Dept of Finance / GovCMS

Deliverable component of

PQRT210001389 GovCMS DXP Proof of Concept Project #1

GovCMS Rules-as-Code Proof-of-Concept Whitepaper

September 2022

Document version: 1.2

Authors: Pia Andrews, Mark Calvert, Alfred Deeb, Con Fountas, Suchi Garg, Emil Jeyaratnam, Phillipa Martin, Stuart Rowlands, Steven Worley

SALSA DIGITAL PTY LTD
Suite 24
3 Albert Coates Lane, Melbourne, VIC 3000
p +613 9910 4099
e info@salsadigital.com.au
w www.salsadigital.com.au

Table of Contents

Background

In April 2022, Salsa Digital submitted a response to the *Whole-of-government GovCMS digital experience (DXP) and content personalisation services* REOI and proposed an innovative Rule-as-Code (RaC) managed service.

While RaC is not typically associated with a traditional understanding of a personalisation service, on closer inspection RaC does meet the core requirement and even goes much further.

Our RaC proposal included an end-to-end RaC service that ranged from Professional Services to help customers break down and codify their rules, through to a hosted, technical solution that integrates with the GovCMS platform.

Dept of Finance approved Salsa's proposal as one of four vendors to progress to the sponsored, proof-of-concept (PoC) phase.

For the Salsa RaC PoC, Finance asked us to address the following questions:

a. Am I up to date with my COVID vaccinations?

b. Am I eligible for a COVID vaccination?

This document outlines the findings and recommendations from our experiences in developing the GovCMS RaC Proof-of-Concept.

Vision

In a presentation in early 2022 to the GovCMS community, Sharyn Clarkson, the Dept of Finance Assistant Secretary for the Online Services Branch, outlined an ambitious, long-held vision she has for citizen-centric user journeys. She described a uni student who was pregnant and was in the middle of a relationship breakdown. Sharyn talked of her vision where government could provide a simple pathway for this person to easily find out about entitlements relevant to her specific needs that transcended government agency boundaries. Sharyn had hoped the structural boundaries preventing user journeys from being mapped would have been addressed in the past 5 years but, alas, very little progress has been made.

Sharyn went on to say the GovCMS program, with its 100 strong agency community, could play a key role in delivering on this vision by working together to put in place more of the pieces needed to realise this vision.

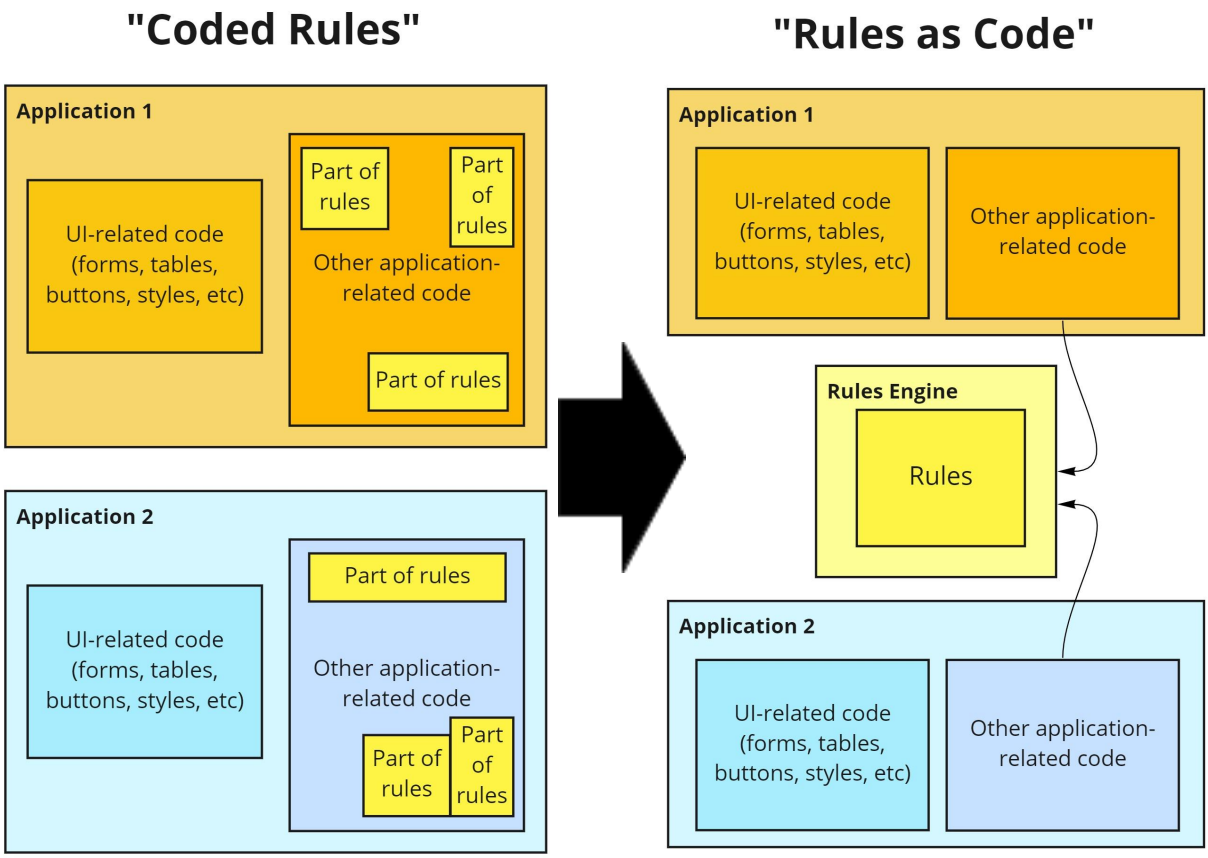
A well-executed Rules-as-Code implementation can deliver on some of Sharyn's vision and support wider democratic institutions and norms.

Rules-as-Code (RaC) is an emerging space within the public sector that drafts and turns legislation and other rules into machine-readable code to provide benefits for government,

citizens and broader society. According to the [OECD-OPSI 'Cracking the Code - Rulemaking for Humans and Machines'] report "RaC proposes to create an reusable, machine-consumable version of some types of government rules, to exist alongside the existing natural language counterpart. This provides a reference implementation for more consistent implementation of rules (especially where they are prescriptive), and to provide a means of traceability back to the legal authorities of decisions made in software. More than simply a technocratic solution, RaC represents a transformational shift in how governments could create, test and use rules, and how third parties could consume them."

RaC provides an easy-to-reuse and legislatively based rules utility that can provide user journeys that transcend government and jurisdictional boundaries, offering a truly citizen-centric or, as Sharyn put it, a 'simple pathway' where people can easily find out about rules and entitlements relevant to their specific needs.

Furthermore, the GovCMS program with its direct access to the 100 strong agency community and its proven success in managing a complex, public facing, shared-service web platform is well positioned to become the central hosting provider for a future whole-of-government RaC platform.



Who is it for?

1. GovCMS as the project sponsor
2. GovCMS customers (government agencies)
3. Australian citizens and residents (end users)

Current State

Pain Points around COVID vaccinations

1. Difficult for users to find the information that applies to them
2. Content duplication with no single source of truth
3. Rules and recommendations changing frequently
4. Changes require:
    - Identifying the source of truth
    - Understanding
    - Interpreting
    - Content creation (multiplied across various agencies and jurisdictions)
    - Many approvals
    - Deployment, publication and change management

Proof-of-Concept Scope

Finance requested that the following use case be addressed for this PoC.

"Using Health.gov.au (COVID vaccination eligibility), **as a site user** after answering a short series of questions, **I am shown** which vaccination options are available to me and the timing that applies before I am eligible, **so that I can** make an informed decision about my healthcare options."

Salsa (with permission from Finance) casted this request as follows:

> Using an demonstration version of a generic health dept COVID website, a site user will find answers to the following questions:
>
> a. Am I up to date with my COVID vaccinations?
>
> b. Am I eligible for a COVID vaccination?
>
> c. Do I have to be vaccinated for my work?

Proof-of-Concept Goals

Delivering Value - the Goals met by this PoC

1. Use a relatively “simple” rules set to explore the value of RaC for the GovCMS community
2. Create a demonstrator that showcases value to citizens/users, value to agencies and value to policy outcomes, namely an easy/quick way for citizens to check vaccination status
3. Explore the value of a single source of truth/centralisation of rules on a common platform (sovereign and secure) that can be used for many services by many agencies
4. Explore the value of exposing rules as an API, i.e. machine readable and actionable
5. Demonstrate multiple UX interfaces integrated with common RaC API
6. Explore benefits of streamlined change management for when rules change
7. Explore the value to user and personalisation options available from RaC into public-facing services and solutions
8. Creating a RaC foundation/pattern to build on, as a module for the GovCMS community to draw upon for websites, services, or policy development

Proof-of-Concept Findings

1. Mapping and codification of rules for the COVID Vaccination use case

The process

The process of mapping the rules can be broken into four main stages:

1. Analysing the rules and extracting key takeaways (key vaccination rules) in logical groupings (in this case, age was the clear grouping)
2. Creating a high-level model of rules/legislation that captures the age groups, the vaccination rules, eligibility criteria, and any calculation logic needed to determine up-to-date and recommended number of doses. This can be mapped as user journey maps/flow charts for each scenario (age, immunocompromised, other disabilities, etc.)

3. Creating a logic/decision model of the final rules mapping to work out 'up to date' status, 'eligibility' status, and 'mandatory' status for working in a particular sector. In this stage, all the necessary Variables and Parameters were defined.

4. Writing test cases defining inputs and expected results. Ideally, the test cases should cover all permutations of variables, focusing on threshold edge cases and crossing boundaries.

5. Writing rules statements that represent the COVID vaccination rules as a simple set of rules in plain English.

This mapping process was done on a Miro board. See ***Appendix 3 - The mapping process*** for sample mappings.

The challenges

The main challenges we faced during the mapping phase were:

- Not working closely with a policy expert from the relevant health authorities in interpreting rules resulted in simplified (and potentially inaccurate) codification of rules for the Proof of Concept. In a typical Rules-as -Code project, the mapping of rules/legislation would be done in close collaboration with a policy expert. This process will highlight any ambiguities and contradictions in the actual policies/legislations, as well as ensure accurate interpretation for codification.

- The realisation that 'Am I up to date?' and 'Am I eligible?' are identical journeys for most people (the exception is for those 30-49, who can be both up to date AND eligible) - we decided to present one user journey to cover both of these questions.

- Inconsistent rules across different sources that made it hard to work out what the final rules were.

- Complex mappings around different types of immunocompromising conditions and other medical issues that could affect risk of serious illness.

- While all sources were dated (which is great) at times it was hard to work out how the source/information related to one another. Was all 'old' stuff no longer relevant? Were some sections still valid but others not?

The outputs

The main output in this case was the Miro board with the rules mapped in both 'journey' format and in variable format showing which parameters make you 'up to date' and/or 'eligible'.

We also created a set of rules statements. Please see ***Appendix 4 - Rules statements*** for the rules statements.

2. OpenFisca Rules Code

The requirements

To provide a structured scalable programmatic interface which can be utilised by many downstream applications to perform calculations and evaluate data presented to idempotent results.

The system would take the rules mapping exercise outlined above which defines regulator requirements for the system, it outlines the parameters required and the variables that will be presented by the system to a requesting client.

The system would need to support effective change management processes and allow clients to choose how to make requests.

The challenges

During the proof-of-concept OpenFisca presented some challenges:

- Adequately capturing complex regulatory requirements and being able to describe them effectively in code
- Complex data modelling requirements to be able to collect enough data to make decisions
- Limited documentation from the OpenFisca open source community
- Vectorial based computation algorithms which are geared toward numerical calculations - requires a change to the way algorithms are written when trying to determine the rules.
- Deployments and code management processes to ensure effective controls are in place when dealing with sensitive computation algorithms

The outputs

The completed proof-of-concept delivered an OpenFisca instance that is geared toward making decisions about COVID vaccination status. The output investigated:

- Effective source code management strategies when dealing with rules-as-code
- Complex data modelling routines; how to effectively accept the data required to make calculations

- API management process and policies to enable truly decoupled development practices
- Strong process for the rules mapping exercise to assist in the translation from legislation to code

Please refer to ***Attachment 02 - About OpenFisca*** for more detail about OpenFisca and how it fits into the overall solution.

