messaging design requires message senders to resend messages for which no acknowledgement has been received and for message recipients to acknowledge all messages	#	1 day		

		received. The efficiency of a messaging channel can be measured by the percentage of messages which are acknowledged on first delivery and for which resends are not attempted. This metric is calculated as the ratio of the number of unique message Bundle.ids (refer to design paper 006 – "Reliable Messaging Infrastructure") processed during the measurement window against the total number of messages received. A continuous improvement objective should be to increase this figure over time				
Identity and Access Management						
57	Claim rate by participant and in total	Measures the activity of participants to the YHCR. The metric is a count of the total number of claims made during the measurement window. This is a primary indicator of availability of the YHCR.	#/s	10 mins		Y
58	Claim rejection rate by participant and in total	Measures the percentage of claims which fail validation for any reason. This measurement is a key security indicator. High rejection rates may suggest a breach attempt.	#	10 mins	10%	
59	Expired token by participant and in total	Measures attempts to use a JWT beyond its expiry time. It is a count of the number of service requests that are rejected during the measurement period. This is an indication that consumer software hasn't been designed to automatically refresh tokens.	#/s	1 hr		
60	Rate of duplicate JTIs detected by	Measures the percentage of claims which fail	#	10 mins	2%	

	participant	because of an attempt to use JavaScript Token Identifier which has been previously used. This measurement is a key security indicator. High rejection rates may suggest a replay attack attempt.				
61	Analysis of reasons for access by participant	The reason for access is presented in a participant claim when requesting authorisation to retrieve data from the SoS. This metric collates the number of requests being made by each participant for each reason for access. This is for informational purposes to help the YHCR understand the use to which it is being put.	#	1 day		
62	Analysis of type of user by participant	The type of user is presented in a participant claim when requesting authorisation to retrieve data from the SoS. This metric collates the number of requests being made by each participant for each type of user. This is for informational purposes to help the YHCR understand the use to which it is being put.	#	1 day		
63	Number of active users by participant and in total	A count of the unique identifiers presented in claims during the measurement window. This a key measurement of the uptake of the YHCR.	#	1 day		
Regional FHIR Store						
64	Number of persisted resources by store	A count of the resources persisted in a FHIR store measured at the end of a measurement window. This measurement assists with infrastructure sizing.	#	1 day		
65	Total data volume persisted by store	The data volume persisted in a FHIR store measured	MB	1 day		

		at the end of a measurement window. This measurement assists with infrastructure sizing.				
66	Resource read rates by store	The number of resources read in a measurement window divided by the total time to service resource read requests which is measured as the time between the receipt of a read requests and the dispatch of results. This is a key performance metric.	#/s	10 mins		Y
67	Data read rates by store	The data volume of resources read in a measurement window divided by the total time to service resource read requests which is measured as the time between the receipt of a read requests and the dispatch of results. This is a key performance metric.	MB/s	10 mins		Y
69	Resource write rates by store	The number of resources written in a measurement window divided by the total time to service resource read requests which is measured as the time between the receipt of a read requests and the dispatch of results. This is a key performance metric.		10 mins		
70	Date write rates by store	The data volume of resources written in a measurement window divided by the total time to service resource read requests which is measured as the time between the receipt of a read requests and the dispatch of results.		10 mins		

		This is a key performance metric.				
71	Request rejection rates	The percentage of requests rejected for whatever reason in the measurement window. This measurement may be an indicator functional or security issues.	#	10 mins	1%	
72	Average request latency	The average time from receipt of a request to request fulfilment during a measurement window. This is a key performance indicator.	#/s	10 mins		
73	Request failure rates	The number of requests for which a response was not completely served during the measurement window. High values indicate a functional problem.	#/s	10 mins		
Consent Management						
74	Number of consent records by policy	A count at the end of the measurement window of the total number of consent records linked to a policy. This metric is for informational purposes.	#	1 day		
75	Resources withheld due to a policy	A percentage of the number of resources which have been withheld during a measurement window because of a data access policy and consent to it. This metric is an indicator of the efficiency of the SoS and may influence future design decisions.	#	1 hr		Y
76	Resources released with qualification by policy	A percentage of the number of resources which have been released but with a qualification over usage during a measurement window because of a data access policy and consent to it.	#	1 hr		

		This metric is an indicator of the efficiency of the SoS and may influence future design decisions.				
77	Percentage of resources tested	The ratio of the number of resources passing through the SoS in a measurement window which have been tested because of a data access policy compared to the total number of resources served. This metric is an indicator of the efficiency of the SoS and may influence future design decisions.	#	1 day		
PIX and Data Availability Service						
78	Patient/client registrations by data provider and in total	The number of PIX registrations during a measurement window by data provider. This is key indicator of compliance by data providers to register patients for which they have a relationship. Low rates indicate selective registration.	#/s	10 mins		Y
79	Failed patient/client registrations by data provider and in total	Percentage of PIX registrations in the measurement window that do not complete for whatever purpose. High values may indicate a compatibility problem between the data provider and SoS or a software fault.	#	10 mins	1%	
80	Number of patients/clients registered by data provider	The total number of registers patients and clients as counted at the end of the measurement window. This is an informational statistic that illustrates usage of the YHCR.	#	1 day		
81	Average number of patient/client linkages	The number of linkages equates to the number of data providers which have had contact with the	#	1 day		

