# LOGO Ministry of Education.JPGProject Closure Report

Delegations Proof of Concept

Last Published Date: 13 October 2015

Document Information

Template Properties

This table describes the details of the EPMO template used to create this document.

|  |  |
| --- | --- |
| Template Name | Project Closure Report |
| Owner | EPMO Manager |
| Current Version | V3.0 |
| Location | http://tetahuhu.moe.govt.nz/policiesProcessesAndTools/Projects/Templates.aspx |
| Date Created | July 2014 |
| Previous Version | V2.0 |

Document Properties

This table describes this version of the document.

|  |  |
| --- | --- |
| Document Name | Delegations Proof of Concept – Project Closure Report |
| Author | Ata, McGregor, Bill Young & Jamie Cavanaugh |
| Owner | James Collier |
| Location | Delegations/MoE |
| Version Number | 1st draft |
| Date Created | 28/09/2015 |
| Last Updated | 28/09/2015 |
| Date Distributed | 28/09/2015 |

Document History

This table provides a history of changes to this document.

| Version No. | Date | Summary of Changes |
| --- | --- | --- |
| Draft 1.0 | 28/09/2015 | Initial draft sent to GCIO, R9, ForgeRock and MoE for comment |
| 1.1 | 13/10/2015 | Section 1.0 Summary and 6.0 Findings added |

Distribution List

This table describes the people who have received this document.

| Name | Title | Group/Team |
| --- | --- | --- |
| James Collier |  |  |
| Colin Wallis |  |  |
| Bill Young |  |  |
| Michael Wilson |  |  |
| Allan Foster |  |  |

Document Review and Sign-off

Document Review

This table lists the review parties and status of the review.

| Name | Title | Review Status |
| --- | --- | --- |
| Stephen Carter | Project Manager | Choose an item. |
| Bill Young | Identify Architect | Choose an item. |
| Patrick Rogers | Programme Manager | Choose an item. |

Quality Assurance

Please have this document signed off, then PDF/scan and store the final version. Please ask the EPMO for advice on specific names. This list may differ depending on oversight and governance components. Customise as appropriate.

This table lists those who have endorsed this document.

| Role | Endorsement | Signed |
| --- | --- | --- |
| Business Owner (Senior User) | I confirm that | Date: |
| [Name]  [Job title and business area] |

Approval

This table lists those who have approved this document.

| Role | Endorsement | Signed |
| --- | --- | --- |
| Sponsor (Project Executive) | I approve this document. | Date: |
| James Collier  Government Chief Architect, DIA |

Document Contributors

The following people authored or contributed to a section of this document:

| Name | Title | Contact Details |
| --- | --- | --- |
| Bill Young | Identify Architect – Findings section | 'Bill.Young@education.govt.nz' |
| Jamie Cavanaugh | Technical Lead – Technical Build and Architecture sections | Jamie.Cavanaugh@datacom.co.nz |
| Lana Traut | Business Analyst / Scrum Master – Demonstration section | LanaT@datacom.co.nz |
| Jana Crighton | Business Analyst / Scrum Master – User stories | [JanaC@datacom.co.nz](mailto:JanaC@datacom.co.nz) |
| Ata McGregor | Senior Business Analyst – Overview and Requirements Sections | [Ata.McGregor@mbie.govt.nz](mailto:Ata.McGregor@mbie.govt.nz) |
| Allan Foster | VIP ForgeRock - – Findings section | Allan.foster@forgerock.com |

Contents

[Project Closure Report 1](#_Toc432686287)

[1. Summary 6](#_Toc432686288)

[1.1. Background 6](#_Toc432686289)

[1.2. PoC Results 6](#_Toc432686290)

[1.3. Access Control vs Delegation vs Relationship 6](#_Toc432686291)

[2. Recommendations 7](#_Toc432686292)

[3. Overview 8](#_Toc432686293)

[3.1. Document Purpose 8](#_Toc432686294)

[3.2. History behind the Proof of Concept 8](#_Toc432686295)

[3.3. What is a Delegation? 9](#_Toc432686296)

[3.4. What is User Managed Access (UMA)? 9](#_Toc432686297)

[3.5. An Agile Approach to the Proof of Concept 9](#_Toc432686298)

[3.6. Resourcing 10](#_Toc432686299)

[4. Requirements Covered 11](#_Toc432686300)

[4.1. Requirements Elicitation Process 11](#_Toc432686301)

[4.2. Coverage 12](#_Toc432686302)

[5. Findings 17](#_Toc432686303)

[5.1. Some Background 17](#_Toc432686304)

[5.2. Results: What worked well 17](#_Toc432686305)

[5.3. Results: What Worked, But Could be Better 18](#_Toc432686306)

[5.4. Results: What Wasn’t Tested 19](#_Toc432686307)

[5.5. Architectural and Design Considerations 19](#_Toc432686308)

[5.6. Conclusion 22](#_Toc432686309)

[5.7. Recommendations 22](#_Toc432686310)

[5.8. Options Going Forward 23](#_Toc432686311)

[5.9. Recommendations 23](#_Toc432686312)

[6. Lessons Learnt 24](#_Toc432686313)

[7. Glossary 25](#_Toc432686314)

[8. Appendix 1: PoC User Stories 28](#_Toc432686315)

[9. Appendix 2: Technical Description PoC Build 34](#_Toc432686316)

[9.1. Introduction 34](#_Toc432686317)

[9.2. High Level Design 35](#_Toc432686318)

[9.3. Identity and Authentication 41](#_Toc432686319)

[9.4. Delegation Considerations 46](#_Toc432686320)

[9.5. Rules and Relationships 56](#_Toc432686321)

[9.6. Implementation Considerations 57](#_Toc432686322)

[9.7. OpenAM Defects 61](#_Toc432686323)

[10. Appendix 3: PoC Demonstrations 63](#_Toc432686324)

[10.1. Immigration demonstration 63](#_Toc432686325)

[10.2. Health demonstration 70](#_Toc432686326)

[10.3. MoE (Careers & NZTA) demonstration 76](#_Toc432686327)

[10.4. MoE (Early Childhood) demonstration 82](#_Toc432686328)

[11. Appendix 4: R9 Delegations 90](#_Toc432686329)

[11.1. Summary 90](#_Toc432686330)

[11.2. Delegations Framework 90](#_Toc432686331)

[11.3. Priority areas mapped to Framework, Requirements and PoC 91](#_Toc432686332)

[12. Appendix 5: Reference Architecture 95](#_Toc432686333)

[12.1. Decentralised 95](#_Toc432686334)

[12.2. Centralised 96](#_Toc432686335)

## Summary

### Background

As an outgrowth of the Result 9 Delegation Capability Project, the office of the GCIO elected to run a Proof of Concept (PoC) project to test the concept of a Common Delegations Capability for shared government use.

Specifically, the PoC project was charged with testing the recently approved UMA (User Managed Access) standard/protocol to:

* Determine if it was fit for purpose to meet the government’s common delegation requirements
* Provide evidence of UMA’s applicability as a common delegation standard
* Determine the potential of a particular implementation (ForgeRock’s OpenAM) of the UMA 1.0 standard

This PoC was attempted by implementing solutions to 4 distinct scenarios contributed by agencies which had been participating in the Delegation Capability Project. In general, the scenarios were based on existing business problems. The last scenario (Education’s child pick-up) was more speculative in nature and was chosen to test specific concepts not addressed by the previous scenarios. In all, the PoC directly tested the top 15 of the 18 priority areas identified by the Delegations Capability Project working group. A good understanding of the remaining 3 areas was obtained even though they were not tested in the PoC.

### PoC Results

The scenarios implemented by the PoC project clearly demonstrate that UMA is a valuable tool when attempting to allow a member of the public to manage “delegations” or access control. That statement comes with significant caveats however.

#### Completeness

The UMA protocol solves the specific problem of a user self-managing access to their resources. Online government services can be expected to frequently need additional access management controls related to these resources. For instance an agency’s business rules prevent access under certain circumstances or when legislation specifically allows access to that resource.

*Agencies consuming a Common Delegation Capability service are almost certain to need access control systems in addition to what is provided by the common service.*

### Access Control vs Delegation vs Relationship

The UMA standard/protocol is designed to give an individual the ability to manage access control of their information resources. Because a delegation generally includes access privileges, we casually equate access with delegation. Similarly, delegations are a specific type of relationship. Some of the scenarios tested in the PoC and the uses identified in the Snapshot Report clearly were more oriented towards relationships than delegations or access control. UMA does not manage relationships in general.

*A well structured approach, above and beyond UMA, will be required to manage complex or abstract delegations and relationships.*

#### Cross Government Delegations

One of the benefits identified in the Snapshot Report of a Common Delegations Capability is the ability to create a suite of delegations across several government agencies in a single act. The standard UMA model would only facilitate a user setting up delegations independently at each agency.

*Parties wishing to implement a collection of delegations across agencies would need a bespoke solution to do so. Participating agencies would need to implement features to support this mode of operation.*

#### Inferred Delegations

An inferred delegation would allow a delegation set up for one purpose to be re-used by different parties in different contexts. The UMA protocol is not designed to behave in this fashion. Bespoke solutions would be required to implement a scheme such as this. It is unclear how a user would understand the eventual extent of a delegation.

There has been no existing policy work identified which would indicate that inferred delegations are desirable.

*Attempts to reuse delegations outside of their original scope should be discouraged until a sound policy base supports the idea.*

#### RealMe

The PoC Delegation Service used RealMe to provide authentication. Because of the conservative privacy approach of the RealMe service, simple things such as associating a name or email address with an individual were problematic and resulted in poor user experience. The addition, in the PoC, of new RealMe services to allow a user to record his/her own personal information and a list of contacts significantly improved the user experience. The use of RealMe in many Result 9 or Result 10 scenarios would also benefit significantly from these services.

These services are not required to be part of RealMe and could reside within a Delegation Service itself.

*An investment in new RealMe services, such as the ones implemented in the PoC, should be encouraged.*

#### Velocity of Change in the IAM Space

UMA is a new standard and will evolve in the near future. Other paradigms for more explicitly managing relationships are forming. Identity Relationship Management (IRM) is under development by the same standards body responsible for UMA (the Kantara Initiative). The state of the art in access control and relationship management can be expected to move quickly over the next 5 years.

*Investments in Access Control and/or Delegation capability should anticipate change in the 5 year timeframe and emphasise flexibility, simplicity and portability.*

#### Policy

A significant number of questions with policy implications were uncovered during the PoC. Answering these questions was outside the scope of the PoC and ‘convenient’ answers were generally assumed. Examples include: Can delegations be re-delegated? How long should a delegation last? Must a delegation be accepted by the delegate?

*Prior to implementing a Common Delegation Capability, the numerous policy questions must be resolved.*

## Recommendations

Assuming the continued advancement of a Common Delegations Capability is desirable, a number of steps should be taken towards this end:

1. Policy questions related to delegations and relationships should addressed by participating parties. The potential impact on end-users could indicate a need to include public consultation in this process.
2. Advance the technical work of the PoC towards a more complete prototype solution. In particular, test the cross-agency scenario identified in the Snapshot Report.
3. Develop a structured approach and architecture for dealing with relationships and complex or abstract delegations.
4. Encourage the development of new RealMe services to allow a user to record and assert personal information and personal contacts.
5. Identify and participate in emerging relationship management paradigms such as Kantara’s IRM to ensure New Zealand government requirements are addressed.

## Overview

### Document Purpose

The purpose of this document is to communication what was discovered in the delegations proof of concept so that informed decisions can be made about delegation systems and standards.

Specifically, the proof of concept attempted to answer the question “Is it viable to produce a delegations service using an UMA protocol”. The need to answer this question came from a number sources across NZ government and was captured by the work of GCIO, Result Area 9, Result Area 10 of public better services programmes.

This document covers:

* **The rationale for what was covered**: explaining what was investigated in the proof of concept and why it was included
* **A record of what was done**: of the scenarios covered, of what was built, of what was demonstrated
* **Considerations and learnings** : to help others who are involved in designing a delegation solution
* **Findings and recommendations**: to show what the key things found and make recommendations going forward.

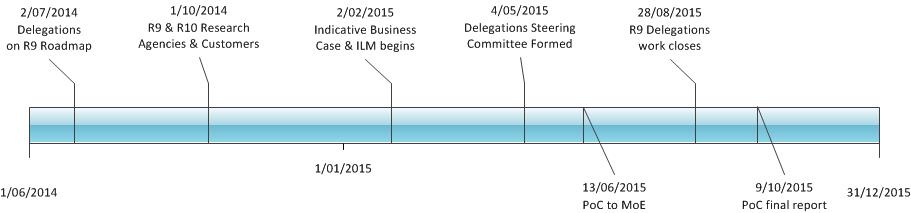
### History behind the Proof of Concept

The ‘Develop Delegated Access Capability’ was a deliverable on the Better for Business Result 9 2014 Roadmap. During the latter half of 2014 the project engaged with customers and agencies, to understand customer’s problems with delegating their responsibilities to third parties when dealing with government. The project formed a working group of representatives from five of the Result 9 agencies (Statistics NZ, ACC, MBIE, IR and MPI) as well as other interested groups within DIA: the GCIO, R10 Federated Service Delivery and RealMe. This was subsequently expanded to include representatives of Ministry of Education and Ministry of Health who were actively exploring the management of customer delegations within areas of their responsibility or oversight.

In February 2015 the R9 Steering Committee agreed that the project should deliver an Indicative Business Case (IBC). The project team then conducted workshops with both agencies and customers to agree an Investment Logic Map (ILM). The Project team developed a comprehensive set of delegation scenarios and subsequently a draft set of high level requirements.

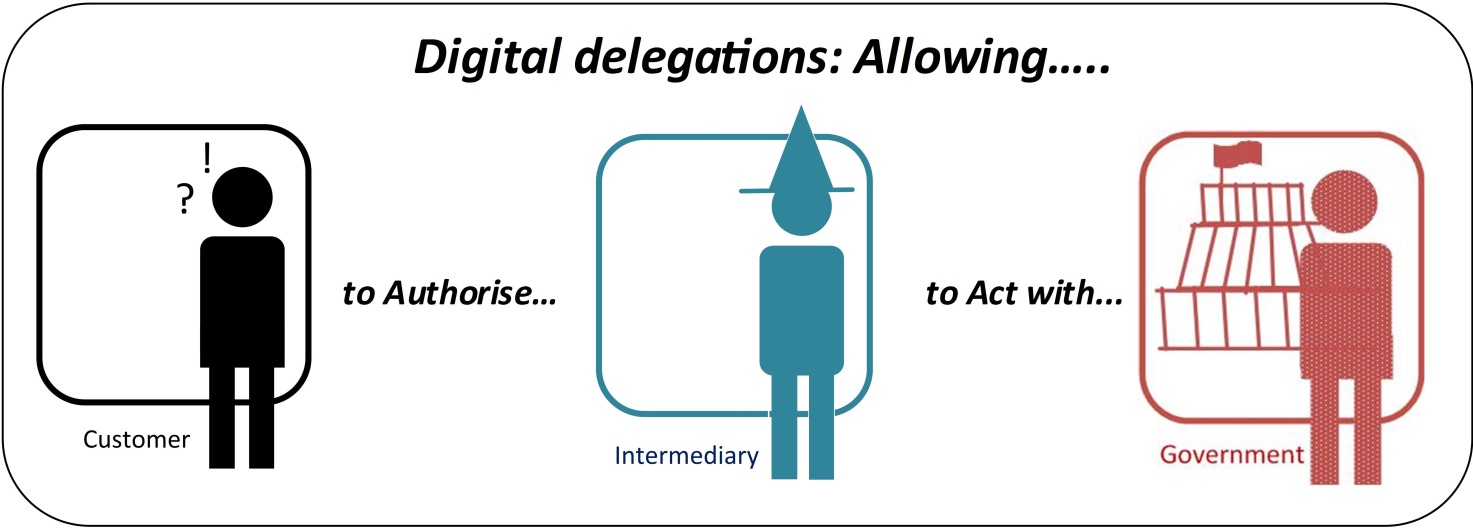
In May 2015 a Delegation Steering Group was formed to reflect the range of interest in a delegation capability broader than the Result 9 agencies. The Steering Group was chaired by the Statistics representative of the R9 Steering Committee.

The project also worked with DIA (GCIO) to manage the delivery of a Proof of Concept (PoC) to test a potential technology standard (UMA – User Managed Access) and related technologies for delivering cross-government delegations. This was intended to help to make real the level of shared capability that would be useful to an interested group of agencies. The R9 Steering Committee directed this work to be handed back to DIA in June 2015, who then partnered with the Ministry of Education to complete the PoC.



### What is a Delegation?

Many businesses and individuals get others to act on their behalf when dealing with government and private sector organisations. For example, a business appoints an accountant to deal with Inland Revenue, ACC, their bank and frequently other agencies as well. An adult daughter may deal with a range of government and health services on behalf of an aging parent. Both businesses and citizens do this because they don’t have the time or skills or, for some individuals, the capacity to do so.

****

### What is User Managed Access (UMA)?

User-Managed Access (UMA) is an OAuth-based access management protocol standard. This standard is suited to delegations as the protocol specifies how a resource owner (that is a delegator) is able to control the authorisation of access to their online services to a requesting party (that is a delegate).

### An Agile Approach to the Proof of Concept

This proof of concept investigated digital delegations against the User Managed Access (UMA) standard, using emerging technology from ForgeRock against a variety of scenarios and capabilities needed by the NZ government

The Proof of Concept was built using an Agile development methodology by MoE, with funding from DIA, utilising resources from both ForgeRock and Datacom. Demonstrations of the Proof of Concept Sprints were presented to a cross government working group through the development cycle. This document records the findings of this Proof of Concept work and is a key deliverable along with the environment and software to Government’s Chief Architect.

A description of the Minimal Viable Product (MVP) was created to define the objectives of the PoC. The following MVP statement was created for the Proof of Concept:

***“A set of demonstrations and documentation to show the Delegations Steering Committee how selected Use Cases that are representative of common government delegation patterns can be implemented using UMA technology, highlighting architectural components, processes, limitations and policy constraints.”***

The PoC was delivered using an Agile approach in six, two week sprints. The first was for technical set up, and the last sprint was for tidy up.

The main sprints were:

* **Immigration:** *Someone overseas wants help to immigrate to New Zealand*
* **Health:** *A patient asks their family member to help them do things on their patient portal*
* **MoE (Careers & NZQA)**: *A job seeker shares their NZQA Record of Achievement (ROA) with a potential employer*
* **MoE (Early child)**: *In the future parents can delegate the pick up their children from school*

The Epics, User Stories and Acceptance Criteria for each sprint along with the demonstrations are shown in the [Appendix 1](#_Appendix_1:_PoC).

### Resourcing

The following team was used to deliver the Proof of Concept build over 6 fortnightly sprints. :

* 0.5 Project Manager
* 0.5 Architect
* 0.25 Senior Business Analyst
* 1.0 Scrum Master/BA
* 1.0 Technical Lead
* 2.0 Developers
* 0.5 Test Manager/analyst
* 1.0 Test Analyst

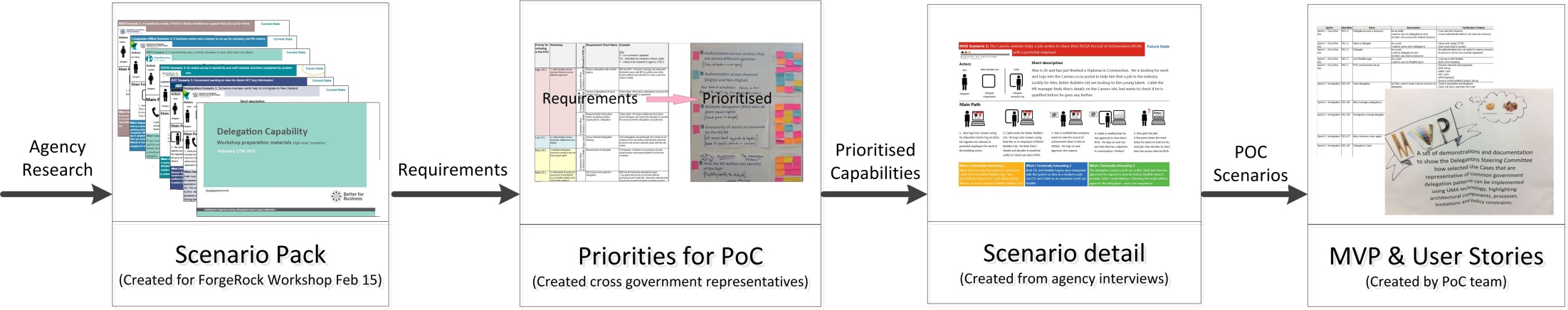
In addition to these project resources the Proof of concept was also supported by:

* The Delegations working group who attended fortnight demonstrations
* Business owners who provided guidance on the scenarios related to their agency
* Forge Rock who provided Input and advice

## Requirements Covered

### Requirements Elicitation Process

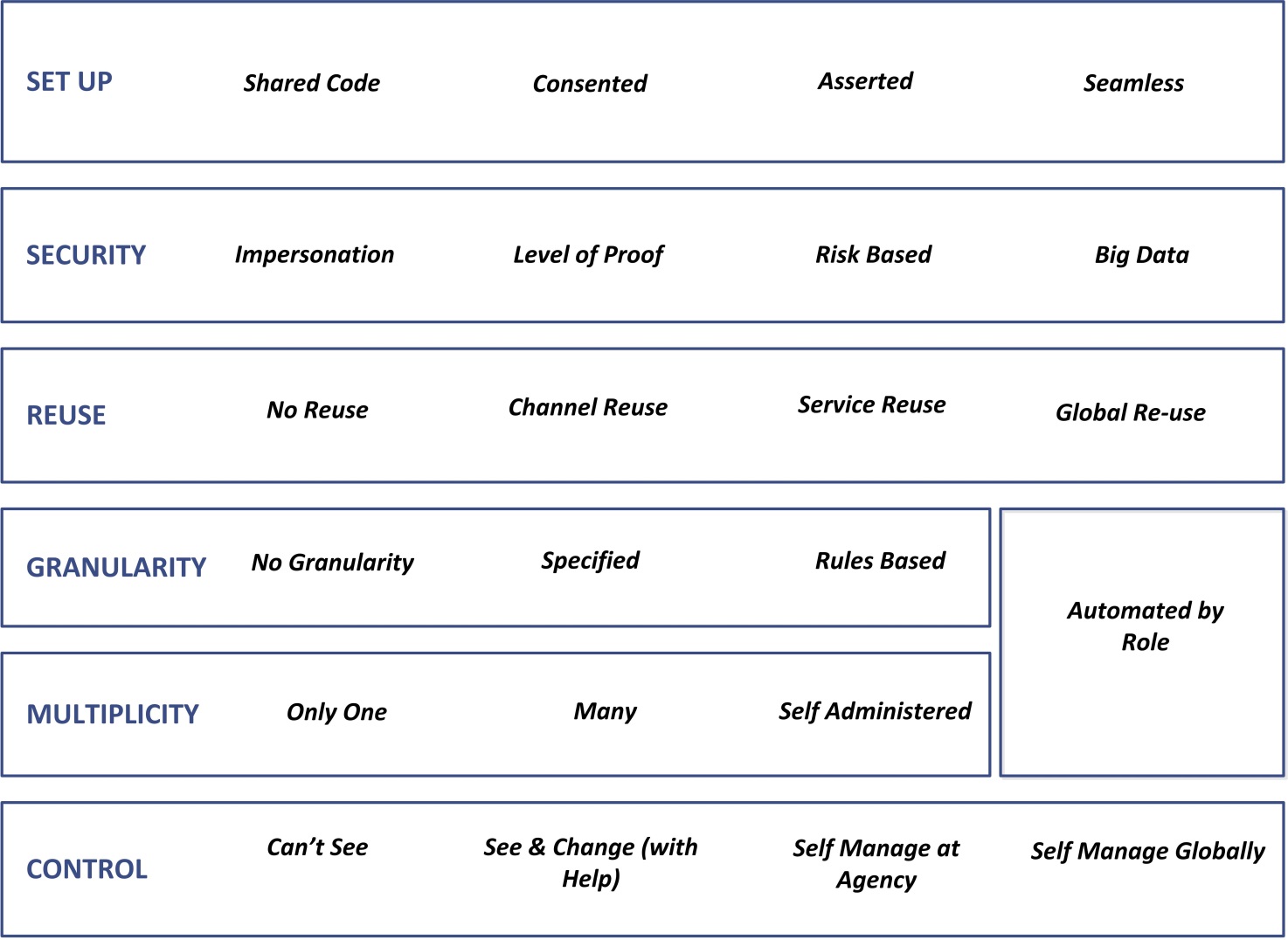
The requirements for the proof of concept evolved out of the R9 Delegations cross agency work. The elicitation process is described below:



1. **Scenario Pack**: Research with agencies led to a set of scenarios that were used in a DIA workshops with ForgeRock in February 2015. These identified a set of initial requirements that were combined with the requirements identified by DIA R10 work.
2. **Priorities for Proof of Concept**: A cross agency working group identified and prioritised what needed to be tested in a Proof of Concept.
3. **Scenario detail**: Further work with Ministry of Health (MoH), Immigration and Ministry of Education (MoE), identified a better understanding of four specific scenarios to be used to test the priority items.
4. **MVP & User Stories**: The Proof of Concept team identified a description of the Minimum Viable Product (MVP), user stories and sprints.