3. Drupal OpenFisca Webform module

The requirements

To provide a flexible, content-managed way for site owners to build journeys and integrate with OpenFisca to perform calculations. The output needed to meet the following requirements:

- Work with the current contributed module set of GovCMS SaaS
- Limit reliance on contributed modules to reduce complexity and maintenance burden of the solution

We needed to ensure that the solution was presented as a framework, rather than an implementation, to allow the patterns and principles learned during the PoC process to influence the future supportability of the solution.

The challenges

The main challenges that were aimed to be resolved:

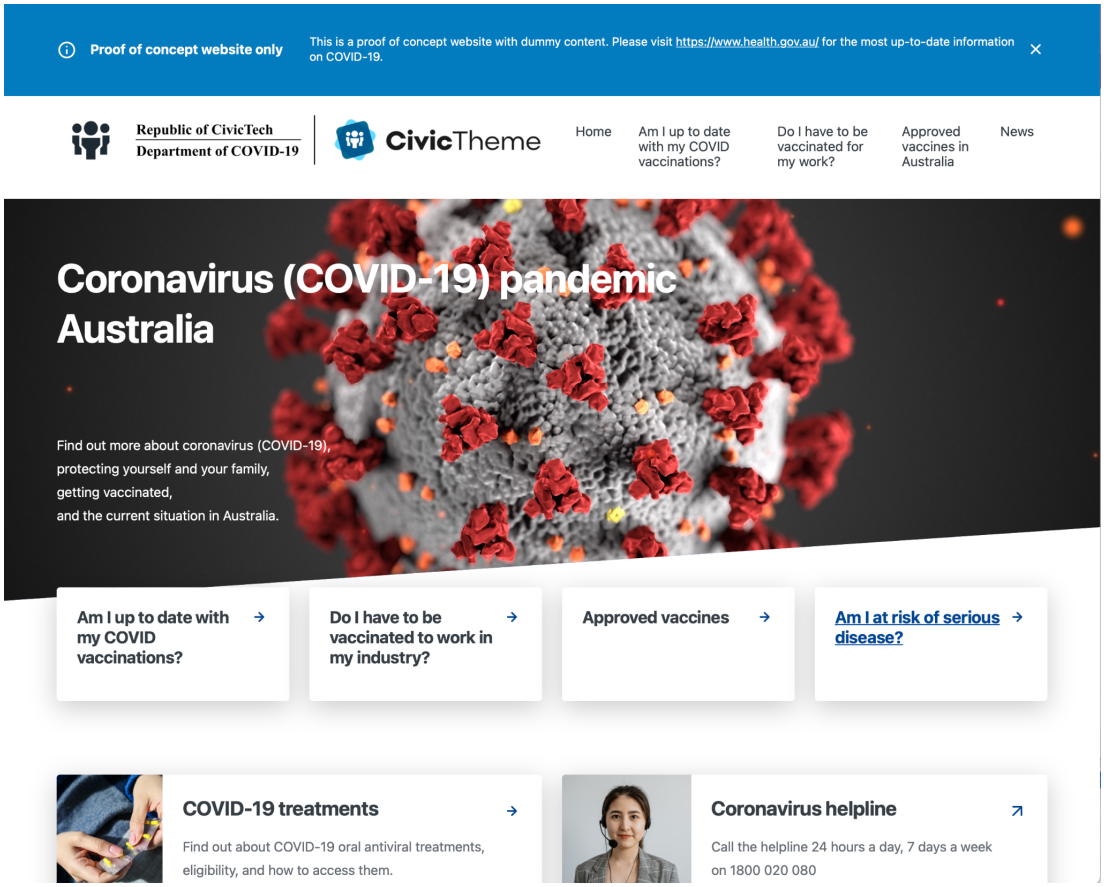
- Role definition of Drupal in the overall solution
- Manage syndicated terms to provide a repeatable relationship system between OpenFisca and many Drupal sites
- User-friendly interface to support ease-of-use when creating user journeys
- Providing configuration interfaces to provide reaction based routing when forms were submitted so relevant data could be surfaced to users
- Tokenization of OpenFisca results to reduce content management complexity
- Ability to relate content to OpenFisca rules to support a more traditional personalisation experience

The outputs

As part of the PoC, Salsa developed two demonstrator government health websites that provide an end-user experience of their vaccination checking pathway.

The websites simulate two pathways that share a single OpenFisca vaccination rule set. The two pathways being:

a. **A generic simulation of a typical COVID/health website** (homepage screenshot shown below).

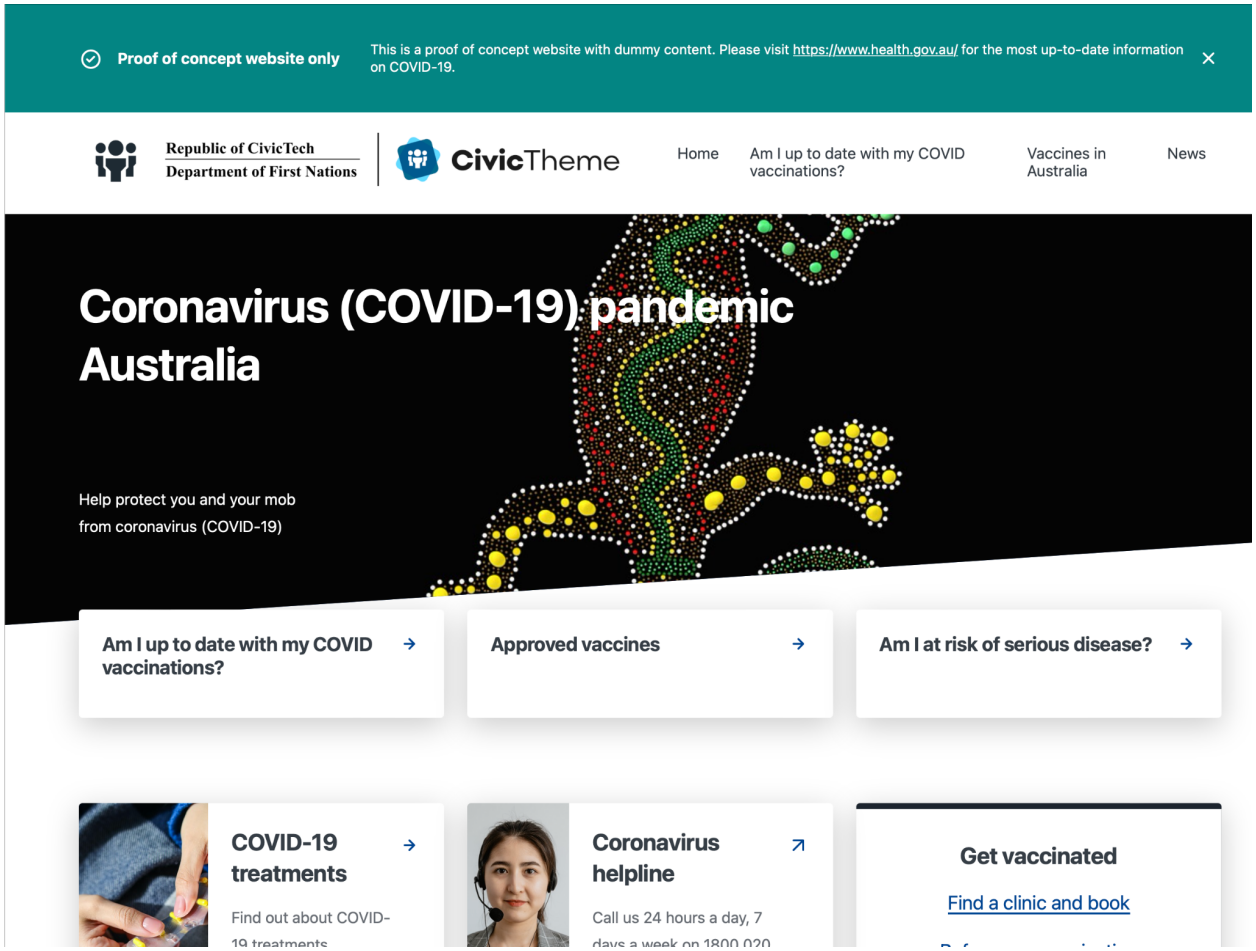


This includes public user journeys for checking 'Am I up to date with my COVID vaccinations' and 'Do I have to be vaccinated for my work' based on a series of responses to questions. It also includes information about eligibility for further vaccinations, vaccines approved for the user along, relevant vaccine booking locations with other useful information.

The rule sets included in this website simulation also demonstrate the ability of the RaC solution to traverse rules from Federal (ie 'Am I up to date..' or 'Am I eligible..') and State (ie 'Do I have to be vaccinated for my work') jurisdictions.

Finally, this website also includes a simulation on rules as they would apply 'today' but also on rules that are scheduled to change at a future point in time. This alternative user journey helps demonstrate the capability of the RaC solution to handle rules that change over time and is also used to illustrate the effort involved in the change management process involved. (For more information refer 6. Change Management section below).

b. **A simulation of a [COVID/health website dedicated to the Aboriginal and Torres Strait Islander communities]** (homepage screenshot shown below).



While this second website shares much of the functionality provided in the generic website, it provides a user journey and results that more closely align with the First Nation community requirements. This helps demonstrate the ability of the RaC solution to provide two UI pathways that share a single OpenFisca vaccination rule set.

4. Hosted Solution incl. Solution Architecture

The requirements

To provide containerised instances of OpenFisca that can be hosted in a Kubernetes-based deployment environment. To ensure that OpenFisca instances can be created easily, a scaffold repository was created that can be initialised to easily deploy additional OpenFisca instances. Due to OpenFisca system requirements, multistage Docker builds were utilised. This allowed installation of large, unchanging dependencies to be cached between builds which dramatically reduced build and deploy times.

To showcase how Drupal and subsequently GovCMS can integrate with OpenFisca, a GovCMS PaaS project needed to be provisioned and deployed on Salsa Hosting infrastructure. This was done utilising the open source scaffolding from GovCMS. To assist with visual design, the open source design system CivicTheme was implemented which enabled the team to focus on building out the OpenFisca integrations rather than building a Drupal theme.

The challenges

- Python dependencies that need to be compiled result in slow build times for OpenFisca, this impacts the ability to quickly deploy the project - to combat this we looked at implementing a multi-stage docker build process which caches the built dependencies resulting in improved build performance.
- Ensuring that the OpenFisca stack could be run in docker hosting environments
- Investigating and scaling OpenFisca instances and preparing the application to run in a highly available ecosystem

The outputs

The outputs for this part of the project:

- OpenFisca scaffold repository
- OpenFisca project deployed to a Lagoon hosting environment
- GovCMS PaaS project deployed to a Lagoon hosting environment
- CivicTheme installed and operating in the GovCMS PaaS environment

5. Security & Privacy

System Security

Salsa's RaC solution does not include submission or capture of personal information, but rather sends anonymised data from the Drupal web form to be processed by OpenFisca. The rules themselves are intended to be open-source and transparent. Data sent to OpenFisca is not stored and is sent via encrypted communication methods (TLS).

The OpenFisca platform itself does not rely on persistent storage (database or other) which considerably reduces the attack surface. It can also run on-premise, granting more flexibility over network rules that inform the security profile of the application.

The OpenFisca and Drupal applications provide API endpoints that are open for public access. These endpoints provide read-only rules calculation capabilities only - they are open for all to integrate with. API access tokens could also be considered as they will offer better

control and visibility into who/what is using the APIs. Either way these should be run behind a WAF with bot protection and rate limiting enabled to better protect the service.

The solution has not been assessed for compliance, however Salsa will work with and provide technical and other relevant security documentation to Finance or individual government agencies on attaining all relevant security accreditations.

Should GovCMS choose to host (open source) OpenFisca on-prem then it will, upon successful security assessment, be covered by their Official:Sensitive accreditation. The Akamai CDN/WAF would also cover the OpenFisca implementation under this scenario.

Similarly if the Drupal webform RaC changes were to be included in the GovCMS distro then the SaaS security accreditation would extend to the RaC changes.

