CDR Engineering Workshop #1

March 2019

Agenda

Welcome - Warren Bradey

Introductions - John Brøndum

1. Delivery Strategy
2. Delivery Schedule

Engineering - Stuart Low

1. Engineering Strategy
2. Engineering Stream Management
3. Modelling of the API Standards
3. Conformance Test Suite
4. Reference Implementation
5. Desktop Sandbox

Conclusion

The Team

Stream	Who	Role
DSB Chair	Andrew Stevens	Interim chair for DSB
Overall	Warren Bradey	Program Director
Overall	John Brøndum	Head of Tech Delivery
Overall	Rob Hanson	Energy
Overall	James Bligh	Lead architect
Overall	*Hiring*	Tech writer
API	*Pending*	API lead
Engineering	Stuart Low	Lead Engineer
Engineering	*Pending*	Engineer
Engineering	*Pending*	Engineer
CX	Michael Palmyre	Lead CX
CX	Eunice Ching	CX researcher
CX	Brett Campbell	CX consultant
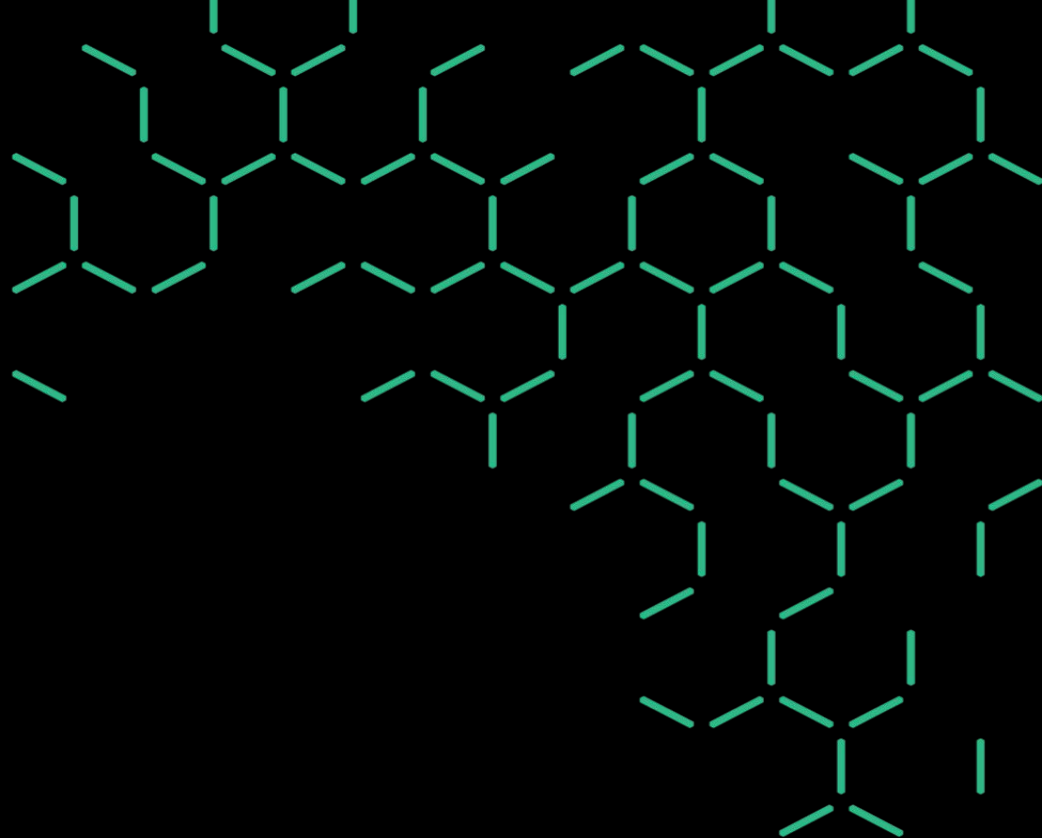
Delivery Strategy

Artefacts and Collaboration

Delivery Goals

Goal	Description	Rationale	Implication
Protection	Protected data sharing	• Consumers are unlikely to consent to data sharing, if CDS doesn't protect them sufficiently	• Secure data exchange • Enforceable data protection
Education	Help the technology community understand the standards through an interactive environment	• Wider and faster CDS implementation	• Need for documentation, demos, sample implementation
Compatibility	Minimise the differences across data holders	• Minimise data recipient implementation complexity by improving consistency across data holder implementations	• Technical validation through automated test script execution and reporting • Operational monitoring
Validation	Deliver the best possible technical standard specifications	• Eliminate technical errors in the specifications • Identify inconsistencies, ambiguities in the specification	• Model data recipient & holder implementation • Continuous testing of the standard
Innovation	Promote new & better services through safely shared information	• Enhance competition delivers better customer outcomes	• Metrics for CDS access and usage • Low cost data access

Components Context

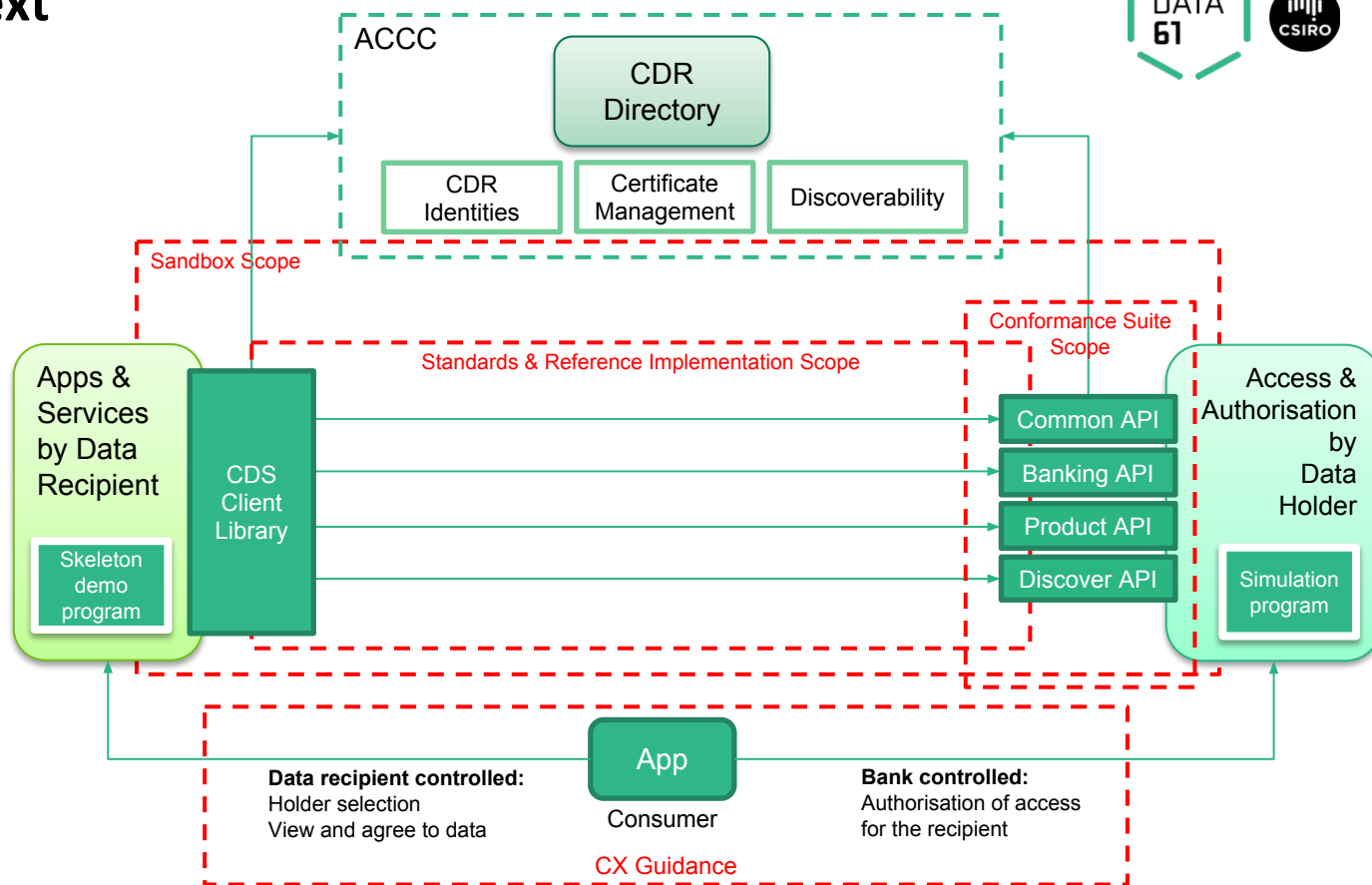Consumer Data Standard documents the data flow between consumer, recipients and holders

CX guidelines provide recommendations for the CDR community on CDS implementation

Reference implementation defines the developer version of the standards

Demo and simulation program provides a skeleton program (no UI) utilising the APIs

Conformance test suite provides data and test cases to test and improve compatibility across data holder API implementations

Sandbox wraps the standard, reference implementation and conformance suite into a deployable software package

