

# Test Data Services White Paper (Australian Edition)

**Eliminating the risks of using** anonymised **Personal Data** with a completely manufactured dataset of the Australian population to facilitate **safe** end-to-end testing of any application.

The logo for Test Data Services features a stylized graphic on the left consisting of three rows of interconnected nodes and lines, resembling a network or data flow. To the right of this graphic, the words "Test Data" are written in a large, bold, blue sans-serif font, and the word "Services" is written below it in a smaller, lighter blue sans-serif font.

**Test Data**  
Services

## Document Produced by

**Paul McLean** BAppSc(CompSc), GradDip(Mgt), CHIA  
Director  
Test Data Services Pty. Ltd.  
Suite 3, 1045 Doncaster Rd  
Doncaster East, Vic 3109, Australia  
P. O. Box 7245, Doncaster East, Vic 3109  
ABN:91635331332(<https://abr.business.gov.au/ABN/View?id=91635331332>)  
[info@testdataservices.com.au](mailto:info@testdataservices.com.au)  
<https://www.linkedin.com/in/paulmclean2/>  
© Test Data Services Pty Ltd 2019, All rights reserved.

## Document History

Version	Date	Author	Comments
1.0	14 <sup>th</sup> September 2019	Paul McLean	Initial Version
1.4	14 <sup>th</sup> November 2019	Paul McLean	Added Data Tagging examples with Python / Locust
1.5	28 <sup>th</sup> November 2019	Paul McLean	Added more Data Tagging and Dataless Testing content
1.6	20 <sup>th</sup> December 2019	Paul McLean	Added 'Population Oriented' Testing content
1.7	24 <sup>th</sup> December 2019	Paul McLean	Subscription Summary Information
1.8	7 <sup>th</sup> January 2020	Paul McLean	Multiple updates relating to email processing.
1.9	9 <sup>th</sup> January 2020	Paul McLean	Various minor edits
1.10	23 <sup>rd</sup> January 2020	Paul McLean	Added initial OpenID Connect (OIDC) Details
1.12	2 <sup>nd</sup> February 2020	Paul McLean	Updated OIDC details
1.13	23 <sup>rd</sup> February 2020	Paul McLean	Added unique random identity and more OIDC calls
1.14	6 <sup>th</sup> March 2020	Paul McLean	Added additional API calls to support OIDC processing
1.15	30 <sup>th</sup> March 2020	Paul McLean	Added additional OIDC detail and 'Functional Testing' bundles and access.



# Table of Contents

Executive Summary .....	3
Proposed Model for Generally Available Test Data .....	4
Population Oriented Testing – of Population Facing Applications .....	5
Dataless Testing.....	6
Summary of Subscription Offerings .....	7
Functional Testing .....	8
Dataset Extensions.....	9
Special Custom Additions to Test Data.....	10
Special Custom API End Points .....	10
Email Domains .....	10
Important note about common FromDomains: .....	11
Tagging and ShowKeys.....	11
Mobile Device Registration .....	12
Use Cases for Test Data.....	13
Interacting with Test Data and Test Data Services .....	18
Data Fetching REST APIs.....	19
Email and Message related REST APIs.....	20
MFA related REST APIs.....	21
OpenID Connect (OIDC), PKCE and related RESTful APIs.....	22
OpenID Connect style APIs .....	23
Supporting APIs to facilitate OIDC and PKCS based testing .....	24
Workload (Performance Engineering) related REST APIs.....	26
Data Tagging REST APIs.....	27
Subscription management REST APIs.....	28
Example API Usage: .....	29
Proposed Subscriptions to Access Test Data (subject to change).....	32
Initial Subscription Pricing (subject to change) .....	33
Real Addresses .....	36
Manufactured Source Documents and their Verification.....	37
Other Manufactured Data .....	40
Detailed Examples of API Usage, using Windows Powershell:.....	43
Some Data Tagging examples with Powershell: .....	46
Detailed Examples of API Usage, using AWS Linux: .....	47
Detailed Examples of API Usage, using MicroFocus LoadRunner: .....	50
Detailed Examples of API Usage, using Python / Locust: .....	52

## Executive Summary

Testing is essential for every system, and testing quality depends on the quality and quantity of the data in the test environment. Test Data Services provides RESTful APIs that allow simulation of millions of Identities, and even supports access to those identities through an OpenID Connect Identity Platform.

To maximize testing quality, many organizations rely on using a subset or sometimes even a full set of Production Data and to mitigate risks associated with using production data, many organizations turn to a Data Anonymization Solution, but not all Data Anonymization Solutions are created equal, but they all confer some risk of deanonymization. If anonymized data is shared beyond the protective boundaries of the organization then it is forever at risk of some future deanonymization attack. To limit this risk, some organizations limit the use of anonymised data to their own internal systems and do not make it available to third-parties to support Integration Testing activities, effectively limiting the quality of testing of system integrations.

This paper proposes an alternative to using anonymised data. While still using real world addresses, we propose using a completely manufactured identity data set, which is internally consistent, and contains relevant links to other identities, devices, email addresses, assets and businesses. When multiple organizations use the same version of manufactured data, and when the data is supported by validation endpoints to verify source identity documents, such as Medicare ID and Driver License, then Integrated Testing can be safely extended across organizational boundaries with zero risk that personally identifiable or confidential information will be exposed, because no personally identifiable data was ever used or loaded into a test environment.

Production data is not static, and test data also needs to change over time. Changes such as births, deaths, marriage, separation, name change, children leaving and returning home happen in real life, and Time-Based RESTful API calls allow the feeding of such data into a test environment. Underlying population attributes are driven by ABS Census data as are historical and ongoing changes to that data.

It is also recognised that there is a 'network effect' in respect of multiple entities using common test data. The more organizations that use the same set of test data, the easier it is to validate interfaces and data flows and to push systems to full scale workloads, activating realistic interface utilization.

To aid in the take-up of widespread usage of this dataset, a [small 25,000 person identity set](#) can be downloaded and used by anyone. Subscription based endpoint validation endpoints will work with these identities, even without a subscription key.

This paper explains the technical details relating to the manufacture, composition and access to an internally consistent dataset of the Australian population to facilitate integrated application testing of applications in all market segments.

## Proposed Model for Generally Available Test Data

The test data will principally comprise Identities, Organizations and Assets. Very simple businesses, such as full time or part time self-employed operators working from a residential address are treated as a special case of a 'Person' as the individual and the business are essentially the same. Businesses, Superannuation funds, Trusts, and other organizational entities are represented as Organizations and Assets such as Motor Vehicles and major items of value that can be used as collateral are stored as Assets.

REST APIs are made available to allow retrieval of Identities, Organizations and Assets for a point in time. Additional REST APIs are made available to support the validation of the source documents for the data, providing an internally consistent ecosystem of related entities.

All REST APIs are locked to a particular dataset with a 4 digit 'Version' number. The first 2 digits denote the major version and the last 2 digits denote the minor version number. Subscriptions relate to a Major version. Various improvements or corrections may be made to data over time. If the changes are relatively minor, then a new minor version of the dataset will be created. Users can then decide for themselves if they want to continue with the original version or use a newer version of data. Major versions will use completely different data, and will not be aligned with other major versions.

Finally, data can be 'tagged' within a subscription, so that particular records can be 'marked' and additional data stored against static records. This custom content can also be shared with others via 'ShowKeys' permitting 'read only' access that data.



## Population Oriented Testing – of Population Facing Applications

A 'Population Facing Application' is an application whose uses are drawn from a population or sub-set of the population. For example, an eCommerce application selling clothing will be used by a specific sub-population, characterised by geography, gender, age, propensity for online engagement and financial capacity, whereas a Government Application providing a key service may have a 'captive audience' that has chosen to interact online rather than waiting in a long line, but both examples can be represented by a 'population' of users.

Most population facing application testing involves significant quantities of data and most of that data comes from one of the following sources:

- Production Data (anonymised or even non-anonymised)
- Ad hoc manually crafted data
- Ad hoc data created using a data creation tool
- AI based data creation, that creates data based on deep AI analysis of production data, with focus on relationships and ratios between data entities.

There is a natural selection bias in all of the above approaches to sourcing data to support testing. Many might dispute this idea with respect to Production Data, but most production data used to drive testing only includes data is found within an production environment database, and does not represent the possibly significant quantity of other data that did not satisfy business rules and was not committed.

When testing an application, the traditional approach has been to curate a set of data to be used to drive test script that expects that data, and takes into account the nuances, simplifications and omissions embedded in that data.

Population Oriented Testing changes the focus of test scripts from being developed with curated data in mind to relying on access to an external population of identities.

When developing a test script to be driven by an external population of identities, the script should be sufficiently complex to deal with an appropriate mix of data situations. For example, they may or may not have a driver's licence, they may or may not have a credit card, they may be under age, they may or may not have a mobile phone number. Their name may be too long or too short.

By driving a properly constructed test script with 'just in time' data supplied from an external population of identities, such as the populations described in this document, the tests executed with those scripts will be more likely to generate workload that is aligned to the real population interactions that a test using curated data and simplified test scripts.

## Dataless Testing

The proposed model for provision and management of test data, as described in this document, allows for testing scripts, scenarios and frameworks to be completely devoid of any test data, and hence be “Dataless”.

The test data within Test scripts effectively form part of the script, and need to be managed carefully. A traditional approach to functional testing would be to ensure a minimum level of test coverage, using the smallest number of interactions, and the least amount of test data.

With access to millions of records, test scripts that simulate people oriented use cases can select random identities in a loop until the entry criteria for the test is satisfied.

For example, if the system under test is an accommodation booking system, then the script may randomly fetch an identity (using /GetRandomPerson) until an identity is retrieved that has an age between 21 and 75 and the person has a current drivers licence.

The UID of the identity would then be saved in a local variable in the script. The test script could then step through the registration process, entering details supplied by the /GetRandomPerson call. Just before submitting the details in the step that will generate an email verification message, a call would be made to /ClearMessage for that person’s email address. After submitting the details, the script would go into a loop, checking for an email with a /GetMessage call, once each 10 seconds. When the message is returned, the user details can be saved using the /SaveUserID call, with the appropriate sub-subscription key. (Each set of Saved UserIDs are keyed by temporary Subscription keys that can easily be generated from any subscription key.)

Another script that conducts searches and makes bookings can fetch an identity with the /GetSavedUserID call, which returns the person’s UID and UserID. Using the UID, the password used in the signup process can be retrieved to continue with the business process.

For related workflow processing, a ‘Message Driven Test’ script would use the /GetNextMessage call (using the appropriate temporary key associated with this process). For each message received, the script needs to decide what it needs to do, based on the contents of the message. If the Application Under Test uses the Email Subject to differentiate message types, then this would be quite simple. The email address can be used to retrieve the original identity and UserID through a series of REST API calls, so the script can log back in, and use the information in the message to progress the workflow.

Note that all of this can be achieved without any test data persisting in the test scripts. A series of special subscription related REST APIs can return the keys that have been created, along with Key Tags, so that even short term Subscription Keys, which are used to manage and tag data, do not need to be hard coded in any scripts.

## Summary of Subscription Offerings

The 2.5 million records of identities are free, and no subscription is required to access them. End-point validation for the first 25,000 identities are also available without need for a subscription. Subscription costs are based on the 'connected' dataset.

Load Testing Subscriptions can be summarized as follows:

- EMAIL:** This subscription allows for processing of emails for the identities in the subscribed dataset, for a pre-nominated FromDomain. The FromDomain is the origin email address domain in the header. Email subscriptions are consumed by each REST API call that is made to clear, check or access email messages and also when inbound emails are received. (The subscription key used when clearing email messages, in preparation for receiving emails is used for consumption of subscriptions when emails are received. Due to the high volume of permitted email interactions in a subscription, the size of email payloads will contribute to subscription consumption, with a unit of subscription being consumed for each 4KB of Base64 Decoded message size – inclusive of all headers.
- Note that inbound emails will be processed even if subscriptions have reached their maximum utilization in a given time period.
- MFA:** This subscription allows the saving of Time Based MFA secrets and the retrieval of current tokens, to facilitate automation of access to MFA protected resources.
- TAGGING:** This subscription allows association of data with identities in the data set to facilitate Dataless testing, offloading the need to manage various data values within test harnesses. For example, it allows the association of 'User IDs' with specific Identities and also allows for sequential or randomized access to those User IDs, eliminating the need to manage lists of User IDs from test scripts, scenarios or harnesses.
- ENDPOINT:** This subscription provides a number of End Points that can be used to validate or return related data using REST API calls that simulate official end points. Note that these APIs may or may not behave exactly like confidential or protected endpoints, but can be used in place of real End Point validators. For example, the ABN lookup is public, and the Test Data Services endpoint looks like the real end point, but the Drivers Licence Check does not proprot to look like the real interface used by identity verification or law enforcement agencies.

**WORKLOAD:** The WorkLoad subscription facilitates Performance Engineering by allowing sophisticated modelling of load and stress tests, and support the execution of those tests with a variety of tools that do not have such capability.

This is achieved by allowing the specification of as many usage or activity models as required in such a way that allows any number of test scripts to make a simple REST API call immediately prior to starting a test iteration, and either proceeding or aborting that iteration on the basis of the data returned from the REST API.

The Workload subscription also includes various API calls to record transaction response times and resource utilization.

## Functional Testing

A special 'Low Volume' subscription can be purchased, that includes all of the above subscriptions, but with much lower transaction limits. A functional testing subscription allows 1M transactions for an annual subscription, rather than 31.5M, but should be sufficient to support typical functional automation implementations, including DevOps style CI/CD. However, it is only able to be used with the Free (2.5 million) Identity dataset.

## Dataset Extensions

In addition to the above REST API related subscriptions, the following subscriptions relate to the underlying data, and special data related REST APIs.

- Standard:** This is the foundation DataSet for the dynamic 25 million Identity Dataset, allowing Point In Time access and some source document references, as well as organisations and employment relationships.
- Finance:** This dataset is intended for Finance and Insurance application usage, and includes motor vehicles owned (or financed) by Identities. It includes a mock PPSR service that mimics basic functionality of the real PPSR service. It is only available as an 'add-on' to the 25 million identity dataset.
- Health:** This subscription is intended for developers and consumers of Health Care services, providing multiple (FHIR/HL7) health records for most of the identities in the dataset. It is only available as an 'add-on' to the 25 million identity dataset.
- Gov:** This subscription includes data and services that would only be used by Australian Government departments and agencies, such as CRN numbers. It is only available as an 'add-on' to the 25 million identity dataset.
- International:** This subscription allows access to API calls to retrieve random identities from a very large pool of Simplified and Minimalist International Identities. A call `GetRandomPersonUSA` or `GetRandomPersonEU` would retrieve a simple Identity, suitable for most eCommerce applications from the USA or the European Union respectively. A call to `GetRandomPersonINT` would return a random identity from any country that is included in the dataset.

## Special Custom Additions to Test Data

It is expected that some organizations will want to use the manufactured dataset, except that they will also want some records added to the data set to align to their own real-world records. For example, an employer with their own super fund may want their corporate details added to the dataset. This can be arranged, so long as their rights to that information are rescinded. Data will not be changed in response to a custom addition, but new records can be added. In this example, one or more super funds may be added, along with the required number of business records and matching reference data. This means that the validation APIs will then return expected results for specific real-world data, simplifying the integration of test data into the client's data domain. Such services will be quoted based on the complexity and volume of the data to be added, but once added, the data will not be removed.

## Special Custom API End Points

It is expected that some organizations will have special API End Point validation requirements that are not included in the starting set of APIs. Such API endpoints can be developed through a quotation based engagement process, but all of the intellectual property developed in the course of any development is retained by Test Data Services, and Test Data Services may decide to include such APIs in the publicly available set of End Point validation APIs.