Having both OpenFisca hosted by GovCMS on-prem and the Drupal webform RaC module in the GovCMS distro would provide full security coverage of the end-to-end RaC solution.

Please refer to ***Appendix 2 - Solution Architecture*** for more details on the solution implementation.

Privacy

No personal identifiable information is submitted to the OpenFisca service. The only data submitted are parameter values required to calculate the end result.

This data is also not stored by the service, it is simply used to provide the one-time calculation and return a result. Any display of personalised content as a result of the calculated response happens clientside in the browser.

Any PII data stored in Drupal via the webform will be stored in a Drupal database using standard Drupal methods and will be covered by either the GovCMS Official Sensitive accreditation for SaaS or by the individual customer agency application security accreditation.

6. Change management process

Cost saving is one of the key benefits of a well implemented RaC solution in that it can reduce the burden of the change management process.

By providing a machine readable, ‘single source of truth’ or reference implementation of legislation and regulation, a RaC system can, by its very nature, make available a transparent structure that facilitates a well functioning change management process. That is, change it once at the source and then manage the dependent systems.

Dependent or downstream systems are those systems that integrate with the OpenFisca APIs (to access Rules Code) and/or Drupal Content APIs (to access Rules Content). These

systems or User Interfaces may extend beyond website UI's to include systems such as Voice devices, mobile or other applications.

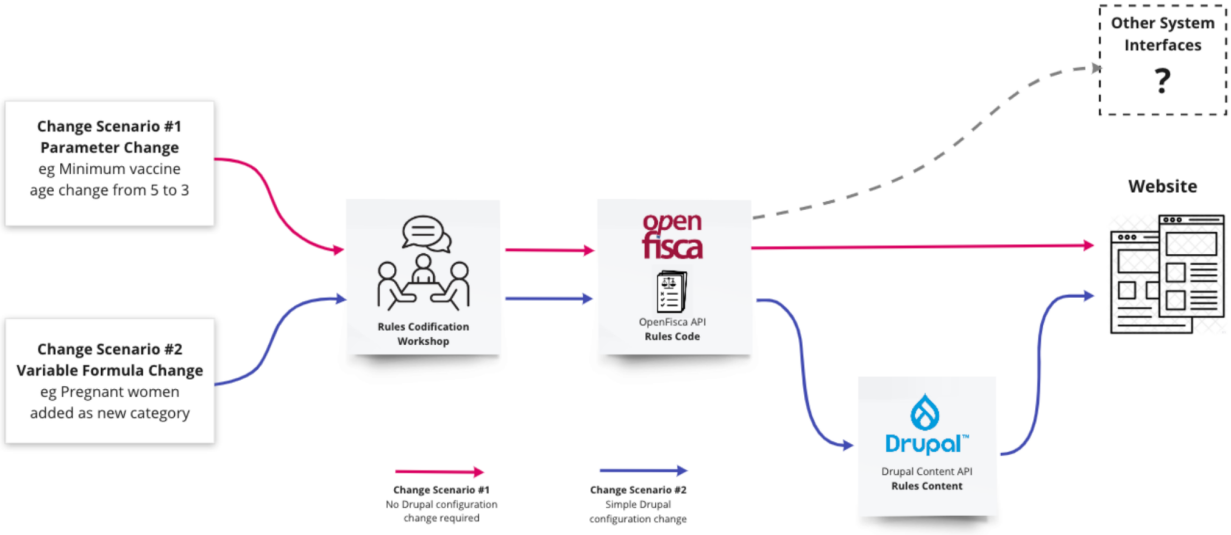
Keeping dependent systems in mind, changes to Rules Code and Rule Content can be managed in a similar way to how a typical software release and deployment process is managed. Once the rules code and content changes are made, release notes are published and deployment dates are set. Stakeholders that manage dependent systems would be responsible for updating their systems in line with release notes and deployment schedules. Care would be taken to ensure backward compatibility is maintained and breaking changes are avoided wherever possible.

Preview environments for OpenFisca can be readily provisioned for testing and other use cases.

The RaC solution that was developed for this PoC, helps demonstrate how a well architected system would help reduce the burden of keeping websites up to date with rule changes. While we focused on website content maintenance effort, similar approaches could be applied to other systems such as Voice and other apps.

To help illustrate the benefits to the RaC change management process, we have listed a series of change scenarios along with the related system components that may be impacted. We've also listed the HR roles that may be required for any given change, as well as a RASCI to help define each role's responsibility within the process.

The list of change scenarios, system components and rules are outlined in ***Attachment 04 - GovCMS RaC PoC - Change Management Matrix***. An example of two scenarios and how they impact downstream systems is shown in the following diagram:



Some of the key highlights outlined in the Change Management Matrix include:

a. Most changes don't require backend or front-end web development

b. Most changes require little or no change to Drupal Rules Form Content (ie Rules Content). In particular, parameter changes are included as variables in Drupal content (forms or text content) meaning that changes flow directly through to the Drupal front end user interface. The simulated **COVID/health website** highlights these Drupal variables in yellow on the results page for easy reference.

c. In most cases, Rules Owners and policy representatives would need to be involved to inform and validate the interpretation of legislation/regulation changes as part of the rules codification process.

d. All rules changes would require an OpenFisca (Python) developer to both update the Rules Code and deploy the OpenFisca application.

To help further demonstrate the change effort needed for some of the change scenarios, the simulated **COVID/health website** includes a an alternative user journey (refer the 'When Rules Change' section on the 'Am I up to date with my COVID vaccinations' page) that helps demonstrate the capability of the RaC solution to handle rules changes.

7. Showcase

A few short videos, showcasing key aspects of Rules-as-Code and the GovCMS Proof-of-Concept are provided below:

- Rules as Code - Proof of Concept Overview
  (Video 2mins)

- *Rules as Code - Delivering a personalised citizen experience for GovCMS*
  (Video 14 mins)

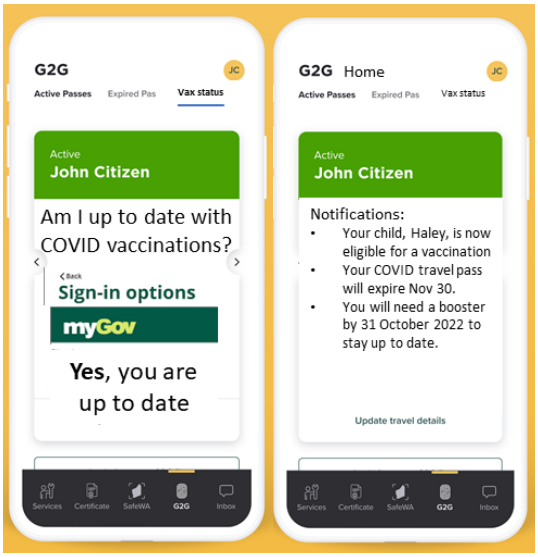
Future Use Cases & Opportunities

Integration with disparate services

In addition to the two web services that are showcased in the PoC, other systems could leverage a common Rules as Code utility. Some examples include the following:

1. **Voice controlled device**s such as Alexa, Google Nest and Apple HomePod. Below is a mockup of someone losing their job and needing a booster to apply for a new job.

2. **Integrations with State and Federal Government COVID Apps** (eg Service WA or Medicare App). A link could be added to the app such as 'Am I up to date with my COVID vaccinations'. The app would send whatever vaccination data it has in store to the RaC utility and only ask supplementary questions as needed, reporting back the status.



3. **Proactive notifications** - systems could be developed to check a citizen's vaccination record using the RaC utility, and proactively send

notifications to individuals. For example, that they may be eligible for a further vaccination.

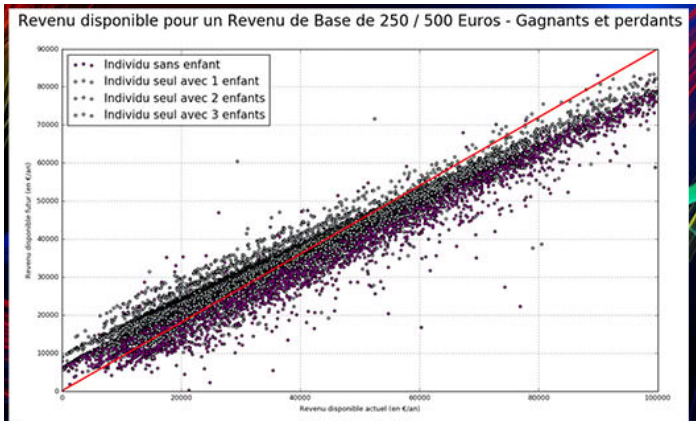
Interactive services

Agencies that use GovCMS may be interested in building more dynamic services where end users can interact with rules in real time, rather than the traditional “submit and response” approach of online forms.

A RaC utility provides the means for any services to interact with rules in real time, with changes to the options happening immediately as the user works through different conditions or questions. Research has shown that the process of filling in a whole form before clicking submit is a stressful experience for users, so providing profile based and iterative approaches to the experience can help encourage and support user confidence to interact with government services.

Policy planning

When the rules are available as a utility, you can use that utility to do other forms of policy planning. Population data can be used to simulate the impacts of change, to model implementation options (and inform resource management, supply chains, etc), and to identify if any unintended gaps or overlaps emerge. COVID status records and patterns of vaccinations could also be used to forecast vaccination purchase quantities and timings, as well as when changes to vaccine advice affect different age groups. Below is an actual example from France that uses rules as a utility (with analysis tools that use Openfisca) with population data to understand the payments impact of social support payment policy changes.

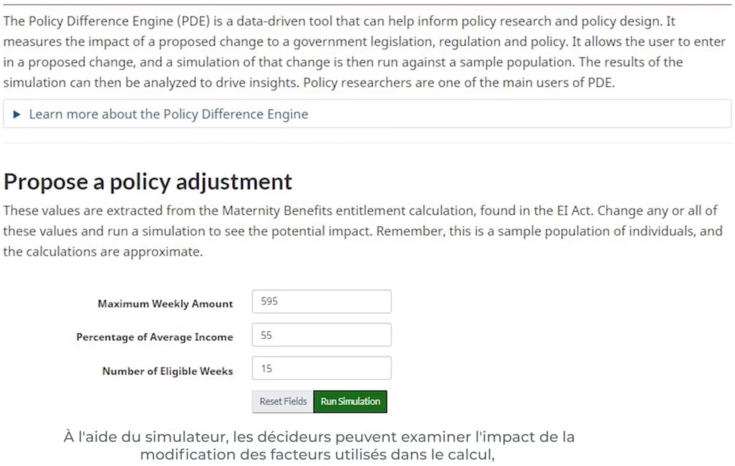


Policy testing

A RaC utility could also support a test driven approach to policy drafting and change over time. In the case of COVID rules, new regulation change scenarios could be tested to help determine impacts before new rules are formalised and published. A test driven approach would also support departments and Ministerial offices to explore changes before making announcements,

and would create a means to engage the public in testing and proposing evidence based policy change over time.

Below is an example of a project in Canada to build a “Policy Difference Engine” on top of a RaC utility, to support policy makers to test and explore changes to policies. A next generation of this could include engaging AI tools (Alexa, Siri, OK Google, etc) to test unlimited permutations to rules, to find the right balance and to maximise and appropriately balance the human, social, environmental and economic benefits from policy change.



https://www.youtube.com/watch?v=w0VKEh-ZrQM

Drupal Smart Content module

The Drupal Smart Content module can be used to build on the RaC user experience and present personalised content at the end of the RaC user journey, using the information gathered during the RaC journey.

The Smart Content module comes with an inbuilt Smart content Block. Using this module, we can create multiple blocks with conditions to hide or show the block. The conditions could be based on the values, say, from cookies, and the cookies can be set using API calls. Once a user fills in some basic details, relevant blocks can be shown to the user not only on the result page but also across the whole site, which helps in maintaining continuity.

A simple implementation of this feature is presented in this video demonstration.

More complex use cases

The GovCMS RaC PoC uses only a very limited set of OpenFisca’s potential capabilities. In this PoC we’re essentially having OpenFisca return Yes/No responses. However, as the name implies, OpenFisca can be used for complex fiscal calculations and modelling.

An example of a more complex use case is the French government Mes Aides service which informs French citizens on their eligibility to 32 social benefits.