ACCC

CDR Directory

CDR Identities

Certificate Management

Discoverability

Sandbox Scope

Conformance Suite Scope

Standards & Reference Implementation Scope

Apps & Services by Data Recipient

Skeleton demo program

CDS Client Library

Common API

Banking API

Product API

Discover API

Access & Authorisation by Data Holder

Simulation program

App

Consumer

Data recipient controlled:
Holder selection
View and agree to data

Bank controlled:
Authorisation of access for the recipient
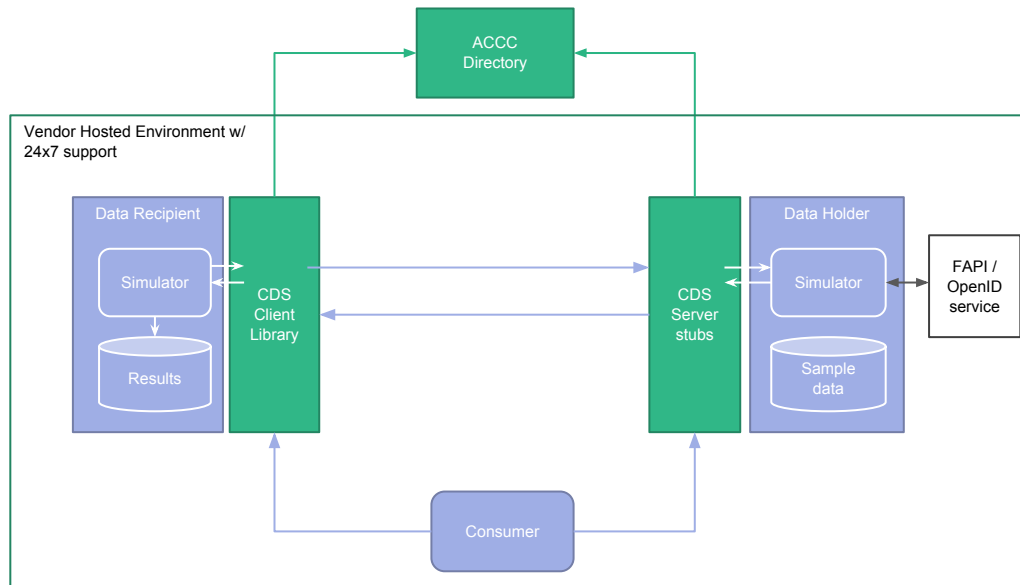
CX Guidance

DATA 61

CSIRO

Sandbox: cloud vs desktop edition

Sandbox – desktop edition is downloadable version of Java reference implementation, Java demo and conformance. Developers will install and run the required software on their developer machines or test environment.

Available to all data recipients and holders

Sandbox – cloud edition is vendor hosted cloud option with a set of pre-configured, integrated development environments.
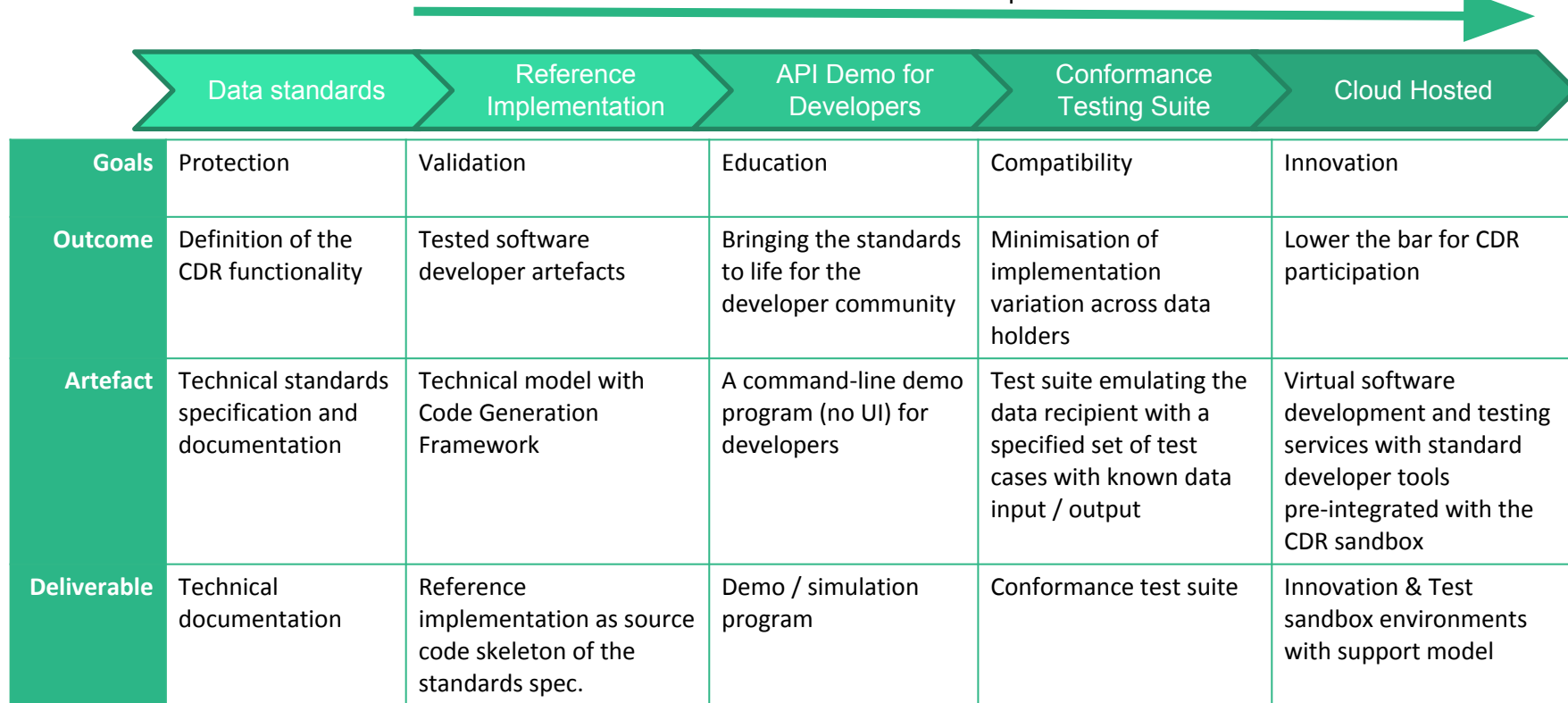
This may form part of a commercial agreement between hosting vendor and data recipients / holders; including support arrangements
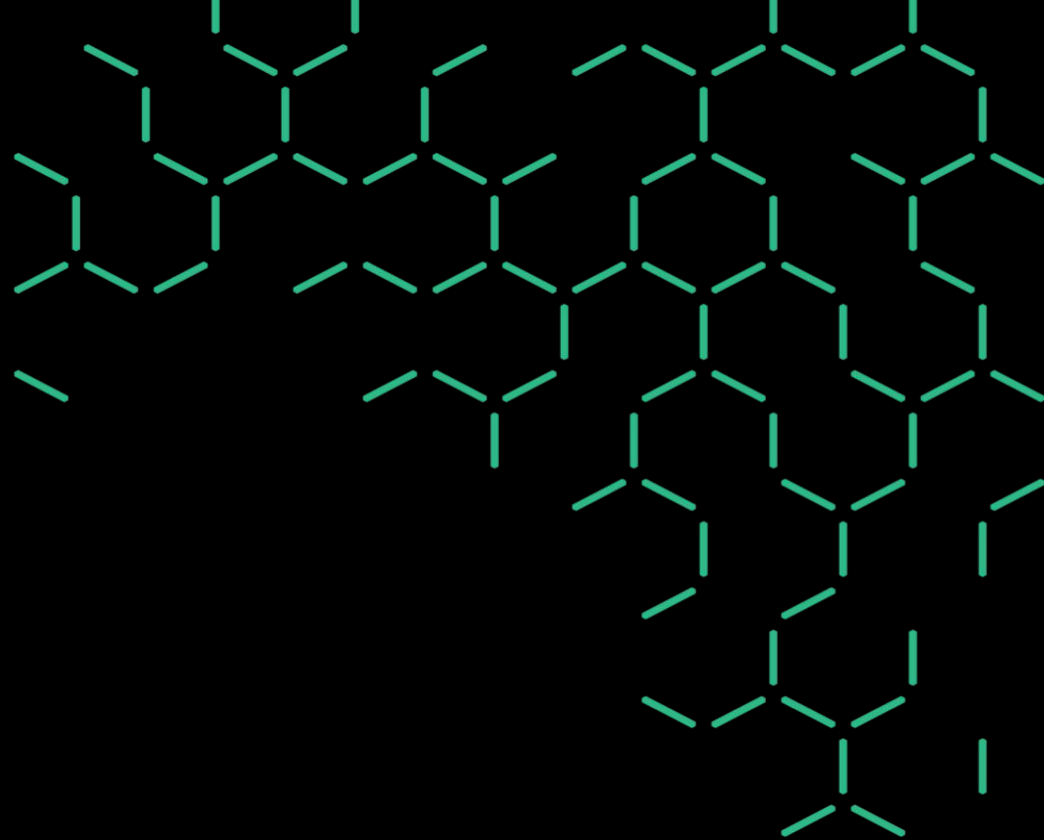
Delivery Dependencies

Sandbox scope →

	Data standards	Reference Implementation	API Demo for Developers	Conformance Testing Suite	Cloud Hosted
Goals	Protection	Validation	Education	Compatibility	Innovation
Outcome	Definition of the CDR functionality	Tested software developer artefacts	Bringing the standards to life for the developer community	Minimisation of implementation variation across data holders	Lower the bar for CDR participation
Artefact	Technical standards specification and documentation	Technical model with Code Generation Framework	A command-line demo program (no UI) for developers	Test suite emulating the data recipient with a specified set of test cases with known data input / output	Virtual software development and testing services with standard developer tools pre-integrated with the CDR sandbox
Deliverable	Technical documentation	Reference implementation as source code skeleton of the standards spec.	Demo / simulation program	Conformance test suite	Innovation & Test sandbox environments with support model

Dependencies illustrated from left to right…

Delivery scoping & partnering options

Deliverables	Who	Governance	Focus	Goal alignment
1 CDR standards	Data Standards Body with support from the CDR community	Data Standards Body Open source license	Standardised data access required to facilitate innovation including CX guidelines	• Innovation • Protection
2 Java Reference Implementation	Data Standards Body with support from the CDR community	Data Standards Body Open source license	Focus on standards maintenance and enable extensions via the CDS code generator and templates	• Validation • Education
2b Other Reference Implementations	CDR community	Open source license and/or commercial agreement	Extend the reference implementation to other and future technology options	• Innovation
3 Java Demo for Developers	Data Standards Body with support from the CDR community	Data Standards Body Open source license with recognition of significant donations	Bring the standard to life for the developers	• Education
3b Other Demo for Developers	CDR community	Open source license and/or commercial agreement	Bring the standard to life for the developers using other technologies	• Education • Innovation
4 Conformance Test Suite	Data Standards Body with support from the CDR community	Data Standards Body Open source license with recognition of significant donations	Aim at the best possible compatibility across data holders	• Compatibility
5 Cloud hosting	Commercial vendors, Data holders	Commercial contract between vendor and CDR participant	Build scale to support many industries and participants; promote innovation and price competitiveness, and support best of breed	• Innovation • Education