## Email Domains

API managed Email interactions will only be possible for subscribers, who will need to subscribe to 'origin' email domains. A subscription will only be permitted to send emails to nominated addresses within their nominated email domains, and emails will only be processed if their origin is a nominated email domain. Each email domain nomination requires a separate subscription.

This email functionality facilitates account signup and activation testing as well as password reset testing, which are tests that have traditionally been very difficult to automate without this type of service.

Email messages will be ignored if they are more than 150KB in size, as the purpose of this email facility is rapid processing of notification emails, rather than transmission of large attachments.

Obtaining response times is a frequent requirement when testing an email interaction, and that is supported by the inclusion of a 'ClearMessage' call, which clears any previous message for that email address from that 'FromDomain'. The time in milliseconds since 1970 is then saved in a database for that combination of email address and FromDomain. When an inbound email for the email address has been received from the FromDomain, the current time is compared with the previous 'clear'

time, and the latency in seconds is calculated, and returned as a JSON field, along with other relevant data.

This means that a test script can make a ClearMessage call immediately prior to initiating an email based action, and then 10 seconds later, check to see if a message has been received. If the message was received by this time, the actual latency would be returned, along with the message body.

## Important note about common FromDomains:

Many systems are developed without completely configuring production like domains for email, which means that multiple users of this service may use a common FromDomain. For example, email generated by AWS services may have a FromDomain of eu-west-1.amazonses.com. If your system is using a default FromDomain, then it is highly recommended that you issue a ClearMessage immediately after successfully reading the contents of the message, to dramatically limit the possibility that someone else will see your message or its contents. Note that even with the free dataset, there are 2.5 million email addresses in use, so the chance that two projects will have a collision with the same email address on the same day is very remote, but reducing that time window down to several seconds makes the possibility of a collision even less likely.

If you are running tests in a cloud environment, and you don't know the domain that is actually generating emails, then you need to generate a transaction from your test environment for an email address that you control, and then visually check that email for the source domain.

## Tagging and ShowKeys

A number of subscription scoped data values can be maintained by subscribers using 'Tags' and that data can be shared with other users, in a 'read-only' manner, via 'ShowKeys'. There are five types of tags, as detailed below:

- **Index Tagging**, which enables the adding of short text strings, that will be used as a unique index, to retrieve a particular identity.
- **Application Tagging**, which enables custom JSON data to be attached to an identity.
- **Data Tagging**, which assigns some application relevant attribute to the identity, such as 'too young to signup', 'no driver licence'....
- **JIT Tagging**, which allows the storage and retrieval, in real time, of key identity data and references. For example, a call to 'SaveUserID' saves a UserID against a unique Identity and calls to 'GetSavedUserID' and 'PopSavedUserID' can get a random Identity or the next Identity from the set of saved identities. Sets of JIT Tagging are supported by using temporary 'sub-

subscription keys' that can be generated from a primary subscription key, allowing unlimited numbers of managed sets of Just In Time data management.

- **FHIR Tagging**, which allows the association of a complete (JSON formatted) FHIR record to be attached to an identity.

The Tagging is very useful for customising the generic identity dataset to a specific application under test, but can be extended to support integrations and collaborative testing with other organizations by the use of ShowKeys.

ShowKeys are effectively a subscription key that is passed as a query parameter in a REST API call to access the custom data that is maintained by a different subscription. The owner of that subscription can request a ShowKey, and users of that Key will see the data of the origin subscription, but the calls to access the data will not impact on API Consumption or Rate Limiting of the origin Subscription, but will draw on the user subscription.

## Mobile Device Registration

Device Identifiers will be associated with each identity that would possibly have a mobile device. This device identifier can be used to register for PUSH messaging to an API service, similar to the Email Address Verification service will eventually be developed to allow retrieval of those push messages via a REST API call.

## Use Cases for Test Data

It's all about testing, and improving the quality of testing while reducing risks arising from the data used while testing. The following use cases highlight way to benefit from using a manufactured data set to support virtually any test.



### *Simple Testing*

Data is required for meaningful testing of even the most simple systems. Having a virtually unlimited supply of test data simplifies testing activities, even if a tiny fraction of available data is used. For example, if a test requires a properly formatted Medicare Number or a Tax File Number, then simply fetching random records using the Free REST API will deliver millions of such values.

### *Continuous Monitoring of Processes Involving Unique Data*

While application performance monitoring has matured and covers many high frequency use cases using a small subset of User IDs or Data Values. Synthetic monitoring does not generally include unique data, such as the registration of a new customer with a previously unseen email address. By combining the random, just in time retrieval of an identity with email validation, it is possible to continuously monitor signup processes in a production environment.

## *High Volume Load Testing of Signup Process for eCommerce Web Site*

The performance and reliability of the Signup process of eCommerce web sites is critical for the operator of the site. By using Test Data Services APIs, the process can easily be tested to levels in excess of a million signups per hour, and involves many of the different offerings in the suite of available REST APIs.

The process first involves a call to /GetRandomPerson so that a random identity can be used in the signup process. The required name, address, email, age, gender and passwords for this identity can easily be selected from the available fields returned by /GetRandomPerson.

During the signup process, a UserID, other than the users email, is often required. This can be selected from any of the fields returned by the /GetRandomPerson call, but an easy option is to combine the First Name with the Serial Number of the identity, such as Lisa1374766.

The email verification process can easily be handled by a call to /ClearMessage, immediately prior to submitting the email details. A few seconds later, the message can be retrieved via a /GetMessage call, which returns the latency, in seconds, from the time the message was cleared until the email was received. Note that the parameters to these message calls include both the target email address as well as the 'FromDomain', so that only messages into the target address from the AUT domain will be available.

The payload of the received message can then be parsed for the activation link or embedded secret, which can then be submitted, in order to complete the signup process.

The UserID for this signed up user can then be saved with a /SaveUser call.

All User IDs saved in this manner go into a 'pool', based on a temporary key. Making a call to /GetSavedUserID will randomly return one of the saved users – for that temporary key. Temporary keys can be generated with a simple REST API, using a Data Tagging subscription key, as often as required, allowing for multiple pools of user ids to be used.

See the Python examples near the end of this document for example code.

## *End to End testing of multiple integrated systems*

Applications are increasingly connected to other applications. End to end testing of multiple applications can be facilitated by using a set of test data that is common to all applications. Integration with various REST APIs to validate addresses, identity and various other information extend data dependencies beyond the traditionally considered scope of a testing engagement. By leveraging a consistent Data Set, along with supporting REST APIs, it is possible to configure multiple application environments in different organizations to participate in combined end to end tests.

### *Just In Time Fetching of Identities for CI-CD testing*

When applications need to validate processes on a continuous basis, such as a Continuous Integration Continuous Deployment pipeline, the ability to fetch a new record or each test is very useful. For example, testing a signup process may require a unique email address and may involve an email validation. Randomly selecting an identity from a set of 25 million for each iteration will provide a 'mostly unique' identity each time, and using the Email Address Verification API will allow validation of the 'one off' asynchronous processes, such as user signup processes.

### *Random variation of field contents / business processes*

By running randomly fetched identities through simple business processes, the mix of unexpected combinations of field contents may generate unexpected errors in the application. For example, a NYC (Know Your Customer) business process may require 100 points of Identification, and someone may supply source document references that would appear, on the surface to be reasonable, but may not actually be acceptable, such as a Tasmania Birth Certificate from 1969. (Birth certificates from TAS can only be electronically verified from 1970.)

### *Penetration Testing*

When penetration testing is undertaken, it is important that the data being protected is not real.

If the penetration test is application oriented, rather than infrastructure oriented, then significant cost savings can be achieved by deploying the entire application stack to the cloud, even when corporate policy for production data does not permit a complete cloud-based deployment. The stack can be tested from the Internet without giving the Penetration Testers access to the Corporate Network and once testing is complete, the environment can be destroyed, saving costs while isolating any test related impacts to the dedicated environment that no other users depend on and without putting any personally identifiable data at risk.

### *Message Driven Testing*

The term 'Data Driven Testing' has been very popular over the past decade, but few tools have been available to support Message Driven Testing. By using the ClearMessage REST API call, provided as part of the email subscription service, it is possible to monitor email addresses associated any identity in the dataset – from a particular 'FromDomain' and access the payload of those messages when they arrive or up to one week after they have arrived. If several messages are received for the nominated email address, they can be viewed in sequence with the PopEmail call. The number of messages received since the last ClearMessage call can be returned with the CountMessages call. This allows the triggering of test scripts and business processes based on the receipt of messages.

Health Informatics is the appropriate and innovative application of the concepts and technologies of the information age to improve health care and health. ( <http://www.e-healthstandards.org.au/ABOUTIT014/WhatIsHealthInformatics.aspx> )



When multiple integrated applications are configured with an internally consistent set of identities, health services identifiers and appropriate validation end-points, it is possible to run substantial end to end tests of systems comprising multiple applications, increasing the quality of service delivery while not compromising the security of personally identifiable data.

### *To Support and Validate Anonymisation Testing*

Some technical solutions need to make various aspects of some data available to third parties. For example, Health Related information may be made available to researchers to support valuable health research. Making such data available usually involves aggregation and anonymization of personally identifiable information, to ensure that information about individuals in a dataset can be derived from the research data set.



Using a manufactured data set, such as is described in this document, allows for transparent verification of the efficacy of anonymization without putting real data at risk, and also provides opportunity for testing and validation of processing without compromising the security of the underlying data.

## Interacting with Test Data and Test Data Services

In keeping with a philosophy that multiple independent teams can collaborate on integrated solution testing, it is imperative that the underlying identity information in a dataset does not change over time. For this reason, new API Endpoints will be published when substantial changes are made to any of the underlying data in a previously published dataset. This will be achieved by releasing a new major version of the data. (Some changes, such as a new expiry date for a Medicare Number or Credit Card may or may not be implemented in a 'static' dataset over time.)

The API endpoints used to access data in the environment are formatted as follows:

[https://api.testdataservices.com.au/v0000S\\_Nnnnnnnnnn](https://api.testdataservices.com.au/v0000S_Nnnnnnnnnn)

[https://api.testdata.email/v0000S\\_Nnnnnnnnnn](https://api.testdata.email/v0000S_Nnnnnnnnnn)

where 0000 is a version number, S is (S or F – indicating Subscription Dataset or Free Dataset) and Nnnnnnnnnn is the Name of the API. Note that some paid subscription based services apply to the 'Free' dataset, but access to any of the data in the 25 million record dataset is only by subscription.

REST APIs, along with their brief descriptions are included in the tables below:

## Data Fetching REST APIs

API Name	Comment
<a href="#">GetRandomPerson</a>	<a href="#">Gets a Random Identity from the full set of identities for the 'Current Point in Time'</a>
<a href="#">GetRandomUniquePerson</a>	Gets a Unique Random Identity from the full set of identities for the 'Current Point in Time', i.e. an identity that has not been retrieved previously for that subscription key.
<a href="#">GetRandomPersonFull</a>	<a href="#">Gets a Random Identity from the full set of identities including recent history of name changes, address changes and employment changes</a>
<a href="#">GetRandomHousehold</a>	<a href="#">Gets a Random household from the set of identities, which will include one or more identities.</a>
<a href="#">GetRandomOrg</a>	<a href="#">Gets a Random Organization from the full set of Organizations</a>
<a href="#">GetRandomSBO</a>	<a href="#">Gets a Random Self Employed Business Owner (businesses that are operated or administered from home)</a>
<a href="#">GetRandomSEHP</a>	<a href="#">Gets a Random Self Employed Health Professional (practice operated or administered from home)</a>
<a href="#">GetRandomBusiness</a>	<a href="#">Gets a Details for a Business, that operates from a commercial address.</a>
<a href="#">GetPersonBioMetrics</a>	<a href="#">Gets a set of (manufactured) biometric (facial recognition) data for the person.</a>
<a href="#">GetPersonCryptoKeys</a>	<a href="#">Gets a set of public and private keys for a particular Person based on the UID of the person.</a>
<a href="#">GetOrgCryptoKeys</a>	<a href="#">Gets a set of public and private keys for a particular organization based on the ABN of the organization.</a>
<a href="#">GetHealthRecord</a>	<a href="#">Gets a full FHIR/HL7 dump for a specific person</a>
<a href="#">GetPersonsAssets</a>	<a href="#">Gets all assets associated with a person (e.g. motor vehicles)</a>
<a href="#">GetRandomAsset</a>	<a href="#">Gets a random asset (e.g. a random motor vehicle in a state)</a>
<a href="#">CheckSuper</a>	<a href="#">Validates a Super Fund by ABN and returns details for that fund.</a>
<a href="#">CheckABN</a>	<a href="#">Validates an ABN and returns ABN Registration details for that business.</a>
<a href="#">CheckMedicare</a>	<a href="#">Validates a Medicare Reference, returning details for that identity</a>
<a href="#">CheckDriverLicense</a>	<a href="#">Validates a Driver Licence</a>
<a href="#">CheckMedicareProviderID</a>	<a href="#">Validates a Medicare Provider ID (find a random provider using GetRandomSEHP)</a>
<a href="#">CheckAHPRA</a>	<a href="#">Validates an AHPRA registration (find a random practitioner using GetRandomSEHP)</a>
<a href="#">CheckPPSR</a>	<a href="#">Simulates a PPSR check for a vehicle</a>

Note: Green text in the above table indicates free access, even without a subscription, for 25,000 of the identities, which are downloadable [here](#) as a CSV file, from the Free Dataset.

## Email and Message related REST APIs.

Note that REST APIs rely on database keys in order to access email records, so all email addresses, for these APIs, are converted to lower case prior to being stored.

API Name	Comment
ClearEmail (was originally called ClearMessage and is deprecated)	Used to clear the latest email and start a passive timer. Note that inbound emails will not be stored unless they are preceded by a call to ClearEmail, so this should be called immediately prior to triggering a process that will eventually generate an email message.
GetEmail	Used to fetch the latest message sent to a particular email address for the subscription, which is based on the originating system domain name. If the email address is related to a TestDataServices identity, then the UID of the matching identity is also returned, so that additional REST API calls can be made to retrieve other information relating to the user, such as user name, password information and any tags that have been attached.
CountEmails	Returns the number of Email messages (but not any contents) since the time the messages for this EmailAddress / FromDomain was cleared, along with the time, in seconds, since it was cleared.
PopEmail	Pops the next message for this EmailAddress / FromDomain, if one or more messages were received since messages were cleared. When all messages have been returned, an error response will be returned indicated that no more messages are available. If the email address is related to an identity, the UID of the identity is also returned, so that additional REST API calls can be made to retrieve other information relating to the user, such as user name and password information.
SubscribeEmailMessages	Making this call will cause all email messages for this email address to be saved against the SubscriptionKey used in this call. It is recommended that a new temporary subscription key be used for each cycle of testing, but only one subscription is active at any point in time for a combination of email address and from domain.
UnsubscribeEmailMessages	Making this call will cause stop further email messages for this email address to be saved against the SubscriptionKey used in this call.
PopSubscribedEmail	Fetches the next email message from the set of messages saved with the SubscriptionKey used in this call. If the destination email address is related to an identity, the UID of the identity is also returned, so that additional REST API calls can be made to retrieve other information relating to the user, such as user name and password information, making it quite easy for a test script to simulate real user interactions that depend on information related to that particular Identity. Note that this call will return the next message in the received sequence, if it has already been returned by a call to PopRandomSubscribedEmail, but if the message has already been returned, it's 'MessageReceived' value will be true rather than false.
PopRandomSubscribedEmail	Fetches a random email message from the subscribed set of messages, based on the SubscriptionKey used in this call. The 'MessageReceived' flag is returned, so that it is clear if the message has already been returned by a 'popSubscribedEmail' call.
ResetSubscribedEmail	Resets the pointer to the next message to be popped for this Subscription key back to the first message, allowing all messages to be re-retrieved in the original sequence.
CountSubscribedEmails	Returns the number of emails saved since the subscription commenced, along with the 'Next Message' pointer and the current status of the email subscription.
ClearSubscribedEmails	Deletes all existing email messages for this subscriptionKey.
SimulateInboundEmail	Simulates the reception of an inbound email for a particular email address from a particular FromDomain. Note that it will not be saved unless a matching call to ClearEmail has already made. This enables testing of test scripts, in anticipation of inbound emails. The Message Text passed into this REST API needs to be sufficiently similar to a real email to trigger required processing, but does not need to contain matching email headers and meta data, as the email address and senders (From) domain will be derived from REST API inputs rather than parsed from the simulated email payload.
SendEmail	This is a protected call, needed special validation before use, but allows sending of emails from any test Identity to an approved email address, allowing for 2-way email interactions.