Opportunities for the GovCMS Program

GovCMS as the central maintainer of the RaC shared platform

The current GovCMS Drupal/platform service offering lends itself to being extended to include the RaC Openfisca/Drupal service. In this scenario GovCMS would not own the content (Drupal forms, web content or OpenFisca rules) but would be the custodian and maintainer of the RaC platform.

A phased, crawl-walk-run approach may look like this:

- **Crawl** - agencies experimenting with silo implementations of the RaC on their own or other 3rd party (compatible) hosting platform to allow time for GovCMS to consider a RaC service model and business case.
- **Walk** - agencies experimenting with silo implementations of the RaC on the GovCMS platform
- **Run** - GovCMS standing-up a RaC marketplace where rule sets including Rules Code and Rules Content, is made available using Drupal Content API's and/or OpenFisca API's. In this scenario, government agencies would be the maintainers of the Rules Code and the Rules Content while GovCMS would provide the technical maintenance of the RaC shared platform.

Rules Code and Rules Content Governance and GovCMS' role

GovCMS could limit their role to just providing the technical maintenance of the RaC shared platform i.e. security patching and upgrading OpenFisca and the Drupal module. The individual agencies would then own, maintain and deploy their own Rules Code and Rules Content on the GovCMS shared RaC platform.

If agencies were the only consumers of their Rules Code/Content then this governance structure could be adequate. However, if an agency's Rules Code/Content was open and available for other agencies or third parties to use, either to incorporate into their own user journeys or for integration with other systems, then higher quality control and governance standards for interoperability would be required.

Rules Code and Rules Content would need to follow an agreed structure and would need a well articulated and controlled version release and deployment process. Locating this central RaC governance role or team inside Finance/GovCMS could also provide the following benefits:

a) a centrally managed OpenFisca/Python resource pool that would accept Rules Statements as inputs from agencies and codify them in OpenFisca. This would alleviate the need for government agencies to hire Python developers and skill them up in OpenFisca.

b) the same team could also manage the release and deployment of platform changes.

RaC policies and interoperability standards could be formulated and agreed by a RaC council or committee made up of key agency representatives or by a RaC community similar to an open source community.

This highly standardised, centrally governed approach would open the way for a RaC 'marketplace' to be formed that would allow Rules Code and Rules Content to be made openly available for agencies across all jurisdictions and for other third parties to use. This would then help realise the vision of providing truly citizen-centric user journeys that transcend government and jurisdictional boundaries.

From a technical standpoint, the centrally governed approach will also help support the proposed microservices platform architecture. While this will help with scaling of the solution, it also requires the central RaC services registry to help users of the shared RaC platform identify which service a rule set belongs to. The RaC team would be tasked with maintaining the RaC services registry.

The ideas outlined here need to be tested and validated through extensive consultation between RaC subject matter experts, GovCMS and other government agencies. Furthermore the technical solution architecture will need to be adapted to meet the requirements of any agreed governance structure.

Recommended Next Steps

Performance testing at scale

The RaC solution must be load-tested on the target platform before it is operationalised.

Salsa is quite confident that the RaC solution can operate at scale based on the architecture of OpenFisca and Drupal JSON:API components as well as real world experience of these components in other contexts. That said, only the execution of a well designed test plan will confirm this.

Pen Testing and securing the RaC solution

The critical nature of the information being delivered by the RaC solution means that it must be protected from cyber threats and potential misuse. This is further emphasised in the OECD RaC Report where Security is listed as one of the six 'Principles For A Successful Rac Approach' (refer to page 14 of the report).

It's expected that as part of Finance's standard risk assessment and due diligence process, the final solution will be pen tested and security assessed. That said, Fisca is a NoSQL solution and is idempotent. This means there are no dynamic exploitable components, greatly reducing the threat vector that the introduction of OpenFiscaposses to the GovCMS platform.

If the final solution was to be fully hosted, on-prem, on the existing GovCMS platform, it could leverage the security measures and practices that are currently in place for the GovCMS SaaS website hosting service. The Official:Sensitive security accreditation could be extended to include RaC solution following a successful assessment process.

The RaC Solution could also be hosted on other Government platforms or on privately operated platforms that have met the required security standards.

RaC Rule set Web Page

Provide ruleset stakeholders such as developers, BA's, content administrators etc, access to technical and other information relating to the ruleset.

Interface for appealing or correcting a coded rule

In the OECD RaC Report , the PRINCIPLES FOR A SUCCESSFUL RAC APPROACH (pg 14) section lists 'Appropriateness and Appealability' as one of the principles. Appealability is about provisioning the ability for users to appeal or correct a coded rule.

This interface could be provided via the RaC Ruleset Web Page.

Public disclosure of rule set code and mapping logic

In the OECD RaC Report, the PRINCIPLES FOR A SUCCESSFUL RAC APPROACH (pg 14) section includes 'Transparency', 'Traceability' and 'Accountability' among the list of principles. This is about documenting and publishing the code along with thinking and decisions underpinning the generation of coded rules from the natural language versions.

The OpenFisca code can also be made publicly available via the code repository. Alternatively, OpenFisca can also generate OpenAPI swagger documentation by default that references the code repository. Here is an example of a French Government rule set
https://legislation.fr.openfisca.org/swagger

This is a fundamental part of delivering open, transparent government. One example of open documentation is the rules statements, which can be publicly available. Please see ***Appendix 4 - Rules statements*** for the rules statements for this PoC.

Further RaC Solution discovery and testing

Based on RaC global case studies, our first hand experience with the GovZero team as well as the experience gained in executing the PoC, we believe an Open Source solution like OpenFisca is fit for purpose.

Furthermore, the natural fit of Drupal in the current GovCMS ecosystem and Drupal being a proven, enterprise-grade CMS and content API again provides confidence that the design of the RaC solution proposed here is the right choice.

That said, Salsa has not explored alternative solutions and has not tested the proposed RaC solution extensively in wider Australian government contexts, or with other systems such as mobile apps. Further discovery and testing would be needed before operationalising this solution for any given GovCMS customer.

Other Points of Interest

Responding to Freedom of Information (FOI) requests - access to historical rules representations

Citizens may lodge FOI requests for information on a historical representation of rules based on their given set of parameters (ie their inputs at an historical point in time). This raises a number of questions such as:

- How would an agency access this information?
- Which RaC system components are relied-on to provide this information?
- Who is responsible for maintaining the verifiable, historical record for any given RaC system component?

In responding to this requirement the following should be considered:

1. OpenFisca and all UI interfaces that integrate with OpenFisca are system components that are in scope.
2. For this PoC a Drupal/GovCMS web UI is in scope.
3. Other 3rd party downstream integration stakeholders (Voice, App, non-GovCMS websites etc) may be unknown.
4. OpenFisca - historical information can be accessed in two ways:
    a. All rule parameters are dated and the dates are retained through history in the code rules. eg a parameter dated 1 Jul 2020 will persist in code
    b. OpenFisca git commits are retained and can be used to rebuild a point in time version of the rules

    Considerations:

    i. The OpenFisca platform owner is responsible for providing access to historical information
    ii. OpenFisca application versions will change over time, which may include backward incompatible/breaking changes. In this instance, access to historical versions of the OpenFisca application needs to be retained. Historical Operating System compatibility issues may also need to be considered.
5. GovCMS/Drupal Web UI - historical information can be accessed in two ways:
    a. Drupal Database and application backups

b. Drupal git commits are retained and can be used to rebuild a point in time version of the rules

Considerations:

i. GovCMS is the platform owner in this instance and is responsible for providing access to historical information.

ii. All combinations of possible rule inputs and related results will need to be accessed for any point in history.

iii. GovCMS/Drupal application versions will change over time, which may include backward incompatible/breaking changes. In this instance, access to historical versions of the Drupal application needs to be retained. Historical Operating System compatibility issues may also need to be considered. This is an issue not only for GovCMS/Drupal RaC instances but for all GovCMS/Drupal instances and is already managed.

6. 3rd party downstream integration stakeholders are the platform owners in this instance and are responsible for providing access to historical information.

7. A RASCI would help clarify to all stakeholders, where their responsibility lies in relation to FOI requests.

Restricting access to future rules that have yet to be made public

Agencies may want to encode, test and 'release' rules that apply for at a future date to OpenFisca but may not want them to be available via the OpenFisca API (ie to the public) until a point in time in the future. Eg a budgeted eligibility provision that is yet to be made public.

Currently OpenFisca allows rule parameters to have a date in the future when they are due to come into effect. However, this future dated rule can be accessed from the time it is released; it can't be hidden.

Two potential options could be considered to meet the requirement for 'embargoed' rules:

a. Deploy the rule on a test environment and then release and deploy the OpenFisca rule at precisely the time it is allowed to be made public.

b. OpenFisca is an Open Source project meaning that we could develop the PR for this change and work with the OpenFisca community to have it implemented. Noting that there is no guarantee that the community will approve this change in which case a workaround would be required.

Considerations:

i. If rule change details are withheld from the public then 3rd party stakeholders may not have time to assess the release notes and undertake testing in time for the deployment. Governance structures will need to be put in place for these scenarios.

Closing

The scope of RaC is very broad, ambitious and touches some unchartered territory. Some questions remain open about the level of Government acceptance of RaC, RaC policies, interoperability standards as well as technical solution direction. At the same time, RaC offers an inspiring vision and exciting possibilities that transcend government agency and jurisdictional boundaries and support core democratic principles such as government transparency and accountability.

Our hope is that this Whitepaper, accompanied by the RaC PoC technical implementation, has proven that a viable RaC utility is not only achievable but also highly beneficial for the citizen and government.

In summary, this Rules-as-Code Proof-of-Concept showcased personalisation services that help citizens filter through somewhat complex vaccination rules to determine their vaccination status quickly and easily.

Using a RaC solution such as the one developed for this PoC, delivers many benefits to GovCMS and its clients, including:

- **A truly citizen-centric approach:** Importantly, traditional personalisation solutions have NOT been built for government. Rather, they're primarily for media consumption or ecommerce, e.g the primary outcomes are Read more or Buy more. Citizens do not want to read more content or buy more products from their government agencies; they want to understand which government services are available to them and when they should be used. They want a low friction solution to let them know what steps they need to take next, or whether they're eligible for something without reading and deciphering complex documents. The RaC approach is truly citizen-centric. Our personalisation solution is about making real change, not following market trends.

- **It's open-source:** Our RaC solution is based on an open source, Python-based rules-as-code OpenFisca platform. Thus, the proposed solution is fully open source, including the repository of rules. This complies with digital service standards, specifically "make source code open". This avoids lock-in, reduces costs/duplication, increases transparency and ultimately enables co-creation with industries, communities and other jurisdictions.

- **It's sovereign:** While OpenFisca was seeded from the French government, it can be built and deployed from repositories in Australia, either on the GovCMS infrastructure or using

other secure Australian based hosting options. This means that Australian government rules, entitlements, legislation, related data, etc, are stored and maintained on Australian soil. Australia's data is kept here, in the hands of the Australian people, our governments and our industry.

- **It's flexible**:
    - **Hosting flexibility**: Agencies can choose whether to host on the GovCMS infrastructure (and leverage all of the IRAP security controls) or host remotely on compliant and compatible infrastructure.
    - **Implementation flexibility**: This implementation is flexible in nature - e.g you may have a single OpenFisca instance running multiple rules, to deliver on the ultimate goal of a whole-of-government solution, that is, one Open Fisca repository with all the government rules independent of agency. Alternatively, we can have separate OpenFisca applications for each ruleset (project) or separate repositories for each agency, or separate repositories for each jurisdiction. This flexibility will allow for the solution to adapt to GovCMS's community needs.
- **It's scalable and resilient (stateless architecture):** The OpenFisca application will run as a stateless solution, having no dependency on database or persistent storage. The solution has OpenFisca running as a Lagoon-enabled project within a kubernetes cluster. This allows for scalability and resilience, benefiting from all the auto-scaling and auto-healing solutions that kubernetes provides, without any scaling bottlenecks (e.g database).
- **It's secure/IRAP certifiable:** This solution is highly secure and does not deal with any personal information or sensitive data and thus represents a low-risk approach. Our solution sends anonymised data from the CMS/ Webform to be processed by the rules engine (OpenFisca). The rules themselves are intended to be open-source and transparent, meaning no data sent or received contains anything of a sensitive nature. The rules engine/platform (OpenFisca) platform can run on-premises, granting more flexibility over network rules that inform the security profile of the application. The solution has not been assessed for compliance, however Salsa is able to work with Finance on attaining all relevant security accreditations, whether it's IRAP or otherwise.
- **Contributing to a movement:** This RaC approach to personalisation will help to contribute to the government-as-a-platform movement, building on the pioneering RaC work being done in France, New Zealand, Canada and NSW. It will bridge the gap between GovCMS and the RaC movement through rich (and simple) integration with OpenFisca.