To be noted …

1. Engineering contributions to the CDS Open Source Project requires the contributor to transfer their Intellectual Property Rights (IPR) to the project under MIT license:

   - Project "pull requests" (aka source donation) can only be accepted if the contributor accepts the transfer of IPR; AND the contributor is the sole Intellectual Property owner of the contribution (e.g. they cannot contribute other legal entities software)

2. Delivery of "conformance testing" will be progressively scoped, as it is a scale from "technical validation" to "interpreting business semantics"

   - The aim is to improve the testing performed such that the final testing activity with the data holders' own test environment is minimised

   - Data holders are encouraged to contribute test data and test cases to the conformance suite to improve the overall quality

3. DSB needs access to a commercial implementation of the FAPI/OpenID technology, as we haven't identified a suitable open source implementation available within our delivery timeframe

   - Vendor may donate software product access for Data61 for dev / test purposes, but Data61 **will not** distribute the software

Delivery Schedule

Tentative Delivery Schedule

Pending CDR legislation & ACCC Directory

Stream	Deliverable	Version	Dates	Dependencies	Dates	Comments
API	Data Standards specification • Common API	v1.0	April ish*	ACCC rules	TBC	Details the personal information APIs
API	Data Standards specification • Banking API	v1.0	April ish*	ACCC rules	TBC	Inclusive of the product reference APIs
API	Data Standards specification • InfoSec profile	v1.0	April ish*	ACCC Directory design	TBC	
CX	Data Standards CX Guidelines: • Language & payloads • Prototype design & build • Prototype testing & consumer interviews	Phase 1	20 Feb 2019	Data Standards (Dec 2018)	18th Dec 2018	Report has been released
Engineering	Reference Implementation: • CDS Java Client Library • CDS Java Server Stub	v1.0	Early May 2019	Data Standards	See above	Estimate 4 months total effort inclusive of CDS Java model, CDS Code Generator, and CDS Templates
Engineering	Sandbox (desktop): • Data recipient CLI simulator using CDS Java Client Library • Data holder CLI simulator using CDS Java Server Stub	v1.0	Early July 2019	Reference Implementation	See above	CLI test suite available, but it has been not tested against other data holder implementation nor their data (aka conformance test suite v1.0)
Engineering	Conformance test suite • Test framework • Technical validation	v1.1	Aug 2019	Data Standards Sandbox (desktop)	See above	Web UI addition Test management functions

Stuart Low

Stuart Low
Lead Engineer

Known as @perlboy

Operating as @csirostu
in CSIRO Data61 capacity

B.InfTech (Inf Systems)
University of Queensland

Experience:

- Blockchain Engineer with Ethereum (Solidity), Truffle Suite, Java & Open Banking APIs
- Formally Head of Innovation, Rabobank Australia & New Zealand
- Senior Distributed Systems Engineer over 20 years through variety of roles including:
  - Datacentre Architect
  - Systems Architect
  - Devops Practice Founder
  - Senior Storage Engineer
  - Senior Systems Engineer
  - Senior Java Developer (and Perl once upon a time)

Projects:

- Blockchain Core Banking Platform, Biza.io
- Geofarmer.io Big Data for Farmers (Internal Startup), Rabobank AU/NZ
- Datacentre Relocation Project, Rabobank AU/NZ
- Enterprise Storage Replacement, Rabobank AU/NZ
- Large-scale 4G ISP Implementation, (Phase 1,2,3), iseek for Virgin Mobile
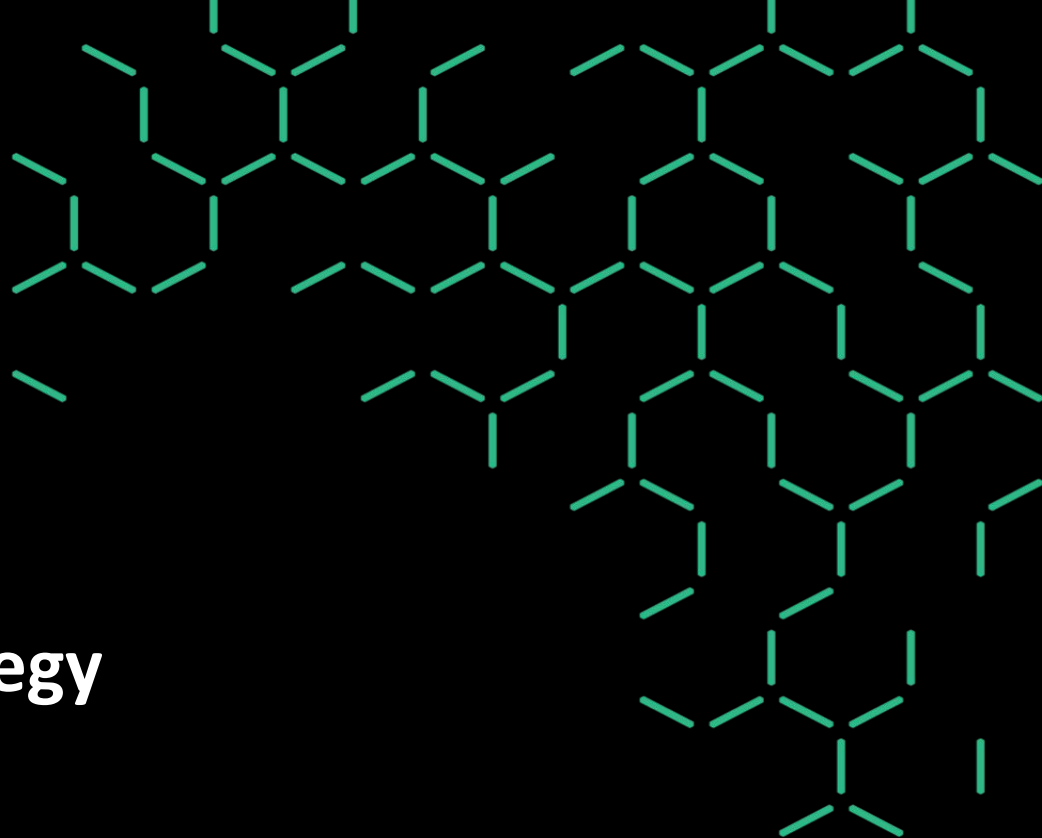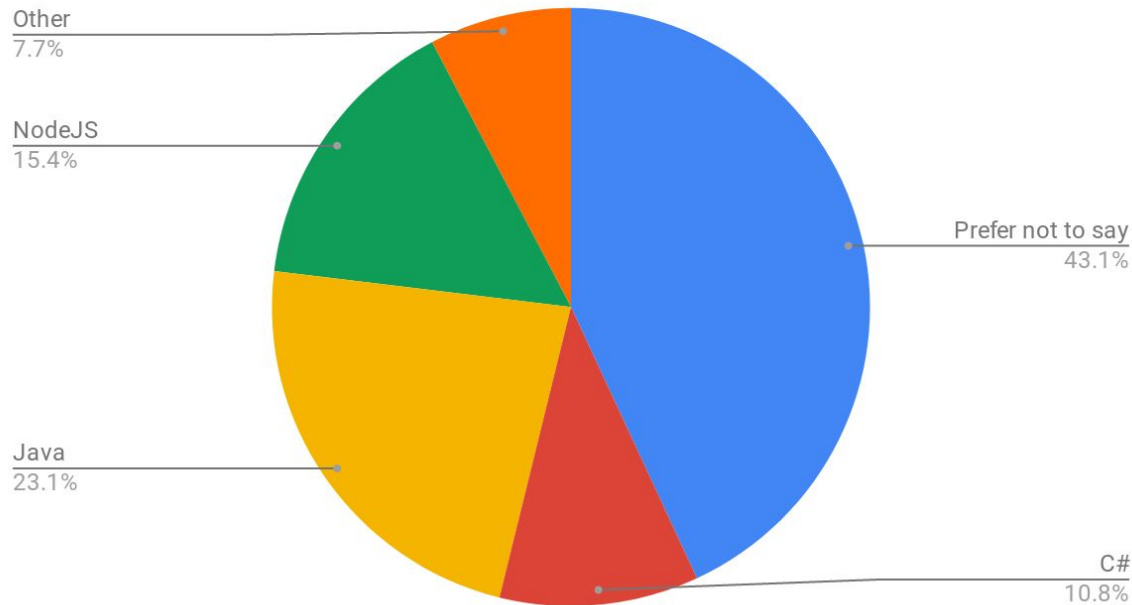
Engineering Strategy

Stuart Low

Goal

To deliver software that is
<u>useful</u>
to participants

Initial Survey Results

CDR Participants: Target Language

- Other 7.7%
- NodeJS 15.4%
- Java 23.1%
- Prefer not to say 43.1%
- C# 10.8%

- **Vendors in use:** Essentially Nil answers

- **Level of Progress:** Most participants either preferred not to answer or are in planning phase

- **How long to implement:** Most participants preferred not to answer. Most of those that did estimated approximately 3 months.

Further Industry Consultation

- We can't help make it easier to implement the CDR if industry doesn't tell us what they want!

- There will be a follow-up survey after these workshops, we would greatly appreciate if you could complete it!

- We will be collating a set of Issues within the Engineering GitHub repository to allow for direct industry feedback

Strawman Disclaimer

We need industry to validate what we are proposing

We will implement...