## MFA related REST APIs.

API Name	Comment
SaveGoogleSecret	Saves a (base32) Google Authenticator Secret Key for an email address for a subscription. The secret is saved as UpperCase, even though many secrets are presented in lower case, as the base32 dictionary does not contain lower case characters. Note that the email address used in this call can be for any email domain but is case sensitive.
GetGoogleAuthCode	Gets the current (6 digit) Google Authenticator code for a particular email address, for a subscription, if a key has already been saved for that email address. Note this also works for Microsoft Authenticator, as Microsoft Authenticator uses the same algorithm as Google. Note that the email address used in this call can be for any email domain but is case sensitive.

## OpenID Connect (OIDC), PKCE and related RESTful APIs.

OpenID Connect is the de facto standard for handling authentication for large populations of users. From traditional web applications to single-page apps to native applications, OpenID Connect provides a template for interoperability that makes it easy to incorporate identity management seamlessly and securely.

These RESTful APIs support an “OpenID Connect” style interface to a population of users. OpenID Connect is already used by most people with they “Login With Google” (or a social media provider) to a web site. The RESTful APIs described in this section allow access to Identities via an OpenID Connect Provider which behaves like Google’s service that is permitted to share limited personal information. By configuring a test environment to the OpenID Connect settings of Test Data Services rather than a real Production Identity Provider, it is possible to test all user interactions that involve OpenID Connect, including the user\_info endpoint requests. However, it is obviously not possible to use the access tokens generated by this OIDC server against a production service, as the tokens will not be recognized by those services.

A real (production) End User interaction with an OpenID Connect Challenge will result in the launching of a new browser window, where the user can be authenticated, but that step is not required for this ‘test’ implementation of OpenID Connect. Prior to an Interaction that requires OIDC authentication, an Identity is ‘pushed’, so that the next OpenID Connect call ‘authenticates’ a previously ‘pushed’ identity. If your testing only requires a signed JWT, then calling `OIDC_authorize_random` returns a JWT for a random identity. Making an ‘Authorization Code’ flow call to `OIDC_authorize` will result in a redirect to the supplied redirect URI (if it matches with the permitted URI against that ClientID). The resulting authorization code can then be transmitted via a backchannel to the Token endpoint, along with the ClientID Secret. If RESTful API calls include relevant Query Parameters, then the authentication process will conform to the PKCE standard, allowing simplified testing of well secured systems.

Application specific claims can be included in generated JWTs by calling `OIDCScopeJWTData` with a key/value pair for a scope for a ClientID for an individual, based on their UID. Only one claim can be added per scope using this method. However, the userinfo endpoint can expose large JSON payloads based on data saved by calling `OIDCScopeUIData`, where the JSON structure passed in a POST can be associated with an Identity for a ClientID. Note that this is very useful for functional testing, where a simple series of REST APIs can be used to setup specific JWT or UserInfo data for an Identity.

The public keys for JWT signature verification for KIDs are shown below:

**kid:** **nIQIP4XNefgvPhzJURuDTefkJVRL9Bqt** is as follows. (Note this key is based on 2048 bit private key, which can be optionally used to sign JWTs as an alternative to the other 4096 bit private key)

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyi31HAKpINh7wNJ10hrq
9sMQdi0ZRzOiRrWanM9IZNoOqNdcptvKV0567jU11TgGDM2yv38iw0EvJQ/HAQB5
j7lGgJFXoAqWMy0pVgeza3h+Cfpj3p3itIH6Z13qu45GEy5sVqzBHRzS4UA2fIPY
dgSU8FKZu5/UBXi1gaRrZQQGoJX21+qPnQpCbA3hTU8L63wKp/48fs8mRdhb6Ok
aBskzE6MRB5smuHJ4JfA9jNpPLw2QPso7A+QO4ZHW351DFUB910Wta43PRkUSzK/
rhwva9JOhD3og54FyDouGOFaGaf4G97y0+azIidYt47w0r8rdt/9MKX78JN9s9KH
xQIDAQAB
-----END PUBLIC KEY-----
```

**kid:** **nOIDC2wieHTHDWkFd2XRxDWRxZeTWBjI** is as follows:

```
-----BEGIN RSA PUBLIC KEY-----
MIICCAgKCAgEAgEfg9NnFi5gOIDC2wie+gHEpRZvmf5JXd7C7CuMGs+f4nJzUwgnL
+DQ265fZgBiTchGm2WLx3yDNs8sMsft160DAsrCRzRGttr8xNTMCRn/LxqhvYEVJ
RPsILk78pUrgoudeUQ3d38mMQDi7GJ5Mo1+HLiobotWA1WVEdTrrpd1k0D5eovK7
E9rGH3MoWahlIID8m8v3VdQvjCBMIn4V0p7nTk+zKyf8MaV5c6nIzepF5zza0ZFG
aJl8s/b4QVjH44XLKN3IxmT4Kh9cgP2FTAoqadlDzb6p/Aiwq9PF87PdjuPDOMXa
s2px6A4L9bZv0Kp8DshMTyvyd8C0hajShIPEDQuhOx4sLox62FQ7bs9GPM3iM6RD
zUbKw2ElNMCY6FnPyNfy6GcA3/x3XrR1YyfI5VuP7Z+ERBxZ384B56/84Un0TNN
46KESByRjpcGd28e61Dwu67I1tJoKGruthEXs3gcwb+ixitssi+SO0oIqAtzN7+L
kNs7rbGQDAxtf0YsjgNe3upn6OAx3LhBbYcaLibTp7xPYUJ+N575W979EBNH0kOb
Ufommc1CsBI2r/JG6/jIpMTi3zpJu9zNOzoolnCahPnvafwNMRqB/e4I1MW1eVxR
7cQDRlml+RwxGCKwxDeTSs0x2/hS8fGqJOj4Q4HCcFte7hrebygkmOkCAWEAAQ==
-----END RSA PUBLIC KEY-----
```

## OpenID Connect style APIs

API Name	Comment
OIDC_authorize	<p>OIDC Authorization Endpoint responsible for ‘authenticating’ a resource owner and issuing authorization codes that can be exchanged for access tokens on the ‘token’ end-point. However, it will also issue ID Tokens as part of the Implicit Flow. The API expects a clientid and an optional nonce. The required clientid is provided by OIDCRegisterApp which simulates a simplified application registration process. If a nonce is provided (by a call to OIDCUserLogin, which must be called prior to calling OIDC_authorize) then the authentication will be returned for the identity supplied to the OIDCUserLogin call. If no nonce is provided, then an Identity will be ‘popped’ from a list of Identities to be authenticated. (see OIDC_PushIdentity for more information). If no Identities are available to be ‘popped’ then a random Identity is authorized.</p> <p>OIDC_authorize will immediately ‘authorize’ the user related to the nonce and the provided ID Token will contain claims related to the scope specified in the OIDCUserLogin call, or a default scope if it is not specified.</p> <p>If no redirect URI is specified, then the IMPLICIT FLOW is processed in response to this request, resulting in an access token and an ID Token without any back channel communication requirement.</p> <p>If a redirect URI is specified, then a AUTHORIZATION CODE FLOW is processed in response to this request, resulting in a redirect to the specified redirect URI containing an authorization token as a query parameter. It is then expected that a separate request be made to the token endpoint (OIDC_token) with the authorization code and the Client Secret (as returned by the OIDCRegisterApp call for that client), after which the requested tokens can be returned.</p> <p>Resulting JWTs will contain minimal claims, including email, email_verified, given_name, family_name and tdsuid as well as the commonly returned (iss, azp, aud, sub, locale, iat, exp and jti) The tdsuid contains the UID uniquely referencing the Test Data Services Identity that has been authenticated, so that all other attributes of that Identity can be retrieved). <a href="#">However, if a call to OIDC_AddScope has been made for the identity (for the matching ClientID), then the claims included in the AddScope call will also be included if the matching scope is in the OIDC authorize request.</a></p>
OIDC_token	This RESTful API returns tokens and refresh tokens according to the OpenID Access protocol.
OIDCRegisterApp	<p>Generates a ClientID, in the form of an Application Registration Code. Google refers to this as an “OAuth 2.0 Client ID” in their offering. A Client Secret is also generated and returned. This information is saved with the SubscriptionKey used in this call, and the subscription is consumed by each interaction with the returned ClientID. This means that multiple instances and stacks can each have their own ClientID. If no Redirect URI is specified, then only “Implicit Flow” authentications can be performed, unless OIDCSaveAuthRedirect is used to specify the permissible redirect.</p> <p>The secret is used in three leg OIDC id_token authentications and can be regenerated by calling OIDCRegenerateSecret.</p>
OIDCRegenSecret	Re-generates a secret for the supplied ClientID. (it is not possible to retrieve a secret, so if the secret is lost or compromised, it can be regenerated.)
OIDCSaveAuthRedirect	<a href="#">Specifies (or overwrites) the permissible ‘Authorized Redirect URI’ for a particular ClientID. It must be in URL encoded format as it is passed as a query parameter.</a>
OIDC_userinfo	<a href="#">This endpoint returns user information based on the scopes used in generating the access token or id token.</a>
OIDC_PublicKeys	<a href="#">This endpoint returns a list of public keys to be used to verify JWT signatures.</a>
openid-configuration	<a href="#">Returns the OpenID Connect Discovery metadata, including relevant end points.</a>

API Name	Comment
OIDC_authorize_random	Same as OIDC_authorize except that a signed JWT is generated for a random identity and no OIDCUserLogin step is required. The only required parameter is the 'client' and the simulated scope is a basic 'profile' request. The call simulates the result of a randomly authenticated Implicit Flow – but without the redirect step, and is useful for API based testing rather than UI or Client Testing where a verifiable token needs to be generated.
OIDC_user_info	This endpoint returns user information based on the scopes used in generating the access token or id token.
OIDC_MockCallback	This endpoint can be used as the redirect URI for any of the OIDC flows for debug purposes for to facilitate API based or Security testing. It expects query parameters for an authorization redirect and will echo those values in the body of its response, as a means to simplify testing.
OIDC_PushIdentity	This RESTful API 'pushes' an identity into structure to be consumed by a subsequent call to OIDC_authorize. <a href="#">It can include a PKCE flag so that the authentication process can support the PKCE flow.</a>
OIDC_PeekIdentity	This call returns the next Identity that would be 'popped' by the next OIDC_authorize call as well as the number of identities that are queued to be 'popped'.
OIDC_PopIdentity	This call pops the next identity, removing it from the list to be used by OIDC_authorize. Making multiple calls to this RESTful API is the only means of emptying the list.
OIDC_RecycleIdentity	This call recycles all of the Identities that had been popped, so that they can be used again. If your situation requires the repeated use of the same subset of identities, then a call to this can be made whenever OIDC_PeekIdentity indicates that no more identities are available to be popped.
OIDC_AddSimpleScope	This call adds a single key value pair to a scope for a particular identity for a ClientID. The single claim will be included in a JWT for this identity for this ClientID if the matching Scope is specified in the initiating request.
OIDC_AddScope	This call uses a POST method to associate a JSON structure with a scope for a particular identity for a ClientID. The supplied claims will be included in any JWT for this identity for this ClientID, if the matching Scope is specified in the initiating request.
OIDC_AddUserInfo	This call uses a POST method to associate a JSON structure with a scope for a particular identity for a ClientID for use by the user_info endpoint, if the initiating request includes the matching scope.
<a href="#">OIDC_SetLoginAge</a>	<a href="#">Subtracts this time (in seconds) from the current time, and includes it with the 'auth_time' claim, to simplify testing of sessions with logins as if they occurred some time earlier.</a>
OIDC_AddAMR	The call uses a POST method to indicate authentication methods to be recorded in the 'amr' (Authentication Methods References) claim. (e.g. [ "pwd", "mfa", "otp", "kba", "sms", "swk", "hww", "fpt", "geo" ] to simplify testing of different authentication methods.

## Claims Supported and Mapped to Underlying Identity Data

Claim	Identity Field
name	Concatenation of GivenName and FamilyName
family_name	FamilyName
given_name	GivenName
middle_name	MiddleName
nickname	GivenName
gender	Gender
birthdate	End-user's birthday (DoBDay, DobMonth and DobYear) is represented as an ISO 8601:2004 YYYY-MM-DD format. The year will be if DoBYear is missing, and will only contain DoBYear if DoBMonth and DoBDay are missing.
Address	Address (this is a free format address representation)
phone_number	For Australian phone numbers: 10 digit unformatted For International

(This table is not yet complete...)

## Workload (Performance Engineering) related REST APIs

API Name	Comment
SaveWorkload	Saves a ONE SECOND slice of the workload specification for a subscription (or sub-subscription), expressed as a percentage of peak workload. Note that each workload profile can be created but never destroyed and is saved against the subscription key that is passed through in the API, so it is expected that every workload specification be saved against a newly created subscription key.
GetWorkload	Gets the current workload information for the current second of the test. Three values are returned in JSON format: RunWorkload (0 or 1, a statistically generated indicator signalling if load should be generated at this moment in time), Workload (a floating point number representing workload percentage at that time) and Comment, giving additional narrative.
StopWorkload	Ensures that all GetWorkload requests return ZERO.
StartWorkload	Starts the workload clock.
PauseWorkload	Pauses the workload clock, so that the current workload being returned stays the same.
ResumeWorkload	Resumes the workload clock, so that the workload progresses according to its specification.
CreateMetric	<p>Creates a new metric and returns a new Key that refers to that Metric. A metric name (Name) is passed to the call and will appear in responses subsequent calls to PutMetric or GetMetric.</p> <p>A time interval (Interval), in seconds, is also expected and represents the number of second for the most granular reporting of metrics.</p> <p>Optional Min and Max parameters will also be accepted.</p>
PutMetric	Writes a value to the current Interval for the Metric based on the Key. The value is a floating point number.
GetMetric	<p>Returns the values for this metric. By default, it returns the metrics for the Previous Time Interval.</p> <p>Optionally, the interval for an Absolute Time (specified in seconds since 1/1/1970) or a Relative Time (Seconds before current time).</p> <p>If the Metric was created with an Interval of 10 seconds, then a call without a time specification would return the following values for the previous (10 second) Interval: Interval Start Time, Interval Duration, Average, Count, Minimum and Maximum.</p> <p>If the metric was created with an optional Low or High value, then the number of measures less than "Min" or higher than "Max" will also be reported, along with the average of Min values and the average of Max values. This allows end user calculation of an average excluding outliers if required.</p>

## Data Tagging REST APIs.

API Name	Comment
GetSubscriptionUsage	Returns the provisioned (paid for) and consumed usage of the service for the current subscription, as well as details on rate limiting use in the Month, Week, Day, Hour, 5min and 10second time buckets.
AddApplicationTag	Saves additional data, in the form of a single text for a particular UID for a subscription. It is expected that this will be used to save application under test (AUT) specific information relating to that particular identity, such as a login id if login ids are generated by the AUT.
AddIndexTag	Saves additional data, in the form of a single text for a particular UID for a subscription, however, it is saved in a manner that allows access by direct lookup, so must be unique.
AddDataTag	Saves additional data, in the form of a single text for a particular UID for a subscription, however it is intended to be a description of the identity record in terms of the Application Under Test, such as 'Too young to sign up in Victoria'. The intention is that this data could be used to navigate complex text flows by giving them context that may not be obvious from the data itself.
AddFHIRTAg	Saves FHIR data, a particular UID for a subscription.
GetApplicationTag	Gets tag set by AddApplicationTag for this UID for this Subscription, unless a ShowKey is also provided, and then it will get the Application Tag that was set by the related subscription.
GetIndexTag	Gets tag set by AddIndexTag for this UID for this Subscription, unless a ShowKey is also provided, and then it will get the Application Tag that was set by the related subscription.
GetUIDByIndexTag	Gets UID set by index value set by AddIndexTag for this UID for this Subscription, unless a ShowKey is also provided, and then it will get the UID identity from the related subscription.
GetDataTag	Gets tag set by AddDataTag for this UID for this Subscription, unless a ShowKey is also provided, and then it will get the Application Tag that was set by the related subscription.
GetFHIRTAg	Gets tag set by AddFHIRTAg for this UID for this Subscription, unless a ShowKey is also provided, and then it will get the Application Tag that was set by the related subscription.
GetAdditionalData	Retrieves previously saved additional JSON data for a particular UID for a subscription.
SubscribePerson	Subscribes a person record in the dynamic (25 million record) Data Set.
SaveUserID	Saves a user ID for a subscription, for later retrieval. This is provided to support test use cases involving a signup process, so that the user id of the successful signup can be saved for later use, eliminating the need to manage that data in the test script. It permits the saving of a text field in association with an Identity UID.
GetSavedUserID	Retrieves a saved user ID for a subscription. The retrieval will return a record from this set of stored UserID that were saved via the SaveUserID call. The records are returned in a completely random fashion, which means that it is possible that the same record will be returned in two successive calls.
PopSavedUserID	Retrieves a saved user ID for a subscription key, like the GetSavedUserID service, but rather than retrieving a random record from the set of saved records, it just returns the Next record, in the sequence that the records were saved. Each record will only be retrieved once, making this service very useful for one-off processing of each record.
RecycleSavedUserIDs	Resets the pointer for messages to be 'popped' by the PopSavedUserID API, so that all of the messages can be returned another time.
SaveValue	Saves a value pair for a subscription. A key expected usage of this service is to store temporary subscription keys that are to be used in various situations or environments.
GetValue	Retrieves a value pair for a subscription. A key expected usage of this service is to retrieve temporary subscription keys that are to be used in various situations or environments.

Note: UserIDs saved with the SaveUserID call can not be deleted. There is no need to delete large numbers of 'SavedUserID' records as they are only guaranteed to be accessible for a week, and a new subscription key can be generated (as often as required) to manage a new set of SavedUserIDs. The SaveValue API can be used to save a subscription key for a particular environment and it can then be retrieved by a test script using the GetValue API call.

## Subscription management REST APIs

API Name	Comment
AddSubscriptionKey	Creates a new subscription key for use with data all services, but particularly relevant to data tagging and email management REST APIs. An unlimited number of these subscription keys can be active at any time and each is used to manage a set of data tags or email messages. Creating a new key, it is possible to include a 'Tag', which will be displayed when viewing SubscriptionKeys using the ShowSubSubscriptionKeys API. Note that care should be taken when creating Subscription Keys if it is possible that they may need to be deleted at a later time. When creating a new SubscriptionKey to give to a team to use, it is important that the provided key is not a 'parent key' as access to a key also confers the rights to delete any keys created by that key.
ViewSubscription	Lists all current subscriptions relevant to the SubscriptionKey used in this call, returning information about the total number of available transactions in each subscription, as well as the number of available transactions in each time period. (e.g. the current hour, or current day)
ShowSubSubscriptionKeys	Returns all subscription keys created with a particular SubscriptionKey. This is useful for managing subscription usage, especially if subscription access needs to be revoked. To see the list if Top Level subscriptions, you must include the SubscriptionUID as a subscription key.
RemoveSubscriptionKey	Removes a particular SubscriptionKey. Note that it a requirement that all SubSubscriptions are removed prior to removing a subscription. Note also that subscriptions can only be removed by 'higher' subscriptions. In other words, the only subscriptions that can be removed are ones that appear in the ShowSubSubscriptionKeys response. The only way to remove the TOP Level subscription key is to use the SubscriptionUID as a Subscription Key. The SubscriptionUID is effectively a 'master key'.

## Example API Usage:

To access the Free REST API to retrieve an Identity for version 0001F of the Data Set, you can simply enter the following into a Browser:

[https://api.testdataservices.com.au/v0001F\\_GetRandomPerson](https://api.testdataservices.com.au/v0001F_GetRandomPerson)

As an alternative, if you are reading this in Hard Copy format, you can scan the following QR code, which visits the same URL.

This will return JSON data, which is not particularly easy for a person to read, but is very easy for software to process.

One of the values returned from this call is a Medicare ID, which is the combination of the Medicare Number (including identifier, checksum and issue number) and the Individual Reference Number (IRN).



In a production setting, that Medicare ID would normally be verified by supplying the entire ID along with name and date of birth details. However, using this API, those details can be retrieved by just supplying the full ID (i.e. the Medicare number with the person position number as the last digit.)

You can try it by replacing the Medicare ID returned in the above example with 20042192443 in the URL below.

[https://api.testdataservices.com.au/v0001F\\_CheckMedicare?Medicare=20042192443](https://api.testdataservices.com.au/v0001F_CheckMedicare?Medicare=20042192443)