### Coverage

The R9 Delegations project identified a framework of key considerations for government delegations. The areas investigated by the Proof of concept are highlighted below:



**Note**: The [Appendix](#_Appendix_4:_R9) gives a fuller description of this framework.

**KEY Built in PoC In report**

The areas prioritised for the Proof of Concept are shown, against the R9 framework, requirements and way they were investigated by the PoC in the table below:

[**Key:** Red=High, Yellow Medium, Blue = Low]

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority Areas**  **Source**: Agency Workshops | **Requirement Name**  **Source**: Agency scenario pack | **Investigated in PoC by...**  **Source**: Jira Proof of Concept | **R9 Framework**  **Source**: R9 Findings |
| 1. Authorisation across services that are across different agencies | Share a delegation with another agency | ***Sprint 4*** *Careers* A delegations created through the Careers website is used at NZQA | **REUSE**  ***Service Reuse*** |
| Assess a delegation set up at another agency. |
| Request further information before accepting another organisations delegation. | Not demonstrated |
| 2. Authorisation across channels (digital and non-digital) | Cross Channel Delegation Sharing | ***Sprint 5*** *MoE*: A delegation set up on a parent portal creates a QR code that the childcare centre can read to confirm who the pickup person is when they arrive at the childcare centre | **REUSE**  ***Channel Reuse*** |
| 3. Multiple delegators (resource owners) who all have equal rights | Shared power to delegate | ***Sprint 5*** *MoE* Both mum and dad can set up and manage delegations for their child. Refer to [3.7 Equal Ownership](file:///C:\Users\McGreggora\Desktop\ProjectClosureReport_PoC%20Delegations(1).docx#_Equal_Ownership) | **MULTIPILICITY**  ***Many*** |
| 4. Granularity of access to resources by the RS/AS (e.g a health portfolio and part of that portfolio) | Set access level against a delegation | ***Sprint 2*** *Immigration*: (Granularity controlled by agency - not delegations system). | **GRANULARITY**  ***Specified*** |
| Access to specific products or services | **Sprint 3** *Health*: Access could be controlled to just notes, or just blood sugar alerts. |
| 5. How the RO defines the identity of the RqP | How to identification digitally who you are delegating to | ***All Sprints***: Use of a rendezvous code. Refer to section [3.3.1 Discovering Users to Share With](#_Rendezvous/Invite) | **SET UP**  ***Consented*** |
| 6. How the RO is managed when it is missing/digitally absent | Assume control when missing digital identify | ***Sprint 2*** *Immigration:* Immigration advisor is set up as the delegate of their client as part of the application creation process. | **SET UP**  ***Seamless*** |
| 7. Managing a group membership claim by an RO or an RqP (outside UMA, but will need to couple with UMA solution– probably on paper) | Authorisation through licensing, registration or membership of a professional body. | ***Not demonstrated– See Findings section*** | **SET UP**  **Asserted** |
| 8. How to present a global view (i.e not a service based view) of a customer’s authorisations, that is manageable to the user | View delegations | ***All Sprints: Delegator could log on to the Delegation system to view.*** | **CONTROL**  **Self-manage global** |
| **Priority Areas**  **Source**: Agency Workshops | **Requirement Name**  **Source**: Agency scenario pack | **Investigated in PoC by...**  **Source**: Jira Proof of Concept | **R9 Framework**  **Source**: R9 Findings |
| 9. How RqP self asserts an authorisation to act (access Resources) without RO consent. Assuming, the RqP is form a professional group trusted by the RS. | Authorisation through licensing, registration or membership of a professional body. | Not demonstrated – [See Findings](#_Findings) section | **SET UP**  ***Asserted*** |
| 10. RqP can sub-delegate to another, including for both asserted or granted authorization. | Delegate to a known business (and known person) | ***Sprints 2***  *Immigration:* advisor to assistant | **MULTIPILICITY**  ***Self-Administered*** |
| Delegate to a known business (but unknown person) | ***Sprint 4***: *Careers*: Business to staff |
| Set up a delegation for a role within a business | ***Sprint 4***: *Careers*: Business to staff |
| Sub-delegate to an another | ***Sprint 5:*** MoE -Mum to grandma to grandpa |
| 11. Client is not created by government and is provided by a 3rd party (i.e. Xero web portal, Apple mobile app, Fit bit or blood sugar device). | Transact through third party software | ***Sprint 3*** *Health*: Both the patient portal and the blood sugar monitor are created by third party vendors. | **REUSE**  ***Channel Reuse*** |
| 12. Gateway for legacy and Greenfields services | Legacy systems | This was not investigated during the proof of concept. Possible approaches to supporting legacy services are:   * Deploying policy access points (i.e. agents) in front of legacy applications * Providing plugins for existing access control systems   The best approach will depend on the amount of change that can be made to legacy applications  For greenfield solutions, integration libraries and collateral (such as integration documentation and checklists) may wish to be provided to ensure smoot on boarding. | Agency specific Implementation issue (Not in framework). |
| 13. Hold RqP request/claim until verified (i.e RO approves or legal authority is recognised) | Delegation held until user consents | ***Sprint 3*** *Health*: Son’s request to view blood sugar is held until mother approves  ***Sprint 4*** *Careers*: Employers request to view Record of Achievement is held until approved. | **SET UP**  ***Consented*** |
| Authorise a delegation with legal proof | Not demonstrated |
| 14. RO is notified only after the RqP has set themselves up with access. | Confirm delegation after it is set up | Not demonstrated – [See Findings](#_Findings) section | **SET UP**  ***Consented*** |
| **Priority Areas**  **Source**: Agency Workshops | **Requirement Name**  **Source**: Agency scenario pack | **Investigated in PoC by...**  **Source**: Jira Proof of Concept | **R9 Framework**  **Source**: R9 Findings |
| 15. RO revokes a policy for a specific RqP, when it was established by the RqP | None identified | ***Sprint 2*** *– Immigration:* Immigrant (who did not set up application) can take ownership of it back from Advisor. | **CONTROL**  ***See & Change*** |
| 16. RS can restrict the number of RqP who can be authorized at any one time, including multiple or unlimited sub-delegates of RqP. (for example IR only allows one accounting firm to act for a RO at any time). | Control how many people can be delegated to concurrently | This was not investigated during the proof of concept. Challenges are:   * How the RS specifies the constraints per resource (does the OAuth 2.0 Resource Registration Set allow additional parameters) * How the AS enforces these constraints. OpenAM may not support these constraints and customisation or workarounds may be difficult.   Note that determining the number of parties that have access can be difficult if delegations can be re-shared, or access is based on relationships or other rules. See [Rules and Relationship](#_Rules_and_Relationships) section. | **MULTIPILICITY**  ***Many*** |
| 17. Expiration of an authorization. (RO set policy/rules and RS set policy/rules). | End delegation at specific time or date | Not demonstrated – refer to  [See Rules and Relationship](#_Rules_and_Relationships) section | **GRANULARITY**  ***Rules*** |
| End a delegation after staff changes |  |
| End a delegation due to age |  |
| Event based delegation |  |
| Delegation Disputes |  |
| 18. Temporary/ Emergency access. Have different rules during an ‘emergency’ period. (e.g initial period during a serious injury claim at ACC or temporary foster care of a child at MSD) | Get access to act for another without their consent in an emergency | This was not investigated during the proof of concept. Options include:   * Implementation on the service agency. In other words, don’t use UMA if the user requests emergency access, have the service agency decide and allow access. * Implementation in the Authorisation Server. This might be done with an additional claim from the Client the AS uses during policy evaluation. From the RS point of view, the request is the same as usual. * Implementation on the service agency, but auditing access using an API on the delegation service.   Things to consider to include auditing and visibility of emergency access to the user, availability in emergencies when the service is unavailable, who determines when emergency access is required. | **GRANULARITY**  ***Rules*** |

## Findings

### Background

#### Semantics

It’s worthwhile to note some things about the language that has been used in the Delegations PoC. This PoC is an outgrowth of the Delegation Capability Project. That project was intended to progress the deliverable identified on the Result 9 Roadmap; ‘Develop Delegated Access Capability’. Throughout the PoC, the “thing” that gets managed by UMA has been variously described as ‘access’, ‘delegation’, ‘relationship’ or ‘authorisation’. In many respects, this interchangeability is harmless. But the reader should remember that

* Access is a by-product of delegation. But access and delegation are not equal.
* Delegation is a particular type of relationship. But not all relationships are delegations.
* Most ‘real-world’ delegations have a very specific context in which the delegation applies.

UMA is a protocol and a standard designed for access control. Equating UMA’s notion of access control with generalised delegation requires a level of abstraction that does not exist in the UMA standard/protocol. This level of abstraction must be dealt with by another element of the delegation system.

#### Assumptions

A number of assumptions were made in the execution of the PoC. These include:

* The Delegation Service would use RealMe for authentication.
* User consent would not need to be repetitive or invasive.
* Privacy and policy issues would have “reasonable” resolution in line with existing services such as RealMe

### Results: What worked well

#### Simple Delegation

The simple scenario of a user creating a single delegation or access privilege using UMA worked simply and in a straightforward manner. The delegation process worked in the most straightforward fashion when the resource owner pre-emptively set up the delegation. This is the central UMA use case.

#### Requesting Party Initiated Delegation

The scenario when a delegation was requested by the would-be delegate (Requesting Party Initiated Delegation) also worked well. Although the user experience is straightforward, considerations would need to be taken because UMA uses patterns that may not be common to many users; i.e. getting an email request from someone to create a delegation.

#### Simple Delegation

The PoC scenario that best matched an UMA solution was the ‘Careers’ scenario. This is because the Client (Careers application) and Resource Server (NZQA) were distinct systems owned by different parties. In the other PoC scenarios, the Client and Resource Server were either the same application or owned by the same party. While UMA can work well in the ‘consolidated RS and client’ situation, the protocol’s strength lies in its ability to deal with distinct RSs and clients.

#### Managing Delegations

The PoC employed two different methods to allow a user to view and manage the delegations they made. The first was centrally, from the Delegation Server itself (initially using the OpenAM Dashboard, but later using a custom User Interface). The second method was from the Resource Server (agency application). While a centralised dashboard provides a global view of delegations, the view from the Resource Server is probably more meaningful to most users.

### Results: What worked, but could be better

#### Constraints Related to Use of RealMe

There are constraints around the use of RealMe Logon service that impact the usability of a Common Delegation Capability.

The policy/privacy backdrop of the RealMe Logon service includes the provision that users should not be asked for their RealMe Logon username. The PoC overcame this obstacle with the use of “Rendezvous Codes”, which are emailed to the intended delegate. However, the use of these codes is less secure than desired.

RealMe Logon service does not provide any identifiers that are meaningful to a person; they are large hexadecimal numbers. This means that a person cannot look at the delegations they’ve created without seeing a meaningless user identifier.

The Delegation Service implemented in the PoC had no way of sending a notification email to users in some circumstances.

In order to give PoC users an acceptable user experience, with useful identifiers, two hypothetical services were added to RealMe: the RealMe Attribute Service (for self-asserted attributes) and the RealMe Trusted Contact Service (to allow a user to create their own list of trusted contacts).

With these services in place, the PoC was able to convey meaningful information about delegations and send notifications.

An alternative to a RealMe Attribute Service would be for the Delegation Service itself to collect information. This was viewed as unacceptable because it creates an additional store of user information that must be protected and maintained.

#### Re-Delegation or Chaining

One of the capabilities identified as useful in the Snapshot Report was the ability to re-delegate or chain delegations. Putting aside the policy questions about the desirability to perform this activity, there were issues around the implementation of chaining.

When an existing delegate wishes to re-delegate and create a delegation chain, the process works as expected and the chained delegate is given access. In cases where Requesting Party Initiated delegations take place, the process is less satisfactory. In this case, the resource owner receives a request to establish a delegation, but without any means to understand the relationship between the existing delegate and the new delegate.

Another concern with re-delegation has to do with lack of understanding that a re-delegation is being used. The UMA authorisation server makes a binary evaluation about whether an individual should be given access to a resource. The fact that the access is the result of a chain of delegations is not passed to the resource server. Neither is the chain of delegation itself. This information is likely to be useful from an audit perspective, possibly needed for the execution of business rules on the resource server and also as part of a forensic trail.

Finally, in situations where a delegate has re-delegated an access permission to a different user, the original resource owner has no knowledge of the re-delegation.

#### Impersonation

In one of the scenarios tested (Immigration), we implemented “Carol acting on Bob’s behalf” as a form of chained delegations. The end result was a case of impersonation, where the Authorisation Server thought Carol was actually Bob and granted access on that basis.

Impersonation is one of the specific behaviours the Common Delegations Capability Project was attempting to eradicate.

#### Self-Asserted Delegation

One the scenarios tested (Immigration) involved a person asserting a delegation to himself from a third party who plays no active part. While this scenario parallels an existing business process, it is not a natural fit for the UMA protocol. The active involvement of the user (resource owner) is required in the UMA protocol.

### Results: What wasn’t tested

#### Delegation Reuse: Cross-agency delegations

The type of cross-agency delegation as pictured in the Snapshot Report is somewhat problematic in an UMA environment. UMA allows a user to define an access control ‘tuple’ between a resource (or a group of resources) at a single resource server (agency application), an authorisation server and the delegate. Creation of this ‘tuple’ requires participation of both the resource server and authorisation server.

As it is not possible to copy, share or reuse this ‘tuple’ between resource servers at different agencies. The reuse identified in the Snapshot Report, would require multiple access control ‘tuples’ to be created.

It would be possible to implement a centralised tool to create multiple UMA delegations. But it would best be confined to specific, well-defined scenarios (such as starting a new company) and would require specific support to be added to resource servers.

#### Delegation Reuse: Inferred Delegations

In this instance, an existing delegation from a resource owner to a delegate for one specific resource would be used to infer a delegation to a different resource, possibly held at a different resource server.

Putting aside questions about the desirability of this behaviour (of which there are many), this situation is similar to the cross-agency delegation situation. A delegation ‘tuple’ cannot be copied or reused in any way.

It may be possible to re-define an existing resource set to change the scope of a delegation. This has not been tested and would only work within a single resource server.

#### Time-Bound Delegations

One of the scenarios was meant to include time-bound delegations. This was not tested as the ForgeRock UMA API did not allow detailed manipulation of the OpenAM policy settings.

### Architectural and Design Considerations

#### Where UMA fits Best

As UMA is specifically designed to give the user (resource owner) control over his/her own information, it is ideal when user control is a requirement in the access control over resources.

The UMA model works best when the “protected resource” is just a simple resource, rather than an abstract thing. In particular, UMA excels at protecting a user’s resource that is made available via a RESTful API.

#### Legacy Systems

UMA could easily act as a binary access control mechanism for legacy systems through the use of an UMA enabling proxy gateway. But the utility of that method would be limited because the legacy systems are likely to have existing provisioning and access control mechanisms which would still have to be satisfied. Legacy systems which are already adapted to sitting behind a proxy gateway to provide access control may be better candidates for using an UMA delegation service, but would have to be evaluated on a case-by-case basis.

#### Mixing UMA with existing 3rd party Access Control Products

Existing web applications could potentially be working with 3rd party access control products. Such systems often work by injecting or modifying HTTP headers, cookie manipulation, etc.. An UMA enabling proxy gateway, as identified above, may also have similar behaviours and create a chain of HTTP manipulation. Many 3rd party products have exhibited sensitivity and instability when used as part of such chains.

#### PEP and PDP Proliferation

The PoC tested using UMA to protect a wide variety of resources, from simple information contained in RESTful web services to more abstract resources such as “authority to act”.

Using UMA as the sole protection of simple resources is straightforward and raises no concerns. As the protected resource becomes more complex, there is a concern about proliferation of Policy Enforcement Points (PEPs) and Policy Decision Points (PDPs).

In the situation where a business application has its own access control rules it is acting as both a PDP and PEP. If that application is also protected by a conventional Web Access Management (WAM) tool an additional PDP and PEP are introduced.

From the perspective of security, it is problematic to have too many PDPs and PEPs. While the “more security is better” rule might seem to apply, a multiplicity of PDPs and PEPs actually makes the end-to-end solution more difficult to understand.

A partial solution to this problem is to consolidate the UMA functionality of a Resource Server and a WAM solution into a single element.

#### Resource Complexity

As “delegation” becomes more abstract and less about direct access control, the bespoke PDP/PEP functionality becomes easily intertwined with business logic and functionality. A lack of separation between “access control functionality” and “business functionality” is a source of potential problems including: security, scalability (in terms of complexity, not performance) and proprietary lock-in.

When more complex and abstract resources are protected by the Delegation System, more structure and implementation discipline will be required to keep business logic and access control logic distinctly separated.

#### Relationships

In many scenarios, the relationship between entities is central to the delegations. In the “Careers” scenario, the relationship between the building company and its employee is required to create the subsequent re-delegation of authority to the employee. UMA itself does not understand or manage relationships. An additional layer of business logic is required in the Delegation Service or the resource server to deal with relationships. Where such relationships should persist is not obvious.

#### Client Systems

Any client system that wishes to access UMA protected resources must be UMA aware and compatible. This is not a large technical feat, but it is a requirement. It would be useful, if providing a Common Delegation Capability based on UMA to provide a set of reference client implementations to facilitate uptake.

#### Multiple Resource Owners

The current UMA 1.0 specification allows a single resource owner. In situations where multiple resource owners are required, an additional level of abstraction is required. While the PoC demonstrates this is clearly possible, such abstractions increase the complexity of the Resource Server with bespoke functionality. Complexity and bespoke behaviour are generally considered undesirable qualities in access control solutions.

#### Non-Person Entities

When one of the parties in an UMA protection scheme is a Non-Person Entity (NPE), additional complexity is required. Since the NPE cannot, presumably, act on its own, one or more individuals need to have authority to act on the NPE’s behalf. This indicates the need for an abstract resource (authority to act) to be protected and some business rule execution to bestow privileges to the individual(s).

#### Time-Bound Delegations

It is not obvious the best way to implement time-bound delegations. The PoC wished to implement the time constraints in the OpenAM policy settings. It could be equally valid to execute time-dependent workflows on the Delegation Server to create/delete delegations. Or, this could simply be considered an additional facet of Access Control that cannot be managed by UMA and should be controlled by a different Access Control system.

#### Centralised vs De-Centralised Deployment

There are three general options for deploying delegation capability:

|  |  |
| --- | --- |
| Deployment Model | Advantages/Disadvantages |
| Agencies build their own delegation capability in a completely de-centralised model. | Simplest model to begin implementation. Highly probable to encounter difficulties when attempting cross-agency services. Duplicated investments across government. |
| Semi-Centralised, grouped by sector. | Would likely meet requirements identified for cross-agency delegation reuse. Less duplicated investments across government. Use of delegation unlikely to be contained within a sector, requiring complex interoperability between sector specific delegation systems. |
| Centralised. | Allows unified view for all of a user’s delegations. Eliminate duplicate investment. Initially more complex to implement with policy work front-loaded. |

#### Policy and Privacy

One of the goals of the PoC was to identify possibly policy related questions for a Common Delegation Capability. Policy questions encountered in the course of the PoC included:

* Should delegations be sub-delegable? If so, under what conditions?
* Do delegations need to be accepted by the delegate (do they form a contract)?
* To what extent is the delegator responsible for the delegate’s actions?
* Can there be a single set of Terms and Conditions for all delegations, or are they specific to the particular delegation?
* How are delegations for a NPE bootstrapped?
* Are there activities where delegated authority does not meet legislated requirements?
* If inferred delegations are allowed, what transparency is required for the resource owner?

It has been assumed that a significant policy effort would be required prior to deploying a Common Delegation Capability.

#### Velocity of change

UMA 1.0 is a new standard and will evolve in the near future. There are other paradigms in development for explicitly expressing and managing relationships. Identity Relationship Management (IRM) is under development by the same standards body responsible for UMA (the Kantara Initiative). It is not clear today how these two standards will relate to each other.

One this is clear; the state of the art in access control and relationship management can be expected to move quickly over the next 5 years. Investments made in the short term should anticipate rapid changes.

### Conclusion

#### Is UMA fit for purpose?

The question remains; is UMA fit for purpose as the basis of a Common Delegation Capability in NZ?

An UMA based delegation service clearly allows more user-centric and user-controlled government services. UMA is also especially useful where more than on agency is involved in service delivery and information must be shared between agencies.

The PoC demonstrates that the UMA model works best when the “protected resource” is just a simple resource, rather than an abstract thing. The “sweet spot” for UMA is for managing access to an individual’s information that is provided via a RESTful web service.

UMA by itself doesn’t solve all problems in access control. There will typically be additional access control mechanisms required. As the complexity and degree of abstraction of the protected resource increases, there will be more access control logic contained outside of UMA.