1. A <u>Conformance Test Suite</u> to test Data Holder compatibility (minimise implementation deviation)

2. A <u>CDS Code Generation</u> tool that outputs <u>Reference Implementations</u> (initially Java) to ensure a firm foundation of DSB foundation

3. A <u>Desktop Sandbox</u> of the Standards API

We won't implement

1. A Cloud Sandbox: This has Operational Support implications and has been deemed outside the scope of the Data Standards Body.

2. A "Graphical Demo App": CX Stream are providing guidelines but, at least engineering wise, it's up to participants to make their own decisions.

BUT
We encourage 3rd Parties to consider doing so

Engineering Stream Management

Stuart Low

Task Management

1. Agile with GitHub Kanban with Fortnightly Sprints

2. Task tracking available publicly in GitHub Projects:
   https://github.com/ConsumerDataStandardsAustralia/engineering/projects/1
   https://github.com/ConsumerDataStandardsAustralia/cdr-codegen/projects/1
   https://github.com/ConsumerDataStandardsAustralia/cdr-models/projects/1

3. All completed tasks are approved by at at least 1 alternate internal reviewer

4. External Issues & Pull Requests are welcomed and automatically enter Backlog status pending integration

Documentation

- Available at: https://consumerdatastandardsaustralia.github.io/engineering/
- Sphinx compiled RESTdown automatically published by Jenkins. Dockerfile for local Docker container execution.
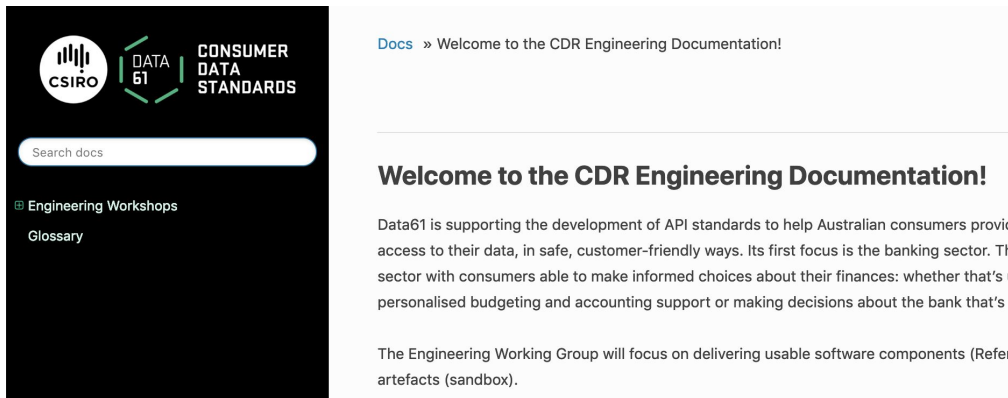
- May change pending Technical Writer (we are hiring!)

Source Control

Growing number of public repositories

- Engineering Doc: https://github.com/ConsumerDataStandardsAustralia/engineering
- CDS Models: https://github.com/ConsumerDataStandardsAustralia/cds-models
- CDS Codegen: https://github.com/ConsumerDataStandardsAustralia/cds-codegen
- Github Base Website: Landing page on github (aka Stu kept 404'ing)

Private Repositories intended to be transplanted to the public artefacts:

- Standards Staging: Internal API discussion
- CDR Development: Scratch documentation of what is in this presentation

Private Administration repositories:

- Ansible rulesets: Repo for our cloud infra

Versioning

- GitLab Flow:
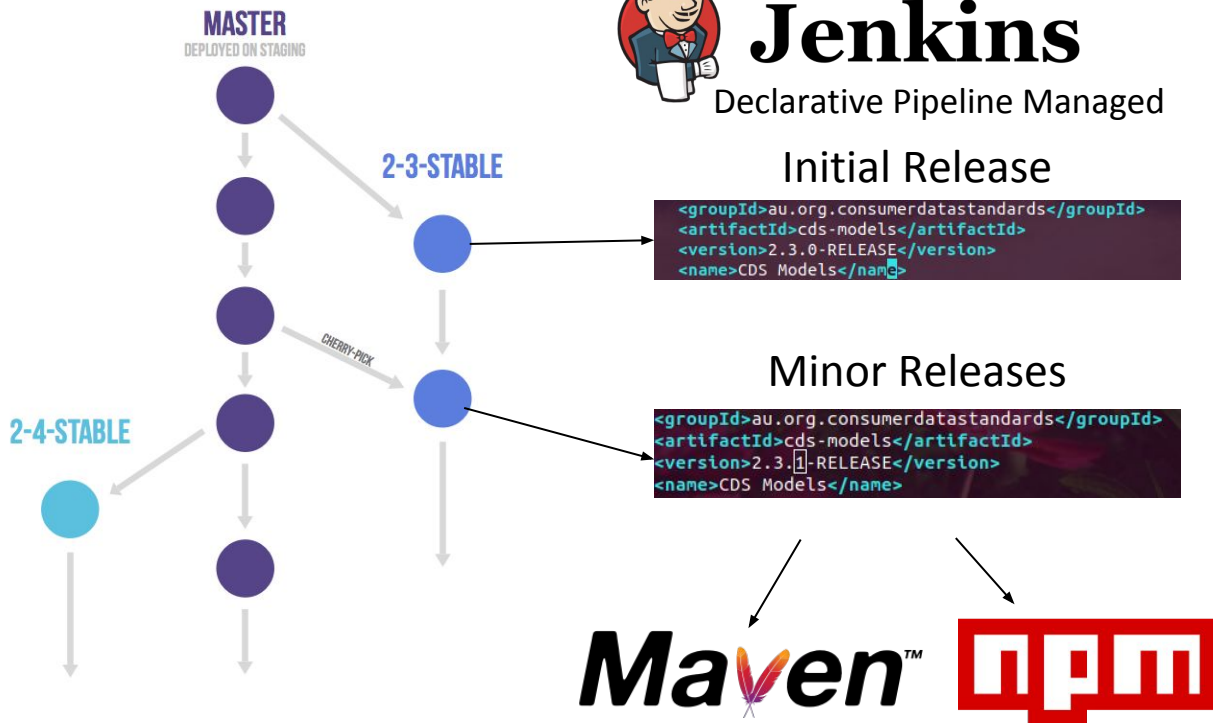 https://docs.gitlab.com/ee/workflow/gitlab_flow.html

- Branches for one or a group of Features

- Mainline branches for official releases with cherry-picked fixes

- Pre 1.0: Building SNAPSHOTS only

Feature Development
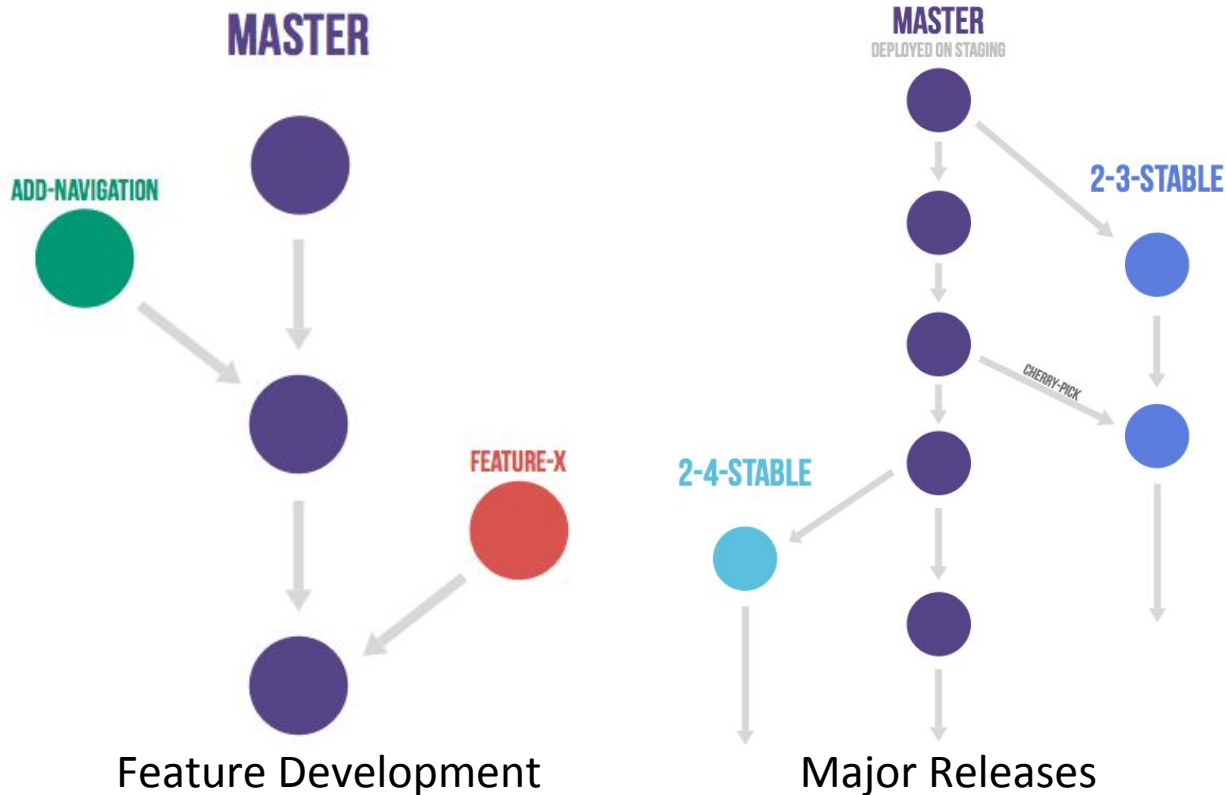
Major Releases

Release Management

- X.Y.0
  Major Version Release

- X.Y.Z
  Minor Point Release (backward compatible)

- Internally Jenkins compiled and where relevant:

  - Published to package/container repositories

  - Triggers build and package of downstream outputs (Reference Implementation results)

  - Executable Jar's published in GitHub Releases

MASTER
DEPLOYED ON STAGING

2-3-STABLE

CHERRY-PICK

2-4-STABLE

Jenkins
Declarative Pipeline Managed

Initial Release

```xml
<groupId>au.org.consumerdatastandards</groupId>
<artifactId>cds-models</artifactId>
<version>2.3.0-RELEASE</version>
<name>CDS Models</name>
```

Minor Releases

```xml
<groupId>au.org.consumerdatastandards</groupId>
<artifactId>cds-models</artifactId>
<version>2.3.1-RELEASE</version>
<name>CDS Models</name>
```

Maven™ **npm**

System Architecture Notes

- Preference to release Docker containers for executables coupled with orchestration using Vagrant.