```
returns{"Medicare":"20042192443","GivenName":"Charlie","MiddleName":"Jessica","FamilyName":"Wong","Gender":"Female","DoBYear":2009,"DoBMonth":7,"DoBDay":28,"Expiry":"11/2023"}
```

Note that the Free Dataset does not include an API to retrieve a Household, and households are not properly represented in the 2.5 million identity dataset, but using the Medicare search gives a limited amount of that functionality. For example, in the above example, the last digit of the Medicare number is a 3, indicating that individual is the third position on the Medicare Care card. By changing the last digit to a 1 or a 2, you can see the details of the first or second person on that same card. For example:

[https://api.testdataservices.com.au/v0001F\\_CheckMedicare?Medicare=20042192441](https://api.testdataservices.com.au/v0001F_CheckMedicare?Medicare=20042192441)

```
returns{"Medicare":"20042192441","GivenName":"Jennifer","MiddleName":"Madonna","FamilyName":"Wong","Gender":"Female","DoBYear":1967,"DoBMonth":10,"DoBDay":30,"Expiry":"11/2023"}
```

[https://api.testdataservices.com.au/v0001F\\_CheckMedicare?Medicare=20042192442](https://api.testdataservices.com.au/v0001F_CheckMedicare?Medicare=20042192442)

returns{"Medicare":"20042192442","GivenName":"Steven","MiddleName":"Allen","FamilyName":"Wong",  
"Gender":"Male","DoBYear":1962,"DoBMonth":10,"DoBDay":15,"Expiry":"11/2023"}

Special subsets of identities are also randomly retrievable, such as self employed health professionals and small business owners. These identities are useful for test cases where business details are required, but the tester does not want to pay for the subscription to access the full and dynamic 25 million identity dataset (where separate businesses, employment relationships and some property ownership relationships also exist). E.g.

[https://api.testdataservices.com.au/v0001F\\_GetRandomSEHP](https://api.testdataservices.com.au/v0001F_GetRandomSEHP)

returns something like{"Address":"560 COGDELL STREET NORTH ALBURY NSW  
2640","CC\_exp\_year":"23","Noun":"law","Code\_40hex":"fallc51c2f063bbfd93374f87610e6b5549e5393","Code\_3digit":"848","PassportExpiry":"09MAY2022","DoBYear":1966,"UID":"80ecfb8b-e881-481b-a8fe-cd4582de42b0","Password4":"eJR7T7D27BAGAGDXKD","AHPRA\_Registration":"PSY0009250707","Serial":130854,"Code\_3alphanum":"MPN","Password1":"uVQQPCND5","Password0":"vS9GVM","Password3":"hHARUL","PassportNo":"QR0361820","Password2":"k]YYT[QCJ2N\*","Phone":"0412977625","Medicare":"46347279241","DeviceID":"c9c6775bd94261e51837a51f5296b295a34562b1","Code\_6alpha":"QEL","MedicareProviderID":"548451YK","DoBMonth":2,"Code\_6digit":"015655","Code\_9digit":"730820958","BusinessName":"MULLIGAN PSYCHOLOGY OF  
ALBURY","LicenceState":"NSW","FamilyName":"Mulligan","PassportPlaceOfBirth":"Nowra-Bomaderry","BusinessACN":"377153332","email":"Vicki.Mulligan.3908@testdata.email","DoBDay":7,"Profession":"Psychologist","MiddleName":"Patricia","Title":"Ms.","AddressID":"GANSW709804664","Code\_40SafeAlpha":"ZPUTBJNAGUUCGEHYNVRUEHRHYZYBVQXGUXCJQSNRE","MedicareExp":"05/2025","LicenceNumber":"113276845","BIP39\_25words":"such ugly hill actress dutch cement swap bike leopard danger square spot december sleep cabbage armed access crouch broom festival genius board review ready rescue","LicenceExp":"15MAY2021","SuperMember":"028165028323","Code\_7alphanum":"BT4F3XS","GivenName":"Vicki","BusinessABN":"36377153332","SuperABN":"39485678420","Gender":"Female","CC\_exp\_month":"04","Verb":"attribute","PassportIssue":"09MAY2012","TFN":"169344020","Code\_12digit":"015909020685","Color":"Sapphire","CC\_num":"4467812634137836"}

[https://api.testdataservices.com.au/v0001F\\_GetRandomSBO](https://api.testdataservices.com.au/v0001F_GetRandomSBO)

Returns a record similar to the above, but may include other types of business owners.

Returns something like {"Address":"81 MANUSU DRIVE MENDOORAN NSW  
2842","CC\_exp\_year":"23","Noun":"memory","Code\_40hex":"7cblc3845a87c77080582e04b61e8fd394efebae","Code\_3digit":"303","PassportExpiry":"13DEC2018","DoBYear":1957,"UID":"ef557bf0-8260-4c6f-a7ea-c7c3a9308a38","Password4":"o1H5HJBV40J8PZJR59","Serial":274910,"Code\_3alphanum":"S0Y","Password1":"sW70XSYDX","Password0":"a5KGLP","Password3":"q2AW87","PassportNo":"FG0177219","Password2":"k)NG1)0Z532","Phone":"0483785325","Medicare":"56050152451","DeviceID":"5fd97951d567d4dc8d4295d1018d7eec733d0a84","Code\_6alpha":"STQ","DoBMonth":9,"Code\_6digit":"009008","Code\_9digit":"163704377","BusinessName":"ROSENBERG ELECTRICALS OF  
MENDOORAN","LicenceState":"NSW","FamilyName":"Rosenberg","PassportPlaceOfBirth":"Melbourne","BusinessACN":"252301241","email":"Dana.Rosenberg.5353@testdata.email","DoBDay":21,"Profession":"Electrician","MiddleName":"William","Title":"Mr.","AddressID":"GANSW718228382","Code\_40SafeAlpha":"KSKRKFRUAUFVQGYBUGEYXKRAQCJXCYMYEBSMWFLC","MedicareExp":"09/2022","LicenceNumber":"318156791","BIP39\_25words":"warm chalk coffee sauce help possible scare gown media solution blame worry donkey crazy brisk enact pepper favorite episode cruel stomach popular ribbon state battle","SuperMember":"019090017665","LicenceExp":"01DEC2025","Code\_7alphanum":"GGA0YH2","GivenName":"Dana","BusinessABN":"75252301241","SuperABN":"72061091450","Gender":"Male","CC\_exp\_month":"06","Verb":"knock","PassportIssue":"13DEC2008","TFN":"738657994","Code\_12digit":"009979032469","Color":"Sangria","CC\_num":"4627012476021076"}

The following QR Codes can be used as a quick way of viewing the output of some APIs:

Check Medicare: 52829829332

GetPerson f1b528da-7adf-4558-a376-65d75bcda740



Check ABN 24798468854

Check Super Fund via ABN 98401566658



Check Medicare Provider 7832416T Check Super Fund ABN 98401566658



[https://api.testdataservices.com.au/v0001F\\_GetRandomPerson](https://api.testdataservices.com.au/v0001F_GetRandomPerson)

[https://api.testdataservices.com.au/v0001F\\_GetPerson?UID=f1b528da-7adf-4558-a376-65d75bcda740](https://api.testdataservices.com.au/v0001F_GetPerson?UID=f1b528da-7adf-4558-a376-65d75bcda740)

[https://api.testdataservices.com.au/v0001F\\_CheckMedicare?Medicare=52829829332](https://api.testdataservices.com.au/v0001F_CheckMedicare?Medicare=52829829332)

[https://api.testdataservices.com.au/v0001F\\_CheckABN?ABN=24798468854](https://api.testdataservices.com.au/v0001F_CheckABN?ABN=24798468854)

[https://api.testdataservices.com.au/v0001F\\_CheckSuper?ABN=98401566658](https://api.testdataservices.com.au/v0001F_CheckSuper?ABN=98401566658)

[https://api.testdataservices.com.au/v0001F\\_CheckABN?ABN=98401566658](https://api.testdataservices.com.au/v0001F_CheckABN?ABN=98401566658)

[https://api.testdataservices.com.au/v0001S\\_CheckMedicareProviderID?MedicareProviderID=7832416T](https://api.testdataservices.com.au/v0001S_CheckMedicareProviderID?MedicareProviderID=7832416T)

[https://api.testdataservices.com.au/v0001F\\_CheckDriversLicence?LicenceState=VIC&LicenceNumber=984770420](https://api.testdataservices.com.au/v0001F_CheckDriversLicence?LicenceState=VIC&LicenceNumber=984770420)

## Proposed Subscriptions to Access Test Data (subject to change)

Data Service Description	Free Data	Paid Standard	Paid Finance	Paid Healthcare	Paid Gov
Number of Identities	2.5 M	25 M	25 M	25 M	25 M
Point-In-Time Identities	Y <sup>1</sup>	Y	Y	Y	Y
Addesses (with GNAF AddressIDs)	Y	Y	Y	Y	Y
Addresses with GPS & Separate Address Fields		Y	Y	Y	Y
Historical (point in past) Identities (including changes in Address and Name)		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
Source Document IDs including Drivers License	Y	Y	Y	Y	Y
Source Document IDs including Birth Certificate		Y	Y	Y	Y
Source Document IDs including Passport	Y	Y	Y	Y	Y
Source Document IDs including Citizenship		Y	Y	Y	Y
Source Document IDs including VISA		Y	Y	Y	Y
Education Summary (with PostNominals)		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
Recent Employment History		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
Preferred Superannuation Fund	Y	Y	Y	Y	Y
Ongoing Updates to Identities (with 'updates' subscription add-on)		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
Identity Check Mock Service		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
PPSR Mock Service			Y		
Motor Vehicles			Y		
Credit History			Y <sup>f</sup>		
<b>Email Processing</b> (Special stand-alone subscription)	Y <sup>2</sup>	Y	Y	Y	Y
Businesses with ABNs	Y	Y	Y	Y	Y
Businesses with ACNs and Trust Structures		Y	Y	Y	Y
Simple (non-historic) ABN Lookup Service	Y <sup>3</sup>	Y	Y	Y	Y
Simple Superannuation Lookup Service	Y <sup>3</sup>	Y	Y	Y	Y
Comprehensive ABN Lookup Mock Service		Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>	Y <sup>f</sup>
Centrelink Reference Number (CRN) Mock Service					Y <sup>f</sup>
Simple Medicare Mock Service	Y <sup>3</sup>	Y	Y	Y	Y
My Health Record FHIR Records				Y	
Simplified My Health Record Mock Services				Y	
Identities with IHI				Y	
Credentialed Health Professionals with HPI-I				Y	
Health Industry Organizations with HPI-O				Y	
Special Pathology Organizations				Y	
Health Records for relevant proportions of the population for various chronic diseases.				Y <sup>f</sup>	

### Notes:

- 1: Only includes individual identities that live in residential homes, excluding residents of barracks, prisons, retirement homes.
- 2: email processing is a special Subscription AddOn that can be added to the Free service or any other paid service.
- 3: Supporting API EndPoints are accessible via subscription – for most records in the Free Dataset. However, API do endpoints work for a [subset of 25K records](#), even without a subscription, to facilitate testing and development of End Point interfaces.
- f: Future implementation – not expected to be available at the launch of the subscription.

## Initial Subscription Pricing (subject to change)

Prices listed below are in Australian Dollars, based on online purchase and full pre-payment. Note, that random access to data in the 2.5 million record set is free, but full access to validation end points is only possible via a subscription.

However, a small subset of [25,000 identity records](#) is available as a CSV download and validation endpoints relating to identities in this dataset will work for these records, even without a subscription key.

This provides an opportunity for testers, developers and students to use a limited subset of data with matching endpoints without incurring any costs.

### Services relating to the **static** 2.5 million identity 'Free' DataSet

Subscription	Daily	Weekly	Monthly	Yearly
Random Access (rate limited to 6 requests/second/IP)	Free	Free	Free	Free
Validation Endpoints	\$40	\$130	\$240	\$2,400
Email Automation (@testdata.email for each Domain <sup>2</sup> )	\$40	\$130	\$240	\$2,400
One Time Code – MFA automation	\$40	\$130	\$240	\$2,400
Data Tagging	\$40	\$130	\$240	\$2,400
Performance Engineering	\$40	\$130	\$240	\$2,400

### Services relating to each offering<sup>1</sup> of the **dynamic** 25 million identity DataSet.

Subscription	Daily	Weekly	Monthly	Yearly
Random Access <sup>1</sup>	\$80	\$260	\$480	\$4,800
Validation Endpoints <sup>1</sup>	\$80	\$260	\$480	\$4,800
Email Automation (@testdata.cloud for each Domain <sup>2</sup> )	\$80	\$260	\$480	\$4,800
One Time Code – MFA automation	\$80	\$260	\$480	\$4,800
Data Tagging	\$80	\$260	\$480	\$4,800
Population Update Feed	\$80	\$260	\$480	\$4,800
Performance Engineering	\$80	\$260	\$480	\$4,800

Note 1: Access to the Finance, HealthCare and Government Offerings are all separate, and each require a separate subscription for access and endpoint validation. However “Email Automation” and “One Time Code” subscriptions are purchased independent of dataset choice.

Note 2: If emails are generated from multiple domains (e.g. preprod.example.com & uat.example.com) then a separate subscription is required for each domain.

## Pricing for Multiple Subscriptions and Outright Purchases:

Item(s)	2.5 million identity	25 million identity
Outright Purchase of data	\$4,800	\$9,600
3 Annual Subscriptions	\$6,000	\$12,000
4 Annual Subscriptions	\$7,600	\$15,200
5 Annual Subscriptions	\$9,200	\$18,400
6 Annual Subscriptions	\$10,800	\$21,600

It is possible to purchase the right for perpetual private use of all data and subscriptions, hosted on your own AWS account, but the AWS Account is expected to be a simple stand alone account. It is possible that execution of these API calls may exceed standard AWS Account Limits, which could impact on other services in the AWS account.

The cost is a one-off payment, equivalent to two years of full access, and includes all files and Cloud Formation scripts to stand up the entire solution in your own standalone AWS account. It includes historic detail but does not include (future) dynamic data updates to the 25 million identity dataset. The cost of one-off payments to other subscription types, such as 'International Data', Email processing, Data Tagging, MFA etc is calculated at two times the annual subscription cost.

Note also that if more API calls need to be made that rate permitted under the rate limiting restrictions of the subscriptions, then it is possible to 'stack' subscription. For example, the MFA subscription allows burst activity of just over 50,000 calls per hour. If bursts of 100,000 per hour are required, then two subscriptions can be combined, or 'stacked'. Testing this service has been validated to ensure that more than 20 subscriptions can be stacked to permit more than 1 million transactions per hour.

## Important Notes on Rate Limiting and API Call limits for Subscriptions:

Each subscription is limited to a total number of calls equal to the number of seconds of the subscription. This means that an annual subscription allows a total of more than 31.5 million calls. Rate limiting is implemented by 'time period buckets' for each subscription, with a (UTC time) calendar based bucket for each Month, Week (starting Monday) and Day, as well as Hourly and 5 minute duration buckets.

The maximum permitted number of calls in in bucket for any subscription is 20% of the maximum permitted total for the next larger 'time bucket'.

This means that an annual subscription allows for higher peak usage than a daily subscription. For example, an annual subscription permits 50,458 transactions per hour, but a daily subscription only permits 17,280 transactions per hour.

However, subscriptions can be 'stacked', so that multiple daily subscriptions can be purchased in the event of a large test volume requirement, as detailed in the table below.

Subscriptions Relating to Static 2.5 Million Identity DataSet								Cost in AUD ex GST	Daily Cost / Peak TPS
Subscription Type	Total Transactions	Max / Month	Max / Week	Max / Day	Max / Hr	Max / 5min	Max Peak TPS		
Annual	31,536,000	6,307,200	1,261,440	252,288	50,458	10,092	168	\$2,400	\$0.04
Monthly	2,628,000		525,600	105,120	21,024	4,205	70	\$240	\$0.11
Weekly	604,800			120,960	24,192	4,838	81	\$130	\$0.23
Daily	86,400				17,280	3,456	58	\$40	\$0.69
Subscriptions Relating to Dynamic 25 Million Identity DataSet								Cost in AUD ex GST	Daily Cost / Peak TPS
Subscription Type	Total Transactions	Max / Month	Max / Week	Max / Day	Max / Hr	Max / 5min	Max Peak TPS		
Annual	31,536,000	6,307,200	1,261,440	252,288	50,458	10,092	168	\$4,800	\$0.08
Monthly	2,628,000		525,600	105,120	21,024	4,205	70	\$480	\$0.23
Weekly	604,800			120,960	24,192	4,838	81	\$260	\$0.46
Daily	86,400				17,280	3,456	58	\$80	\$1.39
Important Notes:	Month, Week and Day boundaries are all based on calendars using UTC								
	Hourly, 5 minute and 1 second limits are based on intervals starting at 00:00:00 each day								
	When multiple subscriptions are active, the subscription with the closest expiry is consumed first.								
Example.....									
If a project needs to run very high load occasionally (a few times a year), then purchasing an annual subscription provides economical access to services for a whole year									
Problem	If a high volume stress test needs to consume API calls at a rate of approximately 100,000 per hour for 2 hours, then the single annual subscription is insufficient, as it supports 50,458 per hour								
Solution	Add three 'daily' subscriptions, for the same Subscription Key, shortly before starting the Stress Test, with each subscription adding hourly peak capacity of 17,280 transactions								
	Giving a total hourly capacity of 102,298 transactions per hour, for up to five hours in a calendar day (based on UTC time zone) at an additional cost of AUD\$120 + GST.								
Alternate Solution	Add two 'weekly' subscriptions, for the same Subscription Key, with each subscription adding hourly capacity of 24,192, giving a total of 98,842 transactions / hour.								

Note that care must be taken to properly estimate the required maximum rate of APIs needed for given Load or Stress Test. For example, each call to Clear and Get an email count as a call, but if the AUT slows, and multiple calls are required to Get a message (while waiting for the email to be generated), then the API usage may exceed initial plans.

## Real Addresses

The addresses included in the 2.5 million record dataset are real addresses, and include concatenated fields along with a GNAF ID. The PostCode is for the address is not always correct (as is the case with real life data) and the address numbering is sometimes not what is expected.

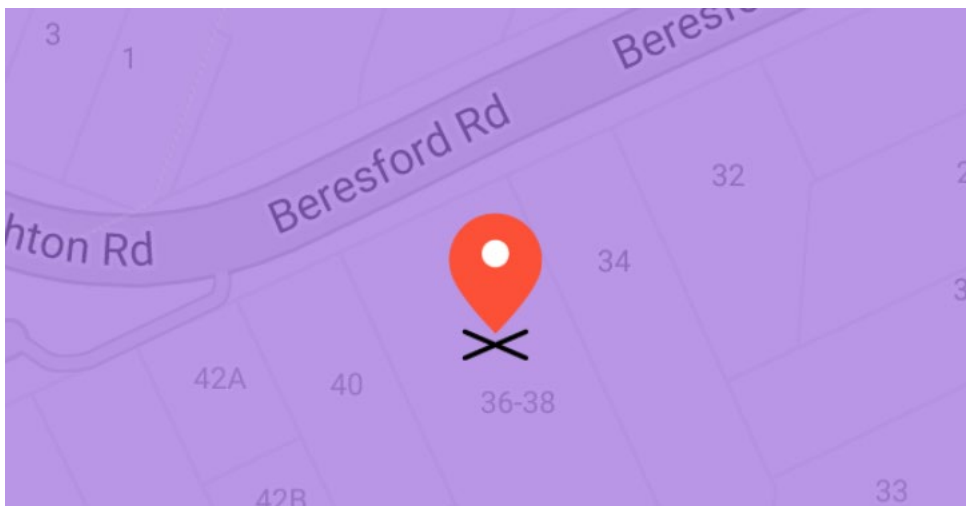
For example: The Identity with a UID of 80892432-8d0d-468e-954f-10aeb4b31608 can be retrieved with the following request.

[https://api.testdataservices.com.au/v0001F\\_GetPerson?UID=80892432-8d0d-468e-954f-10aeb4b31608](https://api.testdataservices.com.au/v0001F_GetPerson?UID=80892432-8d0d-468e-954f-10aeb4b31608)

returns the following:

```
{
  "Address": "36 38 BERESFORD ROAD STRATHFIELD NSW 1816",
  "CC_exp_year": "21",
  "Noun": "fat",
  "BusinessACN": "886654619",
  "Code_40hex": "d312368cfe71cd8eb57a0188b2075607ee8abela",
  "Code_3digit": "038",
  "email": "Michael.Mcauley.9557@testdata.email",
  "DoBYear": "1978",
  "DoBDay": "21",
  "UID": "80892432-8d0d-468e-954f-10aeb4b31608",
  "Profession": "Psychologist",
  "MiddleName": "Robert",
  "Password4": "eD21RS4QWBB7SF03KY",
  "Title": "Mr.",
  "AddressID": "GANSW716727816",
  "AHPRA_Registration": "PSY0003263630",
  "Code_40SafeAlpha": "QGRCURJKKBNPGLRRAGQLJEDZTMBGMHGSULZXDWHF",
  "Code_3alphanum": "GBA",
  "Serial": "303262",
  "Password1": "dCCDE2KJM",
  "Password0": "bZNM6",
  "Password3": "aBM33K",
  "MedicareExp": "10/2021",
  "Password2": "cJASJ&UXG5S=",
  "Medicare": "33940640721",
  "Phone": "0455138757",
  "LicenceNumber": "204026799",
  "DeviceID": "7f7e0ba95c2cca86558fb7888221d6e97611elec",
  "BIP39_25words": "eager flat echo business orient jealous soul swarm enable try subject under desert catch ladder farm bottom ivory dentist response gauge bunker kiss satoshi chair",
  "LicenceExp": "25FEB2024",
  "Code_7alphanum": "CRM3DRB",
  "Code_6alpha": "RKA",
  "GivenName": "Michael",
  "MedicareProviderID": "4666255W",
  "BusinessABN": "88886654619",
  "DoBMonth": "11",
  "Gender": "Male",
  "CC_exp_month": "04",
  "Code_6digit": "016483",
  "Verb": "owe",
  "Code_9digit": "267804490",
  "Code_12digit": "020094004705",
  "Color": "Sangria",
  "BusinessName": "MCAULEY PSYCHOLOGY OF STRATHFIELD",
  "LicenceState": "NSW",
  "FamilyName": "Mcauley",
  "CC_num": "4491021991928551"
}
```

Which includes the address string **36 38 BERESFORD ROAD STRATHFIELD NSW** with a GNAF Address ID of **GANSW716727816**. The actual address is shown in a map below. Unfortunately, most applications expect the address to be either 36 or 38, so it is recommended that when two numbers exist in this field, that either the first or the second is used.



Note that the 25 million identity dataset includes both a concatenated 'free format' address string as well as separate address fields – sourced from the GNAF data source.

# Manufactured Source Documents and their Verification

The method for populating the Identity Dataset used by this service is broadly summarized as follows:

Data is randomly manufactured for every residential address that is of a normal residential occupancy (i.e. not a prison, barracks, retirement home, convent...) based on a statistical model for the real ABS ASGS data for that region. The ABS data not only drives the gender, age, average household size, unemployment rate and high level demographics for the identities created in that region, it also drives the number of related data entities, such as businesses, that operate in that region.

As the key source identity is one's birth certificate, the parents of each identity are reverse connected/engineered rather than manufactured, along with the place of birth, so that regional demographics can be more easily be kept aligned with ABS data. This means that the population relationships in the real world will not be statistically represented in the manufactured dataset, but most Australian Born identities will have a birth certificate that refers to parents that exist within the dataset.

Most Identities in a community do not exist in isolation, but participate in the community. The most obvious example of such participation is employment. Many Identities in the full dataset have an Occupation Code which are assigned using a statistics based approach, based on ANZSCO coding and CENSUS data for the address where the Identity resides.



Identities are verified using a variety of Source Documentation. The process of manufacturing these source documents as well as the verification process for each type of document is included below.

## Birth Certificate

A birth certificate, issued in an Australian State, contains identity information for the individual as well as identity data for the parents of the individual. The data in the full national data set includes an internally consistent set of parent/child relationships, supporting birth certificates, but not all individuals born in Australia can produce a birth certificate, so not all Identities in the full data set will include Birth Certificate information. (see <https://www.service.nsw.gov.au/verify-your-identity-using-myserviceweb-account#australian-birth-certificate> for more information)

State	Electronic Verification Start Year
NSW	1914
ACT	1930
NT	1870
QLD	1953
SA	1944
TAS	1970
WA	1930

## Drivers Licence

Drivers licenses are issued in each State and Territory of Australia, but the name, address and date of birth listed on the Drivers Licence may not match other identity documents. The occasional mismatch of such information is replicated in this data set. Not everyone has a drivers licences and Identities will not have more than one licence.

There are some jurisdiction specific special cases with respect to drivers licenses. For example, names printed on ACT licenses that do not fit are truncated, and the letters TN are added to the license, but electronic verification uses the whole (un truncated) name. Driver licenses are all random 9 digit numbers in this dataset.

## Passport Document Number

Normal Australian Passports (not official or diplomatic) will be included for some Identities, but some of the passports will be expired.

Real Australian Passports may contain partial dates, but in this dataset all identities contain full date of birth. Passport information is not duplicated in the Identity record, so the Date of Birth in the Passport is not separately recorded, but is assumed to be the same as the identity's date of birth. The only Passport specific values recorded in the data is the Document Number, Issue and Expiry dates. Note that some passports in the dataset are expired, to facilitate testing where the expiry duration is important.

Passport numbers will be represented with two letters followed by 7 numbers even though some real world passports only contain one letter with 7 numbers. The passport 'Document Number' will not include any special checksums or formatting requirements but will suffice for simple use against test data validators.

## *Certificate of Australian Citizenship*

Some migrant Identities will have references to a Certificate of Australian Citizenship. The following types are included:

- Certificate of Naturalization-EF (2) prefix, issued 26.01.1955 to 04.02.1970 for issue to female applicants with children
- Certificate of Naturalization-EF (2) prefix, issued 26.01.1955 to 04.02.1970 for issue to female applicants with children included - Back
- Certificate of Naturalization-EM (2) prefix, issued 26.01.1955 to 04.02.1970 for issue to male applicants with children included
- Certificates of Naturalization for Issue to Aliens-EA prefix, issued 26.01.1949 to 04.02.1970 to Certificates issued to minors (under 16 years)
- Certificates of Naturalization for Issue to Aliens-EA prefix issued 26.01.1949 to 04.02.1970 to Certificates issued to minors (under 16 years)
- Certificate of Citizenship issued 06/11/1989 to present – with ACC prefix

## *Medicare Number*

All identities will contain a Medicare Number and families will generally have a single card where each family member has their own number on the card. The last digit in the Medicare Number is the reference for one of the identities on the Medicare Card. This means if the last digit was a 3, then it should be possible to lookup the details for the other identities using the same card with references 1, 2 and possibly 4 or 5.



## Other Manufactured Data

### *Email Addresses*

The set of real world emails includes a huge variety of domains, but all emails included in this manufactured dataset will use a single domain, which is **testdata.email**. Email addresses will, for individuals, comprise a first name, last name and a random number for the free dataset, but the 25 million identity dataset will also include a middle initial. For businesses, email addresses will include the ABN if the business has an ABN. While the leading characters in email addresses are not random and do not have the same character frequency distribution as the population of real world email addresses, they are reasonably well distributed due to the very large number of first names in the manufactured dataset. Many applications index on email addresses, so when testing performance of email related business processes it is important that the indexes are not severely skewed. This has been considered in the manufacture of emails in this dataset.

### *Unique Device / Instance IDs*

Most people in the real world possess multiple mobile devices. Initial implementation of UUIDs in this dataset is limited to a single 40 character ID that will not be aligned with any hardware based format and will be randomly generated. This may limit use where the format of the UUID should contain device specific information.

### *ABN and ACNs*

Businesses and superannuation funds generally have an ABN and ABNs can have ACNs embedded within them. ABNs and ACNs created in this dataset conform to the required formatting and check digits for ABNs and ACNs.

Many businesses operate under a Trust structure, and a significant mix of businesses represented in this Data Set will simulate this type of structure, where the GST ABN will be by a 'Trust' type and will not be directly related to the CAN of the Incorporated Entity.

### *Health Care Professionals*

According to the Department of Health, there are approximately 565,000 registered health professionals operating in Australia. The mix of professionals is represented in the full 25 million identity dataset, but the Free Dataset only includes a small percentage, and only includes those operating as a small, home based business. More information is available at: <https://hwd.health.gov.au/summary.html>

As Health Professionals are required to be registered, AHPRA type registration numbers are included in this dataset. An API endpoint will return information such as is displayed at: <https://www.ahpra.gov.au/Registration/Registers-of-Practitioners.aspx>

## *Small Business Owners*

A number of small business owners are included in the free dataset, but they are all configured as ABN registered private companies, operating from their home address. The full dataset contains a more realistic mix of business structures, but the small business owners (in combination with self employed health professionals) provides many thousands of Business Entities to support testing where business details are required to support testing activities.

## *Secret Words and Passwords*

A series of words, numbers and passwords is included with each identity, and are intended to be a selection that can be used for any relevant requirement as it may relate to any specific project. For some specific requirements, it may be useful to combine several of these values. Some of the fields included are listed below:

- Noun: An English noun that can be used in a secret question/answer situation (e.g. sun)
- Verb: An English verb that can be used in a secret question/answer situation (e.g. grow)
- Color: An color that can be used in a secret question/answer situation (e.g. Desert sand)
- Code\_3digitA 3 digit numeric code (e.g. 128)
- Password0 A password such as cYbhll
- Password1 A password such as QDhrxTcQU
- Password2 A password such as T^HlrCiHtBm%
- Password3 A password such as f)MS[Q<%ukV[BF\*
- Password4 A password such as jEXwqwzDvXaDgwRPKi
- Code\_3alphanum A 3 character alpha numeric code such as 6RK
- Code\_7alphanum A 7 character alpha numeric code 86Y0NS8
- Code\_6digitA should be a 6 digit numeric code, but is actually a 3 digit code
- Code\_9digitA 9 digit numeric code, such as 738408085
- Code\_12digit A 12 digit numeric code, such as 028194305493
- Code\_6alpha A 6 character alpha code, such as ALORWP
- Code\_40SafeAlpha A 40 character uppercase code, not containing I or O characters
- Code\_40hex A 40 character hex code such as 2b6f0cc904d137be2e1730235f5664094b831186

