IT Project Guidance

Design:  
Services to consider for Integration

Version:

0.3

## Description

This document outlines common dependency services relied upon by end user services, to inform subsequent planning, procuring, integration and monitoring activities.

## Synopsis

Systems rely on secondary systems to meet user expectations. These include monitoring, data storage, notification, caching, searching, malware detection, etc.

## Contents

[Description 1](#_Toc149037456)

[Synopsis 1](#_Toc149037457)

[Contents 2](#_Toc149037458)

[Introduction 3](#_Toc149037459)

[Issue 3](#_Toc149037460)

[Risk 3](#_Toc149037461)

[Resolution 3](#_Toc149037462)

[Services 4](#_Toc149037463)

[Hosting 4](#_Toc149037464)

[Database service 4](#_Toc149037465)

[NoSQL Media Datastore service 5](#_Toc149037466)

[Confidential Credential Data store 6](#_Toc149037467)

[SMTP Mail service 6](#_Toc149037468)

[Cache service 7](#_Toc149037469)

[Pub/Sub Service 8](#_Toc149037470)

[Malware Detection Service 8](#_Toc149037471)

[Search Service 9](#_Toc149037472)

[Rule Engine Services 10](#_Toc149037473)

[Workflow Management Services 10](#_Toc149037474)

[Conclusion 10](#_Toc149037475)

[Appendices 11](#_Toc149037476)

[Appendix A - Document Information 11](#_Toc149037477)

[Versions 11](#_Toc149037478)

[Images 11](#_Toc149037479)

[Tables 11](#_Toc149037480)

[References 11](#_Toc149037481)

[Review Distribution 11](#_Toc149037482)

[Audience 12](#_Toc149037483)

[Structure 12](#_Toc149037484)

[Diagrams 12](#_Toc149037485)

[Terms 12](#_Toc149037486)

## Introduction

To decrease development costs, systems integrate with and leverage both 3rd party code libraries and 3rd party services rather than recode what is readily available.

Integrating with 3rd party code libraries decreases development cost, but comes with the responsibility of requiring being monitored, managed, and updated, to keep abreast with the latest features and improvements to security and/or functionality.

Integrating with 3rd party services, by getting an account to a paid service, removes this responsibility. As long as one is paying to keep the account active, the service provider is expected to make available an incrementally improving service, with the least disruption to service users.

When procuring a Software as a Service (SaaS) the vendor takes care of most if not all of these integration needs behind the scenes, with the ability to supersede one or two services with organisation wide services.

For example, they may provide the means to use your organisation’s mail service, and maybe even your organisation’s SEIM and monitoring service.

When developing ones own service, one has to procure these services and integrate with them.

## Issue

As most delivery stakeholders are focused on delivering the key service -- whether it be SaaS or custom developed as a PaaS overing – and omit to plan for the selecting, procuring, integrating and monitoring integrations to 3rd party services.

## Risk

The lack of awareness and planning for these dependency services adds service delivery risk.

## Resolution

This document outlines common dependency services to consider when planning the delivery of a service, reducing delays / non-delivery risks.

# Services

## Hosting

While not an integration in itself, [cloud] hosting is listed here first, as many of the following service integrations are made easier if one is hosting services in environments created within commonly available cloud services.

## Database service

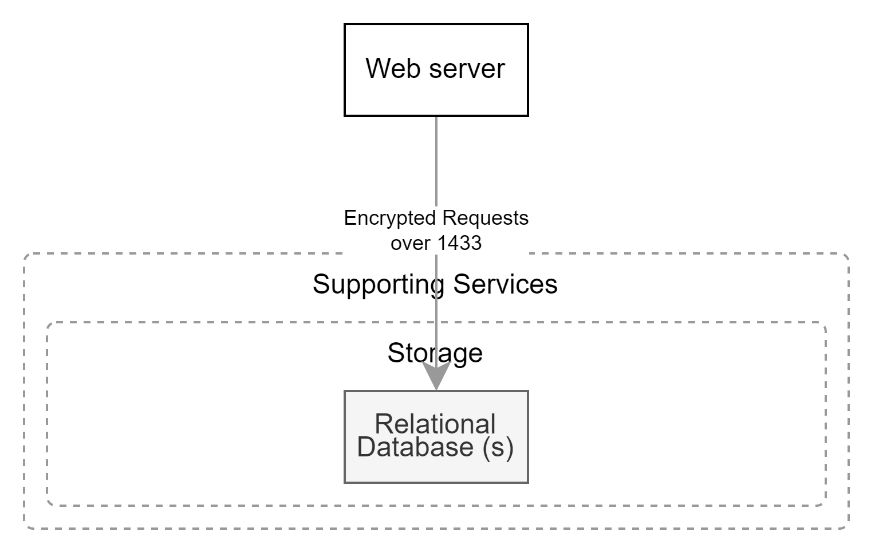


Figure 2: Dependencies on Relational Storage

Systems require integration to relational databases to persist structured relational records.