  This is an opportunity to bring Finance to the table as a contributor, as a student, as a co-creator and as a thought leader to help lay the foundation for RaC in Australia. It's an

opportunity to contribute a small piece, an important building block and then to test, learn, adapt and repeat to deliver a more holistic government solution.

Salsa is committed to help lower the barrier for agencies to experiment and productionise their Rule-as-Code service through our end-to-end RaC service offerings.

Let’s keep talking and doing…

Further Reading/Viewing

1. OECD RaC Report - OECD-OPSI 'Cracking the Code - Rulemaking for Humans and Machines'
2. Rules as code: Regulatory infrastructure for a digital age (Marina Yastreboff)
3. Rules as Code 2.0 Global Plenary
4. Mes Aides service informs French citizens on their eligibility to 32 social benefits.

Attachments

Attachment 01 - About Rules-as-Code

Attachment 02 - About OpenFisca

Attachment 03 - Rules-as-Code Service Delivery Methodology

Attachment 04 - GovCMS RaC PoC - Change Management Matrix

Attachment 05 - About GovZero

Attachment 06 - Gov Zero Aotearoa - Introduction Pack

Appendices

Appendix 1 - Glossary

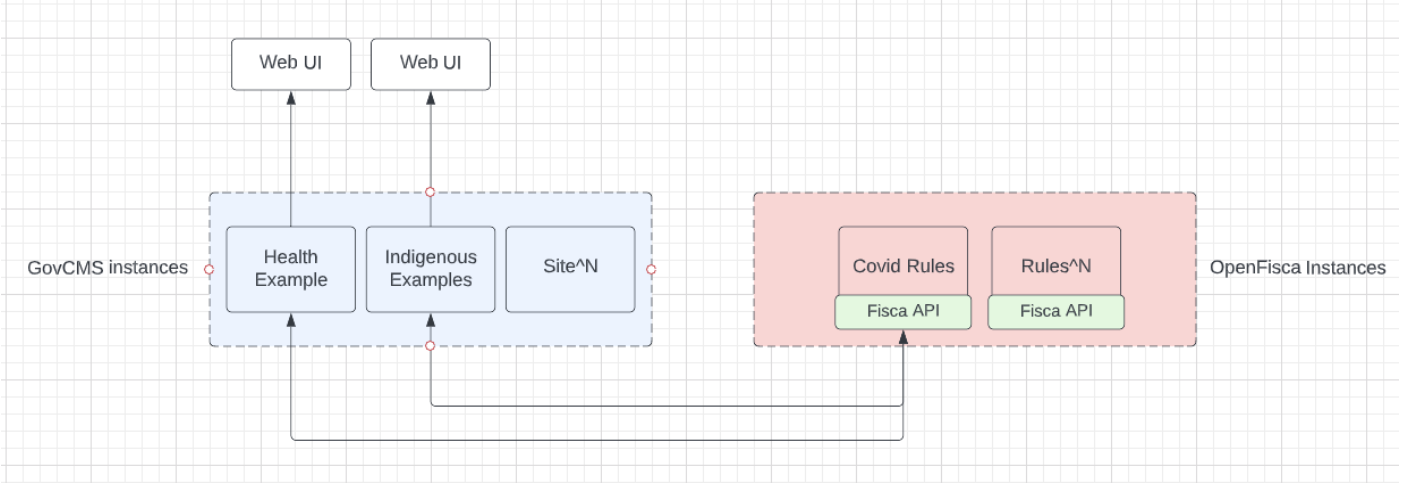
Term	Definition	Additional references
CivicTheme	CivicTheme is an open source design system with a Drupal 9 theme and a component library. CivicTheme was created for government agencies to build Drupal 9 websites faster and at a lower cost. CivicTheme was used to build the UI's for the PoC websites and user journey.	• https://www.civictheme.io/
Concept models	Concept models pull out the concepts and variables identified in the legislation/rules analysis and turn them into a diagram.	• Attachment 03 - Rules-as-Code Service Delivery Methodology
Decision models	Decision models cover the key questions for eligibility	• Attachment 03 - Rules-as-Code Service Delivery Methodology
Drupal Content API	Use of Drupal JSON:API module for exposing all types of content and configuration entities from Drupal in a machine readable format. Drupal content can then be queried using the API endpoints allowing a third-party integration to make API requests based on OpenFisca responses to collect content.	• Appendix 02 - Solution Architecture
Drupal Forms /OpenFisca module	An extension of Drupal Webform module to create new RaC elements that are leveraging the OpenFisca API integration to query the system and present the results.	
OpenFisca	OpenFisca is an open source, lightweight, modular and scalable python-based rules engine platform that is used to encode and serve rules as code. The rules engine is independent of the country or jurisdiction, so can be used to encode legislation of any country / jurisdiction / department. OpenFisca is the rules engine in this solution and provides a structured way to build and test Rules-as-code implementations.	• Attachment 02 - About OpenFisca
OpenFisca API	OpenFisca provides a HTTP API that app developers can use to perform computations, without installing anything locally.	• OpenFisca.org > OpenFisca web API

Parameters (OpenFisca)	A parameter is a property of the legislation that changes over time. Unlike a *variable*, a parameter is not specific to a specific entity (person, household…) eg the amount of the minimum wage; the amount of family allowance per children; Parameters are used in formulas to calculate variable values	• OpenFisca.org > Parameters
Rules-as-Code (RaC)	Rules-as-Code (RaC) is an emerging space within the public sector, which drafts and turns legislation and other rules into machine-readable code to provide benefits for government, citizens and broader society.	• Attachment 01 - About Rules-as-Code
Rules Code	The queryable (Python) code that represent the rules and are accessed via the OpenFisca API.	• Appendix 02 - Solution Architecture
Rules Content	Drupal webforms content (ie questions and user journeys) that provides the user interface for how to interpret and interface with the OpenFisca Rules Code via the OpenFisca API.' Rules Content is available for third party integration via the Drupal Content API.	• Appendix 02 - Solution Architecture
Rules Owner	The role, typically from the Government agency with primary responsibility for a rule set, that oversees and approves all rules mapping and changes on the RaC system	
Rule set	A collection of rules applicable for a given use case. Eg this PoC provides the rule set for COVID vaccination status.	
Rule Statements	Clearly communicated rules expressed in structured, non-legal language that can readily be used as key inputs for developers for coding the rules in OpenFisca.	• Attachment 03 - Rules-as-Code Service Delivery Methodology
Salsa Hosting	A production-ready, Kubernetes/Lagoon based hosting service provided by Salsa Digital. The Salsa Hosting service was used to hosting the PoC Drupal instance.	
Test cases	Test cases look at both eligibility criteria and rule calculations as applied to a specific example/scenario.	• Attachment 03 - Rules-as-Code Service Delivery Methodology
Test-driven	Test-driven development uses the test cases	• Attachment 03 - Rules-as-Code Service

development	(scenarios) to develop code. Developers code for multiple scenarios, building on the code as they go.	Delivery Methodology
Variables (OpenFisca)	A variable is property of a person, or an entity. eg The birth date of a person; the amount of income tax a household has to pay in a year.	• OpenFisca.org > Variables

Appendix 2 - Solution Architecture

Many considerations have been made to assess an ideal candidate for the architecture of the solution. GovCMS is in a unique position with a distributed content authorship model already in place; this can be extended naturally to include a microservice architectural approach for the overall solution.



Why microservices?

Microservices allow each piece of the puzzle to be developed independently. It reduces the overall risk of the solution by separating concerns i.e. rules and legislation into OpenFisca, content relating to legislation curated into Drupal.

Rationale

- OpenFisca is API enabled, this means that to interface with rulesets you need to make HTTP requests
- OpenFisca is written in Python - adding this as a per agency service would increase the burden on teams to have OpenFisca/Python developers as well as Drupal experts
- The rules behind legislation will be the same between agencies; this allows a set of rules to be defined in a central location and provides the ability for many applications to integrate with.
- All changes in OpenFisca require developers as they need to be done in code and redeployed by using a microservice approach you separate content from rules which enables content editors to make meaningful changes without required developers.
- Enable other views into the data (eg. third party applications)

- Discrete smaller instances of OpenFisca that pertain to specific domains of legislation

Separation of concerns

- OpenFisca is responsible for taking user submitted data and making calculations based on that data, this will inform Drupal of a users eligibility and other factors based on the rule sets provided
- Drupal will store content, import taxonomy terms and provide the building blocks to be able to show useful information to end users based on the calculations that OpenFisca makes given user-submitted data.

In the proof of concept, Drupal provides the Web UI by presenting a form that the user can submit. This acts as an entry point for data collection and then Drupal can respond according to the rule evaluations.

OpenFisca's role

OpenFisca is the rules engine in this solution, it provides a structured way to build and test Rules-as-Code implementations. Through the rules mapping process clear pathways are defined which can be translated neatly into OpenFisca code that are then used to power the API.

Variables and Calculations

Variables are the fields that you define in code, these help build your entities and represent your data. The way data is architected in OpenFisca allows you to specify a null value for a particular variable, this will cause OpenFisca to run through the **calculation** related to that variable.

A calculation will have access to all variables present for particular entities allowing API designers to specify required fields ensuring that data is presented in repeatable ways. The calculation methods are idempotent, this ensures integrity and validity of the data; if you make a request with the same variables OpenFisca will respond in the same way.

Parameters

Parameters are configuration values that OpenFisca uses to store time sensitive values. The parameter structure, which looks like:

```
description: Maximum Age of COVID vaccination eligibility for Children (in years).
metadata:
  reference:
  unit: year
values:
  2022-08-01:
    value: 11
```

By referencing a time frame that the parameter is available, parameters can be stored in code over time. Allowing updates and changes to flow seamlessly through the OpenFisca API without requiring downstream change.

Architecture

OpenFisca is a python application that runs via the in-built python HTTP libraries. All rules are loaded into memory on application start ensuring that requests and responses are handled quickly and efficiently. OpenFisca has been architected to support large scale numerical calculations and is suitable for small if-this-then-that type rules definitions.

By dockerising the solution we are able to host the application in a kubernetes environment which enables standard security measures to be placed on the running workloads as well as standard pod scaling processes to ensure that there are enough OpenFisca resources to deal with the request load.

As there are no persistent storage requirements; neither database nor filesystem - OpenFisca is built from the ground up for scale.

Drupal's role

Drupal provides the user interface for how to interpret and interface with the API specifications provided by OpenFisca. There are two main areas that the proof-of-concept focused on; **content administration** and **user interactions**.

A custom Drupal module **webform_fisca** has been developed as part of the proof-of-concept to extend the functionality of core and contributed systems to enable content creators to build Webforms and content that is OpenFisca aware.