## *Superannuation Funds*

Many applications that hold personally identifiable employee data require the entry of superannuation details, as it is a requirement for employers to make Superannuation Guarantee (SG) contributions. To support this requirement, many identities will have a SuperABN field, which refers to the ABN of a Superannuation Fund along with SuperMember which is a 12 digit membership number for that super fund. To obtain the super fund details, one needs to call the SuperLookup service using the SuperABN value. The data returned will be similar to the ABN, Fund Type, Contact Details, USI and Status returned from this current production Super Fund Lookup service:

<http://superfundlookup.gov.au/ABN/View?id=75493363262>. There are 159 Public Offer Super Funds operating in Australia in 2019, so there are 150 Public Offer Super Funds in this dataset.

## *UID – Unique Identifier*

Access to individual identities is via a UID. If you need to retrieve the same identity in the future, then it is necessary to store the UID for the record you wish to retrieve. An example UID is: 08006c52-9543-4bcc-9aa8-d523df08acd6.

If a testing project starts with a Free Version of the API and retrieves a record, then saving the UID will enable the more complete record to be download if a paid subscription is used in the future, but only if the same version of data is being used. (When major revisions to data are made, the entire dataset is recreated, using a new set of UIDs)



## Detailed Examples of API Usage, using Windows Powershell:

Making a call to [https://api.testdataservices.com.au/v0001F\\_GetRandomPerson](https://api.testdataservices.com.au/v0001F_GetRandomPerson) will return something like the following.

```
{
  "Address": "90 TRICKEYS LANE DRUMMOND VIC 3461",
  "CC_exp_year": "23",
  "Noun": "sex",
  "Code_40hex": "61bb16bd7340b7e62ae941f2215b82deab2643fd",
  "Code_3digit": "116",
  "PassportExpiry": "14APR2025",
  "DoBYear": 1963,
  "UID": "f1b528da-7adf-4558-a376-65d75bcda740",
  "Password4": "gQ3EULRK82JNOHUMFR",
  "AHPRA_Registration": "MED0002355001",
  "Serial": 200195,
  "Code_3alphanum": "Z3Q",
  "Password1": "cVMU5L829",
  "Password0": "kA7HJF",
  "Password3": "wY1U0T",
  "PassportNo": "BP0283192",
  "Password2": "v&7KU)CCL9{",
  "Phone": "0451001921",
  "Medicare": "52829829332",
  "DeviceID": "b2b8ac11a7a0b256e54cf44ab47909baa4ee50db",
  "Code_6alpha": "PPX",
  "MedicareProviderID": "7832416T",
  "DoBMonth": 10,
  "Code_6digit": "029469",
  "Code_9digit": "134930711",
  "LicenceState": "VIC",
  "BusinessName": "WEN MEDICAL CENTRE OF DRUMMOND",
  "FamilyName": "Wen",
  "PassportPlaceOfBirth": "Geelong",
  "BusinessACN": "798468854",
  "email": "Kevin.Wen.4855@testdata.email",
  "DoBDay": 9,
  "Profession": "Medical Practitioner",
  "MiddleName": "Charles",
  "Title": "Dr",
  "AddressID": "GAVIC421151231",
  "Code_40SafeAlpha": "UUXFTYZCYMFACGTZTQMYSJTLXAHWDGJCBZWSARMW",
  "MedicareExp": "06/2024",
  "LicenceNumber": "984770420",
  "BIP39_25words": "maze fancy runway shine deal cart vital patrol nice energy chuckle supply plunge become 1
  ayer ritual agree crack mix say when force vessel fresh tree",
  "SuperMember": "001082008969",
  "LicenceExp": "19APR2022",
  "Code_7alphanum": "7ABZC8C",
  "GivenName": "Kevin",
  "BusinessABN": "24798468854",
  "SuperABN": "98401566658",
  "Gender": "Male",
  "CC_exp_month": "09",
  "Verb": "breathe",
  "PassportIssue": "14APR2015",
  "TFN": "768282891",
  "Code_12digit": "004399013197",
  "Color": "Cerulean",
  "CC_num": "4117797707439942"
}
```

This particular identity has a Credit Card, a Drivers Licence, a Passport, a preferred superannuation fund and is a Medical Practitioner with a Medicare Provider ID and an AHPRA Registration ID and is a small business, operating as an ABN registered single director private company with a registered address matching the home address. Like all other identities in this dataset, it also contains a Medicare Number and Address, including a GNAF Address ID and a variety of general purpose codes.

A common requirement for a testing project is a significant number of records that can be loaded into an application and used within the test team.

The field names in the above example can be used to access the field values.

If for example, you wanted a comma separated list of identities with an email address, first name, last name, medicare number, phone number, a free format address and a reasonably strong password, you could enter the following into a PowerShell window.

```
$RequiredNum = 10
$Identities = for ($i=1; $i -le $RequiredNum; $i++) {
    $R=Invoke-RestMethod -Uri https://api.testdataservices.com.au/v0001F_GetRandomPerson
    $Line="$($R.email),$($R.GivenName),$($R.FamilyName),$($R.Medicare),$($R.Phone),$($R.Address),$($R.Password2) "
    $Line
}

"This is a list of records to be used for testing:"
$Identities

"Saving the records into a file (DataForTesting) in the current directory ...."
$Identities > DataForTesting.csv
```

Note that if you try this and encounter a SSL or TLS error, then you will need to upgrade the security of your PowerShell session. This can be done by entering the following command into the PowerShell.

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
```

Running the Powershell command should result in something like the following



```
Windows PowerShell (x86)
>> }
PS C:\Users\paul>
PS C:\Users\paul> "This is a list of records to be used for testing:"
This is a list of records to be used for testing:
PS C:\Users\paul> $Identities
Robert.Livingston.4824@testdata.email,Robert,Livingston,54280459111,0479296735,49 B JONES ROAD KENTHURST NSW 2156,b;7QC-RAMAF{
Steven.Gentle.1334@testdata.email,Steven,Gentle,51931574931,0443742810,UNIT 37 28 BRICKWORKS DRIVE HOLROYD NSW 2142,j=995=A2ZGA{
Dana.Olona.5895@testdata.email,Dana,Olona,65623982531,0491204695,SALVATION ARMY FAMILY STORE 115 117 NEW TOWN ROAD NEW TOWN TAS 7008,m=5EH[QL779%
Easton.Williams.2514@testdata.email,Easton,Williams,20254580611,0441774331,13 46 VICTOR STREET GREYSTANES NSW 2145,s[7D2]HHQML#
Gavin.Almond.8056@testdata.email,Gavin,Almond,42333596341,0460184794,2 WILKINS CRESCENT BURNSIDE HEIGHTS VIC 3023,o>Y6T#2EUKX^
Virginia.Packham.6430@testdata.email,Victoria,Packham,61777634341,0475192964,19 TAYLOR STREET BROMPTON SA 5007,f)C6R#SZL6W=
Evelyn.McSkimming.480@testdata.email,Evelyn,McSkimming,44130000011,0411705190,10 15 COLLIER CLOSE ST HELENS PARK NSW 2560,n{QFD&S2DLW(
Maria.Salaoutis.6558@testdata.email,Maria,Salaoutis,47752829052,0421519004,326 FISHER STREET CLOVERDALE WA 6105,u[GXR%NCJSW]
Ulises.Forrest.8992@testdata.email,Ulises,Forrest,53751443633,0464945012,21 MCLAREN STREET SOUTH FREMANTLE WA 6162,x*RVC^CTA2Q]
Tucker.Salviani.8624@testdata.email,Tucker,Salviani,58444463842,0414974324,UNIT 70 29 70 TAURUS STREET ELMORE VALE NSW 2287,c]19N;VJIBC#
PS C:\Users\paul>
PS C:\Users\paul> "Saving the records into a file (DataForTesting) in the current directory ...."
Saving the records into a file (DataForTesting) in the current directory ....
PS C:\Users\paul> $Identities > DataForTesting.csv
PS C:\Users\paul>
```

By using this example code as a starting point, it is possible to build any list of values, as required, for almost any testing project. It is not necessary to use a very small number. You can request thousands of records using this approach, though it only processes a few per second, so thousands of records will take many minutes to generate.

The following PowerShell screenshot shows the process of Clearing an email address for a particular domain and then shows the results of repeated calls to GetMessage for that email address. GetMessage usually returns the email message within a few seconds of the message being sent from the 'FromDomain' server.

```
Windows PowerShell (x86)
-----
EmailAddress      ClearTime FromDomain      ClearMessage
-----
a12345@testdata.email 1568419246 testdataservices.com.au      True

PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_ClearMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'
200
PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_GetMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'

EmailAddress      ClearTime FromDomain      ClearMessage
-----
a12345@testdata.email 1568419278 testdataservices.com.au      True

PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_GetMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'

EmailAddress      ClearTime FromDomain      ClearMessage
-----
a12345@testdata.email 1568419278 testdataservices.com.au      True

PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_GetMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'

EmailAddress      ClearTime FromDomain      ClearMessage
-----
a12345@testdata.email 1568419278 testdataservices.com.au      True

PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_GetMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'

EmailAddress      ClearTime FromDomain      ClearMessage
-----
a12345@testdata.email 1568419278 testdataservices.com.au      True

PS C:\Users\paul> Invoke-RestMethod -Uri 'https://api.testdataservices.com.au/v00015_GetMessage?EmailAddress=a12345@testdata.email&FromDomain=testdataservices.com.au'

Email Address      : a12345@testdata.email
RecieveTime        : 1568419371
FromDomain          : testdataservices.com.au
html                : <html xmlns:v="urn:schemas-microsoft-com:vml" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:w="urn:schemas-microsoft-com:office:word"
                      xmlns:m="http://schemas.microsoft.com/office/2004/12/omml" xmlns="http://www.w3.org/TR/REC-html40">
                      <head>
                      <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
                      <meta name="Generator" content="Microsoft Word 15 (filtered medium)">
                      <style><!--
                      /* Font Definitions */
                      @font-face
                      {font-family:"Cambria Math";
                      panose-1:2 4 5 3 5 4 6 3 2 4;}
                      @font-face
                      {font-family:Calibri;
                      panose-1:2 15 5 2 2 2 4 3 2 4;}
                      /* Style Definitions */
                      p.MsoNormal, li.MsoNormal, div.MsoNormal
                      {margin:0cm;
                      margin-bottom:.0001pt;
                      font-size:11.0pt;
                      font-family:"Calibri",sans-serif;
                      mso-fareast-language:EN-US;}
                      a:link, span.MsoHyperlink
                      {mso-style-priority:99;
                      color:#0563C1;
                      text-decoration:underline;}
                      a:visited, span.MsoHyperlinkFollowed
                      {mso-style-priority:99;
                      color:#954F72;
                      text-decoration:underline;}
                      span.EmailStyle17
                      {mso-style-type:personal-compose;
                      font-family:"Calibri",sans-serif;
                      color:windowtext;}
                      .MsoChpDefault
                      {mso-style-type:export-only;
                      font-family:"Calibri",sans-serif;
                      mso-fareast-language:EN-US;}
                      @page WordSection1
                      {size:612.0pt 792.0pt;
                      margin:72.0pt 72.0pt 72.0pt 72.0pt;}
                      div.WordSection1
                      {page:WordSection1;}
                      <!--></style><!--[if gte mso 9]><xml>
                      <o:shapedefaults v:ext="edit" spidmax="1026" />
                      </xml><![endif]><!--[if gte mso 9]><xml>
                      <o:shapelayout v:ext="edit">
                      <o:idmap v:ext="edit" data="1" />
                      </o:shapelayout></xml><![endif]><!-->
                      </head>
                      <body lang="EN-AU" link="#0563C1" vlink="#954F72">
                      <div class="WordSection1">
                      <p class="MsoNormal">This is the body of a simple message.<o:p></o:p></p>
                      </div>
                      </body>
                      </html>

remote_ip          : 40.107.136.91
Plain              : This is the body of a simple message.
```