Consider the following when developing this form of storage:

* The database server and database(s) should be created using Infrastructure as Code (IaC) run as an idempotent automation step within a deployment pipeline (developing the instructions in an idempotent manner permits the deployment pipeline to reapply settings without disruption to active users),
* The development of the database schema is also done via an idempotent step, but in a step separate from the Infrastructure as Code (IaC) step required to build the database (and optionally the database server under it),
* If not using a SaaS or PaaS service (i.e., IaaS approach to developing a database), the database should be created on a database server within a subnetwork that only permits traffic to a specific port from the application/web server,
* Databases should be configured to be encrypted,
* Database backups should be configured to be encrypted, in case a user gains access to a prod environment and is able to download a backup for restoration on a different device,
* To avoid the act of reporting having performance impacts it is good practice to plan to use for reporting a database that is separate from the operational database.

## NoSQL Media Datastore service

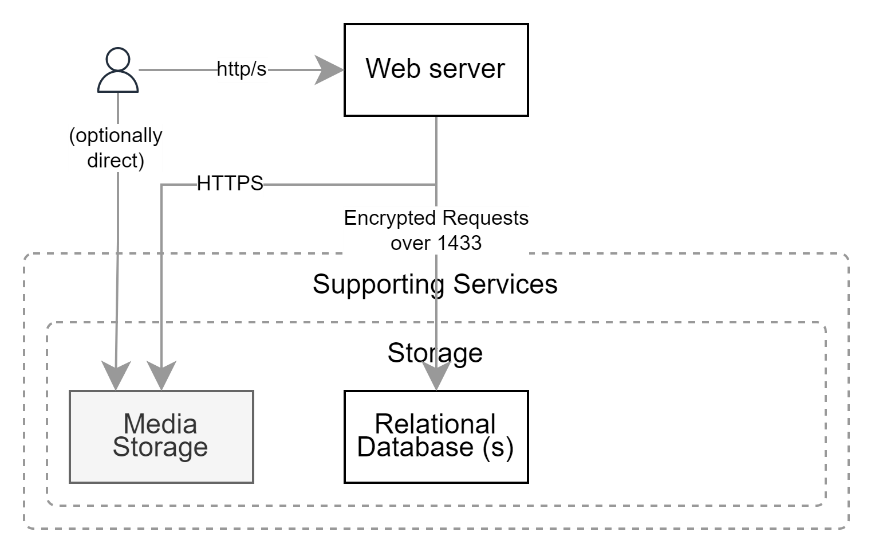


Figure 3: Media storage

While relational databases are fast for the storage of structured data as small relational records, they are not the best solution for persisting media, whether it be part of the system or uploaded and later accessed by end users.

Best practice is to use a NoSQL database for persisting ‘blobs’ of stream-able data (images, sounds, videos, etc.).[[1]](#footnote-2)

Consider:

* Ensuring the datastore is encrypted,
* Ensuring that it only permits secure (HTTPS) communication to it
* Deciding if access to media must be audited (best practice is to audit all user activity)
* Therefore -- and If it can be configured -- limiting access to it from the open web, and only from the webservice itself