- Assumption that public repositories (eg. Maven/NPM) are acceptable

- All compilations conducted on internal Jenkins infrastructure

- Ship Jenkinsfile & Dockerfile definitions with source, publish Docker containers where applicable

  Where possible also include travis-ci.yml's to allow independent compilation verification:
  https://travis-ci.com/ConsumerDataStandardsAustralia

		LAST BUILD	DEFAULT BRANCH	COMMIT	FINISHED
	✓ engineering	# 49	-o- master	⑂ 362dacf ⬈	📅 Passed about 13 hours ago
	✓ standards	# 12	-o- master	⑂ ae5148e ⬈	📅 Passed 8 days ago

Modelling of the API Standards

Stuart Low

Engineering Artefact Overview

Data Standards Body

API Standards

Industry Consultation

CDS Models Library

Code Generator

Conformance Test Suite

Java
Server Stubs
Client Library

Sandbox

Industry Contributions
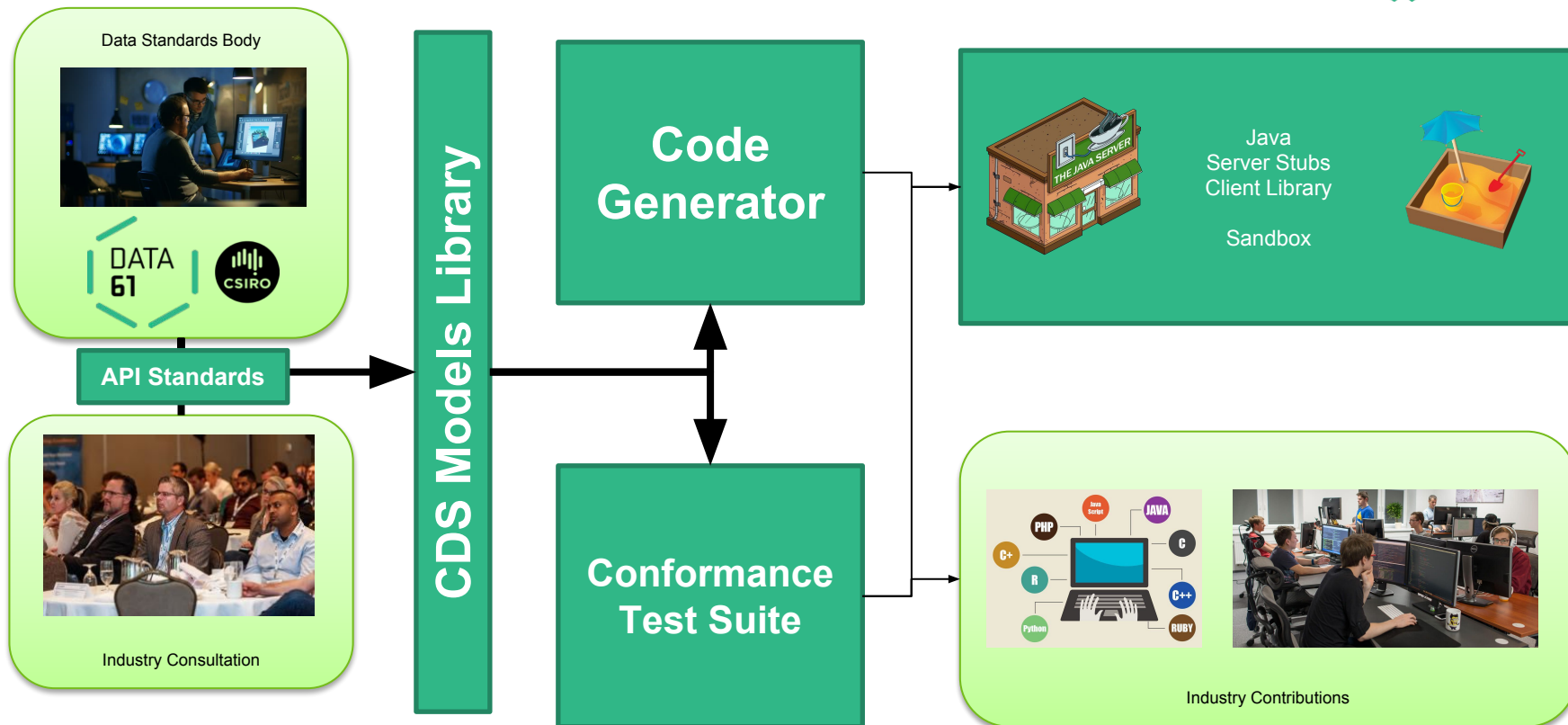
Why not just Swagger / OpenAPI?

What we get

- Generalised API definition standard

- Growing industry adoption

- Active project being worked on continuously (oas3 codegen implementation is becoming a hard fork)

- Multiple language "generic" output support with templating system for expansion

What we don't

- Not CDS Specific

- Inflexible for work outside of specific API definition

- Zero support in Swagger/OpenAPI2 for FAPI driven auth schemes, limited in OpenAPI3

- Specific data typing, custom type definition limited in both support and compatibility (CDS has 18 data types so far)

- Generators deliberately ignore vendor extensions (ie. "x-" prefix attributes). String only tokenisation resulting in array hunting in Mustache templates (ick!)

- (Subjectively) Codebase is pretty ugly….

Bridging Standards API with Code Generation (using Java)

Pros

Issues

- CDS Specific definition standard provides essentially unlimited support for custom data types

- Gradual degradation of "fidelity" in output formats (ie. "best we can do for target X")

- Additional templates can be contributed and/or built in a way that is separated from the API definition itself

- Large amount of access to Java resources

- Can still pass through to swagger-codegen for "alpha release" testing (just don't take it to prod ok? :))

- Requires initial model definition from defined spec

- Data Standards Body commits to ongoing maintenance and support.

Example

```
25  @Section(name = "Common API", tags = { "Customer" })
26
27  public @interface CommonApi {
28          @Endpoint(
29      path = "/customer",
30      summary = "Get Customer",
31      description = "Obtain basic information on the customer that has authorised the current session",
32      requestMethod = RequestMethod.GET,
33      operationId = "getCustomer",
34      responseList = {
35              @EndpointResponse(
36                  responseCode = "200",
37                  description = "Success",
38                  content = ResponseCommonCustomer.class
39              ),
40              @EndpointResponse(
41                  responseCode = "401",
42                  description = "Unauthorised",
43                  content = ResponseErrorResponse.class
44              )
45      },
46      requiredAuth = {
47        @EndpointAuth(
48              type = EndpointAuthType.CDS_FAPI,
49              access = EndpointAuthAccess.READ_ONLY,
50              scope_name = {
51                      Scopes.COMMON_BASIC_CUSTOMER
52              }
53          )
54      }
55      )
56      ResponseCommonCustomer getCustomer();
57
58  }
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/tree/master/src/main/java/au/org/consumerdatastandards/interfaces/api

```
@Documented
@Inherited
@Retention(RetentionPolicy.RUNTIME)
public @interface ResponseCommonCustomer {
        @Property(
            name = "data",
            description = "The Customer Response Data",
                required = true
        )
        public ResponseCommonCustomerData data();

}
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/blob/master/src/main/java/au/org/consumerdatastandards/interfaces/api/responses/ResponseCommonCustomer.java

Example

```java
@Documented
@Inherited
@Retention(RetentionPolicy.RUNTIME)
public @interface ResponseCommonCustomer {
        @Property(
            name = "data",
            description = "The Customer Response Data",
                required = true
        )
        public ResponseCommonCustomerData data();

}
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/blob/master/src/main/java/au/org/consumerdatastandards/interfaces/api/responses/ResponseCommonCustomer.java

```java
public @interface ResponseCommonCustomerData {

    @Property(
                name = "customerUType",
                description = "The type of customer object that is present",
                required = true
    )
    public CustomerTypeEnum customerUType();

    @Property(
                name = "person",
                description = "The Person object of the Customer",
                required = false
    )
    public Person person();

    @Property(
                name = "organisation",
                description = "The Organisation object of the Customer",
                required = false
    )
    public Organisation organisation();

}
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/blob/master/src/main/java/au/org/consumerdatastandards/interfaces/api/responses/ResponseCommonCustomerData.java

Example


```java
public @interface ResponseCommonCustomerData {

        @Property(
                        name = "customerUType",
                        description = "The type of customer object that is present",
                        required = true
        )
        public CustomerTypeEnum customerUType();

        @Property(
                        name = "person",
                        description = "The Person object of the Customer",
                        required = false
        )
        public Person person();

        @Property(
                        name = "organisation",
                        description = "The Organisation object of the Customer",
                        required = false
        )
        public Organisation organisation();

}
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/blob/master/src/main/java/au/org/consumerdatastandards/interfaces/api/responses/ResponseCommonCustomerData.java

```java
public @interface Person {
        @Property(
                        name = "lastUpdateTime",
                        description = "The date and time that this record was last updated by the customer. If no update has
                        required = true
        )
        DateTimeString lastUpdateTime();

        @Property(
                        name = "firstName",
                        description = "For people with single names this field need not be present. The single name should be
                        required = false
        )
        UnicodeString firstName();

        @Property(
                        name = "lastName",
                        description = "For people with single names the single name should be in this field",
                        required = false
        )
        UnicodeString lastName();

        @Property(
                        name = "middleNames",
                        description = "Field is mandatory but array may be empty",
                        required = false
        )
        UnicodeString[] middleNames();