## Some Data Tagging examples with Powershell:

The following code creates a temporary key for Data Tagging purposes, using an active subscription key, and saves it in the variable \$TempKey

```
$TempKey=Invoke-RestMethod -Uri  
https://api.testdataservices.com.au/v0001F_AddSubscriptionKey?ParentSubscription=GEZDGNBVGy3TQOJQGEZDGNBVGy3TQOJQ  
$Key=$TempKey.SubscriptionKey
```

The following code saves 100 randomly selected Identities, along with a sequentially numbered UserID for each of them:

```
$RequiredNum = 100  
$Identities = for ($i=1; $i -le $RequiredNum; $i++) {  
    $R=Invoke-RestMethod -Uri https://api.testdataservices.com.au/v0001F_GetRandomPerson  
    $UID=$R.UID  
    $UserID= "Test" + $i  
    Invoke-RestMethod -Uri  
    "https://api.testdataservices.com.au/v0001F_SaveUserID?SubscriptionKey=$($Key) &UserID=$($UserID) &PersonUID=$($UID) "  
}
```

This request fetches the Identities Randomly

```
Invoke-RestMethod -Uri "https://api.testdataservices.com.au/v0001F_GetSavedUserID?SubscriptionKey=$($Key) "
```

This request fetches the Identities Sequentially (until there are none left)

```
Invoke-RestMethod -Uri "https://api.testdataservices.com.au/v0001F\_PopSavedUserID?SubscriptionKey=\$\(\$Key\) "
```

This request recycles the Identities, so that they can be re-fetched

```
Invoke-RestMethod -Uri "https://api.testdataservices.com.au/v0001F\_RecycleSavedUserIDs?SubscriptionKey=\$\(\$Key\) "
```

## Detailed Examples of API Usage, using AWS Linux:

Fetching a random Person from the 2.5 million identity dataset:

```
curl https://api.testdataservices.com.au/v0001F_GetRandomPerson
```

To easily handle JSON data from the linux command prompt, you can use jq. If you are using an AWS Linux instance, then you can run the following to install jq.

```
sudo yum install jq
```

Once installed you can fetch a random person into a variable, and then isolate out specific fields, and used them, as shown in the following commands:

```
# Get a random person and view the JSON fields on screen
```

```
curl https://api.testdataservices.com.au/v0001F_GetRandomPerson | jq
```

```
{
  "Address": "204 COOLART ROAD MOOROODUC VIC 3933",
  "CC_exp_year": "23",
  "PassportPlaceOfBirth": "Ulverstone",
  "Noun": "estate",
  "Code_40hex": "eedd85550822012b12c01cd3c96ee11a18226821",
  "Code_3digit": "402",
  "PassportExpiry": "17SEP2025",
  "email": "Stefan.Heading.5423@testdata.email",
  "DoBYear": 1979,
  "DoBDay": 16,
  "UID": "975c4d85-8c05-4852-af12-ba38ab8c6aae",
  "MiddleName": "Sam",
  "Password4": "tUDK6YQAVL0S2GWADW",
  "Title": "Mr.",
  "AddressID": "GAVIC421770948",
  "Code_40SafeAlpha": "VXVZLFJQRKJRXBTCDSYNNHRRDHNVSXRGTADEGVC",
  "Serial": 480920,
  "Code_3alphanum": "070",
  "Password1": "iD58CR20A",
  "Password0": "vJ38N2",
  "Password3": "hYFE2P",
  "MedicareExp": "07/2020",
  "PassportNo": "EQ0995604",
  "Password2": "o<RQ1#8V740^",
  "Phone": "0456806471",
  "Medicare": "54438554631",
  "LicenceNumber": "535340937",
  "DeviceID": "23c50e014df65c55de173a4a5000d812987bd14b",
  "BIP39_25words": "already equip follow trumpet skate fiber already mechanic mango vocal
diamond year legend holiday execute gadget wear owner noble turkey pitch lonely gallery vehicle
floor",
  "SuperMember": "000663015910",
  "LicenceExp": "17SEP2026",
  "Code_7alphanum": "RBD5D1K",
  "GivenName": "Stefan",
  "Code_6alpha": "NQP",
  "SuperABN": "23614341507",
  "Gender": "Male",
  "DoBMonth": 3,
  "CC_exp_month": "11",
  "Verb": "seal",
  "Code_6digit": "014097",
  "Code_9digit": "668005795",
  "PassportIssue": "17SEP2015",
  "TFN": "821817528",
  "Code_12digit": "024888015211",
  "Color": "Azure",
  "LicenceState": "VIC",
  "FamilyName": "Heading",
```

```

        "CC_num": "4736882991515457"
    }
}

# Get a random person and save the JSON response in a variable
PersonJSON=`curl https://api.testdataservices.com.au/v0001F_GetRandomPerson`

# Show the email address from the retrieved person
echo $PersonJSON | jq -r '.email'

Susan.Knox.9332@testdata.email

# Save and show the medicare number from the retrieved person
Medicare=`echo $PersonJSON | jq -r '.Medicare'`

echo $Medicare

68133107311

# Check the validity of the Medicare number
curl https://api.testdataservices.com.au/v0001F_CheckMedicare?Medicare=$Medicare | jq

{
  "Medicare": "68133107311",
  "GivenName": "Susan",
  "MiddleName": "Evelyn",
  "FamilyName": "Knox",
  "Gender": "Female",
  "DoBYear": 1945,
  "DoBMonth": 8,
  "DoBDay": 22,
  "Expiry": "03/2021"
}

# Get a random Self Employed Health Professional and save the JSON response in a variable
HealthProfJSON=`curl https://api.testdataservices.com.au/v0001F_GetRandomSEHP`

# Save Provider ID and ABN
Provider=`echo $HealthProfJSON | jq -r '.MedicareProviderID'`

ABN=`echo $HealthProfJSON | jq -r '.BusinessABN'`

# Check the validity of the Medicare Provider ID and ABN
curl https://api.testdataservices.com.au/v0001S_CheckMedicareProviderID?MedicareProviderID=$Provider | jq

{
  "MedicareProviderID": "940206CJ",
  "AHPRA_Registration": "PSY0009373300",
  "FamilyName": "Tuche",
  "GivenName": "Gary",
  "Profession": "Psychologist",
  "Title": "Mr.",
  "UID": "c94bd02c-9df8-4b84-a61a-bf067c17cece"
}

curl https://api.testdataservices.com.au/v0001F_CheckABN?ABN=$ABN | jq

{
  "EntityName": "TUCHE PSYCHOLOGY OF MADDINGTON PTY LTD",
  "AbnStatus": "Active",
  "EntityTypeCode": "PRV",
  "Gst": "2007-01-17",
  "EntityTypeName": "Australian Private Company",
  "AddressPostcode": "6109",
  "AddressState": "WA",
  "AddressDate": "2009-06-28",
  "ABN": "80681958225",
  "BusinessName": []
}

```

Note that the API endpoints in the above example require a subscription, unless the queries are restricted to identities contained in the first 25,000 records of the 2.5 million identity dataset. The subset of 25,000 identities can be downloaded from [here](#) and calls to validation end points for each these records will work, even without a subscription.

# Detailed Examples of API Usage, using MicroFocus LoadRunner:

The following partial screen shot of a VUGen screen shows several API calls, relating to fetching of random data as well as automation of Email and MFA.

```
1 You can now use the Web - HTTP/HTML protocol to record any mobile application.
Solution Explorer
  Solution Untitled
    SignupAndActivityDemo
      Actions
        user_init
        Action
        user_end
      Extra Files
        globals.h
      Runtime Settings
      Parameters
      Recording Report
      Replay Run Results

1 Action()
2 {
3     // visit landing page of AUT
4
5
6     // Fetch random identity, saving key values
7     web_reg_save_param("Email", "LB=\"email\":\\", "RB=\"", "ORD=1", LAST);
8     web_reg_save_param("GivenName", "LB=\"GivenName\":\\", "RB=\"", "ORD=1", LAST);
9     web_reg_save_param("FamilyName", "LB=\"FamilyName\":\\", "RB=\"", "ORD=1", LAST);
10    web_reg_save_param("Phone", "LB=\"Phone\":\\", "RB=\"", "ORD=1", LAST);
11    web_reg_save_param("UID", "LB=\"UID\":\\", "RB=\"", "ORD=1", LAST);
12    web_reg_save_param("Password", "LB=\"Password1\":\\", "RB=\"", "ORD=1", LAST); // this password satisfies the complexity requirements for the AUT
13    web_reg_save_param("Address", "LB=\"Address\":\\", "RB=\"", "ORD=1", LAST);
14    web_rest("GET: https://api.testdataservices.com.au/v0001F_GetRand...",
15            "URL=https://api.testdataservices.com.au/v0001F_GetRandomPerson",
16            "Method=GET",
17            "Snapshot=t288167.inf",
18            LAST);
19
20    // signup page of AUT
21
22
23    // Clear Email, resetting email timer
24    web_rest("GET: https://api.testdataservices.com.au/v0001S_ClearMe...",
25            "URL=https://api.testdataservices.com.au/v0001S_ClearMessage?EmailAddress={Email}&FromDomain=example.com",
26            "Method=GET",
27            "Snapshot=t992746.inf",
28            LAST);
29
30    // Submit Form of AUT
31
32
33    // Get verification email contents (this should be implemented in a loop that waits a pre-configured maximum time)
34    web_reg_save_param("Latency", "LB=\"latency\":\\", "RB=\"", "ORD=1", LAST);
35    web_reg_save_param("ActivationLink", "LB=\"https://", "RB=\"", "ORD=1", LAST);
36    web_rest("GET: https://api.testdataservices.com.au/v0001S_GetMess...",
37            "URL=https://api.testdataservices.com.au/v0001S_GetMessage?EmailAddress={Email}&FromDomain=example.com",
38            "Method=GET",
39            "Snapshot=t41441.inf",
40            LAST);
41    lr_set_transaction("EmailVerification", atof(lr_eval_string("{Latency}")), LR_PASS); // record email latency
42
43
44    // Complete Signup, set password, get MFA Secret Key (save in parameter {SecretKey}) in AUT
45
46    // Save MFA Secret
47    web_rest("GET: https://api.testdataservices.com.au/v0001F_SaveGoo...",
48            "URL=https://api.testdataservices.com.au/v0001F_SaveGoogleSecret?Email={Email}&SubscriptionKey=34feff14fd463d26a40a&SecretKey={SecretKey}",
49            "Method=GET",
50            "Snapshot=t990051.inf",
51            LAST);
52
53    // Initiate MFA Activation
54
55
56    // Retrieve MFA Token
57    web_reg_save_param("ThisToken", "LB=\"ThisToken\":\\", "RB=\"", "ORD=1", LAST);
58    web_reg_save_param("NextToken", "LB=\"NextToken\":\\", "RB=\"", "ORD=1", LAST);
59    web_reg_save_param("SecondsRemaining", "LB=\"SecondsRemaining\":\\", "RB=\"", "ORD=1", LAST);
60    web_rest("GET: https://api.testdataservices.com.au/v0001F_GetGoog...",
61            "URL=https://api.testdataservices.com.au/v0001F_GetGoogleAuthCode?Email={Email}&SubscriptionKey=34feff14fd463d26a40a",
62            "Method=GET",
63            "Snapshot=t846400.inf",
64            LAST);
65
66    if (atof(lr_eval_string("{SecondsRemaining}")) < 2.0) {
67        lr_think_time(2.0); // wait for next token
68        lr_save_string(lr_eval_string("{NextToken}"), "ThisToken"); // switch token
69    }
70
71    // Confirm MFA by using {ThisToken}
72
```

This screen shot shows a fragment of VUGen code used to retrieve details for a particular identity, and then retrieve a Google Authentication Token using the MFA API call.

Solution Explorer

- Solution Untitled
- SignupAndActivityDemo
  - Actions
    - user init
    - Action
    - user end
  - Extra Files
    - globals.h
  - Runtime Settings
  - Parameters
  - Recording Report
  - Replay Run Results

Signup...tivityDemo : Replay Summary

Signup...tivityDemo : Action.c

```

118
119 // Retrieve identity, saving key values
120 web_reg_save_param("Email", "LB=\"email\":\\", "RB=\"", "ORD=1", LAST);
121 web_reg_save_param("GivenName", "LB=\"GivenName\":\\", "RB=\"", "ORD=1", LAST);
122 web_reg_save_param("FamilyName", "LB=\"FamilyName\":\\", "RB=\"", "ORD=1", LAST);
123 web_reg_save_param("Phone", "LB=\"Phone\":\\", "RB=\"", "ORD=1", LAST);
124 web_reg_save_param("Password", "LB=\"Password1\":\\", "RB=\"", "ORD=1", LAST);
125 web_reg_save_param("Address", "LB=\"Address\":\\", "RB=\"", "ORD=1", LAST);
126 web_rest("GET: https://api.testdataservices.com.au/v0001F_GetPers...",
127 "URL=https://api.testdataservices.com.au/v0001F_GetPerson?UID={UID}",
128 "Method=GET",
129 "Snapshot=t790975.inf",
130 LAST);
131
132 // Login using email address and Password (retrieved from API)
133
134
135 // Retrieve MFA Token
136 web_reg_save_param("ThisToken", "LB=\"ThisToken\":\\", "RB=\"", "ORD=1", LAST);
137 web_reg_save_param("NextToken", "LB=\"NextToken\":\\", "RB=\"", "ORD=1", LAST);
138 web_reg_save_param("SecondsRemaining", "LB=\"SecondsRemaining\":\\", "RB=\"", "ORD=1", LAST);
139 web_rest("GET: https://api.testdataservices.com.au/v0001F_GetGoog...",
140 "URL=https://api.testdataservices.com.au/v0001F_GetGoogleAuthCode?Email={Email}&subscriptionKey=34feff14fd463d26a40a",
141 "Method=GET",
142 "Snapshot=t846400.inf",
143 LAST);
144
145 if (atof(lr_eval_string("{SecondsRemaining}")) < 2.0) {
146     lr_think_time(2.0); // wait for next token
147     lr_save_string(lr_eval_string("{NextToken}"), "ThisToken"); // switch token
148 }
149
150 // Complete MFA step of login process by using {ThisToken}
151
152