If UMA, by itself, isn’t a perfect solution for a Common Delegation Capability, perhaps the question should be; is UMA the best current fit to be the basis of a Common Delegation System?

The Technology Options Paper, identified UMA, XACML and proprietary techniques as options to implement a Delegation System. Proprietary solutions can be excluded because they are not future-proof and create vendor lock-in. XACML is a mature standard, but is enterprise-centric (suitable within an organisation) and does not specifically address user-control.

During the PoC development, one pattern was observed. When UMA, by itself, did not solve 100% of a delegation requirement, then both the UMA standard and the underlying OAuth 2 standards were used as a design pattern. This was an unexpected benefit to using the UMA-centric approach.

In addition, UMA demands a context around delegations, which will likely help avoid some possible policy/privacy traps that a less-constrained approach would allow.

In conclusion, no current standard or technology will be a perfect fit for the delegation requirements we have today. The UMA standard/protocol, however, does introduce a much needed element of user-control, provides useful structure and represents the best approach available today to implement a Common Delegation Capability.

### Recommendations

#### Next Steps

There are a number of useful steps that can be taken to advance a Common Delegation Capability.

1. Begin to address the policy questions identified here and in an earlier policy investigation.
2. Develop a PoC solution to the Cross-Agency Delegation reuse scenario identified in the Snapshot report.
3. Work to identify the best place to host a self-asserted attribute and contacts service; either RealMe or within the Delegation system itself. If RealMe is the preferred location, actively develop these services.
4. Develop a structured approach and reference architecture for dealing with complex and abstract delegations. Develop a reference model for access control including delegations.
5. Identify agencies for a delegations pilot. Perform detailed analysis of end-to-end business scenarios.
6. Actively participate in the development of emerging relationship models and standards such as Kantara’s IRM.
7. Begin cross-agency work to establish common semantics and taxonomy around delegations and access control.

#### Recommendations

* + 1. Policy/privacy related ASAP
    2. Incremental approach to UMA (not generic Delegations)
    3. Participate in IRM emergence
    4. Promote uniform government access control model
    5. ReMeAS & ReMeTC
    6. Develop NZ government semantic model for access control, delegation/relationship, other.

#### Options Going Forward

Delegation Service: Incremental vs Big Bang

Other Things to Prepare for future

* IRM
* ReMeAS
* ReMeTC

## Lessons Learnt

The following lessons were learnt using an Agile methodology to build the Delegations Proof of Concept:

|  |  |
| --- | --- |
| **Lesson Learnt** | **Description** |
| Proof of Concept help to clarify and learn | Going through a Proof of Concept (PoC) helped to clarify and refine the problem we were addressing – which had been a challenge given its complexity and the broad range of business contexts. |
| A well-defined Minimum Viable Product (MVP) statement is essential | Creating an MVP statement for the PoC helped to keep everyone on the same track. Focussing on the MVP meant the team and stakeholders were less likely to get distracted and stuck in all the detail and complexity that surrounds delegations. |
| Feedback is essential | Receiving feedback early from stakeholders was essential in iterating and developing the PoC in the right direction. |
| Demo early and regularly | In order to gain feedback from stakeholders early and regularly, it was important to show them something they were able to give feedback on. |
| Consider the audience for the demo | The demo must be tailored to the audience. Some of the specific things we learned were; it’s not always about showing software, don’t demo too fast, not all audience members will feel comfortable telling you that they don’t get it. |
| Business owners need to make decisions quickly | Having business owners for each scenario that were able to make decisions quickly when questions came up was essential to completing each sprint. |
| Commit to daily stand ups | During this PoC process, communication and keeping everyone on the same page was essential. Daily stand ups proved an efficient way to disseminate information within the team and an opportunity to address any issues as they arose. |
| Utilise a visual workspace | Our scrum board was an effective way for the team and stakeholders to see what the team was working on and the progress being made on these items. |
| Invest in automated testing | Even though the PoC was done over a very small amount of time, the benefits of investing in an automated testing suit were clear. It saved us a lot of time, and also allowed us to pick up bugs very quickly. |
| Maintain some level of documentation as you go | This makes things much easier when reaching the end of the PoC and a closing report is required. |

## Glossary

|  |  |
| --- | --- |
| Term | Description |
| AAT | Authorisation API Token. An OAuth token specified by UMA that is required to evaluate access toa protected resource. |
| ACC | Accident Compensation Corporation |
| Acceptance Criteria | A definition of the outcome expected before the user story can be considered completed. |
| Agile | A term used to describe a set of software development methods that allow for rapid development of work software. This Agile process used Epics & User stories managed via a backlog, Kanban board & daily stand ups, two weekly sprints with sizing using Fibonacci scoring, and two weekly demonstrations and retrospectives. |
| API | Application Programme Interface. Usually refers to services offered by an application that other applications can invoke |
| AS | Authorisation Server. Specified by UMA Controls access to the protected resources. |
| AWS | Amazon Web Service – the cloud based service hosting the proof of concept environments |
| Bearer Token | A token that by itself authenticates a user. Whoever provides the token can act as the user it represents. |
| Careers | A website service delivered by the Ministry of Education providing assistance to connect from education and training to employment. |
| Client | Specified by UMA. The application that is accessing the protected resource on behalf of the requesting party |
| DIA | Department of Internal Affairs |
| Epic | A grouping of related user stories, generally into a high level capability of functionality. This project used Epics to represent Scenarios, and a scenario/epic was delivered in one Sprint. |
| FSD | Federated Service Delivery (an approach to cross agency delivery of services through R10) |
| iCMS | RealMe’s igovt Context Mapping Service. Used for different agencies to refer to a specific user without using a common identifier |
| ILM | Investment Logic Map (Methodology and tool to present the problems and benefits of an initiative, often used to support a strategic business case) |
| GCIO | Governments Chief Information Officer |
| Identity Provider | Service that is the authoritative source of identity. Usually refers to a SAML 2.0 or OpenID Connect server. |
| IR (IRD) | Inland Revenue, Inland Revenue Department |
| Jira | A software application that was used by the project team to manage their Epics, User stories, Acceptance Critera, and test information for the Proof of Concept. |
| LDAP | Lightweight Directory Access Protocol. A protocol to interact with a directory (data store). Commonly refers to the directory itself. |
| MBIE | Ministry of Business Innovation & Employment |
| MoE | Ministry of Education |
| MPI | Ministry of Primary Industries |
| MVP | Minimal Viable Product (a method used to describe what must be delivered within a development process) |
| NZQA | New Zealand Qualification Authority |
| OAuth | A widely adopted standard for authorisation. |
| OAuth Token | A string representing authorisation. Is often provided during web service calls |
| OpenAM | ForgeRock access management product |
| OpenDJ | ForgeRock LDAP product |
| PAT | Protection API Token. An OAuth token specified by UMA that is required to manage resources protected by the Authorisation Server |
| Permission | A fine grained rule specifying what kind of access is granted (e.g. read, write, etc) |
| PoC | Proof of Concept |
| Policy | A rule that allows access to a certain resource. |
| QR code | A machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a smartphone. |
| R9 | Result area 9 of NZ governments better public services programme focused on better services for businesses |
| R10 | Result area 10 of the NZ governments better public services programme focused on better services for individuals |
| RealMe Attribute Service (RAS) | A fake service created by the proof of concept to demonstrate scenarios that would be improved with additional attributes, possibly provided by RealMe |
| RealMe login service | The RealMe service that provides authentication services to service agencies. |
| RealMe Verified Identity | In addition to authentication, RealMe can provide passport level verified identity data to service agencies. A RealMe verified identity a user’s account that has been verified to this level. |
| Refresh Token | An OAuth token that is usually long-lived but cannot be used as an access token. can be exchanged for a current access token |
| Rendezvous Code | A code used to invite or grant access to other users in a system. Usually transmitted electronically, e.g. via email or SMS. |
| Requesting Party (RqP) | Specified by UMA. The user attempting to access a protected resource. |
| Resource | A record or action that UMA is used to protect |
| Resource Owner (RO) | Specified by UMA. The user who owns the resource |
| RPT | Requesting Party Token. An OAuth token specified by UMA that represents access to a protected resource |
| RS | Resource Server. Specified by UMA. Holds the protected resources |
| SAML | Security Assertion Markup Language. A security protocol used by RealMe to provide information to service agencies. |
| Service Agency | The clients and resource service that utilise the Delegation solution’s services |
| SIAM | Student Identity and Access Management system. A prototype project for the Ministry of Education |
| Sprint | An agile term use in this project to indicate the work complete and demonstrated each fortnight. |
| SSO | Single-Sign On. The ability for the user to only enter their credentials once, and have access to multiple systems. |
| UI | User Interface. The visual interface that users interact with. |
| User Stories | An technique (often used in Agile processes) for articulating a requirement from a user’s perspective. Typically it takes the form (As a <User type> I want to <aim or goal> so that I can <rational for requirement>. |
| UMA | User Managed Access. A standard based on OAuth to allow users to manage access to resources across different services |

Appendix 1

Proof of Concept User Stories

## Appendix 1: PoC User Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Identifier** | **Story** | **Description** | **Verification Criteria** |
| Sprint 1 - Out of the box | DEL-5 | Delegate access a resource | As an entity  I want to use my delegated access  So that I can access the shared resource. | - I can view the resource  - I must authenticate before I can view the resource |
| Sprint 1 - Out of the box | DEL-4 | Name a delegate | As a user  I want to name who I delegate to. | - name user using OOTB  - User must exist in system |
| Sprint 1 - Out of the box | DEL-3 | Delegate | As a user  I want to delegate access  So that I can share a resource. | - An authenticated user can select to share a resource  - A resource can be successfully registered |
| Sprint 1 - Out of the box | DEL-2 | Use RealMe login | As a user  I want to use my RealMe log in. | - I can log in with RealMe  - Basic error handling |
| Sprint 1 - Out of the box | DEL-1 | POC environment set up |  | - Domain selected and registered  - DNS set up  - SAML certs  - SSL certs  - AWS deployed  - Source control/ jenkins/ project set up |
| Sprint 2 - Immigration | DEL-62 | Invite delegation | As Bob I want to invite Carol to receive my delegation | - Code is generated and displayed  - Carol can log in and enter the code  - Code can't be reused  - Note IAM things  - Minimal validation around code  - Carol has an account beforehand |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Identifier** | **Story** | **Description** | **Verification Criteria** |
| Sprint 2 - Immigration | DEL-60 | Alice manages delegations | As Alice  I want to manage my delegations  so that I can edit delegations | - Invite code to Alice  - Alice logs in and accepts invite at immigration  - Alice can view application  - Alice can access 'manage delegations' |
| Sprint 2 - Immigration | DEL-28 | Immigration change delegate | As Immigration  I want to change the delegate  so that the application can continue | - Dave exists in system  - Bob and Carol can no long access the application  - Dave can access the application  - Create admin user that logs in using RealMe to manage delegations   Note: Highlight how admin operators authenticate in the real world |
| Sprint 2 - Immigration | DEL-27 | Alice chooses a new agent | As Alice  I want to choose a new agent  so that they can complete my application. | - Bob and Carol no longer have access  - Eric has access  - Alice can delegate by code or ID |
| Sprint 2 - Immigration | DEL-26 | Delegate to Carol | As Bob  I want to delegate to Carol  so that she can follow up on the application. | - Link on immigration app (sharing)  - Bob finds Carol  - Bob shares with Carol  - Carol will log in and view the application using the temporary ID  - Carol gets a view and edit role, because she's not an adviser, she can't delete or submit an application. |
| Sprint 2 - Immigration | DEL-24 | Begin visa application on behalf | As Bob  I want to begin a visa application for Alice  so that Alice can immigrate. | - Bob must log in with RealMe  - There is a fill in on behalf button  - Bob can fill in the visa application form  - A policy is created from Alice's client ID to Bob  - Bob can view the application |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Identifier** | **Story** | **Description** | **Verification Criteria** |
| Sprint 3 - MoH | DEL-23 | Notification | As Bailey  I want to be notified when my mom's blood sugar spikes. | - Bailey can register interest in blood sugar  - Check delegation  - Testers can generate new blood sugar values and trigger a notification  - Bailey is notified by text (SMS watcher)  - Separate delegation to prescriptions |
| Sprint 3 - MoH | DEL-12 | Bailey logs in to patient portal | As Bailey  I want to log in to the portal and do things so that I can help my mom | - Bailey has access to Aroha's prescriptions  - Bailey can request a repeat  - GP can see request in doctor portal |
| Sprint 3 - MoH | DEL-11 | Delegate to Bailey | As Aroha  I want to delegate my access to Bailey  so that he can do things for me | - Share by code  - Log in to DS with RM  - Home portal (see prescriptions, click share all, email invite code)  - Bailey accepts delegation |
| Sprint 3 - MoH | DEL-10 | Set up patient portal | As Aroha  I want to set up my patient portal account  so that I can delegate access. | - Health portal  - I can log in not using RealMe  - Create basic health profile (prescriptions, notes)  - GP can enter prescriptions |
| Sprint 4 - MoE | DEL-184 | Careers account set up | As A I want to set up my Careers account so that I can find a job. | - A can set up an account  - A logs in using ESIAM  - A selects type of jobs they're interested in |
| Sprint 4 - MoE | DEL-183 | Notification and Approval of ROA request | As A I want to be notified of requests to view my ROA so that I can approve who sees my ROA. | - A gets a notification  - A can approve request  - Notification contains B not C (from the company)  - A can view delegations |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Identifier** | **Story** | **Description** | **Verification Criteria** |
| Sprint 5 - MoE 2 | DEL-282 | All delegation stored in one place | As the parent portal  I want to find the delegation  so I don't have to store it. | -Parent Portal asks Delegation System for delegation  -Parent Portal doesn't store any delegation |
| Sprint 5 - MoE 2 | DEL-234 | ECP verifies the authorisation | As Early Childhood Centre I want to check the person so that I know they have the authority to pick the child up | - ECP has verified the person is authorised to pick the child up |
| Sprint 5 - MoE 2 | DEL-233 | Grandparent delegates to someone else | As a grandparent I want to delegate the authority so someone else can pick the child up. | - Grandparent logs into the parent portal  - Grandparent gives the access to someone else to pick the child up |
| Sprint 5 - MoE 2 | DEL-21 | Time bound delegation \*\*\*SPIKE\*\*\*. | As a parent/guardian  I want the delegated authority to be time bound  so that the delegate can only pick the child up on that day. | - Parent/guardian enters the time band  - when time band expires - the delegate has no access to pick the child up |
| Sprint 5 - MoE 2 | DEL-20 | Change delegations made by another parent/guardian | As a parent/guardian  I want to change a delegation made by another parent/guardian  for a child. | - Parent/guardian can cancel a delegation made by another person.  - Parent/guardian can create a new delegation to pick the child up. |
| Sprint 5 - MoE 2 | DEL-18 | Delegate (grandma) receiving authorisation | As a delegate  I want to receive authorisation  so that I can pick up the child. | - Delegate (grandma) receives an invitation code  - Delegate (grandma) login/creates an account in parent portal  - RealMe authentication is used to log in |
| Sprint 5 - MoE 2 | DEL-17 | Delegate to grandparent | As a parent/guardian  I want to delegate the authority to pick up my child  so that someone else can pick them up. | - Mum logs into parent portal  - Mum gives an access to grandmother to pick up the child |
| Sprint 6 | DEL-351 | ESIAM Name | As an ESIAM user I want the Carrers to know my name when I log in. | - ESIAM name displayed in Careerss Portal and Delegation dashboard |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Identifier** | **Story** | **Description** | **Verification Criteria** |
| Sprint 6 | DEL-338 | Seamless logon between Health and Delegation | As a user I can to login only once using Realme  so I can access health portal and delegation service. | - A and B logs with Realme - A and B not required to log in again when sharing or delegating |
| Sprint 6 | DEL-336 | Polish the demo |  |  |
| Sprint 6 | DEL-322 | Sharing delegation with a company | As Alex I can see that  I delegate to Better Builder. | - Caleb specifies he works for Better Builders  - Caleb has access on behalf of BB to ROA  - Alex see BB in Delegations System  - Email comes from Better Builder  - Caleb gets notified when Alex shares his ROA |
| Sprint 6 | DEL-310 | Trusted contacts | As a user I want  to see my trusted contacts  so I can easily delegate. | - Set up relationships @ RAS  - Health scenario - I can choose RAS contact  - Parent Portal scenario - I can choose RAS contact |
| Sprint 6 | DEL-290 | Real name displayed in OpenAM | As a user I want to  see real name in Open AM dashboard  so I can see who I delegated to. | - User sees the real mane in OpenAM  - User sees the real name in the delegation email he receives  - Integrate across all the scenarios |
| Sprint 6 | DEL-289 | Manage dashboard | As a use I want to log on to one place  so I can manage all my services. | - Create new delegation service dashboard  - User is able to view delegations  - User is able to revoke delegations |

Appendix 2

Technical Description

Proof of Concept Build

## Appendix 2: Technical Description PoC Build

### Introduction

#### Overview

This document contains technical information about the Delegations proof of concept run by the Ministry of Education that finished in September 2015.

Other documents will give an outline of the goals of the proof of concept and the high level outcomes that were achieved.

This document is focussed on capturing some of the detailed learnings from the project, include some design decisions. The intended audience are technical people considering implementation of a (possibly centralised) Delegation solution using UMA in the New Zealand (RealMe) environment.

#### Proof of Concept

Note that there are several design decisions and development approaches that reflect the fact this was a proof of concept, rather than a production ready project.

The goal was to demonstrate user experiences and uncover technical issues specifically around the implementation of UMA. In particular, non-function requirements were not explored.

#### Scope

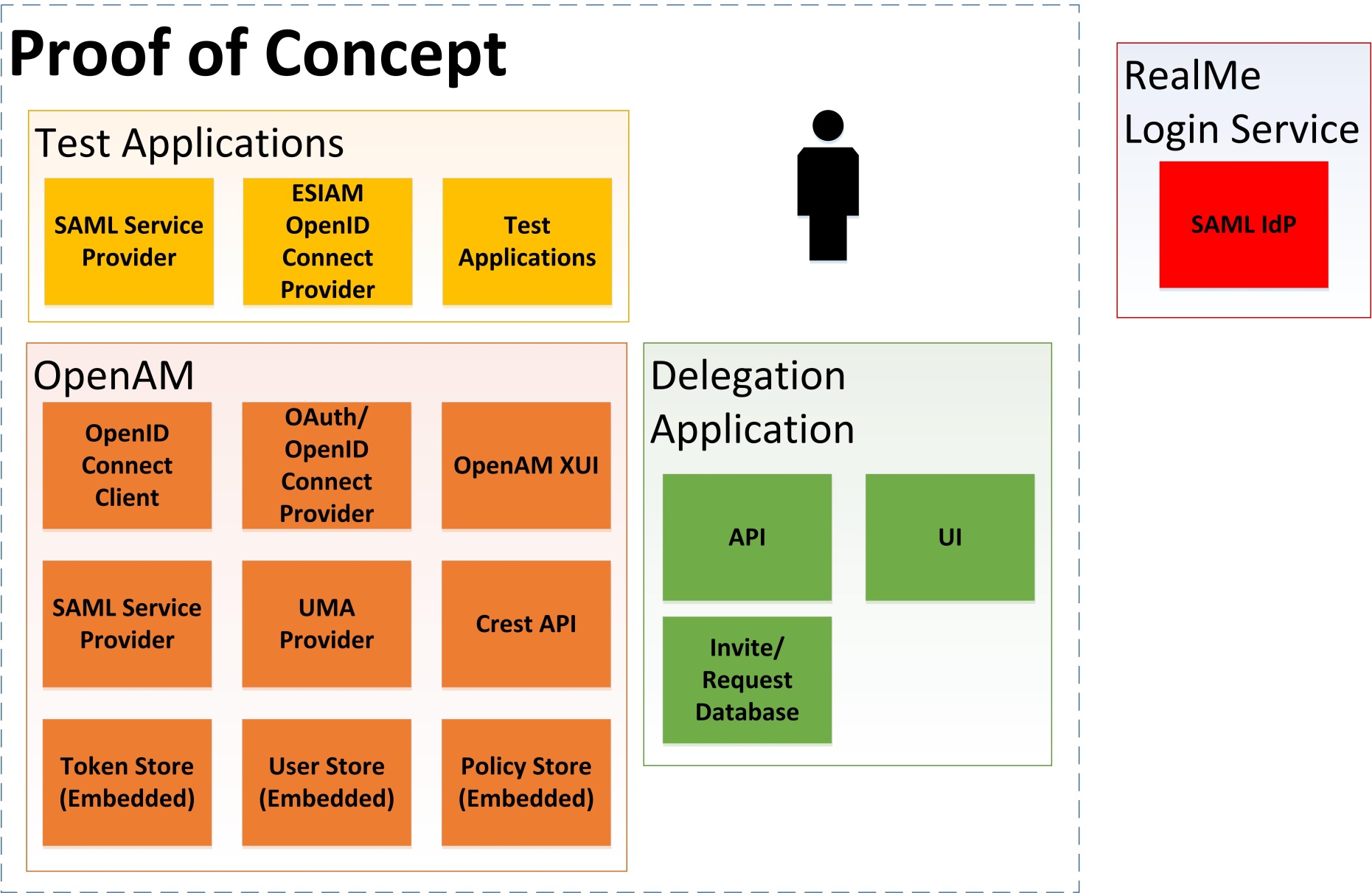
This document does not include:

* Detailed descriptions of UMA, OAuth, RealMe or other technologies or services used
* Detailed design, for example class and sequence diagram. The design is discoverable from the source code. Also, design for a production system may differ significantly.