## Confidential Credential Data store

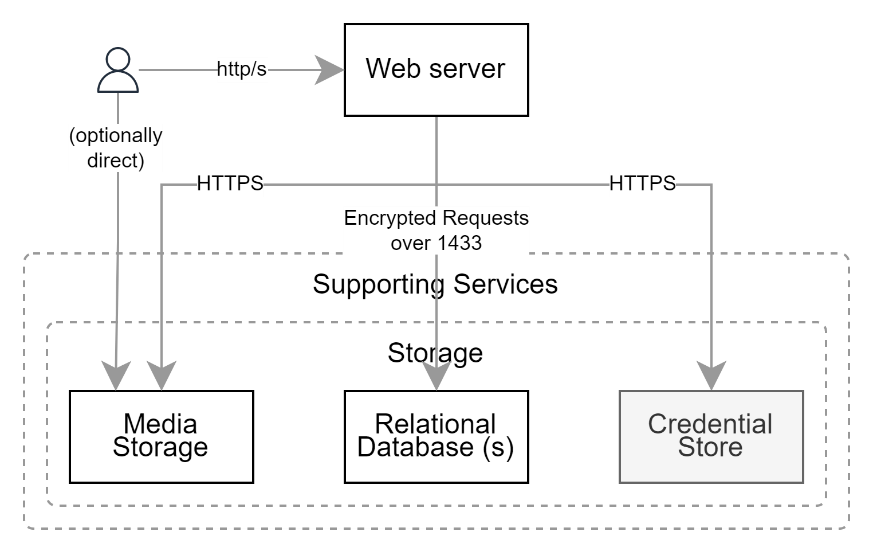


Figure 4: Confidential Credential Datastore

While relational databases are generally thought of first when considering the storage needs of systems, a key service to consider is the one used to persist confidential integration information, accessible only from the service, and the deployment pipeline agent that put them there.

Note:  
Using a keystore instead of a system’s configuration file (“config file”) provides better security in case an agent gains access to the webservice device.

## SMTP Mail service

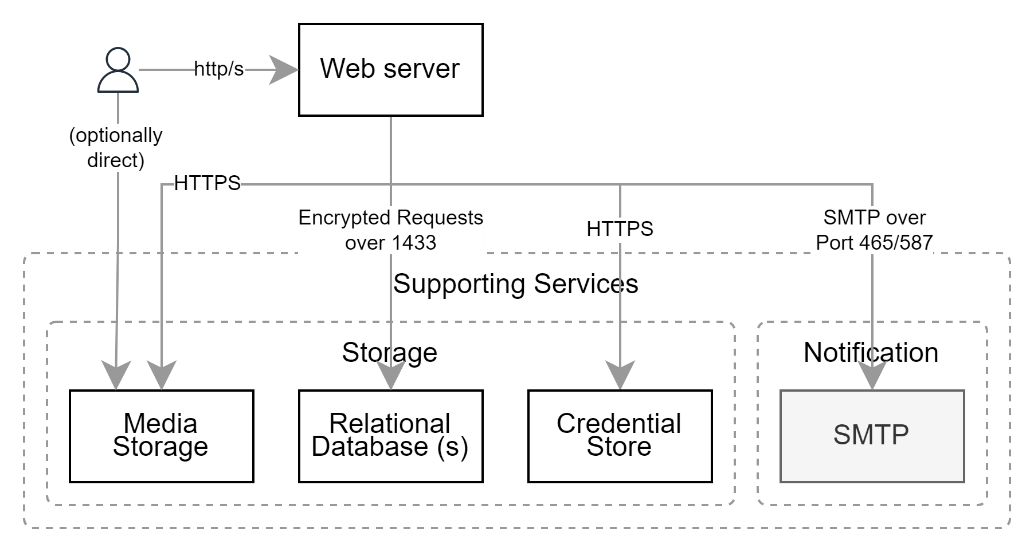


Figure 5: SMTP Mail Service Dependency

Mature systems benefit from integrating with mail services to send notifications outside the system.

Use cases include but are not limited to notifying users as follows:

* They have received an invitation to a role within a group (e.g.: Member) or associated to a resource (e.g.: Collaborator).
* Their role has been changed,
* A resource they are responsible has not been looked at for a while – maybe it needs looking at to decide it is still current, or action needs taking,
* Etc.