Content Administration

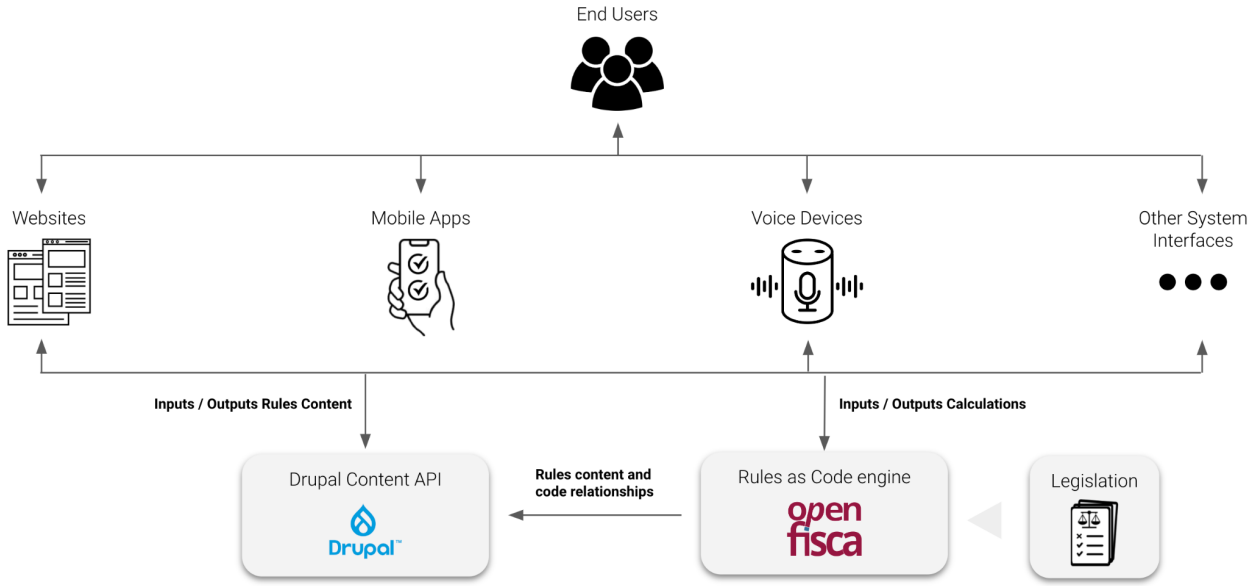
The primary focus of content administration is to identify a robust way to relate content to decisions that OpenFisca can make. This allows non-technical users to create content and allocate priority and importance based on data from OpenFisca.

The **webform_fisca** module provides the mechanisms for:

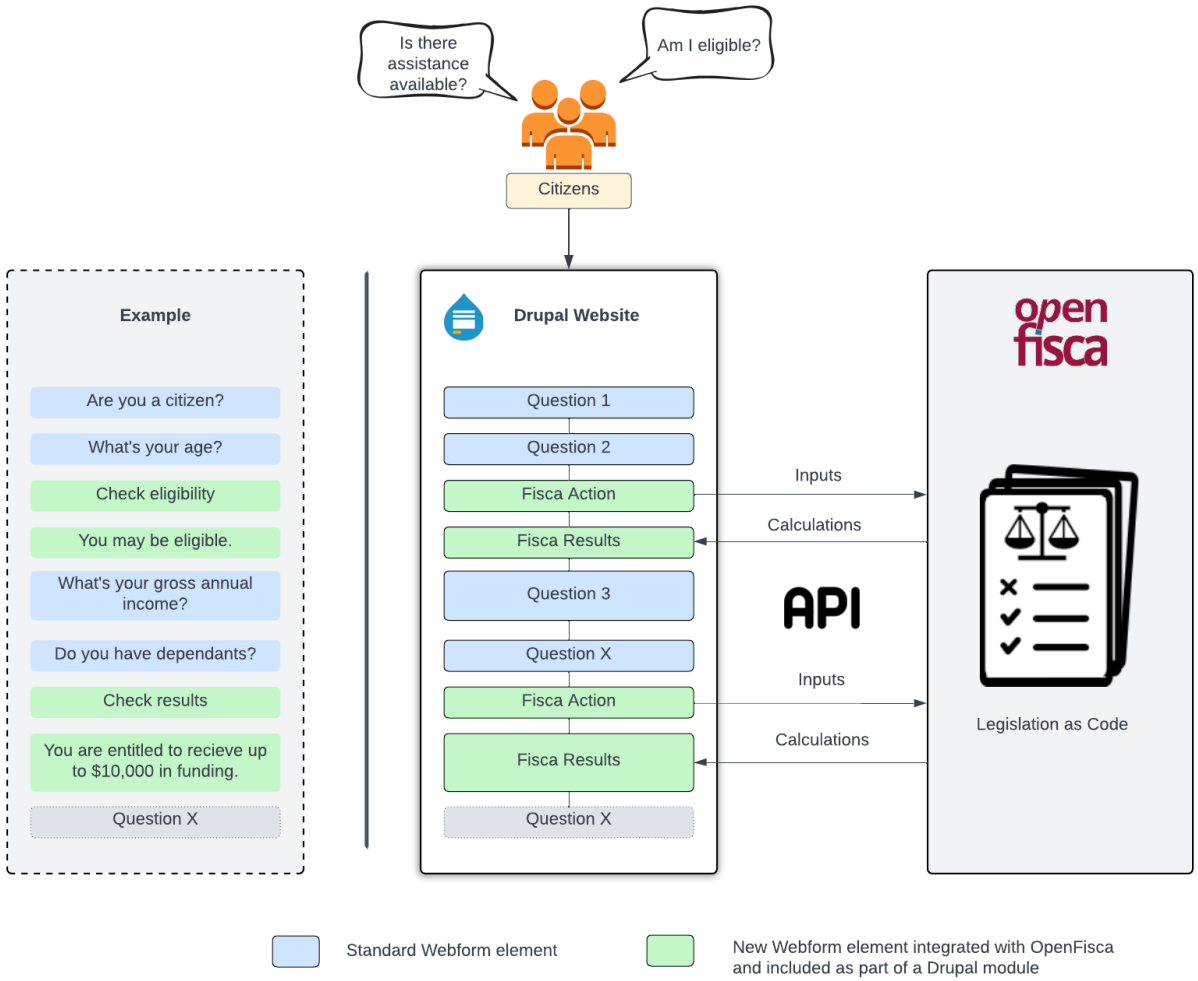
- A vocabulary that will house all terms from connected OpenFisca instances, this vocabulary can then be added to content types during standard site building processes.
- A Syndication process to create terms for each Drupal site, this will interrogate all connected OpenFisca APIs using their Swagger definition to automatically manage terms based on **variables** that have been defined in OpenFisca

This creates a loose relationship between OpenFisca and Drupal; both systems can operate correctly without this process; however this syndication process augments the content creation process and enables a robust content relationship model.

Overview



User Interface



webform_fisca

The **webform_fisca** module has been developed to provide a standard mechanism to integrate with OpenFisca. This module depends on Webforms and provides a number of extensions to the Webform module to support this.

Component	Description
Submit handler	The Webform submit handler is responsible for collecting the submitted webform data and constructing a recogniseable API request for OpenFisca.

	The submit handler also provides branching logic support to allow the webform to redirect to different result pages based on the OpenFisca response.
Field configuration	Integration with OpenFisca to collect **variable** names, this is stored as third party configuration for the webform fields and allows content creators to map fields to OpenFisca variables.
Taxonomy Syndication	Create taxonomy terms based on the variables contained in OpenFisca to enable content editors to easily relate content to decisions. This coupled with the **smart content** module (or similar) can be utilised to provide a more traditional personalisation experience.
Rules Content	Rules content indicates user journey's and provides a configuration interface for how a webform will respond to an OpenFisca response.

Future Considerations

Content API

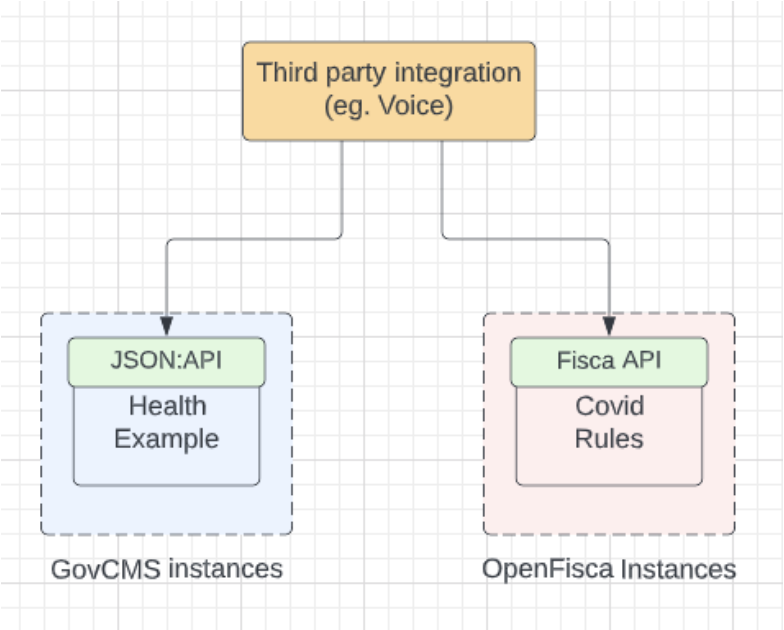
GovCMS is in a unique position to adopt a microservice mesh when defining content APIs. Each site can enable the required Drupal modules and begin exposing content to external systems. This forgoes the need to syndicate content between projects and allows them to consume data directly from source.

The Drupal JSON:API module is capable of exposing all types of content and configuration entities from Drupal in a machine readable format.

Third-party integrations can integrate with Drupal to collect information regarding:

- Drupal webform configuration will enable integrations to use defined user journeys and will prevent content duplication between integrations.
- Drupal content can be queried for using the API endpoints; this allows a third-party integration to make API requests based on OpenFisca responses to collect content.

The flexibility of the Drupal JSON:API module will enable third-party applications to make decisions based on their requirements; they can integrate with both API endpoints or one as needed.


Third party integration
(eg. Voice)
JSON:API
Health
Example
Fisca API
Covid
Rules
GovCMS instances
OpenFisca Instances

Appendix 3 - The mapping process

The mapping process enables rules and legislation to be codified into a rules engine (such as OpenFisca). The process interprets and transforms rules/legislation written by legal drafters and/or policy experts into a format that can be readily converted into machine-consumable legislation.

The process should identify gaps, ambiguities and contradictions in the logic of the actual legislation and should be done in close collaboration with policy/legislation experts.

The process consists of the following recommended stages:

1. Initial analysis of the rules

   This process extracts the rules in scope into a summarised, plain english representation. This step can be performed in a word processing document or a model diagramming tool such as Miro.

2. High level model

   Following the initial analysis, a high level model is created that captures any entities (persons, groups, etc.), legislative requirements, the eligibility logic, and the calculation logic for the rules being codified. This model can resemble a user journey or flowchart. This step typically identifies the questions that will require user input. The model is usually created using a diagramming tool such as Miro.

3. Logic/decision model

   The high level model is then converted into a logic/decision model that more closely matches machine-consumable legislation. In this step, we identify the user variables and parameters needed to meet the legislative requirements, as well as capturing obligations and developing test cases. The model is usually created using a diagramming tool such as Miro.

4. Test cases

   To enable easier testing and verification of the implemented rules, it is recommended that test cases are provided for OpenFisca developers. The test cases should define the input variables and expected outcomes. The test scenarios should cover all combinations, and focus on inputs that purposefully cross thresholds. If appropriate, changes to legislative parameters should be incorporated also.

5. Rules statements

Rules statements take complex rules and legislation and represent them as a simple set of rules in plain English. This step can be performed earlier in the process, but it is most effective when done at the same time as writing code. In a typical Rules-as-Code project, these statements will be tested and verified by a policy/legislation expert from the relevant agency.

During the discovery phase of the proof-of-concept project, our business analysts examined the existing, COVID vaccination rules from these publicly available sources:

- *Health.gov.au > Stay up to date with your COVID-19 vaccines*
  https://www.health.gov.au/initiatives-and-programs/covid-19-vaccines/getting-your-vaccination/stay-up-to-date
- *Health.gov.au > Who can get vaccinated*
  https://www.health.gov.au/initiatives-and-programs/covid-19-vaccines/who-can-get-vaccinated
- *Health.gov.au > ATAGI statement on defining 'up-to-date' status for COVID-19 vaccination*
  https://www.health.gov.au/news/atagi-statement-on-defining-up-to-date-status-for-covid-19-vaccination
- *Health.gov.au > ATAGI statement on the use of a 3rd primary dose of COVID-19 vaccine in individuals who are severely immunocompromised*
  https://www.health.gov.au/news/atagi-statement-on-the-use-of-a-3rd-primary-dose-of-covid-19-vaccine-in-individuals-who-are-severely-immunocompromised

Note: During a 'normal' RaC process, we'd work with the relevant agency or authority to fully understand the requirements and ensure all rules are correctly captured and interpreted. For this PoC we only used a specific set of resources.