### High Level Design

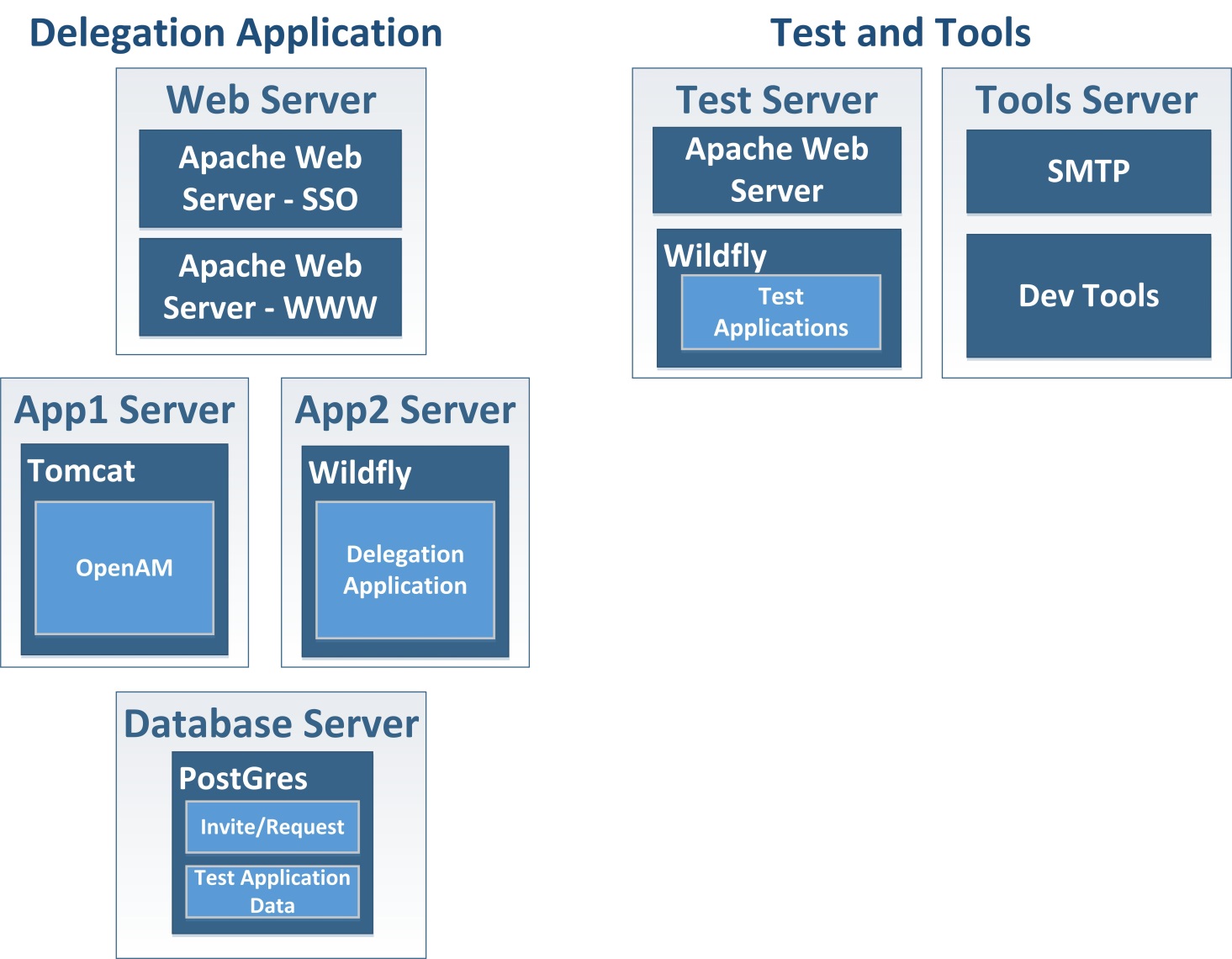
#### 8.2.1 Proof of Concept Logical Architecture

The following diagram shows the logical architecture of the proof of concept



#### 8.2.2 Proof of Concept Deployment

The proof of concept was deployed to Amazon Web Service’s (AWS) Sydney region, within the Ministry of Education’s account managed by Datacom. Development tools were reused from the existing Ministry of Education’s AWS deployment from the SIAM prototype.



This deployment was the same for the two environments used during the proof of concept:

* Proof of concept development (pocdev) environment – used by the sprint team and regularly updated
* Proof of concept (poc) environment – used for demos.

#### 8.2.3 Components

8.2.3.1 Versions

The versions of major third party software used within the proof of concept are:

|  |  |
| --- | --- |
| Component | Version |
| OpenAM | 13.0.0-SNAPSHOT |
| Tomcat | 7.0.54 |
| Wildfly (JBoss Application Server) | 8.2.0 |
| Apache Web Server | 2.4.6 |
| CentOS | 7.1 |
| PostgreSQL | 9.4 |

Several libraries are used by the Java applications (Delegation application and demo applications). These can be found in the maven build files from the source code.

8.2.3.2 OpenAM

The UMA features in OpenAM were not available in a full release at the time of the proof of concept. However, ForgeRock make OpenAM latest development code and nightly builds available. ForgeRock provided the team with indications of which builds were suitable for use during the proof of concept.

These development builds were mostly stable, although some defects were found and raised in ForgeRock’s defect tracking system (See section 8.7 for a list of defects). ForgeRock either provided fixes, or the team used workarounds to deliver the required functionality.

For the purposes of achieving the proof of concepts goals, OpenAM contained the basic functionality required to implement UMA, and ForgeRock provided indications of possible future features.

These gave the team a rough idea of the features and limitations that would need to be considered for a production deployment.

The features of OpenAM that were used:

* + - UMA Authorisation Server
    - OAuth Authorisation Server
    - OpenID Connect client (for ESIAM integration)
    - SAML Service Provider (for RealMe integration)

OpenAM provides its functionality through its XUI (browser interface) or CREST API (Common REST API).

8.2.3.3 Delegation Application

The Delegation application provided two core pieces of functionality:

|  |  |
| --- | --- |
| Functionality | This included |
| UI for centralised use cases | * Creating and completing rendezvous codes * Administration of delegations (Dave from Immigration) * A lightweight dashboard to explore a replacement for the OpenAM dashboard |
| API endpoints. For more information see section 0. | * Wrapper for OpenAM policy endpoints (for example, policy and user management) * Support for service agency delivered UI (for example, rendezvous codes) * Headless user management * Faking of seamless login and iCMS functionality (for simplicity) |

8.2.3.4 Demo App Implementations

To support demonstration of the scenarios, a number of demo applications were created. These included:

* + - Immigration New Zealand application
    - Health Portal
    - Careers New Zealand application
    - NZQA (for storing and retrieving records of achievement)
    - ESIAM (for authentication of education sector users)
    - Parent Portal

These applications were built as minimally as possible to demonstrate the required functionality.

Towards the end of the proof of concept, some of the scenarios were revisited to provide an idea of what an improved experience might look like. Because of this, most of the demo applications have a ‘demoed’ version, and an ‘improved’ version that is available.

The difference between these versions can elsewhere in the documentation.

8.2.3.5 ESIAM and RealMe Attribute Service

Deployed on the test server are ESIAM and the RealMe Attribute Service. These are fake services that demonstrate the use of possible external services to enhance the delegation solution capability. These are discussed in later sections.

8.2.3.6 Development Tools

Where possible, the development team reused tools and infrastructure from the Ministry of Education’s SIAM prototype. This includes:

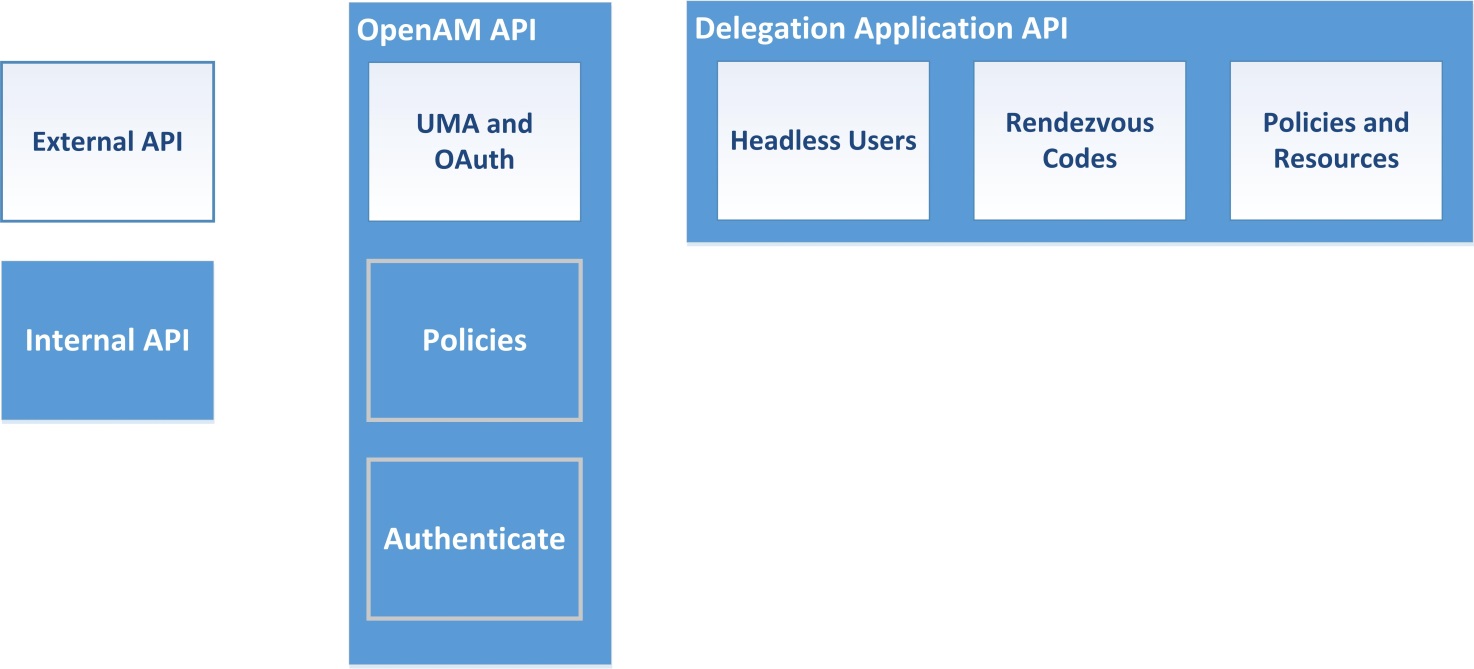
* + - GIT source control
    - Jenkins continuous integration server
    - Selenity test automation tools
    - Archiva release management tool
    - Email trap server

The team used Vagrant in their local developer environments and changes in the code trigger automated builds. Nightly deployments and automated tests keep the proof of concept development (pocdev) environment up to date, with deployments to the proof of concept (poc) environment as necessary.

#### 8.2.4 Delegation Service APIs

This section outlines the APIs that are provided by the Delegation Application, either for use by the demo applications, or to support its own functionality.

A visual representation of the APIs is shown below:



Note that for the proof of concept only basic functionality was implemented. Items to be considered for a production deployment would be:

* + - Correct layout and definition of the API. Perhaps following RESTful principles.
    - Correct authentication and authorisation. The main focus of these APIs is on providing functionality, and therefore trusts the consuming application to only make calls for the right users. Proper authentication should be considered (possibly using UMA or OAuth).
    - Correct use of identifiers. Often, Delegation Application identifiers or RealMe FLTs are used in API calls. This is not appropriate outside of a proof of concept. See section 0 for more information.

8.2.4.1 OpenAM UMA and OAuth APIs

OpenAM provides out of the box APIs required by UMA and OAuth specifications. These APIs are used by the demo applications to:

* + - Obtain required tokens (PAT/AAT)
    - Register UMA resources
    - Authorise access to protected resources

8.2.4.2 Delegation Application – Headless Users

Headless users are used in the Immigration (Alice) and Parent Portal (Jonny) scenarios. These demo applications are provided an API that allows them to create headless users and perform other operations, such as obtaining required tokens and managing their delegations.

For the proof of concept there was no authorisation performed by these endpoints. A production deployment would need to consider this.

8.2.4.3 Delegation Application – Rendezvous Codes

The Delegation Application provides an API for applications to generate and complete rendezvous codes. This is used by the Health Portal.

8.2.4.4 Delegation Application – Policies and Resources

The Delegation Application provides a number of APIs to view and modify UMA Policies and Permissions. The operations are:

* + - Create Delegations – used for headless cases and where delegations are created by the demo application (e.g. Parent Portal uses its own rendezvous system)
    - Replace Delegations – used by the Delegation Application itself for administrators (‘Dave’) to replace a user’s delegations
    - View Policies and Resources – used by the Parent Portal to show current delegations.

Note that some of these APIs use OpenAM’s raw policy endpoints to obtain the policy information. The queries supported by these APIs are limited and are not scalable. Alternative ways of accessing this information may be needed for a production ready deployment.

8.2.4.5 OpenAM – Policies

Some of the Delegation Application APIs required access to the UMA policies created by OpenAM. These APIs made use of both the OpenAM UMA policy endpoints, also used by the OpenAM XUI dashboard, and the raw policy endpoints. External applications would not be expected to use this API.

8.2.4.6 OpenAM – Authenticate

In order to perform actions on the OpenAM policy endpoints an authenticated session must be provided. Where this session isn’t provided, the Delegation Application creates a session using the OpenAM authenticate endpoint. External applications would not be expected to use these endpoints.

#### 8.2.5 User Interfaces

The Delegation Application contains the following user interfaces

|  |  |
| --- | --- |
| **User interface** | **Description** |
| OpenAM Dashboard | OpenAM provides an out of the box dashboard for viewing and managing a user’s delegations. The proof of concept used the OpenAM product while it was in development so there were minor issues and some missing functionality.  It is not clear whether this dashboard would be suitable for a production deployment, because of the limited ability to customise it. |
| Delegations Dashboard | To explore what a custom, centralised dashboard might look like, a custom UI was created using the APIs provided by OpenAM. |
| Rendezvous Creation and Completion | The rendezvous UI was created for the Immigration application to create and complete rendezvous codes outside of the demo application by redirecting the user. This is cumbersome, and a better approach might be to make use of the APIs, the way the Health Portal does. The Rendezvous completion screens are also used by headless users to ‘claim’ their entity. |
| Administration | A very basic administration UI allowed the Immigration administrator to replace the delegation for a user. The support requirements for a centralised delegation application are not clear and it is likely that more functionality and UI would need to be developed to meet these requirements. |

### Identity and Authentication

#### 8.3.1 Identity

There are no specific requirements from UMA to store information about a user. The only requirements are that the users are able to authenticate and obtain the various tokens. This allows very little information to be stored about each user.

Where the AS uses an external Identity Provider for authentication, such as RealMe, it is possible that no personal information is captured by the system. This provides a high level of privacy and removes the need to manage user data in the solution. However there are significant impacts that are outlined below.

A significant limitation of OpenAM with respect to the proof of concept is the identifier centric approach that OpenAM has to managing UMA resources and sharing with other users. In the RealMe environment, there is no easy way to obtain a trusted identifier for users, and this makes sharing with users difficult. The workarounds for this were to use rendezvous codes and to explore the usability of RealMe provided identifiers.

8.3.1.1 Finding Users

The most obvious issue discovered with identities in a RealMe based delegations environment is the difficulty of finding a particular user to share with. Due to the privacy centric approach adopted by RealMe, there are no useful identifiers that one user can use to refer to another.

In a non-RealMe environment, alternatives could be username, email address, or perhaps searching a directory for a user.

Instead, the approach used often in the proof of concept is to provide a rendezvous code. The user who wishes to delegate obtains a code that must be provided to the intended delegate. The code may be provided in person (shown in the Immigration scenario), or via email (shown in the Health and Parent Portal scenarios).

The security of the rendezvous code was not extensively looked at.

A more radical alternative was implemented in the final sprint. The concept of ‘RealMe Contacts’ where users could maintain a list of RealMe accounts for other users they cared about was introduced. This allows users to specify other users from their contact list to receive delegations.

While the setup of these contacts may ultimately be similar to a rendezvous, there might be value in reusing these contacts across other services. Also, the ability to take advantage of verified identity data to support the linking process could be useful.

8.3.1.2 Trusting Users

Once a delegation has been setup, UMA (and OpenAM) provide the ability for a user to view who they have currently delegated to. However, because RealMe does not provide information (unless the user is identity verified) there is no simple way to display who currently has access. Two approaches were discussed during the proof of concept

* + - Using a ‘RealMe Attribute Service’ that provided self-asserted information about a user. This would have less confidence than a verified RealMe identity, but may be more available.
    - Allowing the user to specify a nick name for each delegate they interact with.

The first option was implemented during the final sprint to show the possible user experience if this information was available.

8.3.1.3 Notifications

A seemingly small, but significant challenge is the ability to notify users when certain events occur in the system. However, if the system does not collect email address or phone number, no notifications can be sent.

One option is to collect this information, but this adds yet another place for users to provide and maintain their details.

During the final sprint, this information was made available from the ‘RealMe Attribute Service’ to show the positive user experience, should this feature be implemented by RealMe.

#### 8.3.2 Headless Users

A case implemented during the proof of concept is the ‘headless’ user. This user is an entity within the system with no credentials. They may represent:

* + - A real person that cannot or does not need to interact directly with the solution. For example, an immigration applicant that has asked an advisor to create and manage their application for them, or a child who’s parent’s manage delegations of the child’s resources.
    - A non-person entity, such as a company. The company cannot log in, instead agents act on behalf in various contexts.

This pattern allows delegations to be represented between headless users in the same manner as non-headless (normal) users. This is useful because access to the resources can be evaluated by the Client and Resource Server without knowing the type of user that owns the resource.

It is also useful if the person represented by the headless user decides to claim the account and begin managing the delegations themselves. This was demonstrated in the Immigration scenario if Bob specified Alice’s email during the creation process. Alice would receive an email with an invite to link to the headless account, but Bob could carry on as normal regardless of whether Alice accepted the invite. It may also happen if the account represents a child who takes control of their account when they come of age.

Managing these users and their delegations requires APIs and UIs for the correct people to manage the delegations. Who can create and manage these accounts is a difficult question that is discussed in the rules and relationships section below.

#### 8.3.3 Privacy

There are several privacy considerations that have been brought up during the proof of concept. Some may be genuine policy related issues, and others may be users’ expectations of privacy.

The high level discussion around privacy is not addressed here.

The considerations raised during technical implementation were:

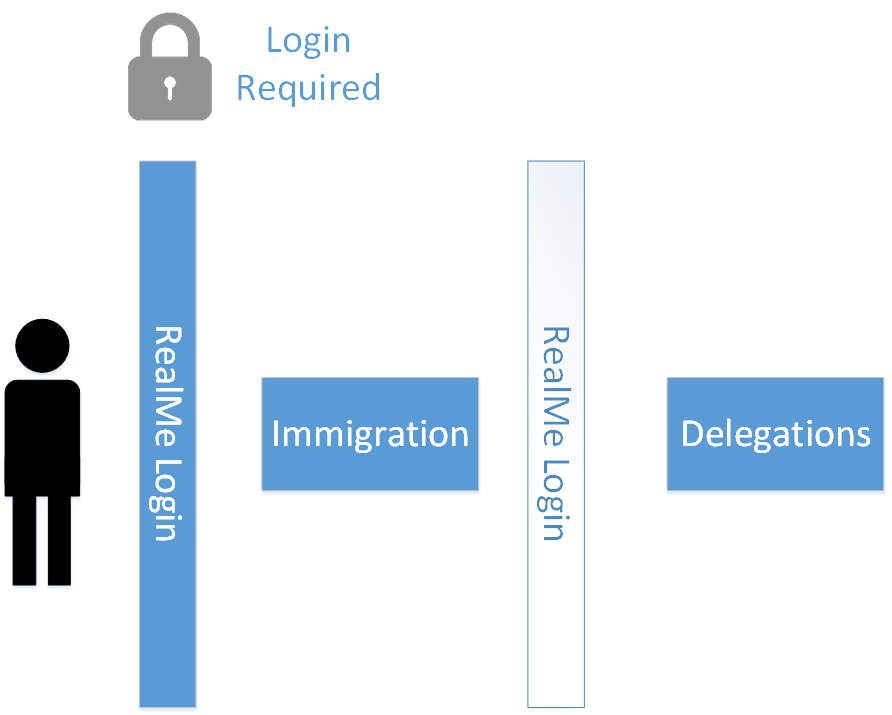
* + - Ability to search for other users, for the purpose of creating a delegation
    - Identifiers used within the system and displayed to other users
    - Notifications to users
    - In a centralised model, the ability for administrators of one application to see a user’s delegations across other applications.

#### 8.3.4 Authentication

The proof of concept demonstrated different options for authenticating users in the implemented scenarios. The different combinations are outlined below.

8.3.4.1 Agency and Delegation Solution Using RealMe

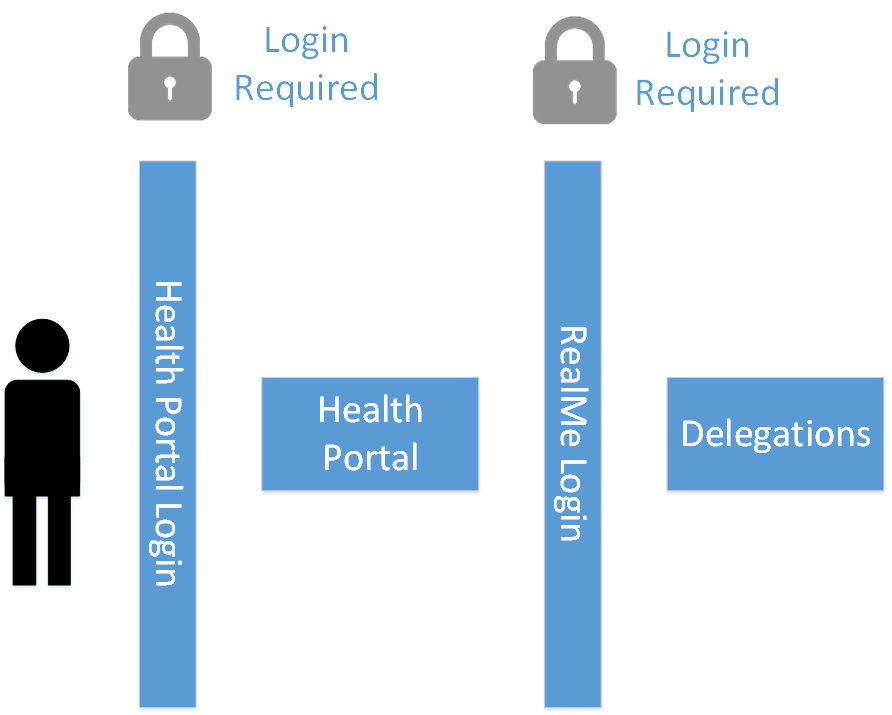
This was demonstrated in the Immigration scenario. All users authenticated using RealMe. As you can see from the following diagram, the user needs to only log in once. Seamless login, or iCMS can be used to interact with the delegations solution.



In this case the user has a single RealMe login account that allows them to authenticate to both the agency and Delegation Solution. For API calls, the RealMe iCMS service can be used to identify the user. For UI redirects, the RealMe Seamless Login Service can be used to avoid the user logging in multiple times per session (effectively SSO). Descriptions of how the iCMS and Seamless Login Service work is outside the scope of this document.