## Cache service

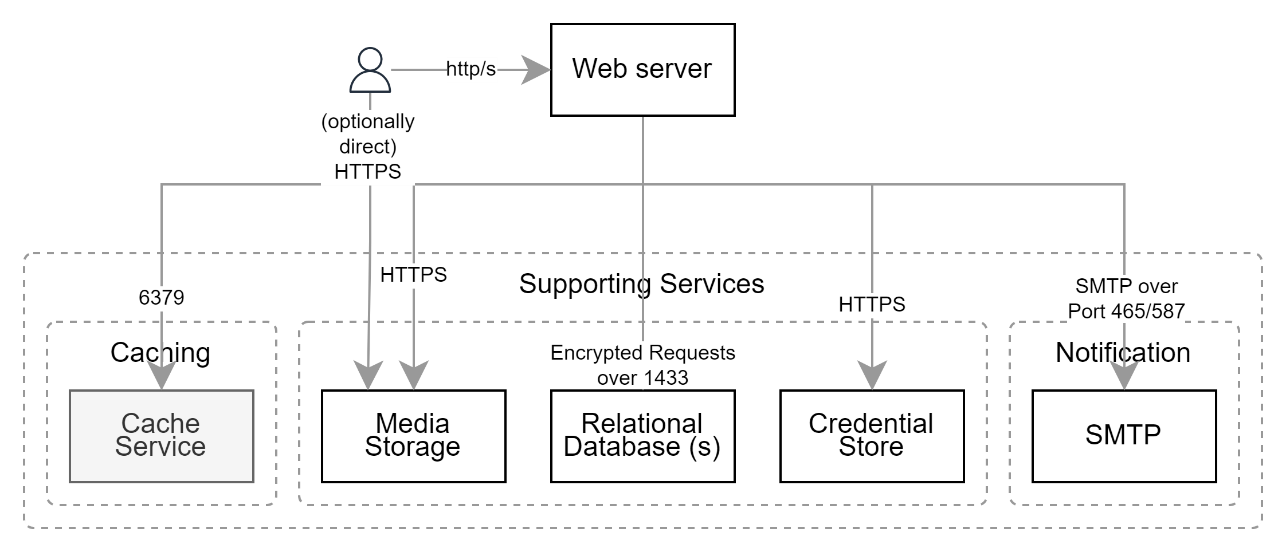


Figure 6: Caching Service Dependency

In-memory caching, shared between webservers, improves availability of the service while reducing device infrastructure costs. It does this by caching results of slow changing information, to reduce the number of queries to any datastore.

Consider:

* Best practice is to cache information closest to the user, to not only reduce the number of queries to datastores, but also remove the need to recompute information prior to formatting it. [[2]](#footnote-3)
* Developing different cache timeouts for different purposes (e.g., 1 second, 5 seconds, 30 seconds),
* Avoid caching information longer than 30 seconds as it can cause unintended latency to the updating of results, in turn increasing support calls and costs,

## Pub/Sub Service

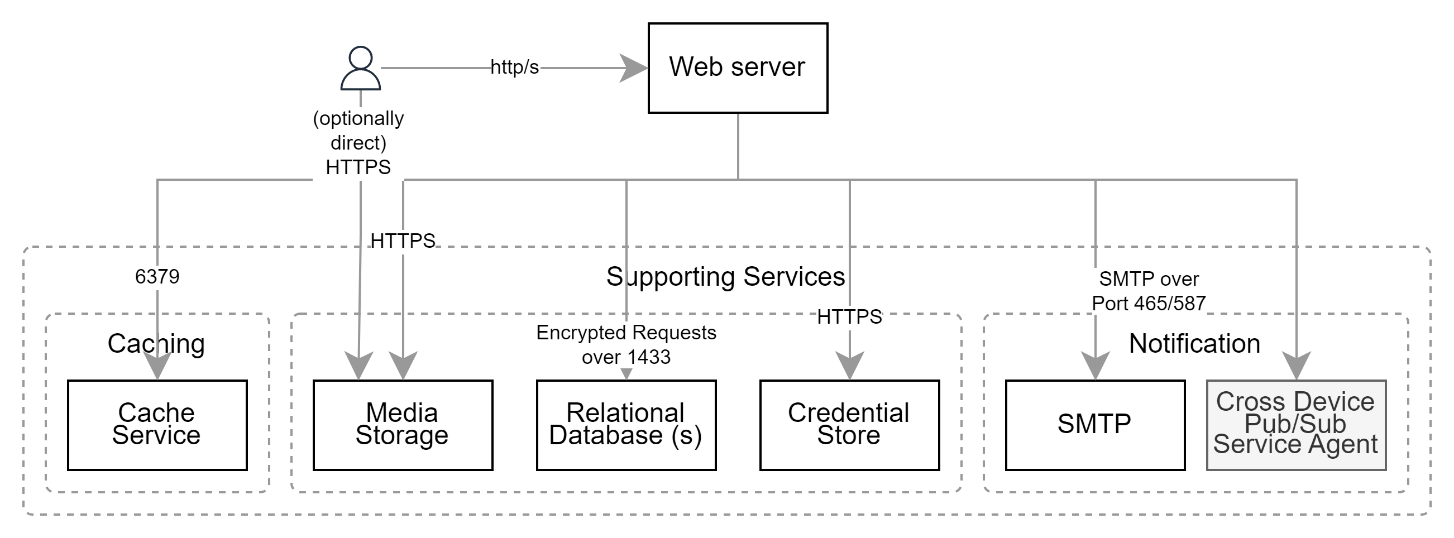


Figure 7: Pub/Sub Service Dependency

Current best practice is to develop systems in a highly decoupled manner, to improve testability, modularity and maintainability. A key practice is to use a publish/subscriber service to communicate between components and devices.