        @Property(
                        name = "prefix",
                        description = "Also known as title or salutation. The prefix to the name (e.g. Mr, Mrs, Ms, Miss, Sir
                        required = false
        )
        UnicodeString prefix();

        @Property(
                        name = "suffix",
                        description = "Used for a trailing suffix to the name (e.g. Jr)",
                        required = false
        )
        UnicodeString suffix();

        @Property(
                        name = "occupationCode",
```

https://github.com/ConsumerDataStandardsAustralia/cds-models/blob/master/src/main/java/au/org/consumerdatastandards/models/types/customer/Person.java

Conformance Test Suite

Stuart Low

www.data61.csiro.au

Conformance Test Suite Requirements

- Performs verification of Data Holder implementations

- Needs to be accessible to industry participants

- Needs to establish an extensible framework to allow continuous use case expansion

- Accepts uploading of "intended state" to validate data mapping is accurate

- Needs to provide a test report that is parseable and industry accepted

QFPL Validation Prototype (fka conformance)

- Developed by Queensland Functional Programming Lab in Haskell, packaged as a "black box" Docker container

- Early stage validation prototype containing:
  - Strongly typed data model of the Standards API
  - Alpha Mock Data Holder
  - Initial test suite (partially by proxy of it being Haskell)

- Resulted in:
  - 70+ pieces of feedback on the Standards API
  - 300+ code commits

- Ignited internal discussions around "What is Conformance?"
- Repository renamed to *validation-prototype*
- QFPL to provide "light touch" maintenance

Conformance Test Suite Context
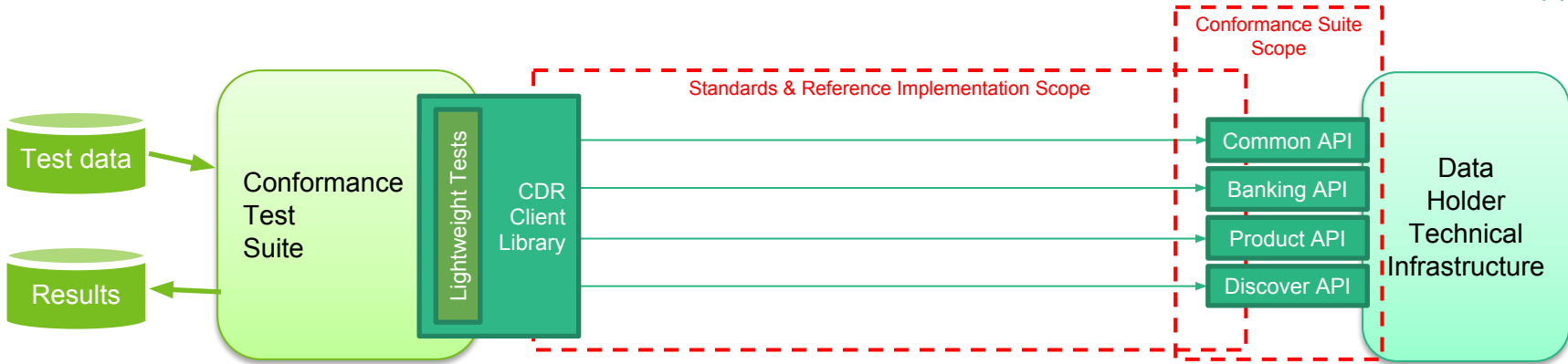
- Java based container with a Test Harness
- Validates the Standards API Model with Unit Testing during packaging
- Extends on Lightweight Tests to be included in the Java Reference Implementation
- Focused on validating Data Holders utilising Data Recipient client library
- Accepts "intended state" test data and provides results in a parseable format

What is Conformance?

Standards Definition	API Conformance	InfoSec Conformance
Payloads	Endpoints	FAPI Authentication
Versioning	Attribute Mapping	Scopes
Endpoints	Headers	ACCC Directory
Error Handling	Versioning	
	Data Types	
	Discoverability	
	Industry Use Cases	Industry Use Cases

Code Based Compliance

Guideline Based Compliance

CX Guidelines

What is Conformance?

Area	Description	Relevant Decision Proposals
Standards: Payloads	Payloads and Attributes consistently named and structured	Proposal 12 + Payload Proposals
Standards: Versioning	Is Versioning consistent and does it allow for graceful degradation	Proposal 04
Standards: Endpoints	Are Endpoint paths consistent and compliant	Proposal 08, 11
Standards: Error Handling	Are all relevant Errors sufficiently described	Proposal 11
API: Endpoints	Does a target API contain all the required Endpoints	Multiple
API: Attribute Mapping	Does each API Endpoint correctly map attributes (ie. first_name matches First Name etc)	Multiple
API: Headers	Each API Endpoint accept and/or provide defined headers	Proposal 02
API: Versioning	Each API Endpoint gracefully degrade based on Version specification	Proposal 04