8.3.4.2 Agency Using Local Authentication

This was demonstrated in the Health scenario where the health portal had local authentication. The following diagram shows the result:



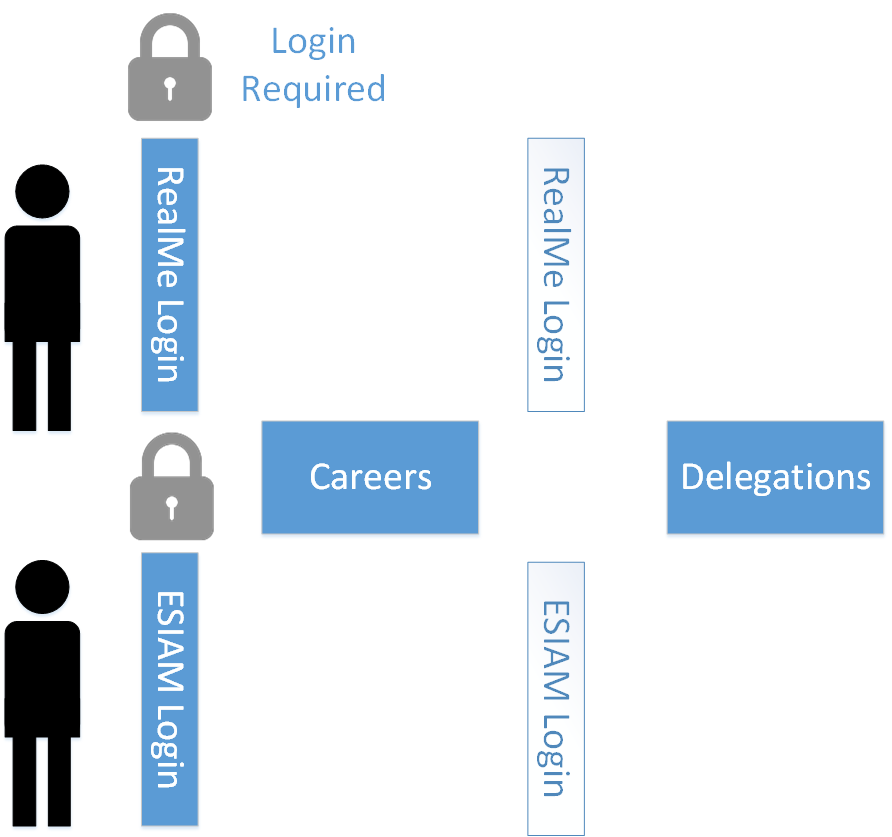
An agency using local authentication provides the least desirable user experience. The reasons for this are:

* + - New users have to create local agency accounts and RealMe accounts during the same flow
    - Existing local agency users are required to create RealMe accounts to delegate, but if delegations are managed through APIs at the agency, then they may never need the RealMe account again.
    - To use the APIs provided by the Delegation Solution, the agency needs an identifier to use during the calls. For the proof of concept, a static ID was provided, but a production deployment would require some sort of federated ID and possibly a mapping service.

Note that if the user doesn’t need to interact with a centralised delegation solution then these do not apply. For example, if the application uses an API to manage delegations and obtain tokens then using the local authentication may be adequate. However, this removes the ability for the user to manage their delegations centrally.

8.3.4.3 Agency and Delegation Solution Using Multiple but Shared Authentication

This scenario was demonstrated in the Careers scenario. The job seeker could authenticate using ESIAM, and the employer could authenticate using RealMe. The following diagram illustrates:



Both users only need to log in once, similar to the first scenario. This demonstrates that the authentication used by the resource owner and the requesting party can be independent.

For the proof of concept, each user either had a RealMe login, or ESIAM login, not both. In a production deployment, the ability for a single user to link (federate) multiple authentication methods should be considered.

#### 8.3.5 Session Management

Session management in a shared environment has a significant impact on user experience. The goal is for a user to navigate between related systems without having to unnecessarily re-enter their login details.

One approach to this is to use a central authentication server, such as OpenAM, as the holder of a user’s session. The user can then leave the session open and automatically sign in to different services. There are disadvantages to this approach. These include:

* + - Security concerns of leaving an session behind
    - Complicated to implement global log out
    - Navigations not including the central server may not have a seamless experience.

An alternative approach, provided by RealMe, is the Seamless Login Service. This allows service providers to manage a user’s session locally, and provides a means for the user to be redirected away without leaving a session behind. Instead, a token (SAML assertion) is passed between service providers and RealMe. However, this requires all services to use RealMe.

Using RealMe is probably not an unrealistic requirement in a shared environment. The Health scenario showed a poor user experience when multiple authentication providers are used for the same person (The Careers scenario used two authentication providers, but each user only used one).

Session management is less of an issue in a non-shared environment, or where APIs are used to provide delegations functionality on the service agency. In this case, a federation or identifier mapping tool, such as the iCMS, is required.

#### 8.3.6 Delegation Identifiers

There is a need for the demo applications to identify users, for example, during API calls. Where the application uses RealMe, the iCMS can be used to provide the user’s FLT. We can also assume a similar mechanism would be available for ESIAM.

However, there are two cases where this is not sufficient:

1. The Health Portal does not use RealMe so cannot use the iCMS
2. Headless users (companies, children and users such as Alice in the immigration scenario) do not have a RealMe account so there is no identifier for them

Therefore, sometimes the Delegation Application may need to provide an identifier. For the Health Portal, an identifier can be obtained using an OAuth scope (forcing the user to log in again). For headless users, an identifier is provided through the API when they are created. In both these cases for the proof of concept, the identifier is **not** pairwise (i.e. it is the same for all applications).

This may not be appropriate in a real environment but demonstrates the need and purpose for such an identifier.

### Delegation Considerations

#### 8.4.1 Consent

There are two key UMA tokens that represent user consent and are required to manage resources and determine access. These are the Protection API Token (PAT) and the Authorisation API Token (AAT). The out of the box behaviour in OpenAM to obtain these tokens is to:

1. Authenticate the user
2. Obtain consent from the user

Using RealMe’s Seamless Login Service it is simple to remove the need for the first step. However, the second step is still jarring for the user. Early on in the demonstrations it was decided to remove the need to capture consent. The type of consent required, and the means to capture it should be considered for a production ready deployment.

#### 8.4.2 Discovering Resources

One of the key challenges that was highlighted during the proof of concept is understanding how users should discover resources they have access to.

The two high level approaches are:

1. Store the set of resources a user can access on the Client
2. Query the Delegation Solution for the set of resources a user can access

The issue with the first approach is that this essentially duplicates the delegations on the Client and the Delegation Solution.

The issue with the second approach is that it seems to make the second half of the UMA flow redundant. In other words, why ask the Delegation Solution for a list of resources a user has access to, and then once the user selects the resource, check whether the user has access? They should have access, because it was in the list of resources they have access to. UMA doesn’t provide guidance on how this problem should be solved. For the proof of concept, a few different approaches were implemented:

|  |  |
| --- | --- |
| Approach | Example |
| **Resource Identifiers** | For example, knowing the application ID, user ID or other identifier to find the resource. This was implemented in the Immigration scenario.  This is simple to implement, but provides a poor user experience. |
| **‘Favourites’** | For example, the service agency records resources the user may have access to. This was implemented in the Health Portal.  This could be considered duplication of delegations, and has the added disadvantage that if a Resource Owner shares the resource using the Delegation application (i.e. not through the service agency) the delegate may not have a way of accessing the resource because it won’t be on their friends list. |
| **Search** | For example, the user can search for the resource they wish to access. This was implemented in Careers.  This depends on the policy of the service agency on the ability to find other people’s resources and may not be desirable.  However, this does provided a good user experience, and combined with the ‘Favourites’ approach could be very useful. |
| **Querying the Delegation Solution** | For example, the service agency can query the delegation solution for resources you have access to, and then display these. This was implemented in the Parent Portal when showing which children you have the ability to pick up.  The biggest disadvantage to this approach is that it appears to make the final UMA authorisation call redundant. The service agency has asked for a list of resources the user can access, why re-check access once the user selects one? |

#### 8.4.3 Discovering and Identifying Users

Another key challenge highlighted during the proof of concept is how important it is to be able to discover and identify users. The proof of concept doesn’t provide a general solution to this issue; instead it highlights interactions that require confidence in identity. The areas of delegation that require confidence in identity are:

1. Discovering users to share with
2. Viewing which users have access to resources
3. Notifying users of events

8.4.3.1 Discovering Users to Share With

The out of the box behaviour of OpenAM is to use identifiers, such as username or email, to share with users. However, because of the limited information provided by RealMe this isn’t feasible.The following approaches were either implemented or discussed during the proof of concept:

|  |  |  |
| --- | --- | --- |
| **Approach** | **Comment** | |
| Rendezvous /Invite | This appears to be the most obvious solution, and applicable in most scenarios. This essentially involves the Resource Owner specifying access to a resource and one time code being generated. This code can be provided to a delegate to claim this access.  This is a common pattern in identity and access management systems.There are several ways of distributing the code, this could be:   * Providing the code to the resource owner, they can distribute it themselves. This might be in person, or over the phone * Emailing the code to the delegate. The resource owner must provide the email address in this case   Additionally there are different approaches to implementing this shown below: | |
| Centralised rendezvous system – Centralised UI | The first approach was implemented in the Immigration scenario. |
| Centralised rendezvous system – Service Agency UI | The second approach was implemented in the Health scenario, and has the benefit that the agency can customise delivery of the one-time code. For example, the email can be customised and when the delegate accepts the delegation, the delegation can also be presented in an agency specific manner. |
| Service Agency rendezvous system | The third approach was implemented in the Parent Portal and allowed different types of delegations to be created depending on who generated the invite. This was needed because of the equal ownership required.  The security of the rendezvous code has not been extensively assessed. There may be some valid security risks and mitigations considered for a production deployment. |
| Direct Sharing | This is the out of the box method for identifying users in OpenAM. The Resource Owner specifies a user they wish to have access. Because users in RealMe have no usable identifier this isn’t feasible. | |
| Previous Delegates | An interesting approach that was not implemented would be to allow Resource Owners to easily share with users they have previously shared with. This would probably be a valuable feature, but doesn’t solve the issue of delegating to someone the first time. | |
| RealMe Contacts Service | Another interesting approach that was implemented in the ‘improved’ demo applications, is to imagine a service offered by RealMe to have a list of contacts that a user interacts with. Service agencies or the Delegation Solution could use this service to allow users to choose someone to delegate to. This approach has not been fully explored, and the focus was on providing a good user experience when identifying users, rather than considering the feasibility of such a service. | |
| External Register | An approach that seems to have been discussed in the working group, but not implemented, is to allow users to search through external registries for delegates. This could be searching of a professional accreditation database to find accountants or lawyers for example.This has not been explored during the proof of concept. | |

#### 8.4.3.2 Identifying Users

Once a delegation has been created, it is important to be able to know with confidence who the delegate it. However, due to the limitations of RealMe, there is no identifiable information about the user.

Solutions to this issue has not been extensively explored in the proof of concept, but some possible solutions are:

* **RealMe Attribute Service**: Some attribute service provided by RealMe, possibly combined with the RealMe Assertion Service (providing verified identity data) could provide identifying information for users. Additionally, in the final sprint, the concept of RealMe ‘Contacts’ was implemented. This allows a user to set up a list of contacts using RealMe, and providing these contacts to service agencies.
* **User Managed Aliases:** An option put forward by ForgeRock would be to allow users to manage a set of aliases for users they delegate to. This could be used to help users identify who they have shared with, but may not provide a strong level of confidence.
* **Agency Information**: One practical option may be to display delegations at the service agency. The agency could then use information stored in their applications to identify users. This works around some of the privacy issues and possibly leverage existing processes to collect information, but relies on all agencies maintaining sufficient identity information.

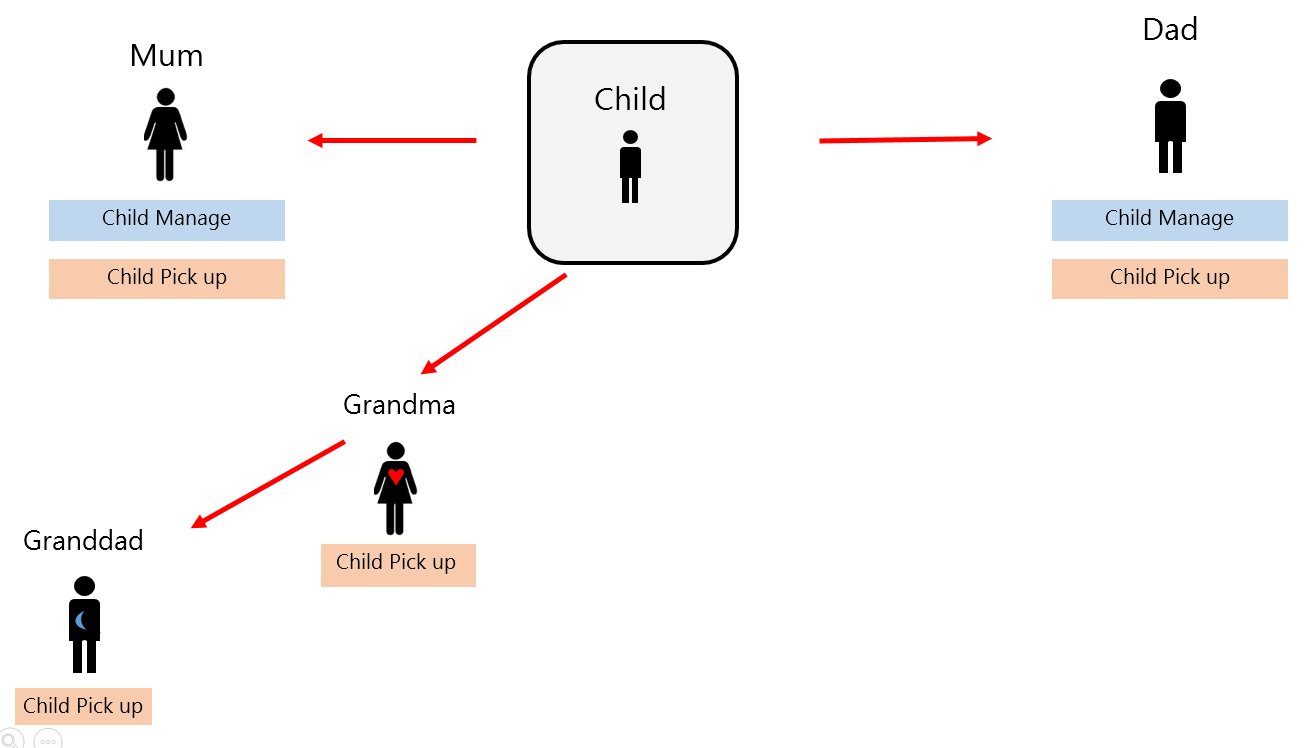
8.4.3.3 Notifications

There are several user flows that require notifying the user of certain events. It is most likely that these will be over email, however RealMe doesn’t provide a user’s email address. The options to solve this are:

1. Have service agencies generate emails. This assumes that most agencies collect email addresses, and was implemented in the Health scenario. This has the added benefit that notifications can be customised to the user’s current scenario and context
2. Have a centralised service provide email addresses for users. This was implemented in an conceptual ‘RealMe Attribute Service’ during the proof of concept
3. Have a centralised notification system where services such as the Delegation solution can generate emails and identify recipients by their RealMe identifiers (FLTs).

#### 8.4.4 Delegation Chains

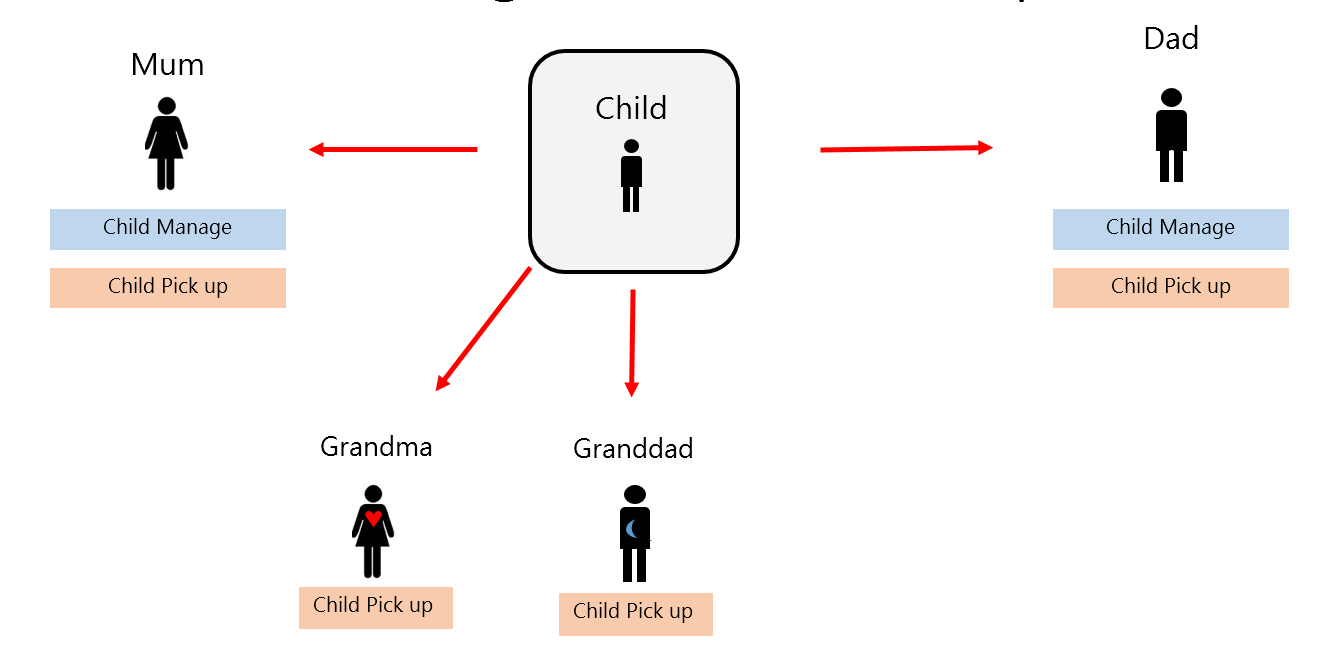
One of the concepts shown during the proof of concept is the ability to re-share a delegation with other people. OpenAM allows this feature to be globally enabled or disabled. It cannot be enabled per resource, or per delegation. Delegation Chains allow users to share any resource they have been given access to with anyone else. OpenAM’s out of the box behaviour is to hide these re-shares from the resource owner. Using the Parent Portal example:



Grandma has been given access by Mum. Grandma has then shared her access with Grandad. The consequence of this is that Mum and Dad have no way of knowing that Grandad now has access. In some cases this may be desirable. For example, if I share access to my tax records with an accounting firm, I may not need to know each individual that has access.

It is worth noting that if Grandma lost access, then the chain is broken and Granddad can no longer access the resource.

An alternative would be to allow Grandma to create a new delegation from the child for Grandad:



In this design, Mum and Dad can see that Granddad has been given access. But if Grandma is revoked, Granddad still has access. This approach is more cumbersome to implement and would require implementation outside of OpenAM.

Essentially, the requirements of the delegation, such as who should be able to see it or re-share it, and what should happen when access is revoked should be considered when designing a delegation model for a given use case.

#### 8.4.5 Automatic Chaining

There are several use cases where users wish to automatically re-share delegations with others. This might happen because:

* The entity that receives the delegation is a non-person entity (e.g. a company). This also includes people that have no online interaction (e.g. a child). In this case the delegation must be passed to a real person in order to be useful. This is shown in the Careers scenario.
* The entity that receives the delegation doesn’t perform the actions. For example, an assistant is required to have access to all of the delegations. This is required for the Immigration (Bob to Carol) scenario.

This is not currently supported in OpenAM, and is one of the reasons that the Careers scenario was not completed during its sprint. For the improved version, this was implemented using workarounds to demonstrate the intended user experience.

For a production deployment, it is not clear how this would be implemented, but it is likely that it would be driven by rules and relationships. These rules and relationships may or may not be part of the Delegation solution. ForgeRock have discussed the possibility of adding in extension hooks to facilitate this functionality.

Also, there are options in how these rules and relationships are stored. For example, it isn’t clear whether the delegation chain is implied - rules and relationships should be evaluated at authorisation time, or whether delegation chains should be created as relationships and resources are updated – the delegation chains always reflect the current access for a given resource.

These options are demonstrated below.

8.4.5.1 Automatic Chaining Based on Relationships



In the above case, Alex shares his Record of Achievement (RoA) with Better Builders. There is a relationship between Calen and Better Builders that means Better Builders automatically re-shares this with Caleb. Caleb can then access the RoA, and if Alex revokes Better Builders access, the chain is broken and Caleb no longer has access.

An advantage of this solution is that Caleb can see all of the resource he may have access to. This is also scalable in terms of performance – evaluating access is simply traversing the chain to the resource owner.

A significant disadvantage of this is that the chains must be maintained to match the relationships at all times. Events that affect the chain include:

* New resources being shared with Better Builders
* Caleb’s relationship with Better Builders changing or being terminated
* Additional users form relationships with Better Builders and require access to Better Builders’ delegations

8.4.5.2 Chaining Based on Relationship Evaluation



In this case, Alex shares his RoA with Better Builders, but there is no automatic share to Caleb. Instead, when Caleb attempts to access the resource his relationships are evaluated to see if he has access through one of these.

The disadvantages of this include:

* Evaluating a large number of relationships may not be performant at scale
* It may not be clear what resources Caleb has access to at any point in time

#### 8.4.6 Requestor Initiated Delegations

Requestor initiated delegations is a feature of OpenAM that provides an alternative to creating delegations other than the resource owner sharing directly with a delegate.

UMA itself does not specify how delegations should be created, however it does not prevent the Authorisation Server from implementing several ways of achieving this.

Requestor initiated delegations are used in the Careers scenario, and an example is:



In the first instance, Caleb attempts to access Alex’s resource, but is denied. An approval request is sent to Alex.

Once Alex approves the request, Caleb can then access the RoA.

However, there is a significant issue with this for the Careers scenario. In the above diagrams, Caleb (who works for Better Builders) has attempted access, and therefore Alex is asked whether Caleb should have access. But Alex doesn’t know who Caleb is. Instead, Alex should be asked to approve access for Better Builders.

There are two solutions to this:

1. Caleb makes the request as Better Builders. This is effectively impersonation, and is not desirable.
2. The Delegation Solution uses relationship information to know that Caleb represents Better Builders and that the request should come from them.

The second solution is similar to the automatic chaining solution in the previous section, in that it requires some sort relationship information, either within the Delegation solution or externally available.

Ideally, Alex should receive a request to Approve Better Builders access, then Better Builders should re-share this access to Caleb (see the previous section) or other employees of Better Builders.

Even if there were a mechanism to translate requests from Caleb to Better Builders, there are complicating factors, such as if Caleb works for multiple companies, how does the system know which one Caleb is acting on behalf of?

The proof of concept has not solved these issues, but has highlighted the need for rules and relationships to be considered.

**8.4.6.1 Centralised vs Agency Implementations**

There are options on how and where pending requests can be implemented. These include:

1. Centralised Pending Requests – Centralised UI
2. Centralised Pending Requests – Agency UI using Centralised API
3. Agency Pending Requests

The Careers scenario used the first approach, where Alex received an email from the Delegation Solution and approved the request from a central UI.

The Health scenario used the third approach, where emails and requests came from the Health Portal, and Alex approved access. A centralised API was used to create the delegation once approved.