Initial analysis of the rules

Based on the above web pages, we pulled out the rules in a summarised, plain English format. We brought these across into Miro (see the screenshot below). The logical grouping for the rules was based on age. As you can see from the screenshot, the number of doses for 'up-to-date' status was pulled out for each age group. Additional information was then dropped next to each age group, and copied and pasted if relevant to more than one age group. This included information about when someone should get an additional dose, the recommended vaccine (e.g. Pfizer, Moderna, etc.) and rules around timeframes (e.g. you need to wait 3 months after a COVID infection before getting a booster).


Dose by age
5-12 yro
2 doses = up to date
5yro and over: Extra dose IF severely immunocompromised
If have had Covid, can't have dose within 3 months =up to date in that 3 months
If have had Covid should get next scheduled shot 3-4 months after positive test
12-15yro
2 doses = up to date
3rd primary dose from 2 months (and no later than 6 months) after
Pfizer recommended as booster
16 and over
2 doses + booster (3 doses) = up to date
dose 3 from 3 months after 2nd dose
'overdue' if a booster has not been received within 6 months of completing their primary schedule
3 months after
4th dose recommended if medical condition that increases risk of severe Covid illness
5th dose recommended if immunocompromised
Pfizer recommended as booster for 16 & 17yro
18+ can have Pfizer or Moderna as booster (3rd & 4th doses)
30-49
Can have 4th dose if they want
If have had Covid, can't have dose within 3 months
4th dose recommended if medical condition that increases risk of severe Covid illness
50+
4th dose recommended
If have had Covid, can't have dose within 3 months

We also started looking at some of the other rules across different groups, such as aged care residents, Aboriginal and Torres Strait Islanders and people with disabilities living in shared residential accommodation. However, for the purpose of this PoC we focused solely on up-to-date status (and eligibility, which is intrinsically linked to 'up to date' status). We further simplified the interpretation of the rules for the PoC by grouping together the following health issues:

- Severely immunocompromised
- Disability with significant or complex health needs
- Complex and/or multiple health conditions that increase the risk of severe COVID

Codifying the rules

There are several stages to further prepare rules for codifying. Not all of these are strictly required, but following this process enables developers (who may not have a strong understanding of the underlying legislation) to understand the rule logic, as well as highlight gaps and issues in the legislation. The stages consist of::

- Create rules statements
- Create a high level model that maps the rules via a user journey/flow chart
- Create a decision model that maps rules to a logic flow

As part of any of these processes you'll also need to start noting the variables, the user input required to populate the variables, and relevant parameters (properties of the rule/legislation that changes over time). This can be done by the business analyst or in conjunction with an OpenFisca developer.

Rules statements

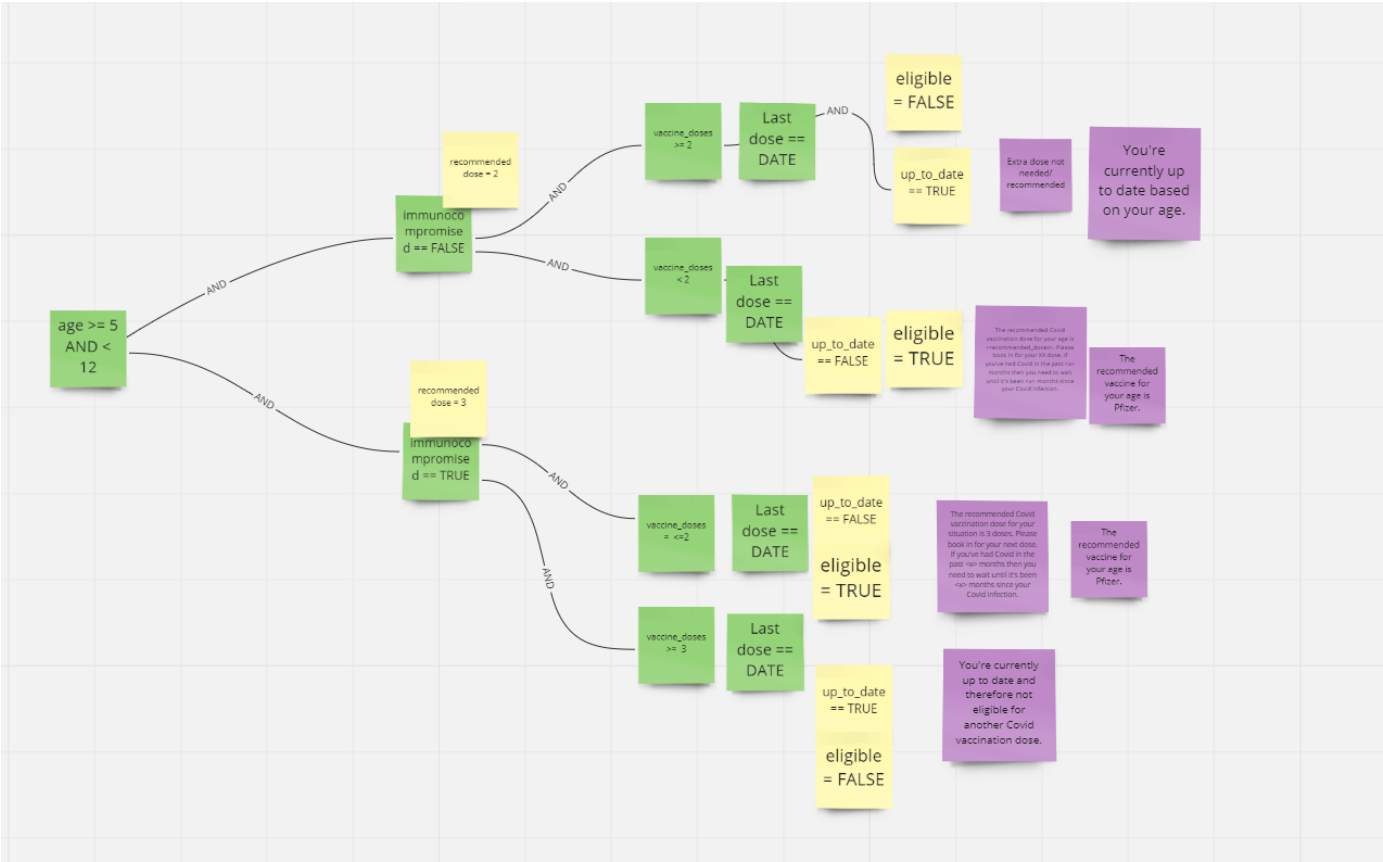
Rules statements are simple statements of the rules. For example:

- The recommended dose for 5-11 year olds is 2.
- The recommended dose for 5-11 year olds who are severely immunocompromised is 3.

For a full breakdown of the rules statements for this COVID vaccinations PoC, please ***Appendix 4 - Rules statements***.

High level model - Creating user journey maps/flow charts for each scenario

Another option to codify rules is to map a user journey. This method was relevant for COVID vaccinations given age was a logical way to group/classify different rules. The screenshot below provides a user journey for someone aged between 5 and 11 years old.



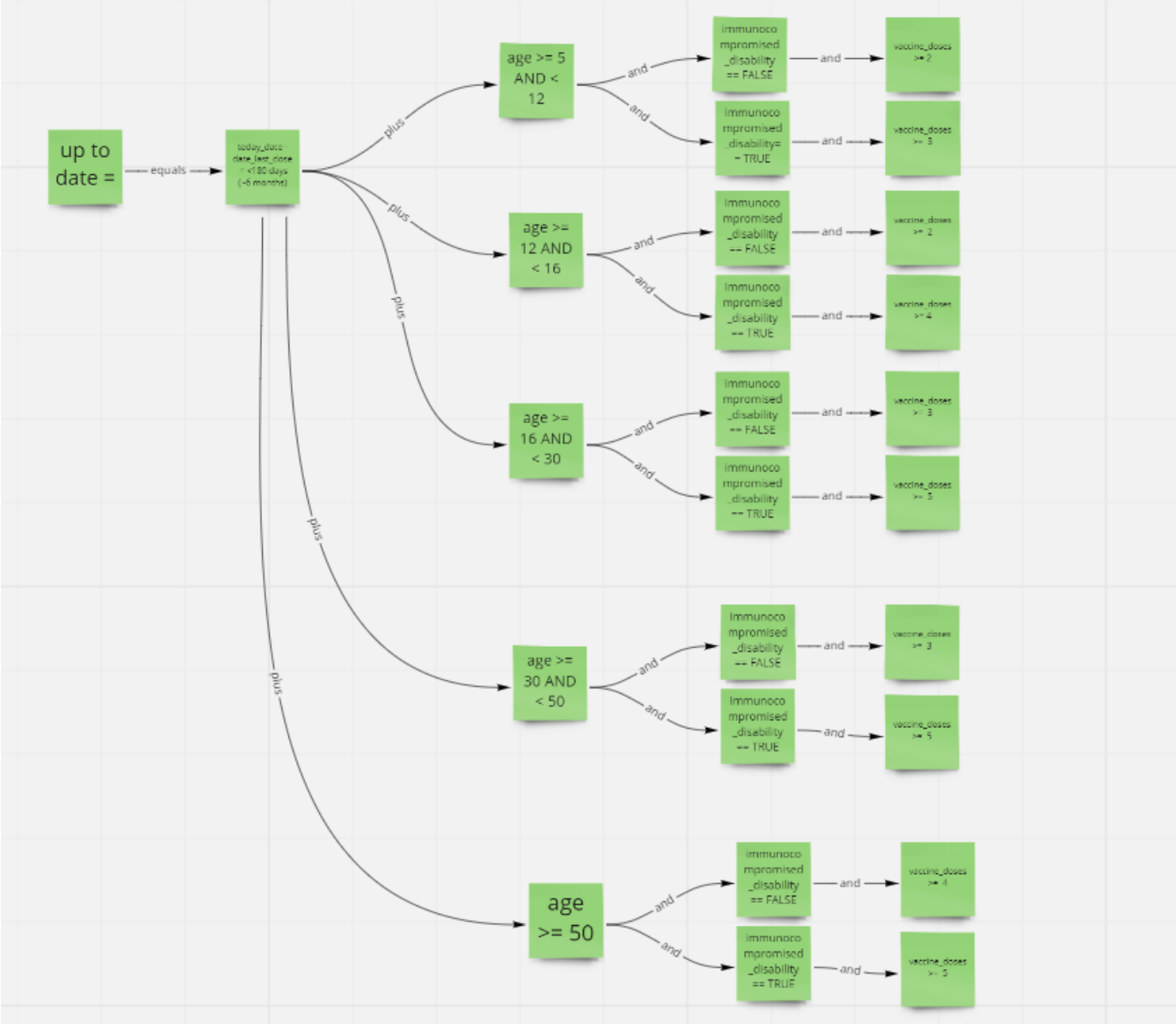
This process was done for all age groups to represent the user journey flow through the rules.