API: Data Types	Each API Endpoint correctly demonstrate intended Data Types	Proposal 13
API: Discoverability	Does the API support the Discoverability definition	TBD
InfoSec: FAPI Authentication	Authentication & Authorisation server provides correct FAPI headers	Decision 35 + TBD
InfoSec: Scopes	Are all scopes available and enforced for all endpoints	TBD
InfoSec: ACCC Directory	Do Endpoints comply and enforce ACCC Directory admission criteria	TBD

Conformance: Industry Use Cases

- **What?**
 We are inviting contributions from industry participants

- **Why?**
  Data Holders have the best understanding of "edge cases" in the delivery of their data sets. A group effort is likely to get the greatest coverage that is **useful.**

- **How?**
  Data61 welcomes and invites contributions of conformance use cases in either:
    - Formalised Test Case document (template to be released)
    - Code based test case contributions (via pull requests)
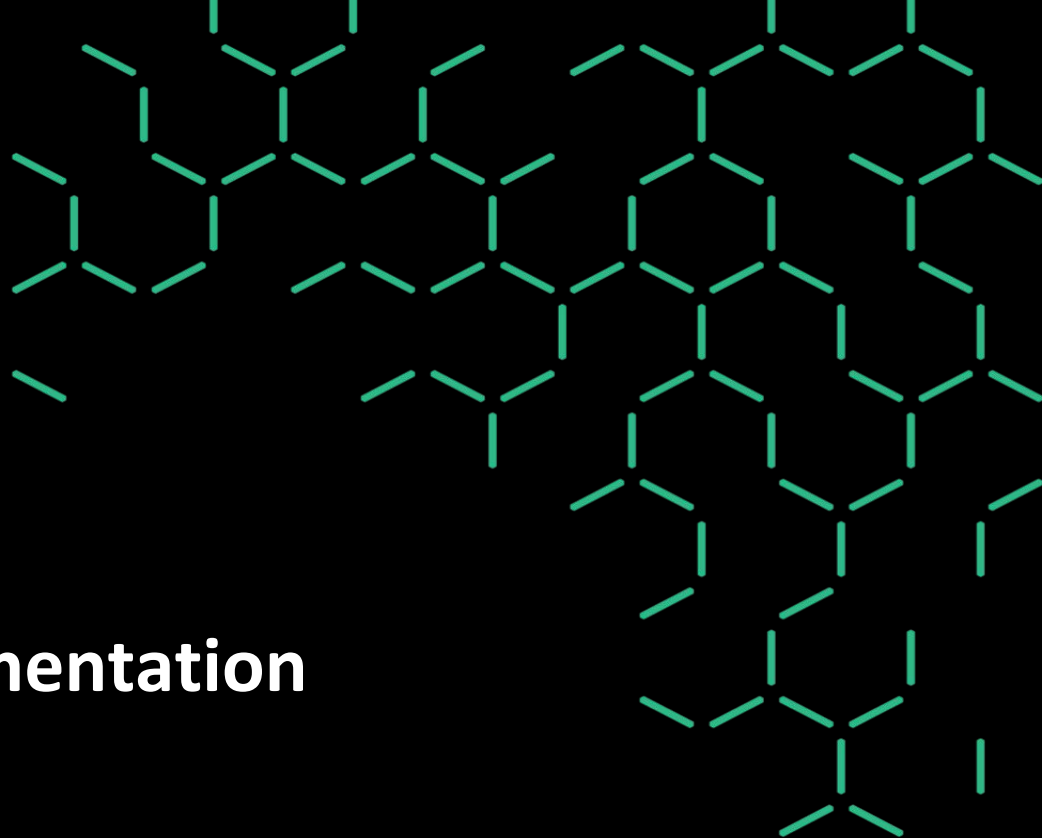
Reference Implementation

Stuart Low

Reference Implementation Requirements

- Provide full coverage of the Standards API for both Server & Client implementations.

- Provide consistent implementations in multiple languages, initially Java with additional languages based on industry feedback and/or contributions

- Available publicly and available in standard package formats

- Should consider potential for vendor specific backup/restore formats for commercial API Gateways

- Allow for community contributions to be made

Reference Impl Requirements: Data Holders

- Provide a skeleton of the Standards API for Data Holders to integrate their business logic

- Aim to be agnostic on deployment method by supporting implementation of multiple languages and multiple commercial vendor products (such as API gateways)

- Make it easier to pass Conformance Testing Suite compliance by demonstrating the Standard within a Data Holders familiar environment

- Aim for maintainability within an evolving Standards API

- Provide consistency across industries so that as the Consumer Data Standards evolve the same software suites can be utilised

- Integrate with the ACCC Directory technical requirements

# Reference Impl Requirements: Data Recipients

- Provide a consistent client library of the Standards API so that Data Recipients can focus on their business logic

- Aim to be agnostic on deployment method by supporting implementation of multiple languages and multiple commercial vendor products

- "Smoke tested" using Conformance Test Suite on Data Holders

- Aim for maintainability within an evolving Standards API

- Graceful degradation of versioning to allow smooth internal library transition

- Integrate with the ACCC Directory technical requirements
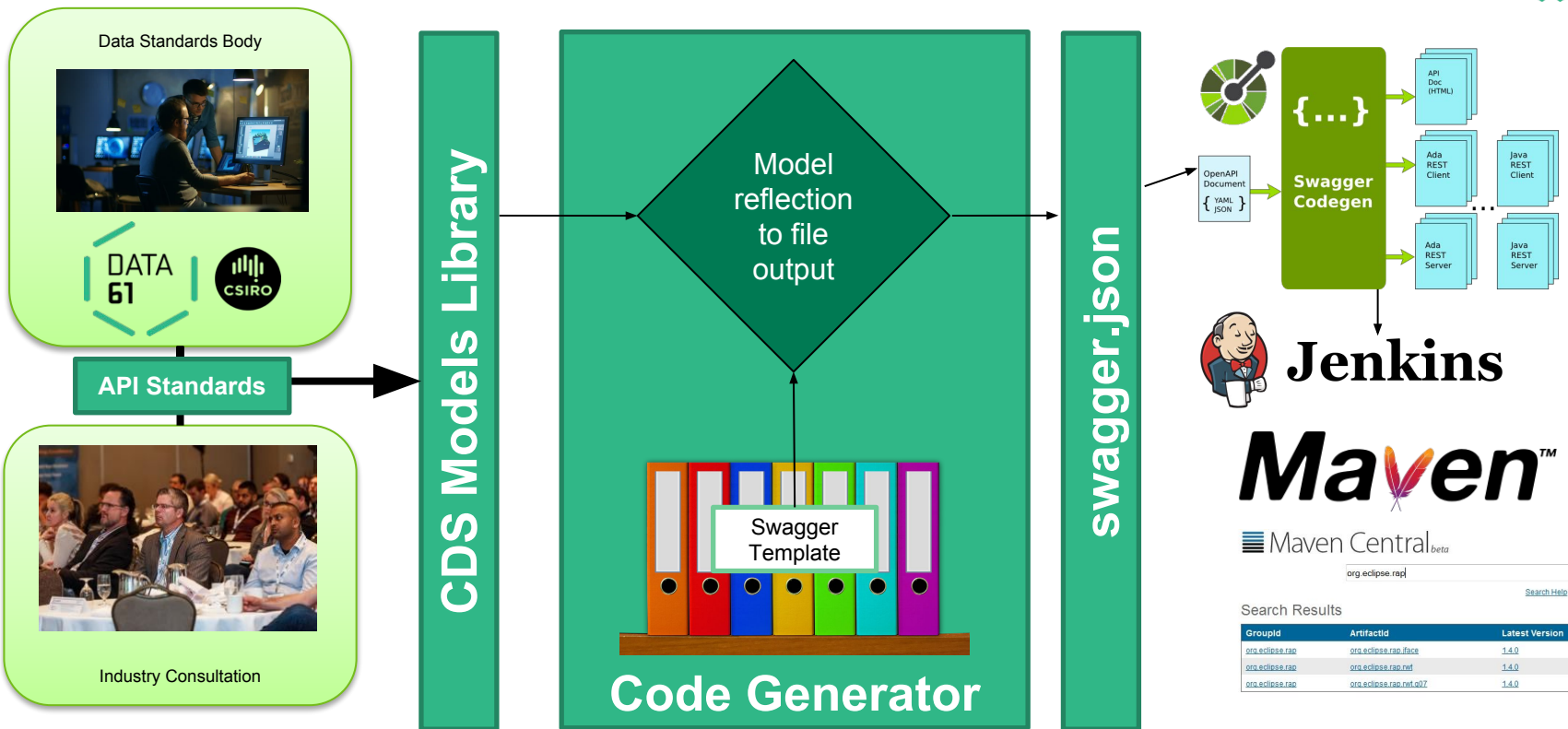
Reference Implementation Context
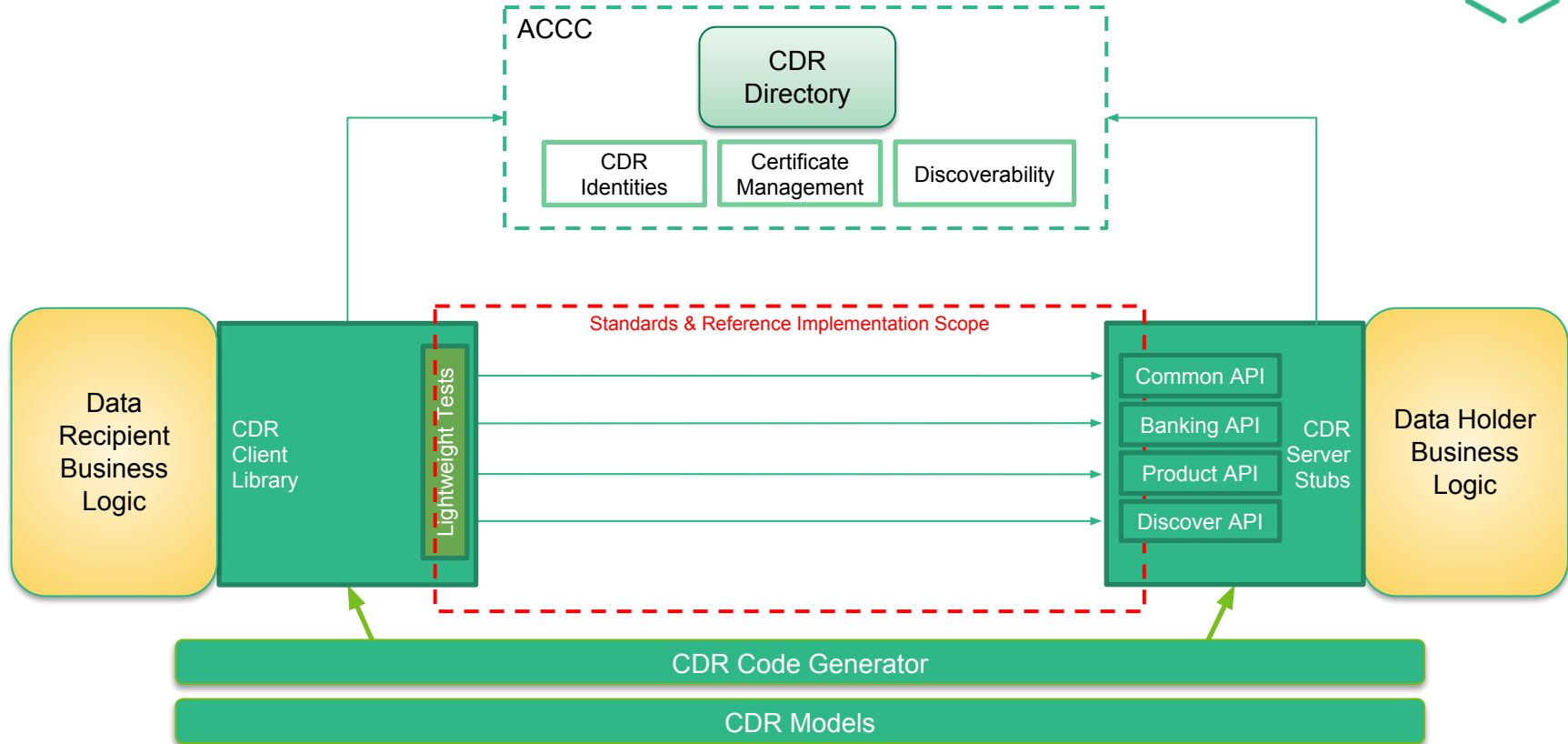
Reference Implementation Generation Workflow (Java)
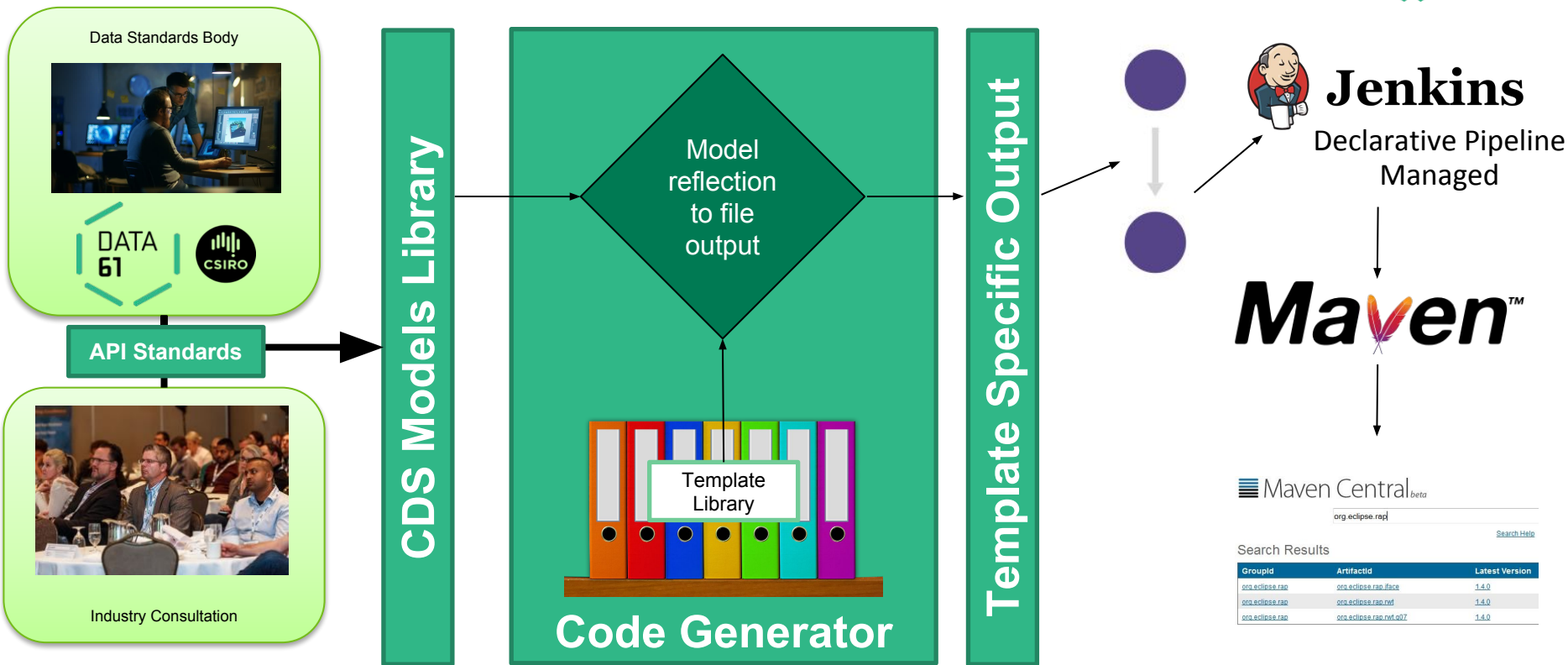
Reference Implementation Generation Workflow (Swagger to "Alpha Java")

Reference Implementation Template

- Apache Velocity based templates, very common templating library, you are likely to have received many emails from it

- Access to a data structure representative of interface definitions

- Template to File mapping either in Java POJO/YAML/JSON (tbd)

- Modelled roughly on swagger-codegen patterns