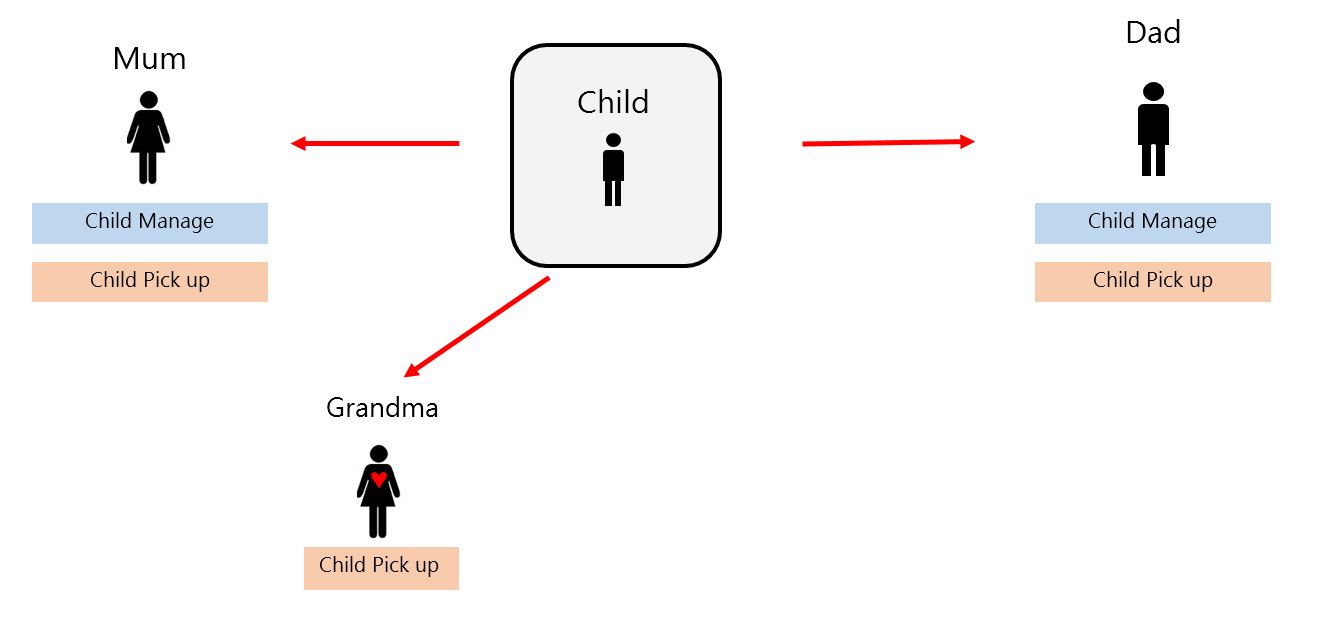
The second and third options provide the opportunity for users to manage their delegations within the context of the resource. This may be desirable. Ultimately, this choice should be consistent with the approach taken for other features of the solution, such as creating rendezvous codes and managing delegations.

#### 8.4.7 Equal Ownership

One of requirements the proof of concept aimed to explore was equal ownership of resources. UMA itself does not permit resource owners to share ownership of a single resource. However, in looking into the scenarios that required it, a solution was found.

The example used is for the Parent Portal scenario where both parents have equal ability to delegate who can pick their child up. A similar scenario would be for multiple directors delegating resources for a business.

For both of the above cases, creating a headless user that represents the true owner (e.g. the child or business) solves the issue. The ability to pick the child up is owned by the child. While this might not be true legally, in terms of a logical model it fits well. The parents are then given authorisation to manage this resource on behalf of the child. This can be done using delegations as well. The following diagram demonstrates the concept:



By giving Mum and Dad access to the ‘Child Manage’ resource, they can equally manage the child’s delegations.

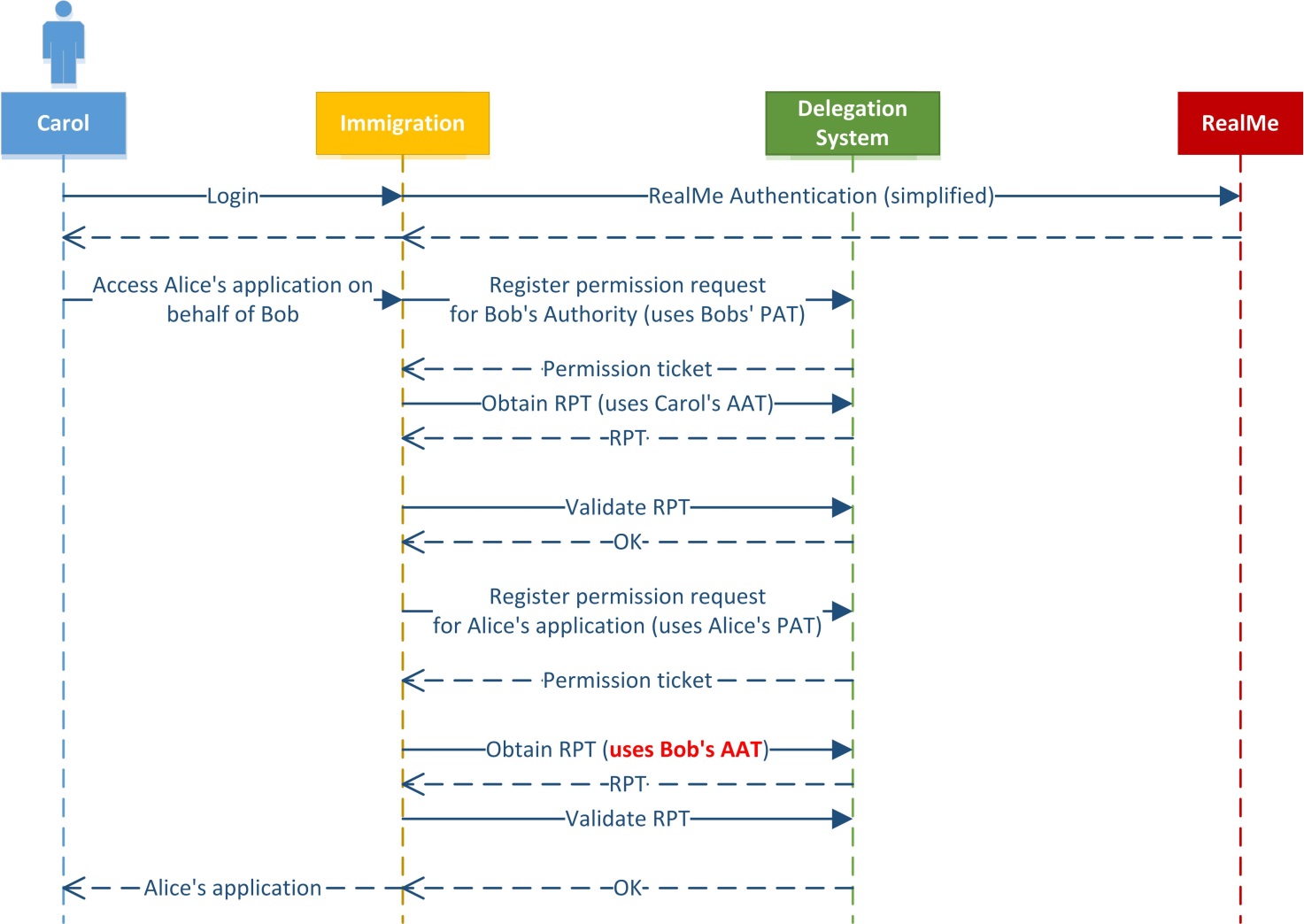
An alternative would have been to one of either Mum or Dad own the resource and grant the other access to this. But this would have a few drawbacks:

1. The child is not represented in the system. Any resources that logically belong to the child will clutter up the Mum or Dad’s account.
2. Assuming Mum was the primary owner, then Dad would not really be equal.

#### 8.4.8 Authority Delegation

Early in the proof of concept the Immigration scenario required Carol, Bob’s assistant, to access applications that Bob had access to. This was implemented by granting Carol access to a resource of Bob’s that represented his authority.

When Carol attempted to access a resource it would first evaluate whether she could access Bob’s authority, and then whether Bob had access. The following diagram demonstrates the evaluation:



As you can see, this requires the use of Bob’s AAT even though he is offline. In fact, OpenAM recommends an additional OpenID Connect id\_token is passed when obtaining an RPT, and this id\_token is not available offline.

This turns out to be a form of impersonation (Carol is attempting to impersonate Bob), and is not recommended for production deployments. One of the goals of UMA is to eliminate the need to impersonate others. Instead, a combination of Delegation Chaining and relationships based rules to automatically create the chains would be more suitable.

#### 8.4.9 Push Notifications

A similar issue to the Authority Delegation was found when performing push notifications. See the diagram below.



As you can see, this requires the use of Bailey’s AAT even though he is offline. This is also a form of impersonation (the Health Portal is attempting to impersonate Bailey). This would not be recommended for a production environment.

This makes push notifications difficult to implement. In a discussion with ForgeRock, some alternatives were proposed, but not extensively explored. Some options are:

1. Use notification channels where Bailey is online. For example web sockets on a mobile device. In this case, the device (with Bailey online) would regularly be recreating the channel, checking access each time.
2. Only assess Bailey’s access when Bailey signs up for subscriptions and have the Health Portal manage the revocation of this.

These solutions require more analysis and there may be other, more appealing options.

#### 8.4.10 Resource Granularity

During the proof of concept the most common type of resource was a generic resource for a user that represented access to an aspect of their account. For example, their current immigration application, their health prescriptions, etc.

Having multiple resources of the same type was not implemented, but is conceivable. For example, having tax records for different years, or different attributes for an account.

One type of resource that was implemented that would not be recommended is the Immigration resource that represents a user’s authority. This was intended to allow assistants to access applications granted to their boss. However, this is a form of impersonation and is not ideal.

#### 8.4.11 Resource Scopes

UMA allows resources to have multiple scopes to further define access granularity. The simplest example is the ability to read and/or write to a resource. The proof of concept did not use multiple scopes, mainly due to a lack of richness in the demo applications. Future work in this area would need to understand when best to use multiple scopes versus multiple resources.

#### 8.4.12 Time-Bound Delegations

During the Parent Portal scenario it was planned to include the ability for the delegation to be time-bound. For example, only for the next week.

Because OpenAM’s UMA implementation uses OpenAM’s policy engine, the same types of policies used for access control can theoretically be used for delegation. This includes time-bound policies.

However, the OpenAM UMA API did not allow for complex polices to be created. Instead the alternative was to create the raw policies using OpenAM’s generic policy API. Unfortunately this did not work as expected, as policies created through the generic policy API do not work as expected with the UMA policy API. An issue was raised with ForgeRock (OPENAM-6739).

Given the scope of the proof of concept, it was decided to not implement a workaround for this. Instead, a production deployment that required time-bound delegations would either need to:

* Make use of a fix (if provided) from ForgeRock
* Investigate a workaround. For example, creating the policy using the UMA API, then modifying it to include the time bound condition.

Other considerations of time-bound policies would need to be made. For example, cleaning up old policies and notifications of expiry. These were not explored during the proof of concept.

#### 8.4.13 Resource Descriptions and Names

There are several aspects of resources that the resource server can define, and may be visible to the user. This includes:

* + - Name
    - Type
    - URL
    - Icon
    - Scope descriptions

The visibility and impact of these attributes has not been fully explored in the proof of concept. It is likely that a production deployment would make recommendations to resource servers on rules and best practices for naming these attributes.

#### 8.4.14 Resource Server and Client Coupling

For all but the Careers scenario, the Resource Server and Client are the same application. This may mean not all of the benefits of UMA are fully realised, and that there may be simpler alternatives. It wasn’t within the scope of the proof of concept to consider these alternatives.

### Rules and Relationships

The Immigration and Careers scenarios highlighted a need for a rules and relationships component or system. The core requirements highlighted by this proof of concept are:

* + - Allow users to request access to resources on behalf of another entity
    - Grant access to resources that have been shared with an entity based on rules and relationships
    - Ensure users have the correct visibility of who can access their resources
    - Ensure users have the correct visibility of what resources they can access
    - Provide tools for managing the rules and relationships required

This needs to be investigated at a higher level, before exploring the possible technical options.

### Implementation Considerations

#### 8.6.1 Centralised UI vs Centralised API

The proof of concept explored two options for interacting with a user in a centralised delegation solution:

1. Centralised UI – the user is redirected to the centralised service when interaction is required
2. Centralised API – the user mainly interacts with the service agency UI, but the service agency uses a delegations API to perform delegation related tasks.

The immigration scenario used the first option, and the Health and Parent Portal scenarios used the second.

Neither approach is clearly better than the other, they each have their own advantages and drawbacks. The main factors to consider are:

* + - User experience – is it jarring to be moved to different sites?
    - Cross government experience – is there value in the user being aware that the delegations are across government?
    - Implementation effort – is there significantly more effort for agencies to implement a UI?
    - Complexity – each approach requires a technical feature of the centralised service, either the ability to seamlessly redirect between sites, or provide identifier mapping services for API calls. Do either of these approaches add unnecessary complexity?

For each type of interaction that might be required the UI vs API approach should be considered. It would be recommended that at a minimum the APIs are built, and the UI can either be built centrally or locally as required.

Overall, the question of Centralised UI vs Centralised API relies on high level discussions about the purpose of a centralised delegation service.

#### 8.6.2 OpenAM Configuration, User and Token Stores

OpenAM provides an embedded LDAP server that can be used for configuration, users and tokens. However, it is recommended for production that at a minimum the user and token stores are deployed to standalone instances of OpenDJ.

For the proof of concept, the embedded store was simpler and worked with no customisation. Using a standalone instance of OpenDJ for a production deployment is well understood and there is little benefit in demonstrating this.

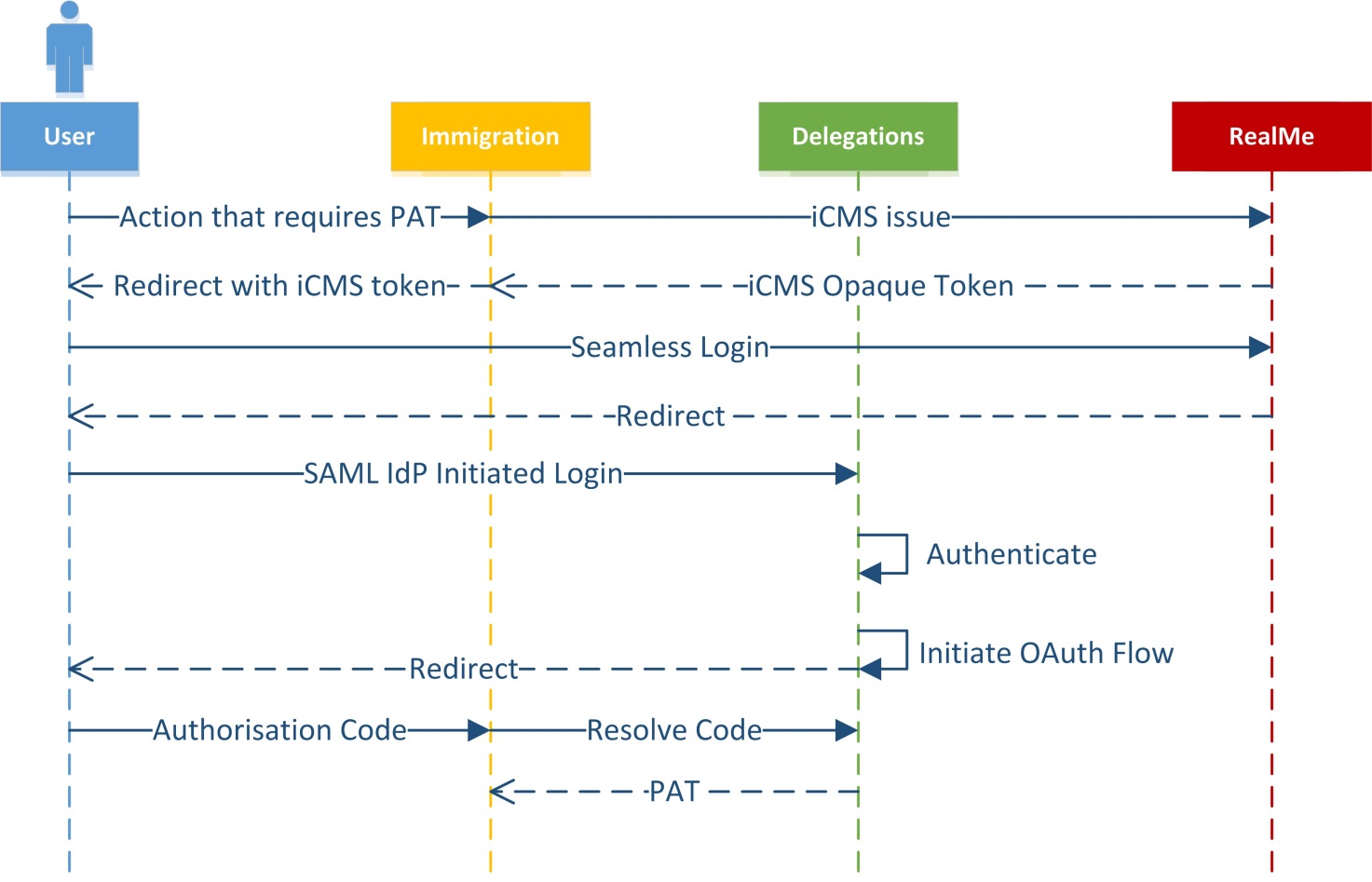
As part of defining the architecture for a production deployment, consideration should be made for standalone instances of OpenDJ for OpenAM’s user and token stores. See the OpenAM documentation for more information.

#### 8.6.3 Tokens

UMA relies heavily on OAuth tokens. Obtaining these tokens and storing them has a significant impact on the user.

8.6.3.1 Obtaining Tokens

Most scenarios in the proof of concept initiated an OAuth authorisation code to obtain a token. This involves redirecting the user to the delegation server with the correct request. To improve the user experience, the user would be redirected using Seamless login. However, this is a fairly complicated and brittle implementation. The flow is shown below:



While this works, there are a number of potentially unnecessary redirects, and it relies on the Delegations solution being able to initiate an OAuth flow to Immigrations.

Given that in this flow it was assumed that the user doesn’t need to consent, and the authentication at the service agency is sufficient, a simple option would be to provide an API to obtain tokens. This would look like the following diagram:



Note, that it may not be possible to return a PAT directly in this case. However, an authorisation code may be returned, and then resolved by the service agency instead.

8.6.3.2 Storing Tokens

Storing tokens and managing their lifetimes is important for security purposes. From the work on the proof of concept it appears the guidelines for each UMA token type is:

|  |  |
| --- | --- |
| Type | Lifetime |
| PAT | Lasts as long as the resource is available (long lived). However, if the token is no longer available, it is difficult to have the Resource Owner obtain a new one, because they are offline during most of the flow. |
| AAT | Lasts as long as the requesting party may wish to access resources (long lived). However, if not available, the requesting party could be asked to obtain a fresh token. |
| RPT | Lasts only for a short period of time to access the resource. |

Note that the PAT and AAT may use refresh tokens to increase security while stored for a long period of time. According to the UMA spec, the RPT cannot use refresh tokens.

All of the above tokens are bearer tokens, so care must be taken to ensure they are stored and transported securely. If a malicious user were to obtain one, they could impersonate actors in the system.

#### 8.6.4 Governance

There are probably several governance requirements that have not been explored in the proof of concept. For example:

* Auditing of users’ actions, and making these audits available to the correct parties
* Reporting of activity and state within the solution

OpenAM does provide some of these features out of the box, however there would likely be significant analysis and development required to implement these.

#### 8.6.5 Support and Administration

In developing a production solution, there are probably several support and administration scenarios that would need to be delivered. OpenAM provides some functionality, however there would need to be consideration for:

* The type of UI or API required to provide the functionality
* The scope of the functionality required
* The privacy and security requirements – especially if a cross-government solution was implemented.

#### 8.6.6 Non Functional Requirements

The proof of concept did not explore non-functional requirements of a production solution. However, the components within the solution are scalable and have been proven in high volume deployments. For example, according to ForgeRock, OpenAM can support millions of policies.

However, some of the workarounds used to provide the functionality are not scalable, for example, searching UMA policies. Practical alternatives would need to be investigated for a production deployment.

#### 8.6.7 Workarounds

8.6.7.1 Seamless Login

Seamless login is a critical part of delivering a good user experience in a cross domain RealMe environment.

There are two types of scenarios where RealMe’s Seamless Login Service provide a better user experience. Firstly, when a user is required to authenticate to different systems, they are only required to enter their credentials once. They can authenticate to additional systems using their session at the first system.

Secondly, if a scenario requires redirecting between two or more systems (and returning) during a flow (for example to collect OAuth tokens, complete invites etc.) one approach is to use Single Sign On –effectively leaving multiple user sessions open. This however may be confusing for the user and multiple sessions can be insecure if the user doesn’t correctly log out. Seamless login gives each system the ability to ‘hand off’ the user closing the session and redirecting them away.

While Seamless login is a straightforward process, for the Proof of Concept, it was decided to fake seamless login rather than add additional complexity to the applications. This is possible because the test applications and the delegation system are in the same privacy domain and therefore share FLTs.

There are some limitations to Seamless login that are well understood, and the proof of concept doesn’t do anything outside of these limitations. (For example, the relay state is a maximum of 80 characters).

This workaround has little impact on the proof of concept, but enables smoother demonstrations and flows that are more representative of the likely final outcome.

8.6.7.2 API Authentication

There are several APIs provided by the delegation system that require a temporary session in OpenAM. Either the user is identified by a trusted client, or the user is interacting in some other means.

To create the session the API application authenticates the user using an OpenAM module. Usually this would either be an appropriate standard module or a custom authentication module. However, for the proof of concept it was simpler to use the standard username/password module and ensure that all users have the same OpenAM password.

This workaround will have very little impact on the proof of concept.

Another workaround was required to return an identifier for the user to the Health application (because they don’t share authentication methods a common identifier is needed). Ideally an OpenID connect token would be returned by due to an issue in OpenAM it was simpler to create a custom scope and return this.

8.6.7.3 RealMe Identifiers

In the improved experience that was explored in the final sprint, a RealMe Attribute Service was created. This provides user friendly identifiers so users can find and identify users they have shared with.

However, OpenAM currently only uses the user ID for display, so this usable attribute had to be stored as the user’s UID. This leads to issues if the name contains spaces or mixed case. However it is still important to demonstrate the usefulness of RealMe providing such an identifier.

In a production deployment, the identifier would probably be stored in a different attribute, and OpenAM would ideally support using that attribute for display purposes (or the OpenAM interface not used). Other challenges with the RealMe attribute service include:

* + - Naming of headless users
    - Allowing users to have nick names for contacts (display of this in multiple places)
    - Using RealMe attributes or contacts within service agency flows might not be clear where these come from.

8.6.7.4 Automatic Sharing – Careers

As part of the improved Careers scenario, the team needed two things:

* When Caleb requests Alex’s record, the request appears to come from Better Builders
* Once Better Builders receives the delegation, it is automatically shared with Caleb

Because these features aren’t supported in OpenAM, these were implemented with workarounds in order to demonstrate what a complete experience would be.

This involved:

* Making the initial request as Better Builders. This is impersonation, and not recommended in a production deployment.
* Automatically sharing when a user approves access. This was done by calling out to careers from the dashboard when a user approves access. This wouldn’t be feasible in a production deployment because:
  + The delegations application might not rely on Careers to find out who works for Better Builders
  + The automatic sharing only works on request based (approval) access.