Decision model - Final rules mapping based on requirements

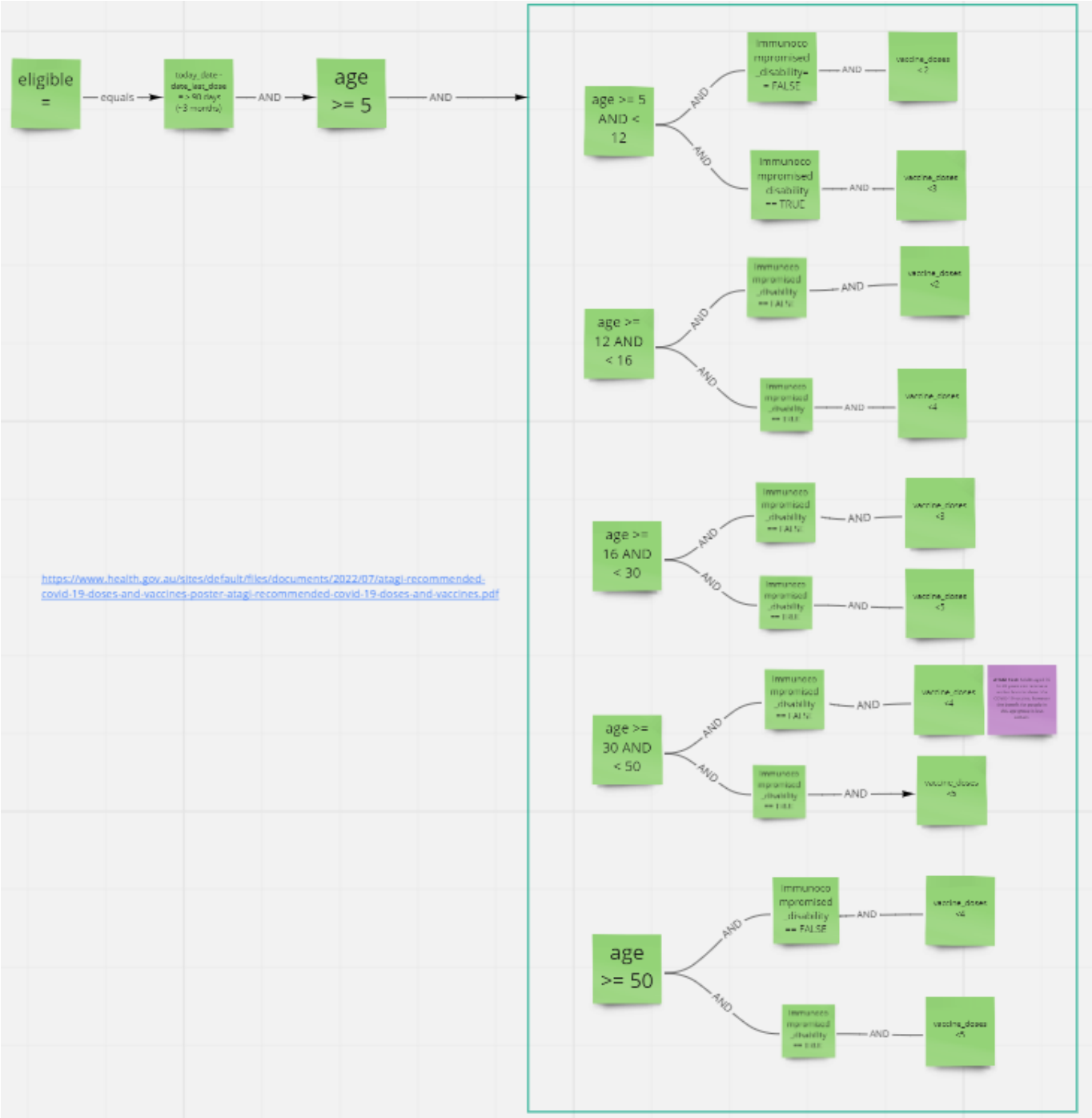
While a developer could potentially code the rules based on rules statements or the user journey mapping, in our case we created a final rules mapping centred around the main deliverables/definitions of:

1. ‘Up to date’ status
2. ‘Eligibility’ status
3. ‘Mandatory’ status for working in a particular sector

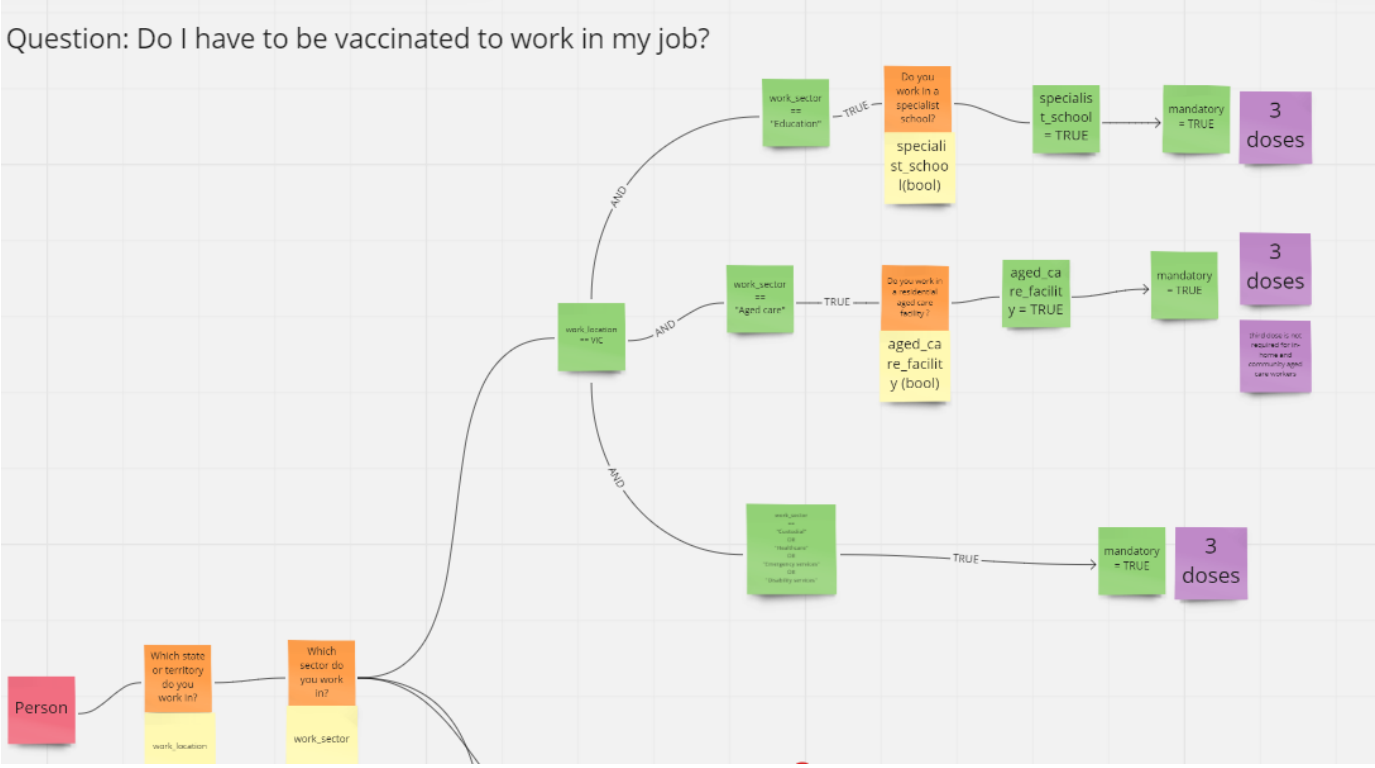
Below is the final rules mapping for up-to-date status.



Below is the final rules mapping for eligibility.



Below is the final rules mapping for COVID vaccinations in workplaces (in this case Victoria only).



Defining the necessary variables and parameters

During the mapping process, it is useful to to define the necessary variables and parameters. This can be done either as a specific task or concurrently with the rules mapping (doing this in the decision model stage is most efficient).

OpenFisca distinguishes between variables (input variables and formula variables) and parameters. Input variables are typically properties of a person or entity, such as a person's age or the number of vaccination doses they have had. Formula variables are where the results of rule calculations are stored and outputted. For example, when we calculate whether a person is up-to-date with their vaccination doses, the result is returned via a formula variable (ie. covid_vaccination_up_to_date). Finally, parameters are properties associated with the rule or legislation, such as the minimum age a person is eligible for vaccination.

Examples of input variables we identified are shown in the table below.

Question/term	**Variable**
Age	age (number)

Date of last vaccine dose?	last_vaccine_dose (date)
How many vaccine doses have you had?	vaccine_doses (bool)
Severely immunocompromised OR disability with significant or complex health needs, OR complex and/or multiple health conditions that increase the risk of severe COVID-19	immunocompromised_disability
Which state or territory do you work in?	work_location
Which sector do you work in?	work_sector
Do you work solely in a private residential (home) setting	private_home_only
Do you work in a specialist school?	specialist_school
Do you work in a residential aged care facility?	aged_care_facility
Are you a disability worker in an education setting?	disability_worker_in_school
Do you work for NSW Health?	NSW_health_worker

There are more than 40 parameters implemented for the proof-of-concept. Examples of key parameters used are listed below: .

Question/term	**Parameter**
Minimum age for eligibility	min_age_of_eligibility
Recommended number of doses	recommended_doses (there are different recommended dose parameters for different age groups)
COVID infection wait time	infection_eligibility (number)
Recommended doses for work	work_recommanded_doses (there are different recommended dose parameters for different states and work sectors)

Appendix 4 - Rules statements

IMPORTANT NOTE: These representations are our understanding of the rules as at Jul/Aug 2022. These are examples only and should not be relied upon for medical advice. Normally subject matter and policy experts would be part of the rules codification process - for the purpose of this POC publicly available information was used and interpreted with this involvement.

Rules statements provide a summary of the rules/law in plain English and in terms that can be coded.

The following rules statements represent the COVID-19 vaccinations rules we used for this PoC. In some cases we have simplified the logic, e.g. combining the health conditions of 'severely immunocompromised' with other health conditions that may require the person to have additional doses.

Recommended doses

- If a person has had the recommended number of doses they are considered 'up to date'.
- The recommended dose for 5-11 year olds is 2.
- The recommended dose for 5-11 year olds who are severely immunocompromised is 3.
- The recommended dose for 12-15 year olds is 2.
- The recommended dose for 12-15 year olds who are immunocompromised or have a disability with significant or complex health needs, or have complex and/or multiple health conditions that increase the risk of severe COVID-19 is 3.
- The recommended dose for people 16-49 years old is 3.
- The recommended dose for 16-49 year olds who are immunocompromised or have a disability with significant or complex health needs, or have complex and/or multiple health conditions that increase the risk of severe COVID-19 is 4.
- The recommended dose for people 50 years old and over is 4.
- The recommended dose for people 50 years old and over who are immunocompromised or have a disability with significant or complex health needs, or have complex and/or multiple health conditions that increase the risk of severe COVID-19 is 5.
- If it's been more than 6 months since someone's last dose, they are not up to date.

Eligibility

- Children under 5 (4 and under) are not eligible for a vaccination.

- People aged 5-29 and 50+ are only eligible for a vaccination dose if they're not up to date.
- People aged 30-49 years old who have had the recommended doses for their situation are also eligible for an additional vaccine dose (winter booster).
- Anyone who's had COVID in the past 3 months is not eligible for a vaccination - they need to wait until after 3 months to get their next dose (if eligible otherwise)

Vaccines

- The recommended vaccine for people aged 5-17 is Pfizer.
- The recommended vaccines for people aged 18 and over are Pfizer and Moderna.
- People aged 18 and over can have Astrazeneca in certain circumstances and can have Astrazeneca as booster (3rd & 4th doses) if they can't have the Pfizer vaccine for medical reasons or have had 2 doses of the AstraZeneca vaccine previously
- People aged 18 and over 18+ can have Novavax in certain circumstances and can have Novavax as a booster (3rd & 4th doses) if no other COVID-19 vaccine brand is suitable for that individual.

Mandatory vaccination for work

- If a person works in the "Education" sector in Victoria and works in a specialist school, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Aged care" sector in Victoria and works in a residential aged care facility, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Custodial" sector in Victoria, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Healthcare" sector in Victoria, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Emergency services" sector in Victoria, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Disability services" sector in Victoria, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the "Disability services" sector in Western Australia and does NOT work solely in a private residential (home) setting, then it is mandatory for them to be vaccinated (3 doses)

- If a person works in the “Healthcare” sector in Western Australia and does NOT work solely in a private residential (home) setting, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the “Aged care” sector in Western Australia and does NOT work solely in a private residential (home) setting, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the “Education” sector in New South Wales and is a disability worker in an education setting, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the “Aged care” sector in New South Wales and works in a residential aged care facility, then it is mandatory for them to be vaccinated (2 doses)
- If a person works in the “Aged care” sector in New South Wales but does not work in a residential aged care facility, then it is mandatory for them to be vaccinated (3 doses)
- If a person works in the “Healthcare” sector in New South Wales and works for NSW Health, then it is mandatory for them to be vaccinated (2 doses)
- If a person works in the “Healthcare” sector in New South Wales and does NOT work NSW Health, then it is not mandatory for them to be vaccinated
- If a person works in the “Disability services” sector in New South Wales, then it is mandatory for them to be vaccinated (3 doses)