```

Output

Replay

Replay status failed View summary Started at: 29/10/2019 9:43:27 AM Elapsed time: 00:00:01

```

Action.c(14): x-amz-cr-pop: mELw\r\n
Action.c(14): x-amz-cf-id: 5Vzilyi_ck6cHCGenE6NpJ3bzA0zh8TRginiC09AyWuVnTUrT6uSQ==\r\n
Action.c(14): \r\n
Action.c(14): t=581ms: 1423-byte response body for "https://api.testdataservices.com.au/v0001F_GetRandomPerson" (RelFrameId=1, Internal ID=1)
Action.c(14): {"Address": "20 CARNE STREET MOULAMEIN NSW 2733", "CC_exp_year": "23", "PassportPlaceOfBirth": "Townsville", "Noun": "marriage", "Code_40hex": "b3b6e454ccfb337f100f1fb3250de13fb9d07", "Code_3digit": "497", "PassportExpiry": "19JUN2020", "email": "Jasmin.White.7644@testdata.email", "Action.c(14): DoByYear": "1991", "DoDay": "30", "UID": "bb433424-ed5f-4a61-afb4-4003b7e6f9a5", "MiddleName": "Teres", "Password4": "dNAR17UPEV4E8338FA", "Title": "Ms.", "AddressID": "GANSW712804156", "Code_40Saf", "Alpha": "GZWRH0XEKFNFSDDHNLMDRTCLKSJWVWVJTYCGQHV", "Serial": "449157", "Code_3alphanumeric": "7PG", "Action.c(14): "Password1": "jHNY6PZZ1", "Password0": "gU4LFJ", "Password3": "KK6FQQ", "MedicareExp": "05/2023", "Action.c(14): "PassportNo": "L40867389", "Password2": "i5E6F-dFKM8", "Phone": "0485162684", "Medicare": "33337100051", "LicenceNumber": "668360736", "DeviceID": "bd7c0685e25e5c2d1166fbb9cd7aefdb977885c", "Action.c(14): "BIP39_25words": "bone concert crater gorilla smoke hockey magic relax person pepper box el", "Action.c(14): "LicenceExp": "27JUN2025", "SuperMember": "025231004429", "Code_7alphanumeric": "F4K7RNL", "G", "ivenName": "Jasmin", "Code_6alpha": "JEB", "SuperABN": "99518512859", "Gender": "Female", "DoBMonth": "11", "CC_exp_month": "01", "Verb": "correct", "Code_6digit": "006438", "Code_9digit": "632500990", "Action.c(14): "PassportIssue": "19JUN2019", "TFN": "756310809", "Code_12digit": "027201017136", "Color": "Ros", "Action.c(14): e", "LicenceState": "NSW", "FamilyName": "White", "CC_num": "4291411300992038"}
Action.c(14): Notify: Saving Parameter "Address = 20 CARNE STREET MOULAMEIN NSW 2733".
Action.c(14): Notify: Saving Parameter "Email = Jasmin.White.7644@testdata.email".
Action.c(14): Notify: Saving Parameter "UID = bb433424-ed5f-4a61-afb4-4003b7e6f9a5".
Action.c(14): Notify: Saving Parameter "Password = jHNY6PZZ1".
Action.c(14): Notify: Saving Parameter "Phone = 0485162684".
Action.c(14): Notify: Saving Parameter "GivenName = Jasmin".
Action.c(14): Notify: Saving Parameter "FamilyName = White".
Action.c(14): web_rest("GET: https://api.testdataservices.com.au/v0001F_GetRand...) was successful, 1423 body bytes, 473 header bytes [MsgId: MWSG-26386]
Action.c(24): web_rest("GET: https://api.testdataservices.com.au/v0001S_ClearMe...) started [MsgId: MWSG-26355]
Action.c(24): Notify: Parameter Substitution: parameter "Email" = "Jasmin.White.7644@testdata.email"
Action.c(24): The request to server https://api.testdataservices.com.au/v0001S_ClearMessage?EmailAddress=Jasmin.White.7644@testdata.email&FromDomain=example.com" is done with headers:
Action.c(24): :method = GET
Action.c(24): :scheme = https
Action.c(24): :path = /v0001S_ClearMessage?EmailAddress=Jasmin.White.7644@testdata.email&FromDomain=exam

```

# Detailed Examples of API Usage, using Python / Locust:

Example Load Test Script for Locust, that invokes 5 REST APIs, showing how data resulting from one API call can be used in a subsequent call along with a screen shot of Locust Load Test.

```
from locust import HttpLocust, TaskSet

def login(l):
    SubKey='RajQcaJIVcgESzJeRRtquQ48sfCtYzDd'

def logout(l):
    l.client.get("/v0001F_GetRandomPerson")

def GetRandomPerson(l):
    l.client.get("/v0001F_GetRandomPerson")

def SaveUserID(l):
    SubKey='RajQcaJIVcgESzJeRRtquQ48sfCtYzDd'
    resp=l.client.get("/v0001F_GetRandomPerson")
    UserID=resp.json()['UID']
    UserID = 'test' + str(resp.json()['Serial'])
    SaveRequest="/v0001F_SaveUserID?SubscriptionKey=" + SubKey + "&UserID=" + UserID + "&PersonUID=" + UserID
    l.client.get(SaveRequest, name="/v0001F_SaveUserID")

def RegisterUser(l):
    EmailKey='34feff14fd463d26a40a'
    SubKey='RajQcaJIVcgESzJeRRtquQ48sfCtYzDd'
    resp=l.client.get("/v0001F_GetRandomPerson")
    UserID=resp.json()['UID']
    Email = resp.json()['email']
    ClearEmailRequest="/v0001S_ClearMessage?EmailAddress=" + Email + "&FromDomain=testdataservices.com.au&SubscriptionKey=" + EmailKey
    l.client.get(ClearEmailRequest , name="/v0001S_ClearMessage")
    GetEmailRequest="/v0001S_GetMessage?EmailAddress=" + Email + "&FromDomain=testdataservices.com.au&SubscriptionKey=" + EmailKey
    l.client.get(GetEmailRequest , name="/v0001S_GetMessage")

def GetUserID(l):
    SubKey='RajQcaJIVcgESzJeRRtquQ48sfCtYzDd'
    GetRequest="/v0001F_GetSavedUserID?SubscriptionKey=" + SubKey
    RetrievedUser=l.client.get(GetRequest, name="/v0001F_GetSavedUserID")
    UserID=RetrievedUser.json()['PersonUID']
    RetrieveRequest="/v0001F_GetPerson?UID=" + UserID
    l.client.get(RetrieveRequest, name="/v0001F_GetPerson")

class UserBehavior(TaskSet):
    tasks = (GetRandomPerson: 2, RegisterUser: 1, SaveUserID: 1, GetUserID: 5)

    def on_start(self):
        login(self)

    def on_stop(self):
        logout(self)

class WebsiteUser(HttpLocust):
    task_set = UserBehavior
    min_wait = 500
    max_wait = 1500
```



LOCUST

HOST  
https://api.testdataservice  
s.com.au

STATUS  
**RUNNING**  
4000 users  
[Edit](#)

SLAVES  
**12**

RPS  
**6279**

FAILURES  
**0%**



Reset  
Stats

[Statistics](#) [Charts](#) [Failures](#) [Exceptions](#) [Download Data](#) [Slaves](#)

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/v0001F_GetPerson	65958	0	80	89	26	3023	1349	1820
GET	/v0001F_GetRandomPerson	53429	0	130	148	48	1145	1350	1493.1
GET	/v0001F_GetSavedUserID	66130	0	120	127	51	1710	76	1820.9
GET	/v0001F_SaveUserID	13374	0	78	85	44	845	75	380.1
GET	/v0001S_ClearMessage	13437	0	63	70	24	1511	3	382.6
GET	/v0001S_GetMessage	13418	0	72	78	28	795	137	382.3
Aggregated		225746	0	110	112	24	3023	749	6279

The following python commands show various REST API calls and their responses.

# Import necessary modules to support HTTP Requests and JSON handling

```
import requests
```

```
import json
```

# Make a request to Get a Random Person, saving the results in the variable RP then show it's contents

```
RP=requests.get('https://api.testdataservices.com.au/v0001F_GetRandomPerson');
```

```
RP.text
```

```
{'Address': '177 OLD MILL ROAD WOLUMLA NSW
2550', 'CC_exp_year': '21', 'Noun': 'pressure', 'Code_40hex': '1cc5d01a7442c48ca78d3e96a0093b7d9179473a', 'Code_3digit': '83
4', 'email': 'Richard.Soldat.5586@testdata.email', 'DoBYear': '1956', 'DoBDay': '14', 'UID': '94bfb8a8-4cc4-471c-9b8c-
8f583a930879', 'MiddleName': 'Bill', 'Password4': 'dJ62DGVLYD5DJZKZR7', 'Title': 'Mr.', 'AddressID': 'GANSW712563668', 'Code_
40SafeAlpha': 'HSFPDVTXZTXZLLJQJSRZMDQXEPAGFTUSQYCWJUST', 'Serial': '1604887', 'Code_3alphanumeric': 'QLV', 'Password1': 'a7VADJR
NK', 'Password0': 'v2M4JL', 'Password3': 'uU5TAF', 'MedicareExp': '12/2022', 'Password2': 'n (J15)21YVJ!', 'Phone': '0408555131
', 'Medicare': '39770889351', 'LicenceNumber': '805665074', 'DeviceID': '8a3d713d100ced2802080e50b4532ed2c8d2895f', 'BIP39_
25words': 'senior already bonus play taxi mimic boring glue demise ostrich initial shed raw fancy below frost lend
parrot dynamic celery firm cage daughter crush
fog', 'SuperMember': '000176017935', 'LicenceExp': '13SEP2021', 'Code_7alphanumeric': 'LRLFJTN', 'GivenName': 'Richard', 'Code_6a
lpha': 'ODE', 'SuperABN': '32531094229', 'Gender': 'Male', 'DoBMonth': '12', 'CC_exp_month': '05', 'Verb': 'lock', 'Code_6digit': '
012255', 'Code_9digit': '655205635', 'TFN': '398085445', 'Code_12digit': '024089015752', 'Color': 'Peach', 'LicenceState': 'NS
W', 'FamilyName': 'Soldat', 'CC_num': '4262598064187089'}
```

# Convert the Random Person details (RP) into JSON format, and save as RPJ, then extract various values

```
RPJ=RP.json()
```

```
UID=RPJ['UID']
```

```
Medicare=RPJ['Medicare']
```

```
UserID='TestUser'+str(RPJ['Serial'])
```

```
Email=RPJ['email']
```

```
GivenName=RPJ['GivenName']
```

```
FamilyName=RPJ['FamilyName']
```

```
Password=RPJ['Password2']
```

```
'About to process Identity ' + UID + ' with email of ' + Email + ' and name of ' + GivenName + ' ' + FamilyName + '
by signing that user up with a UserID of ' + UserID
```

```
'About to process Identity 94bfb8a8-4cc4-471c-9b8c-8f583a930879 with email of Richard.Soldat.5586@testdata.email and
name of Richard Soldat by signing that user up with a UserID of TestUser1604887'
```

```
>>> RPJ=RP.json()
>>> UID=RPJ['UID']
>>> Medicare=RPJ['Medicare']
>>> UserID='TestUser'+str(RPJ['Serial'])
>>> Email=RPJ['email']
>>> GivenName=RPJ['GivenName']
>>> FamilyName=RPJ['FamilyName']
>>> Password=RPJ['Password2']
>>> 'About to process Identity ' + UID + ' with email of ' + Email + ' and name of ' + GivenName + ' ' + FamilyName + ' by signing that user up with a UserID of ' + UserID
'About to process Identity 94bfb8a8-4cc4-471c-9b8c-8f583a930879 with email of Richard.Soldat.5586@testdata.email and name of Richard Soldat by signing that user up with a UserID of TestUser1604887'
>>>
```

## # Create temporary keys for Data Tagging, then save and retrieve a UserID

```
SubKeyResp= requests.get('https://api.testdataservices.com.au/v0001F_AddSubscriptionKey?SubscriptionKey=
dNQNhjq2GeWJ9DNQuaatZ7shdQKSQ2Uz');

SubKey=SubKeyResp.json() ['SubscriptionKey']

SubKey

'9I2htdRXn5YFjwbxknvPNL9ALXscgDzc'

requests.get('https://api.testdataservices.com.au/v0001F_SaveUserID?SubscriptionKey=' + SubKey + '&UserID=' + UserID
+ '&PersonUID=' + UID)
<Response [200]>
RU=requests.get('https://api.testdataservices.com.au/v0001F_GetSavedUserID?SubscriptionKey='+SubKey)

RU.text

{'PersonUID':"94bfb8a8-4cc4-471c-9b8c-8f583a930879","UserID":"TestUser1604887"}
```

## # Save a few more and retrieve some, randomly, from the set of saved IDs

```
for i in range (0,5):
    RU=requests.get('https://api.testdataservices.com.au/v0001F_GetSavedUserID?SubscriptionKey='+SubKey)
    FetchIDj=RU.json()
    FetchUID=FetchIDj['PersonUID']
    RP=requests.get('https://api.testdataservices.com.au/v0001F_GetPerson?UID=' + FetchUID);
    RP.text
'Saved {"PersonUID":"7967d517-b615-49f9-b94c-d93alac9b378","UserID":"TestUser1662389"}'
'Saved {"PersonUID":"eab3a302-0d09-4238-bb0d-f73cbd7a70ea","UserID":"TestUser1261822"}'
'Saved {"PersonUID":"0e41b48e-708e-46e7-8cf2-3d0f0b56754a","UserID":"TestUser739719"}'
'Saved {"PersonUID":"3bd09b84-4652-48b2-88ae-61fa4f49c723","UserID":"TestUser77027"}'
'Saved {"PersonUID":"albae41b-dcef-413c-a1c4-a0942fd8e267","UserID":"TestUser1567065"}'
```

## # Randomly Retrieve 5 UserIDs from the saved set, displaying email and password used for each of those users.

```
for i in range (0,5):

    RU=requests.get('https://api.testdataservices.com.au/v0001F_GetSavedUserID?SubscriptionKey='+SubKey)

    FetchIDj=RU.json()

    FetchUID=FetchIDj['PersonUID']

    UserID=FetchIDj['UserID']

    RP=requests.get('https://api.testdataservices.com.au/v0001F_GetPerson?UID=' + FetchUID);

    RPJ=RP.json()

    Email=RPJ['email']

    Password=RPJ['Password2']

    'Retrieved User ' + UserID + ' with email of ' + Email + ' and password of ' + Password

'Retrieved User TestUser2449057 with email of Christopher.Pontague.2199@testdata.email and password of f]H0G]H6RR9^'
'Retrieved User TestUser77027 with email of Sergio.Wood.7024@testdata.email and password of t<CVV@X178J]'
'Retrieved User TestUser1307763 with email of Patrick.Carroll.4184@testdata.email and password of d$TH8$F64BB^'
'Retrieved User TestUser2198299 with email of Lisa.Winch.1493@testdata.email and password of t@38N(AANP6%'
'Retrieved User TestUser1307763 with email of Patrick.Carroll.4184@testdata.email and password of d$TH8$F64BB^'

>>> for i in range (0,5):
    RU=requests.get('https://api.testdataservices.com.au/v0001F_GetSavedUserID?SubscriptionKey='+SubKey)
    FetchIDj=RU.json()
    FetchUID=FetchIDj['PersonUID']
    UserID=FetchIDj['UserID']
    RP=requests.get('https://api.testdataservices.com.au/v0001F_GetPerson?UID=' + FetchUID);
    RPJ=RP.json()
    Email=RPJ['email']
    Password=RPJ['Password2']
    'Retrieved User ' + UserID + ' with email of ' + Email + ' and password of ' + Password

'Retrieved User TestUser2449057 with email of Christopher.Pontague.2199@testdata.email and password of f]H0G]H6RR9^'
'Retrieved User TestUser77027 with email of Sergio.Wood.7024@testdata.email and password of t<CVV@X178J]'
'Retrieved User TestUser1307763 with email of Patrick.Carroll.4184@testdata.email and password of d$TH8$F64BB^'
'Retrieved User TestUser2198299 with email of Lisa.Winch.1493@testdata.email and password of t@38N(AANP6%'
'Retrieved User TestUser1307763 with email of Patrick.Carroll.4184@testdata.email and password of d$TH8$F64BB^'
>>>
```