### OpenAM Defects

The following defects were raised during the proof of concept

|  |  |
| --- | --- |
| ForgeRock JIRA | Description |
| OPENAM-6739 | Creating UMA policy as amadmin doesn't show in user's resources |
| OPENAM-6666 | Re-shared resource that is revoked by resource owner, re-shared user still has access |
| OPENAM-6546 | Deleting UMA resource does not delete policy and breaks XUI |
| OPENAM-6544 | Duplicate UMA resources after sharing |
| OPENAM-6543 | need\_info is returned from when attempting to get an RPT |
| OPENAM-6406 | Cannot authenticate using MSISDN module in nightly build |
| OPENAM-6384 | XUI: Sharing resource twice (with another user) fails |
| OPENAM-6374 | Registering UMA resource sometimes gives error |
| OPENAM-6385 | Revoking access to individual resource using XUI fails |

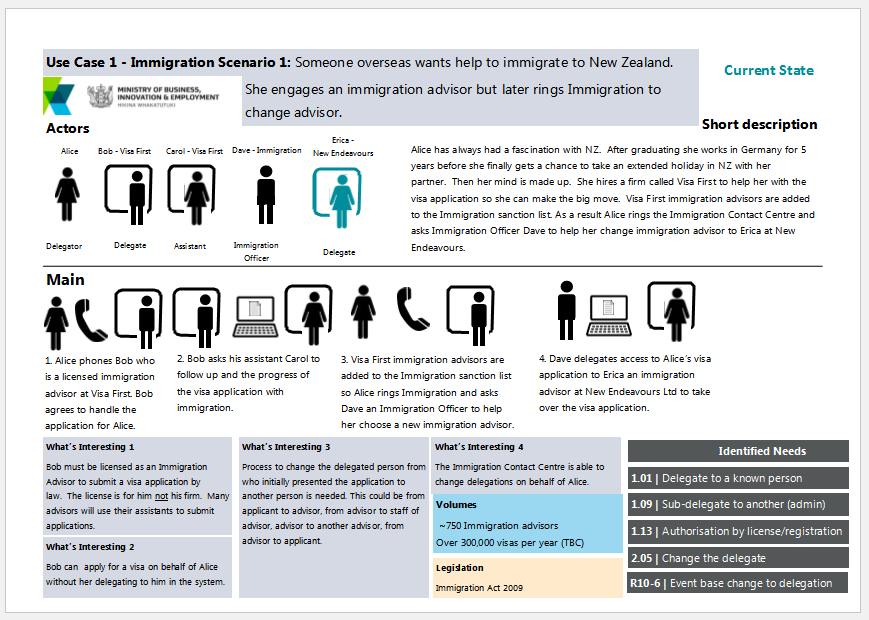
Appendix 3

Proof of Concept Demonstrations

## Appendix 3: PoC Demonstrations

### Immigration demonstration

**Scenario overview:** *Immigration advisors are able to create applications on behalf of applicants that do not have an online presence. Applicants can create an online presence if the wish and take over the management of their application.*

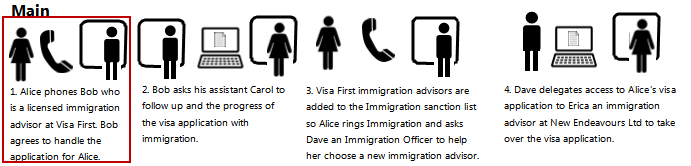


**Scenario Break down**

The main flow above is broken down into each of the individual steps, with screenshots and written steps of the process.

**Step 1**: *Bob creates a visa application on behalf of Alice*

* The key point in this step is that Alice does not have an online presence with Immigration. This is the headless user case.

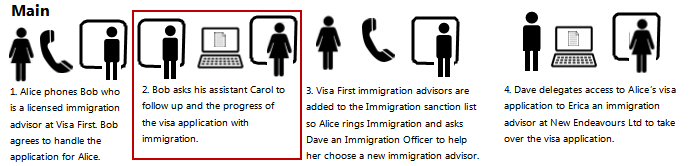


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I create an INZ application on behalf of an applicant?   1. Log in to the INZ demo application as an immigration advisor. 2. Click on the ‘Create an application for someone else’ link. 3. Enter the applicant’s email address. 4. An application will then be created with a temporary ID. |
|  |
|  |

**Step 2**: *Carol checks on the progress of Alice’s application on behalf of Bob*

* The key point here is that Bob has delegated his authority to his assistant Carol. Carol can see all applications that Bob has access to.

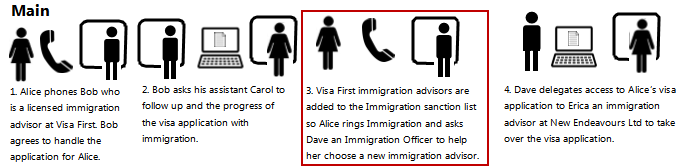


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | As an immigration assistant, how do I find an application?   1. Log in to the INZ demo application as a user. 2. Click on the ‘Search’ link. 3. Using the ‘search on behalf’ functionality, enter the ID of the advisor you are working on behalf of, and the ID of the application you are looking for and click the search button. |
|  |

**Step 3**: *Alice asks Dave to help her choose a new immigration advisor*

* The key point here is that immigration administrators have access to all applications.

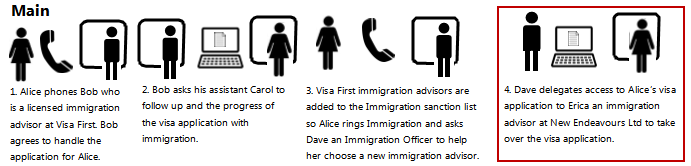


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | As an immigration administrator, how do I find an application?   1. Log in to the INZ demo application as an INZ administrator. 2. Search for an application. The search results display resource ID. 3. Copy the resource ID, then log out and return to the POC links page. |
|  |

**Step 4**: *Dave delegates access to Alice’s application to Erica, the new immigration advisor*

* The key point in this step is that immigration administrators have the ability to change delegations on behalf of applicants.

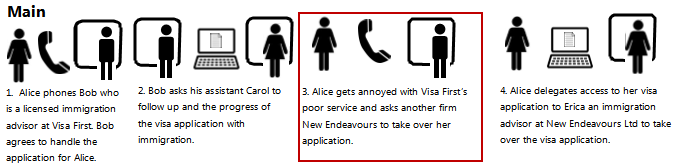


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | As an immigration administrator, manage the associated delegations on behalf of someone?   1. Go to the ‘AS admin app’. 2. Enter the resource ID and click share, an invite code will be displayed on the screen. 3. Give this code to the new INZ advisor (Erica). This process revokes access to the previous delegate/sub delegate (Bob and Carol). |
|  |

**Alternate step 3**: *Alice finds a new immigration advisor to take over her visa application*

* The key point here is that although Alice began the scenario without an online presence, she can at any stage create one with the email that was sent to her when her application was created.

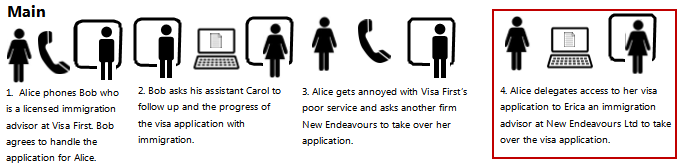


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | As an applicant, how do I get access to manage my application?   1. Click on the link in the email you received when you application was created for you. 2. Your application will then be linked to your account. |
|  |

**Alternate step 4**: *Alice delegates access to her visa application to Erica*

* Once Alice has an online presence, she can manage who has access to her application by managing the delegations.

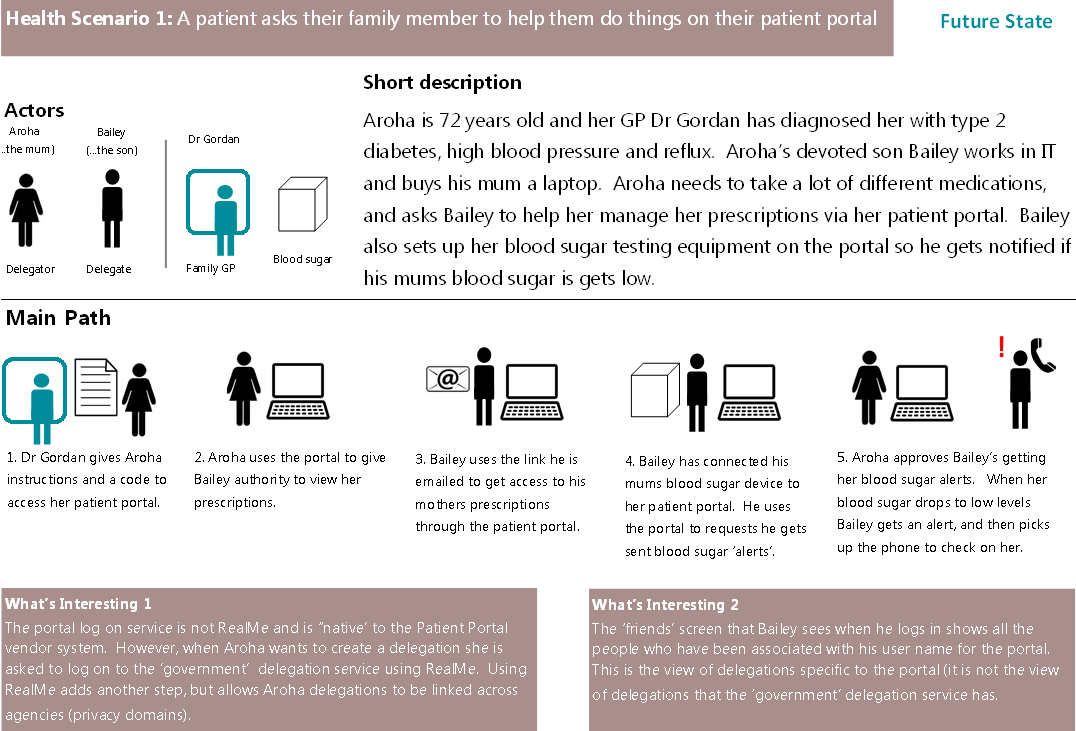


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | As an applicant, how do I manage who can access my application?   1. From the INZ demo application home page, click on the delegate access link. 2. Click on the ‘share via code’ button. 3. Give the rendezvous code to the person you wish to delegate access to. |
|  |

### Health demonstration

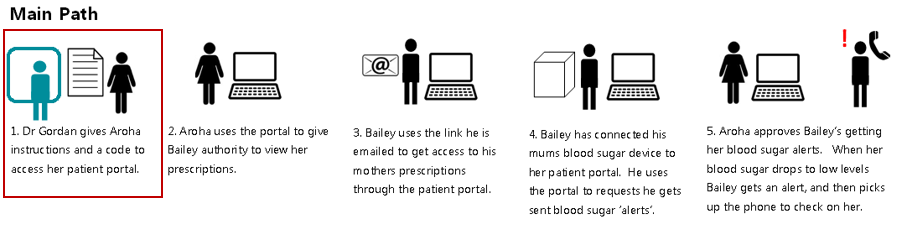
**Scenario:** *A patient is able to delegate access to a family member and a family member is able to request access to a resource of the patient.*



**Scenario break down**

The main flow above is broken down into each of the individual steps, with screenshots and written steps of the process.

**Step 1**: *Dr Gordan gives Aroha access to her patient portal*

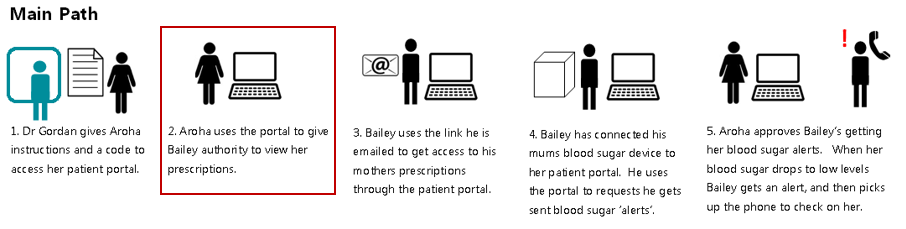


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I get access to my patient portal?   1. Visit the health portal demo application and click on the ‘Register’ button. 2. Enter the code given to you by your doctor. 3. Log in with RealMe, you will now have access to your patient portal. |

**Step 2:** *Aroha lets Bailey view prescriptions*

* This is a demonstration of a standard delegation using either an email address or RealMe contact information.

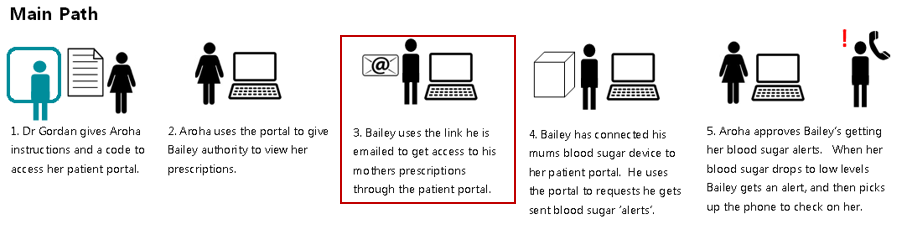


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I delegate my prescriptions to someone?   1. Log in to the Health Portal demo application. 2. Click on the ‘View My Prescriptions’ link. 3. Click one of the two sharing options. 4. Using RealMe login and access the delegations service. 5. Enter email address of a person you would like to share your prescriptions. 6. Click on ‘Share’ button, a success page will be displayed. An email request has been sent to the person you wish to share your prescriptions with. |
|  |

**Step 3**: *Bailey gets access to his mother prescriptions*

* This is a demonstration of how someone could accept a delegation.



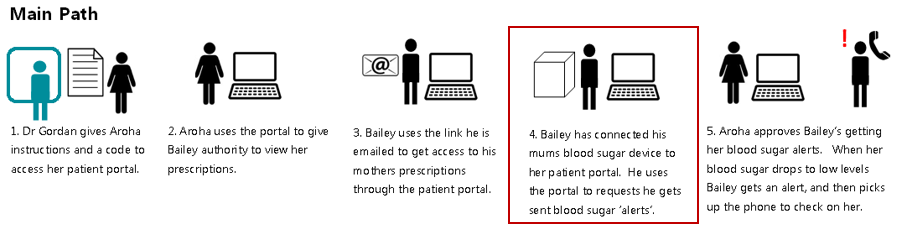
**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I accept delegation to manage someone’s prescription?   1. Click on the URL link in the email request sent to you, to get access to someone’s prescriptions. 2. Log in to the Health Portal 3. An ‘Invite Completed’ message will be displayed. 4. Return to Portal Home page and the relationship with the friend is displayed. |

|  |  |
| --- | --- |
|  |  |

**Step 4**: *Bailey asks for his mother’s blood sugar alerts*

* Bailey can request access to the blood sugar alerts resource. This can only be done if Bailey has already been delegated access to a resource that Aroha owns, in this case Bailey has access to her prescriptions.

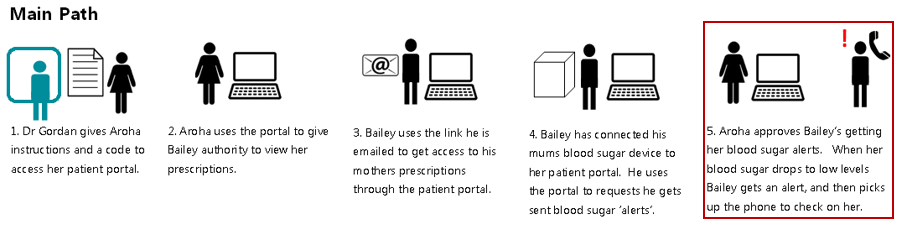


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I register interest in blood sugar notification?   1. Log in to the Health Portal demo application. 2. Click on your friend’s name. 3. Enter your mobile number and click on ‘Request Notification’ button. 4. A confirmation page will be displayed. |

**Step 5**: *Aroha approves Bailey’s request for blood sugar alerts*

* This is an alternative to delegating access. In this case, the delegation was initiated by Bailey in the form of a request and all Aroha has to do is approve it.

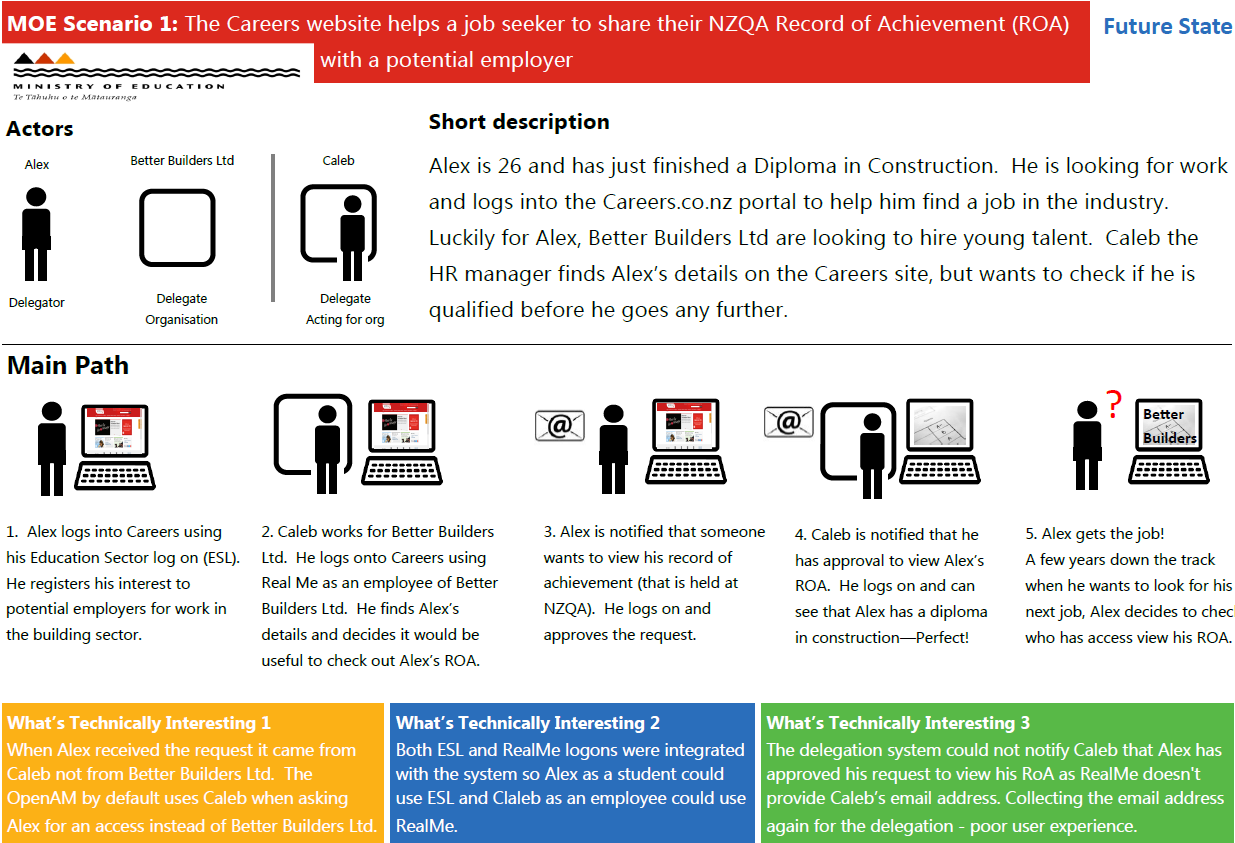


**Screenshots and Process Steps**

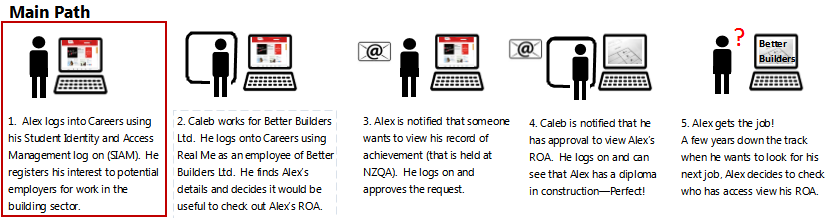
|  |  |
| --- | --- |
|  | How do I delegate my blood sugar notification to someone?   1. Log in to the Health Portal demo application. 2. A message with Request Approval link is displayed 3. Click on the link and approve the request. 4. You are redirected back to Home Page, the request has been approved. |
|  |

### MoE (Careers & NZTA) demonstration

**Scenario:** *The Careers website helps a job seeker to share their NZQA Record of Achievement (ROA) with a potential employer*



**Step 1**: *Alex creates Careers account*

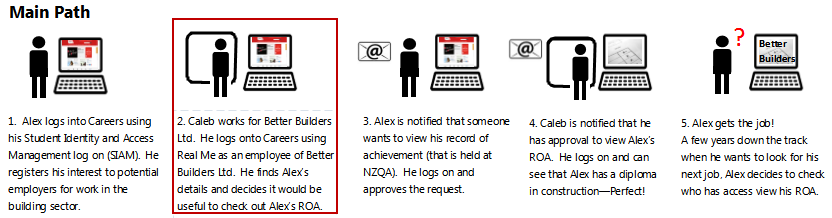


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I set up my Careers account as a job seeker?   1. Log in to the Career Portal demo application using ESL (Education Sector Logon). 2. Enter your Full name and select an industry you would like to work in. 3. Click Save, your selection is displayed on your Careers Home page. 4. Click on “Manage my profile” to change your industry selection. |

**Step 2**: *Caleb registers interest*

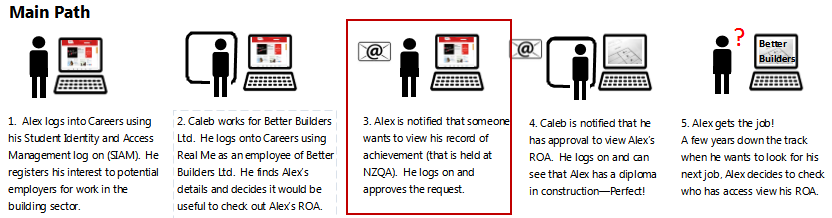
* The key point here is that the interest in Alex is on behalf of Better Builders. So Alex will see that Better Builders would like access to his ROA, not Caleb.