```
1   #parse( "common/shared.vm" )
2
3   {
4     "swagger": "2.0",
5     "info": {
6       "title": "Consumer Data Standards",
7       "description": "API sets created by the Australian Consumer Data Standards to meet the needs of the Consumer Data Right",
8       "version": "1",
9       "license": {
10        "name": "MIT Licence",
11        "url": "https://opensource.org/licenses/MIT"
12      }
13    },
14    "host": "data.provider.com.au",
15    "basePath": "/cds-au/v1",
16    "schemes": [
17      "https"
18    ],
19    "consumes": [
20      "application/json"
21    ],
22    "produces": [
23      "application/json"
24    ],
25    "paths": {
26  #foreach($section in $api.sections)#foreach($endpoint in $section.endpoints)
27      "$endpoint.path()": {
28        "get": {
29          "summary": "$endpoint.summary()",
30          "description": "$endpoint.description()",
31          "operationId": "$endpoint.operationId()",
32          "tags": [
33  #foreach ($oneTag in $section.definition.tags())
34            "$oneTag"#if( $foreach.hasNext ),#end
35  #end
36
37          ],
38          "responses": {
39  #foreach($endpointResponse in $endpoint.responseList())
40            "$endpointResponse.responseCode()": {
41              "description": "$endpointResponse.description()",
42              "schema": {
43                "$ref": "#/definitions/$endpointResponse.content().getSimpleName()"
44              }
45            }
```

SUPER BETA POC

https://github.com/ConsumerDataStandardsAustralia/cds-codegen/blob/master/src/main/resources/swagger/swagger.json.vm

DATA 61 · CSIRO

Desktop Sandbox

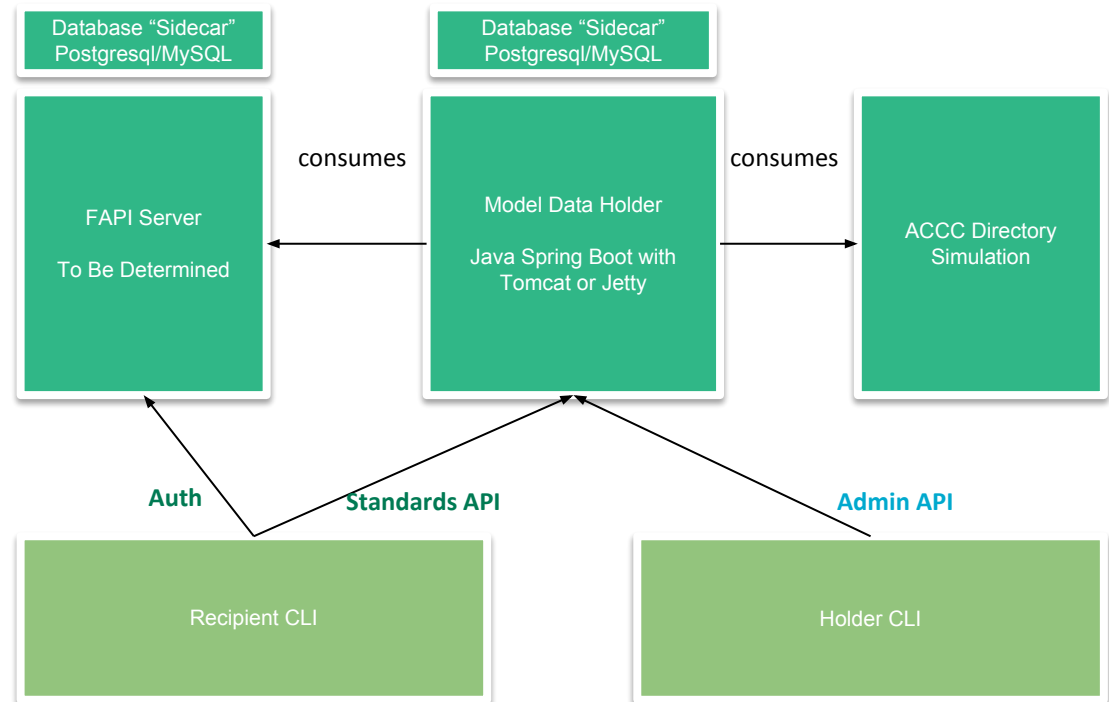Stuart Low

Desktop Sandbox Requirements

- Demonstrate the Standards from a Data Provider perspective

- Demonstrate the Standards from an Infosec Standpoint

- Demonstrate the Standards from a Data Recipient perspective

- Simulate the ACCC Directory

- Allow a Holder/Recipient developer to replace individual Desktop Sandbox components

Sandbox Context
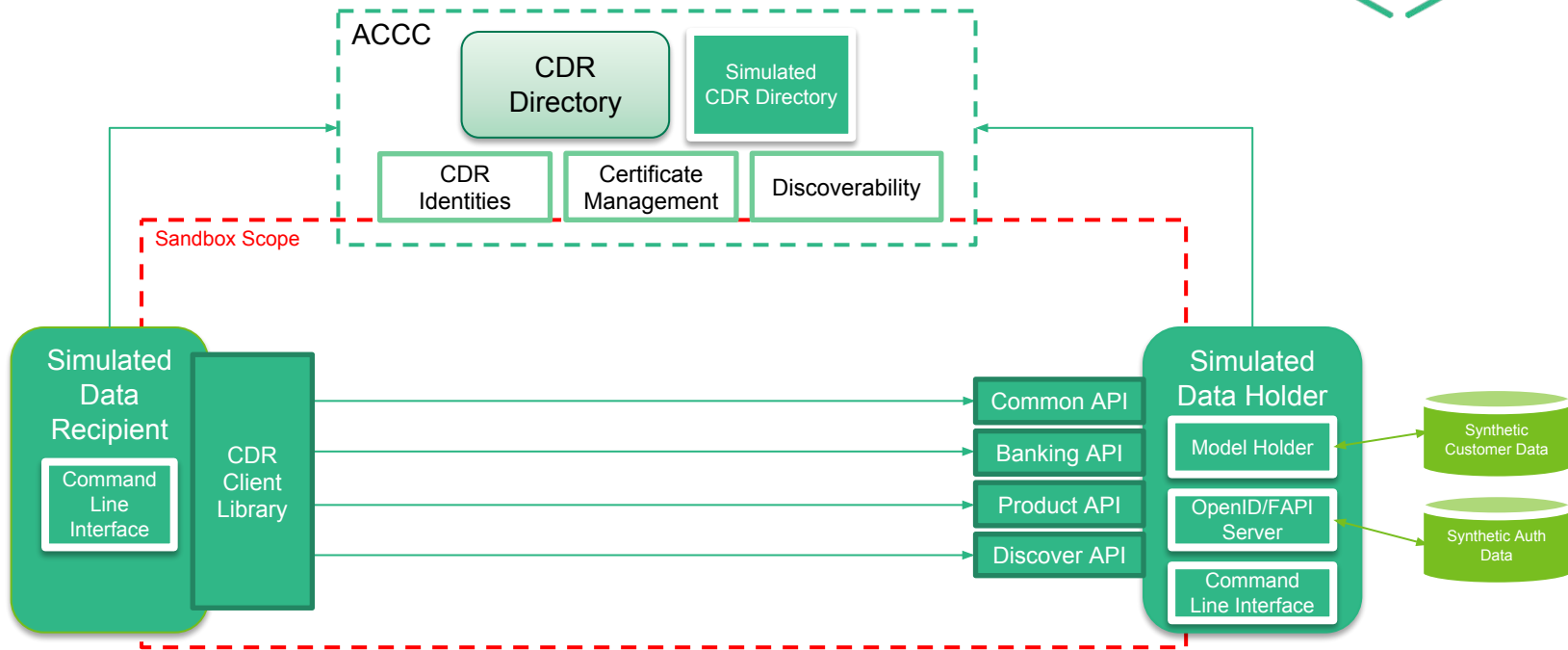
Sandbox Container Architecture

- 5 "Live" Containers ("vagrant up"):
  - Model Data Holder API
  - ACCC Directory Simulation
  - FAPI Server
  - 2 x Ephemeral Databases

- 2 x CLIs ("docker run recipient-cli"):
  - Recipient interaction
  - Holder admin

Database "Sidecar" Postgresql/MySQL

Database "Sidecar" Postgresql/MySQL

FAPI Server

To Be Determined

consumes

Model Data Holder

Java Spring Boot with Tomcat or Jetty

consumes

ACCC Directory Simulation

Auth **Standards API** **Admin API**

Recipient CLI

Holder CLI

Cloud Sandbox Options

Reiterating: *"This has Operational Support implications and has been deemed outside the scope of the Data Standards Body."*

Options:

- Deployable image of "instant" sandbox from vagrant up
- Persistent database deployment with GUI of Admin API
- Vendor specific CDR simulation (that passes Conformance!)

Finish