## Malware Detection Service

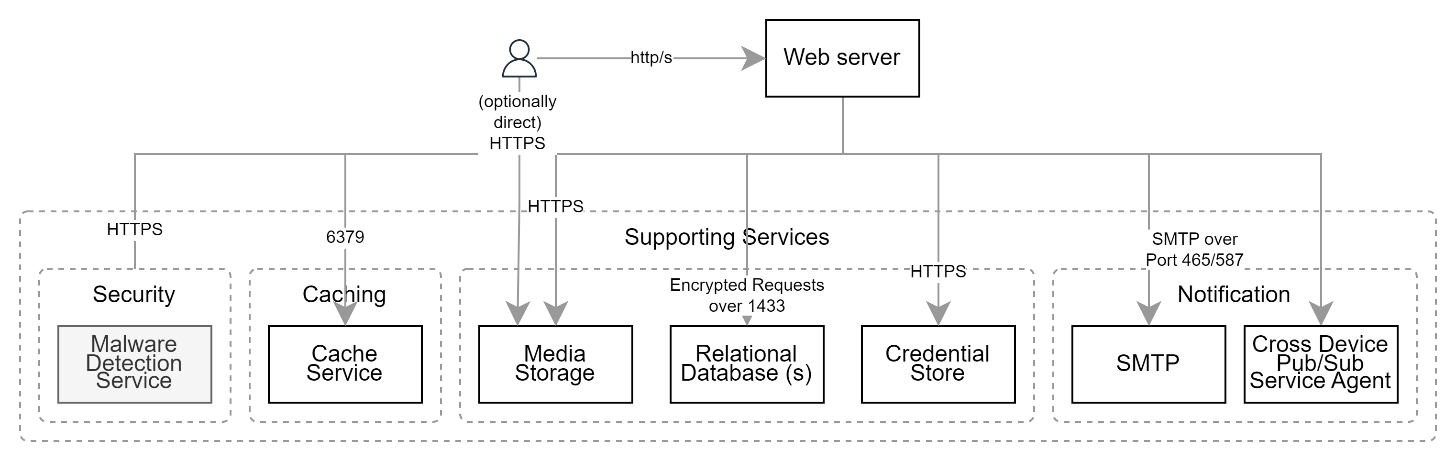


Figure 8: Malware Detection Service Dependency

Even for the simplest systems that don’t manage uploaded documents or images, it is surprising how early the ability for end users to upload media is required – and by extension, malware detection is required.

Simple examples of such cases include but are not limited to system admins wanting to update privacy statements or data usage statements without redeploying the system, uploading logos for the banner, footer, etc. System Users may want to upload an image of themselves for their avatar image, etc.

To protect against malware being uploaded, enterprises prefer using an overall WAF service to protected the services they use. While such approaches improve security in a a general way, there are many historical lessons that one should be resilient in one’s own right[[3]](#footnote-4)[[4]](#footnote-5)[[5]](#footnote-6).

Perimeter WAF based protection can work, but poor system design can produce a relatively poor user experience when the WAF denies the request, without the UX able to notify the user as to why.

Secondly, WAFs don’t stop the latest malwares until they have been updated. Which implies that without a means to scan the data store in which malware infected media has been uploaded will continue to infect downloaders of the media, indefinitely.

Malware that infects users inflicts reputational damage to the service and organisation that is providing the service.

A more sophisticated solution is to incorporate a malware detection service into the system that a) watches uploads – even if the WAF was supposed to be doing this already – and b) regularly scans previously uploaded media using malware detection patterns it acquired after the media was uploaded.