		patient/client. This metric is measured at the end of the measurement window. This is an informational statistic that illustrates usage of the YHCR.				
82	Latency for PDS tracing	A average of the time from connecting to the Spine Mini Service Provider for a trace to receive a response averaged during the measurement window. This is a key performance indicator.	ms	10 mins		Y
83	Permanent tracing error rates by data provider and in total	The percentage of PDS traces which result in a permanent error leading to the patient registration being rejected. Data providers are expected to trace patients before registering contact with the SoS. High failure rates indicate that a provider is not compliant with this requirement.	#	1 hr	1%	
84	Transient tracing error rates	The percentage of PDS traces that returned a transient error and needed to be repeated. A high or rising number could indicate a problem with the national SMSP service.		10 mins		Y
85	Rate of data availability requests by data consumer and in total	The number of requests made during the measurement window against the data availability service. This is a key indicator of service availability and activity at a data consumer.		10 mins		Y
86	Efficiency of data availability	The percentage of data availability requests which		1 hr		

	requests by data consumer	return a positive response. This is an indicator of the value of the YHCR to a data consumer. Very low numbers mean that the YHCR is only surfacing data for a small percentage of patients or clients.				

Appendix 2 – Maturity Matrix

Section	Narrative	Consultative	Draft	Normative
1 Introduction 1.1 Purpose of this Document	X			
1.2 Interaction Patterns	X			
1.3 Relationship of this Document with Other Standards	X			
1.4 Intended Users of the This Document	X			
2 Performance and Scalability 2.1 Sizing Assumptions and Estimates of Demand		X		
2.2 Performance Objectives 2.2.1 Synchronous Query		X		
2.2.2 Asynchronous Query		X		
2.2.3 Subscriptions		X		
2.2.4 Transactional Messaging		X		
2.3 Scalability in the Cloud		X		
2.4 Performance Testing 2.4.1 Synchronous Query Testing			X	
2.4.2 Soak Testing			X	
2.4.3 Testing of Asynchronous Patterns			X	
2.4.4 Reuse of Performance Test Tooling	X			
2.5 Options for Improving the Elasticity of Input / Output Bound Components 2.5.1 Use of an Object Store with Offline Indexing		X		
2.5.2 Dynamic Sharding		X		
3. High Availability	X			
3.1 GCP Regions and Zones	X			
3.2 Objectives for High Availability in the Cloud		X		
3.3 GCP Resource High Availability Configuration 3.3.1 Compute Engine				X
3.3.2 Kubernetes Engine				X
3.3.3 Cloud SQL				X
3.3.4 Cloud Storage				X
3.3.5 Fire Store				X
3.3.6 Load Balancer				X

3.3.7 DNS				X
3.3.8 Cloud Tasks				X
3.3.9 Memorystore				X
3.3.10 Key Management Service				X
3.4 Core and Subsidiary SoS Services		X		
3.5 High Availability by Environment		X		
3.6 Availability Targets		X		
3.7 Planned Maintenance			X	
4. Backup and Recovery 4.1 Introduction		X		
4.2 Data Content of the SoS	X			
4.3 Approach to Backup		X		
4.4 Data Recovery Procedures and Contingency Plans 4.4.1 Exporting Data Off-Cloud		X		
4.5 Validation of Backups and Recovery Rehearsal		X		
4.6 Disaster Recovery		X		
5 Monitoring and Alerting 5.1 Monitoring Defined	X			
5.2 A Monitoring Architecture			X	
5.3 Scope and Requirements for Monitoring 5.3.1 Performance Monitoring		X		
5.3.2 Availability Monitoring		X		
5.3.3 Data Quality Monitoring		X		
5.3.4 Security Monitoring		X		
5.3.5 Profile of Usage Monitoring		X		
5.3.6 Integrity of Usage Monitoring		X		
5.3.7 Infrastructure Monitoring		X		
5.3.8 Cost Monitoring		X		
5.4 Out-of-the-box GCP Tooling 5.4.1 Stackdriver			X	
5.4.2 Kubernetes Monitoring Engine Dashboard			X	
5.5 Custom Monitoring 5.5.1 Metrics		X		
5.6 Data Quality Analysis Tooling 5.6.1 Custom Rules		X		
5.6.2 Recording of Data Quality Test Results		X		
5.6.3 Capture of data for comparative analytics		X		
5.7 Forensic Analysis Tooling		X		

6 Release Management 6.1 Source Code Management				X
6.2 Use of the GCP Container Registry		X		
6.3 The Release Controller		X		
6.4 Maintenance of Environment Build Scripts		X		
6.5 Integration of Release Management and Change Management Processes		X		
Appendix 1 - Metrics		X		