**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I set up my Careers account as an employer?   1. Log in to the Career Portal demo application using RealMe login. 2. Enter your full name and the name of the company you work for and select save to complete your account profile setup. 3. Click on “Manage my profile” to update your name.   How do I register interest to view a candidate’s Record of Achievement?   1. Log in to the Career Portal demo application using RealMe. 2. Click on ‘Search for job candidates’ link to. 3. Select an industry and click ‘Search’ button to view a ‘Potential candidates’ page with a list of job seeker names. 4. Click on the candidate name to register interest to view his/her ROA, success page is displayed and email is set to the candidate. |
|  |
|  |

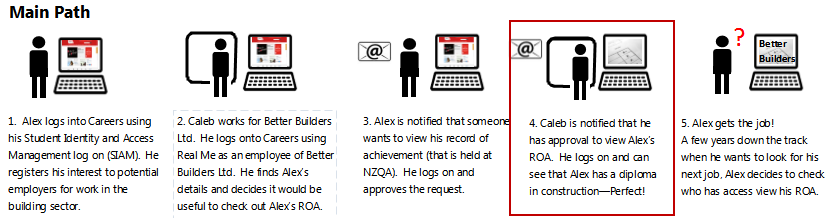
**Step 3**: *Alex accepts request*



**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I give someone access to my record of learning?   1. Click on the link in the email you received. 2. Log in with ESL. 3. Approve the pending request(s) |
|  |

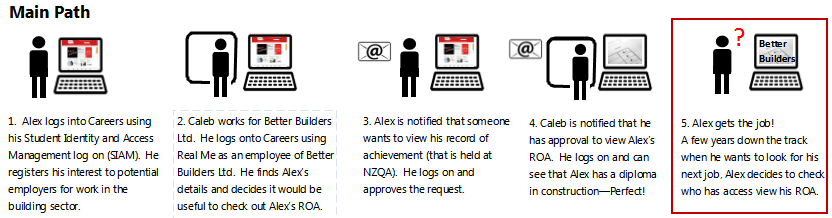
**Step 4**: *Caleb views Alex’s Record of Achievement*



**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I see ROAs that I have been granted access to?   1. Once you have received a notification email, login to the Careers site using your RealMe login. 2. Click on the candidate’s name. 3. The candidate’s record of learning will be displayed. |
|  |
|  |

**Step 5**: *Alex views a history of what he has delegated (shared)*

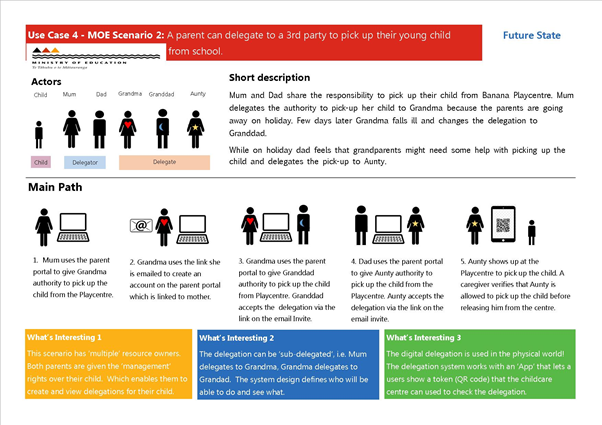


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I see who has access to my ROA?   1. Log in to the delegations dashboard. 2. Click on your record of achievement resource. 3. Your resource and the people that have access to it will be displayed. |

### MoE (Early Childhood) demonstration

**Scenario:** *A parent gives a permission to pick up his/her child to someone else. Early childhood centre can verify the person.*

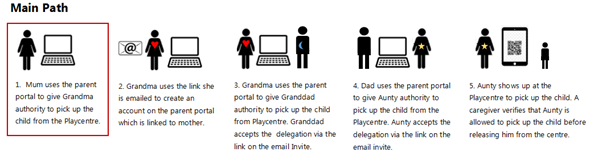


**Scenario break down**

The main flow above is broken down into each of the individual steps, with screenshots and written steps of the process.

**Step 1:***Mum gives an authority to pick up her child to Grandma*

* This step shows ‘multiple’ resource owners. Mum and Dad both are given the management rights to manage the pick up of their child.

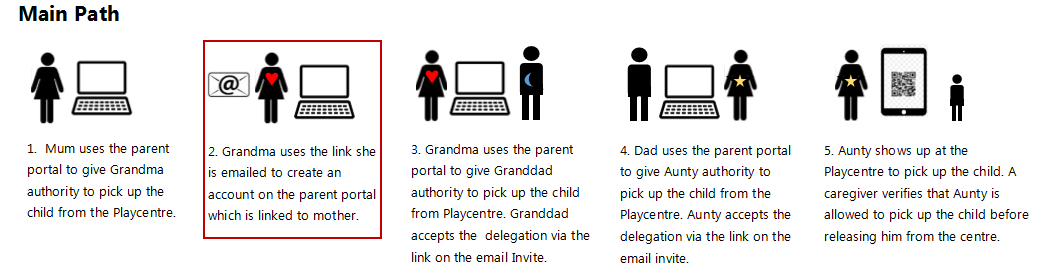


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I delegate authority to pick up a child?   1. Log in the Parent Portal using RealMe. 2. Click on ‘Manage authority to pick up’ to view existing authorisations or request a new authority. 3. Enter name and email address of the person you would like to have an authority to pick up your child. 4. A success message will be displayed when you click on ‘Delegate’ button. |
|  |
|  |

**Step 2:** *Grandma creates Parent Portal account*

* This steps shows the creation of ‘sub-delegation’ where mum delegates the authority to pick up her child to Grandma. In this case, mum can see dad and grandma are set up to pick up the child.

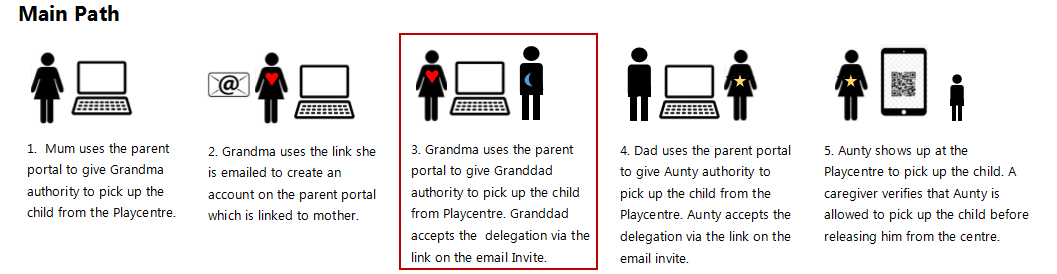


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I accept the delegation to pick up someone’s child?   1. Click on the URL link in the email invitation sent to you by the delegator. 2. Login using RealMe to access the Parent Portal account. 3. Set your profile image by taking photo or using the default image, click on ‘Save’ 4. Parent Portal home page will be displayed with the child’s name and links to available functions. 5. User is linked to the person who sent the delegation invitation |
|  |
|  |

**Step 3:** *Grandma gives the authority to Grandad*

* This step shows Grandma creating sub-delegation by inviting Grandad to have access to pick up the child. In this case only Grandma can see Grandad’s authority. Mum and Dad can’t see this delegation.

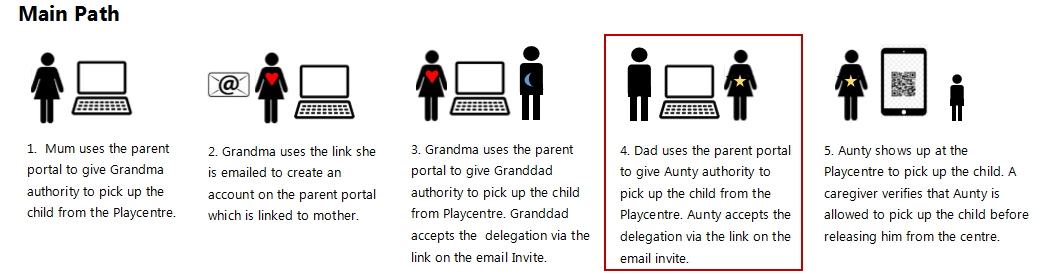


**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I delegate further the authority to pick up a child?   1. Log in the Parent Portal using RealMe. 2. Click on ‘Manage authority to pick up’ to view existing authorisations or request a new authority. 3. Enter name and email address of the person you would like to have an authority to pick up your child. 4. A success message will be displayed when you click on ‘Delegate’ button. |
|  |
|  |
|  |
|  | Grandma can see the delegation she has given to Grandad. |
|  | Mum can only see the delegation she gave to Grandma but she can not see the delegation Grandma gave to Grandad. |

**Step 4:** *Dad gives the authority to Aunty*

**Key points:**



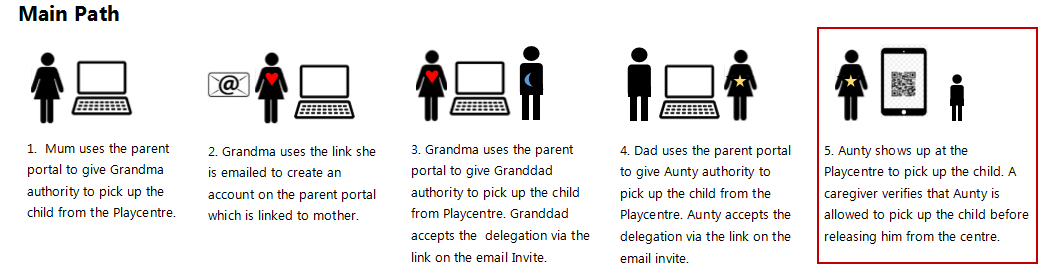
**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I delegate authority to pick up a child?   1. Log in the Parent Portal using RealMe. 2. Click on ‘Manage authority to pick up’ to view existing authorisations or request a new authority. 3. Enter name and email address of the person you would like to have an authority to pick up your child. 4. A success message will be displayed when you click on ‘Delegate’ button. |
|  |
|  |
|  | How do I accept the delegation to pick up someone’s child?   1. Click on the URL link in the email invitation sent to you by the delegator. 2. Login using RealMe to access the Parent Portal account. 3. Set your profile image by taking photo or using the default image, click on ‘Save’ 4. Parent Portal home page will be displayed with the child’s name and links to available functions. 5. User is linked to the person who sent the delegation invitation   Note: Dad can see his and Mum’s delegation of the authority. Aunty only sees her authority. |
|  |
|  |

**Step 5:** *Aunty can prove she is authorised to pick up the child*

**Key points:**

* This step shows Aunty using a token (OR code) representing the digital delegation. The childcare centre uses an ‘App’ to check the delegation.



**Screenshots and Process Steps**

|  |  |
| --- | --- |
|  | How do I verify myself at the ECP when picking up the child?   1. Log in the Parent Portal using RealMe. 2. Click on ‘Get pick up code’ to view the QR code and permission status is Allowed. 3. Present the QR code to a staff at the Playcentre.   How does Early Childhood Playcentre verify the authority to pick up a child?   1. Using a QR scanner on your device, scan the QR code which a person picking up a child presented to you. 2. Check Access page is displayed. 3. Verify the permission and the photo of a child and the person pick up the child. |
|  |
|  |
|  |

## Appendix 4: R9 Delegations

### Summary

The key findings of the R9 Delegations Project can be found in the Delegation Capability August 2015 Snapshot report. This report summarises the customer research, agency research, problem and benefit analysis, policy and technical Findings.

The key findings were:

* Delegations are common for businesses and individuals
* Setting up delegations costs business customers millions of dollars
* The challenges of delegation are largely hidden from end business customers but are acutely felt by professional intermediaries (lawyers, accountants and advisors)
* It is common form Intermediaries (delegates) to act for the same business with many different government agencies and services.
* Government agencies have common needs for managing delegations, but each agency’s delegation capabilities are at widely different stages of maturity.
* There are a range of technology options that support a delegation capability.
* The key enabler for reuse of delegations is the establishment of agreements and standards supporting authorization rules to establish trust across agencies and service providers.

### Delegations Framework

The R9 Delegations project identified a framework of key considerations for government delegations. The areas investigated by the Proof of concept are highlighted below:



**KEY Built in PoC In report**

### Priority areas mapped to Framework, Requirements and PoC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Priority Areas**  **Source**: Agency Workshops | | **R9 Framework**  **Source**: R9 Findings | **Requirement Name**  **Source**: Agency scenario pack | **Investigated in PoC by...**  **Source**: Jira Proof of Concept |
| High  (19) | 1. Authorisation across services that are across different agencies | **REUSE**  ***Service Reuse*** | Share a delegation with another agency | ***Sprint 4*** *Careers* A delegations created through the Careers website is used at NZQA |
|  | Assess a delegation set up at another agency. |
|  | Request further information before accepting another organisations delegation. | Not demonstrated |
| Low  ( 6 ) | 2. Authorisation across channels (digital and non-digital) | **REUSE**  ***Channel Reuse*** | Cross Channel Delegation Sharing | ***Sprint 5*** *MoE*: A delegation set up on a parent portal creates a QR code that the childcare centre can read to confirm who the pickup person is when they arrive at the childcare centre |
| Med  (18 ) | 3. Multiple delegators (resource owners) who all have equal rights | **MULTIPILICITY**  ***Many*** | Shared power to delegate | ***Sprint 5*** *MoE* Both mum and dad can set up and manage delegations for their child.  Refer to [3.7 Equal Ownership](file:///C:\Users\McGreggora\Desktop\ProjectClosureReport_PoC%20Delegations(1).docx#_Equal_Ownership) |
| Med  (15 ) | 4. Granularity of access to resources by the RS/AS (e.g a health portfolio and part of that portfolio) | **GRANULARITY**  ***Specified*** | Set access level against a delegation | ***Sprint 2*** *Immigration*: (Granularity controlled by agency - not delegations system). |
|  | Access to specific products or services | **Sprint 3** *Health*: Access could be controlled to just notes, or just blood sugar alerts. |
| High  ( 22 ) | 5. How the RO defines the identity of the RqP | **SET UP**  ***Consented*** | How to identification digitally who you are delegating to | ***All Sprints***: Use of a rendezvous code. Refer to section [3.3.1 Discovering Users to Share With](#_Rendezvous/Invite) |
| High  (19 ) | 6. How the RO is managed when it is missing/digitally absent | **SET UP**  ***Seamless*** | Assume control when missing digital ID | ***Sprint 2*** *Immigration:* Immigration advisor is set up as the delegate of their client as part of the application creation process. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Priority Areas**  **Source**: Cross Agency & Forge Rock Workshops | | | **R9 Framework** | | **R9 Requirement**  **Source**: Agency scenario pack | **PoC User Stories**  **Source**: Jira Proof of Concept |
| Med  ( 13 ) | 7. Managing a group membership claim by an RO or an RqP (outside UMA, but will need to couple with UMA solution– probably on paper) | | **SET UP**  ***Asserted*** | Authorisation through licensing, registration or membership of a professional body. | | Not demonstrated– See Findings section |
| Med  ( 18 ) | | 8. How to present a global view (i.e not a service based view) of a customer’s authorisations, that is manageable to the user | **CONTROL**  ***Self-manage global*** | | View delegations | ***All Sprints***: Delegator could log on to the Delegation system to view. |
| Med  ( 11 ) | | 9. How RqP self asserts an authorisation to act (access Resources) without RO consent. Assuming, the RqP is form a professional group trusted by the RS. | **SET UP**  ***Asserted*** | | Authorisation through licensing, registration or membership of a professional body. | Not demonstrated – [See Findings](#_Findings) section |
| Med  ( 18 ) | | 10. RqP can sub-delegate to another, including for both asserted or granted authorization. | **MULTIPILICITY**  ***Self-Administered*** | | Delegate to a known business (and known person) | ***Sprints 2***  *Immigration:* advisor to assistant |
|  | | Delegate to a known business (but unknown person) | ***Sprint 4***: *Careers*: Business to staff |
|  | | Set up a delegation for a role within a business | ***Sprint 4***: *Careers*: Business to staff |
|  | | Sub-delegate to an another | ***Sprint 5:*** MoE -Mum to grandma to grandpa |
| Med  ( 16 ) | | 11. Client is not created by government and is provided by a 3rd party (i.e. Xero web portal, Apple mobile app, Fit bit or blood sugar device). | **REUSE**  ***Channel Reuse*** | | Transact through third party software | ***Sprint 3*** *Health*: Both the patient portal and the blood sugar monitor are created by third party vendors. |

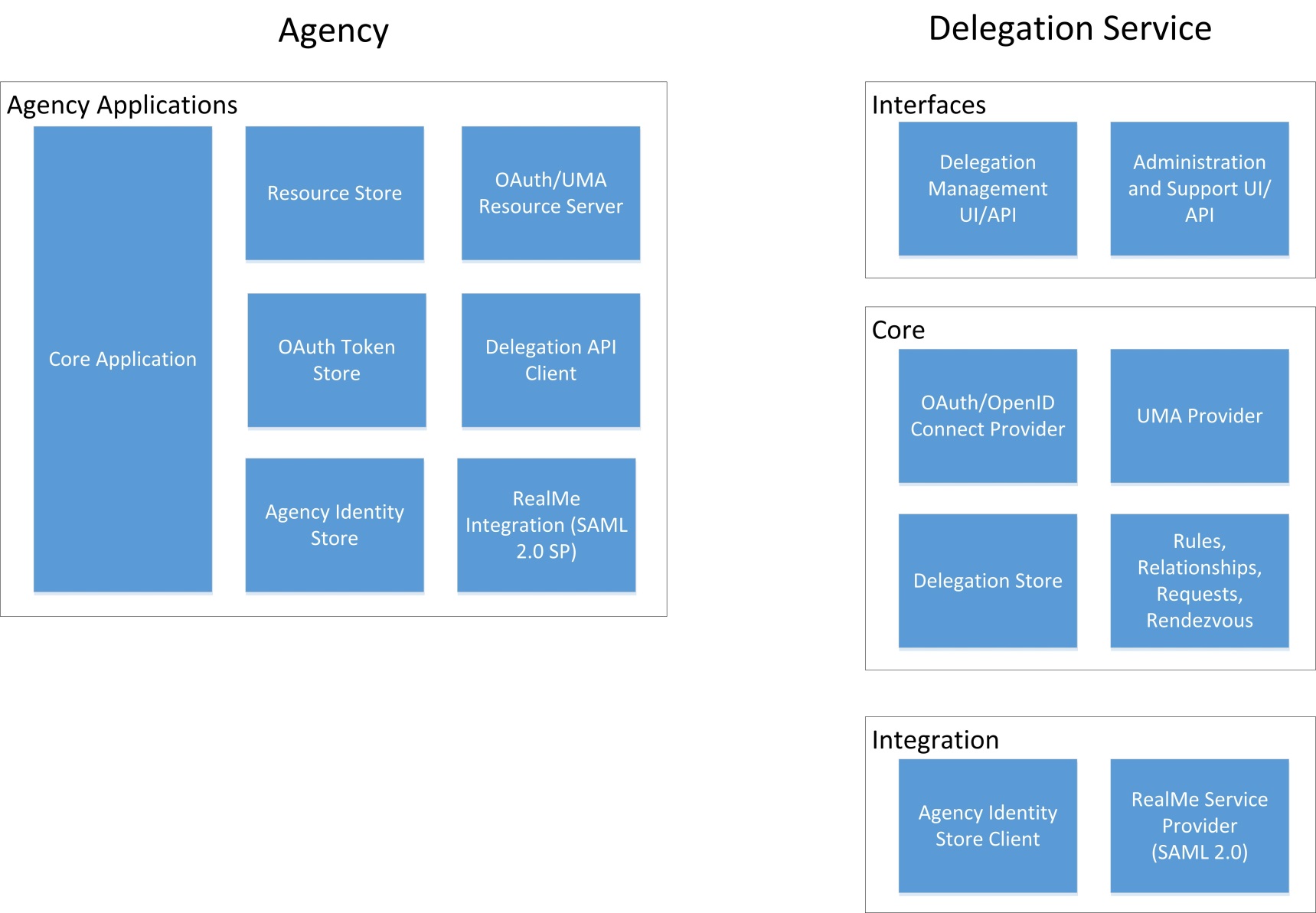
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Priority Areas**  **Source**: Cross Agency & Forge Rock Workshops | | **R9 Framework** | **R9 Requirement**  **Source**: Agency scenario pack | **PoC User Stories**  **Source**: Jira Proof of Concept |
| Low  ( 9 ) | 12. Gateway for legacy and Greenfields services | Agency specific Implementation issue (Not in framework). | Legacy systems | This was not investigated during the proof of concept. Possible approaches to supporting legacy services are:           Deploying policy access points (i.e. agents) in front of legacy applications           Providing plugins for existing access control systems  The best approach will depend on the amount of change that can be made to legacy applications  For greenfield solutions, integration libraries and collateral (such as integration documentation and checklists) may wish to be provided to ensure smoot on boarding. |
| Low  ( 8 ) | 13. Hold RqP request/claim until verified (i.e RO approves or legal authority is recognized) | **SET UP**  ***Consented*** | Delegation held until user consents | ***Sprint 3*** *Health*: Sons request to view blood sugar is held until mother approves  ***Sprint 4*** *Careers*: Employers request to view Record of Achievement is held until approved. |
|  | Authorise a delegation with legal proof | Not demonstrated |
| Low  ( 4 ) | 14. RO is notified only after the RqP has set themselves up with access. | **SET UP**  ***Consented*** | Confirm delegation after it is set up | Not demonstrated – [See Findings](#_Findings) section |
| Med  ( 18 ) | 15. RO revokes a policy for a specific RqP, when it was established by the RqP | **CONTROL**  ***See & Change*** | None identified | ***Sprint 2*** *– Immigration:* Immigrant (who did not set up application) can take ownership of it back from Advisor. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Priority Areas**  **Source**: Cross Agency & Forge Rock Workshops | | **R9 Framework** | **R9 Requirement**  **Source**: Agency scenario pack | **PoC User Stories**  **Source**: Jira Proof of Concept |
| Med  ( 10 ) | 16. RS can restrict the number of RqP who can be authorized at any one time, including multiple or unlimited sub-delegates of RqP. (for example IR only allows one accounting firm to act for a RO at any time). | **MULTIPILICITY**  ***Many*** | Control how many people can be delegated to concurrently | This was not investigated during the proof of concept. The challenges for this would be:   * How the RS specifies the constraints per resource (does the OAuth 2.0 Resource Registration Set allow additional parameters) * How the AS enforces these constraints. OpenAM may not support these constraints and customisation or workarounds may be difficult.   Note that determining the number of parties that have access can be difficult if delegations can be re-shared, or access is based on relationships or other rules. Refer to  [See Rules and Relationship](#_Rules_and_Relationships) section |
| High  ( 21 ) | 17. Expiration of an authorization. (RO set policy/rules and RS set policy/rules). | **GRANULARITY**  ***Rules*** | End a delegation after a specific time or date | Not demonstrated – refer to  [See Rules and Relationship](#_Rules_and_Relationships) section |
|  | End a delegation after a staff change |
|  | End a delegation due to coming of age |
|  | Event based delegation |
|  | Delegation Disputes |
| Low  ( 6 ) | 18. Temporary/ Emergency access. Have different rules during an ‘emergency’ period. (e.g initial period during a serious injury claim at ACC or temporary foster care of a child at MSD) | **GRANULARITY**  ***Rules*** | Get access to act for another without their consent in an emergency | This was not investigated during the proof of concept. Options for this include:   * Implementation on the service agency. In other words, don’t use UMA if the user requests emergency access, have the service agency decide and allow access. * Implementation in the Authorisation Server. This might be done with an additional claim from the Client the AS uses during policy evaluation. From the RS point of view, the request is the same as usual. * Implementation on the service agency, but auditing access using an API on the delegation service.   Things to consider when choosing an approach would include auditing and visibility of emergency access to the user, availability in emergency situations when the delegation service is unavailable, who determines when emergency access is required. |

## Appendix 5: Reference Architecture

### Decentralised

The logical representation of the architecture for a decentralised agency implementation of a delegation service is shown below.



### Centralised

The logical representation of the architecture for a centralised government implementation of a delegation service is shown below.