## Search Service

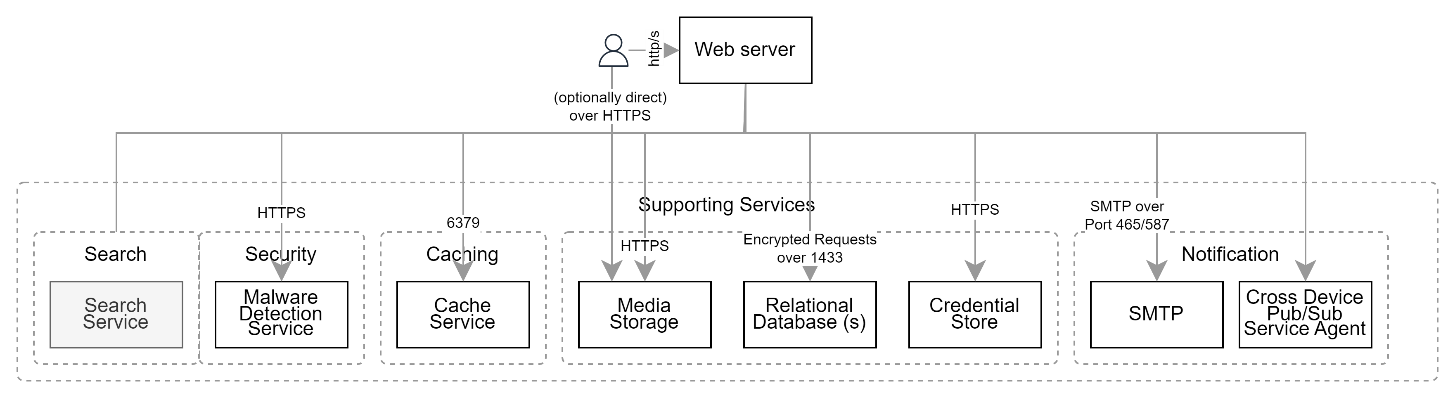


Figure 9: Search Service Dependency

Users use services primarily to find information as opposed to post new information.

Users benefit from having the means to search for information in a forgiving, phonetic based manner, able to find the correct results if query information is incorrectly spelled[[6]](#footnote-7).

## Rule Engine Services

System designs may include a reliance on a 3rd party rule engine service.

Note:  
Rule engines are often included for the misguided objective of permitting non-developers (e.g., stakeholder and system analysts) to upload new logic to a system without waiting for a redeployment, nor a reliance on developers. The reasons this approach is no longer considered best practice are several. For one, its often based on a misdiagnosis of existing manual deployment strategies -- concluding they are time consuming and expensive -- rather than recognising that they are simply not automated. Secondly, BAs generally do not develop pipeline run test rigs, and therefore lead to an introduction of errors in logic that are not caught before they are deployed.

## Workflow Management Services

The development of workflows to manage resources is recommended. Workflows can be developed in code by developers working to meet target tests, or they can be developed in “low code/no-code” within a dedicated workflow management service.

There are advantages and disadvantages to both approaches.

## Conclusion

In general, custom systems integrate with and rely on a wide set of services to meet current user expectations of service.

Appendices

Appendix A - Document Information

### Author & Contributors

Sky Sigal, Solution Architect

### Versions

0.1: Initial document

0.3: Added diagrams

### Images

[Figure 2: Dependencies on Relational Storage 4](#_Toc149037263)

[Figure 3: Media storage 5](#_Toc149037264)

[Figure 4: Confidential Credential Datastore 6](#_Toc149037265)

[Figure 5: SMTP Mail Service Dependency 6](#_Toc149037266)

[Figure 6: Caching Service Dependency 7](#_Toc149037267)

[Figure 7: Pub/Sub Service Dependency 8](#_Toc149037268)

[Figure 8: Malware Detection Service Dependency 8](#_Toc149037269)

[Figure 9: Search Service Dependency 9](#_Toc149037270)

### Tables

**No table of figures entries found.**

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Sandy Britain, Enterprise Architect |  |
| Amy Orr, Data Domain Architect |  |
| Rodney Snell, Business & Technical Lead |  |
| Matt Duguid, DevOps Engineer |  |
| Russell Campbell, Project Manager |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project’s Glossary.

##### IT

: acronym for Information, using Technology to automate and facilitate its management.

##### ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

1. Note that the relational database can still be used to persist structured, indexed, metadata *about* the media blob. [↑](#footnote-ref-2)
2. One effect of this is that the cache will contain multiple copies of information, one per culture-language. [↑](#footnote-ref-3)
3. [Maginot Line - Wikipedia](https://en.wikipedia.org/wiki/Maginot_Line) [↑](#footnote-ref-4)
4. <https://en.wikipedia.org/wiki/Siegfried_Line> [↑](#footnote-ref-5)
5. [Perception and reality of perimeter security effectiveness - Help Net Security](https://www.helpnetsecurity.com/2015/04/16/perception-and-reality-of-perimeter-security-effectiveness/) [↑](#footnote-ref-6)
6. Select and use services that can adapt phonetics based on the user’s cultures, such that ‘*wh*’ sounds differently in English-NZ from ma-NZ. [↑](#footnote-ref-7)