3325 NZRIS (New Zealand Research Information System)  
Data Provider REST API Specification

Version: v2.05

Date: 8/6/2020

# Document control

## Version history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description of change |
| 11/12/18 | 0.1 | Ruan Malan | First draft started |
| 21/12/18 | 0.2 | Ruan Malan | Vendor feedback incorporated.  First draft provided for initial NZRIS data provider review. |
| 4/02/219 | 0.3 | Ruan Malan | Name change to NZRIS  Added all attributes for all entities from data specification  Reduction in suppression operations  Added some sample values  Spell checked. |
| 7/02/19 | 0.3.4 | Ruan Malan | Small alignments fixed with data specification  Person Academic Record Qualification description  Asset pool was missing Supplier Organisation Type and Supplier Organisation Industry  Fixed an incorrect single digit sample NZBN value  Mandatory protection requirements on ApplicationReview and Person  In ApplicationRequestedResource made optional maximumResourceQuantity and resourceValue |
| 15/02/19 | 1.0 | Ian Baker | Completed edits for v1.0 FOR REVIEW public release. |
| 16/04/19 | 2.0 | Ian Baker | v2.0 release containing changes related to the v2.0 NZRIS Data Specification.  Please note versioning change to keep API release versions in sync with major releases of the Data Specification. |
| 14/06/19 | 2.01 | Ian Baker | V2.01 release containing changes and revisions post internal reviews. |
| 16/07/19 | 2.02 | Ian Baker | V2.02 release – fixes and changes for the changes made to the V2 Data Specification – particularly relating to the submission entity and related methods. |
| 17/12/2019 | 2.03 | Melissa Fordyce | V2.03 release – revised swagger (in JSON and YAML) inserted into document, following custodian application development |
| 8/6/2020 | 2.05 | Melissa Fordyce | V2.05 release – revised swagger (in JSON and YAML) inserted into document, following custodian application development.  Note version 2.04 is included in this version. |

The changes made between v2.03 and v2.05 are as follows:

* Base path for API updated from “/services” to “/services/nzris/v1”
* URL updated from “/submissions/{submissionId}” to “/SubmitData”
* Mandatory headers added – Content-Type, UserKey, NZRIS-org-nz-Request-Id
* Success response code updated from 200 to 202
* Macrons added to Vision Mātauranga labels in Award Granted and Project entities.

Contents

[1. Document control 2](#_Toc1132413)

[1.1 Version history 2](#_Toc1132414)

[1.2 Figures and tables 4](#_Toc1132415)

[2. Introduction 5](#_Toc1132416)

[2.1 Purpose 5](#_Toc1132417)

[2.2 API definition 5](#_Toc1132418)

[2.3 Timeline 5](#_Toc1132419)

[3. Terms and definitions 7](#_Toc1132420)

[4. Getting started 8](#_Toc1132421)

[4.1 REST primer 8](#_Toc1132422)

[4.2 Introduction to the NZRIS data provider API Structure 8](#_Toc1132423)

[4.3 Introduction to IDs and namespaces 10](#_Toc1132424)

[4.4 API publishing 10](#_Toc1132425)

[4.4.1 Prototype API 11](#_Toc1132426)

[4.4.4 Production Website 11](#_Toc1132429)

[4.5 API versioning scheme 11](#_Toc1132431)

[4.5.1 Version updates and management 11](#_Toc1132432)

[4.6 First tranche API Authentication 11](#_Toc1132433)

[4.6.1 Machine to machine trust 11](#_Toc1132434)

[4.6.2 Website authentication 12](#_Toc1132435)

[4.6.3 Federated identity 12](#_Toc1132436)

[4.7 How to sign up to the NZRIS data provider API 12](#_Toc1132438)

[5. Technical features of the NZRIS Data provider API 13](#_Toc1132439)

[5.1.1 HTTP headers 13](#_Toc1132440)

[5.1.2 HTTP status codes 15](#_Toc1132441)

[5.1.3 API characteristics 18](#_Toc1132442)

[6. API overview 20](#_Toc1132443)

[6.1 Data Provider API Inventory 20](#_Toc1132444)

[6.2 API Authorisation 24](#_Toc1132445)

[6.3 API – Detailed specification 25](#_Toc1132446)

[7. Data Definition 26](#_Toc1132447)

[7.1 Logical data model 26](#_Toc1132448)

[7.2 API resource model 27](#_Toc1132449)

[7.3 Sample payloads 28](#_Toc1132450)

[8. Tools, Support and Resources 29](#_Toc1132451)

[8.1 Test harnesses 29](#_Toc1132452)

[8.2 URLs 29](#_Toc1132453)

[Appendix A – Detailed API swagger definition Change history 30](#_Toc1132454)

[V0.3 change log 30](#_Toc1132455)

[Breaking changes 30](#_Toc1132456)

[Non-breaking changes 31](#_Toc1132457)

[Description only updates (no operation or field change) 31](#_Toc1132458)

[V0.2 change log 31](#_Toc1132459)

## Figures and tables

[Figure 1: Simplified RSI data model 26](#_Toc441543)

[Figure 2: Permission Model 27](#_Toc441544)

# Introduction

## Purpose

The target audience of this document is anyone who writes software that integrates with the NZRIS data provider Application Programming Interfaces (APIs) in order to provide data into NZRIS.

The purpose is to:

* Provide a specification (interface behaviours and payloads) for the new NZRIS system Data Provider Application Programming Interface (API),

This draft is intended to elicit feedback from the initial NZRIS data providers.

This document does **NOT** cover the NZRIS public query API, which has different security profile and interaction dynamics and delivery timing considerations. For consistency, the same underlying entity structures will be used where viable however some changes may be required.

Chapters 2-4 are intended for a broader audience and the chapters thereafter are targeted at technical audience.

## API definition

The actual API specification format used is Open Application Interface Specification (OAIS v2.0) :





Apart from browsing in environments (see “**4.4 API publishing”**), this definition is the easiest and most accurate way to review the API content, and can be imported at <http://editor.swagger.io/>

This document explains the approach to the API but does not attempt to reproduce all details in the API definition (as this would be verbose).

The Swagger definition can be auto converted to other format like JSON schema and RAML or WADL as required, or downloaded from the environments.

Please see section **“Appendix A – Detailed API swagger definition Change history”** for information on the maturity of this specification and its change history

## Timeline

Following the v1.0 release in February 2019 which was intended to garner feedback from integration partners and vendors, we released v2.0 to reflect both that feedback and necessary changes made to the v2.0 NZRIS Data Specification.

Version 2.0 was updated further to v2.01 to represent minor changes following the MBIE internal review process.

The specification was updated to version 2.03 in December 2019, and 2.05 in June 2020. Both updates were made following the development of the NZRIS system infrastructure, to ensure that this operates effectively.

Assuming no major changes, v2.05 version will act as the stable version for the NZRIS public release.

Thereafter the only changes likely are from issues discovered during the design/build phase, to reflect the implementation design decisions, to fix defects or changes to match solution/product capabilities.

Such changes will be applied only as necessary and version controlled.

# Terms and definitions

The following table contains the glossary of the common terms, definition and acronyms used in this document.

| Term | Definition |
| --- | --- |
| API | Application Programming Interface |
| DTD | Document Type Definition |
| ETag | Entity Tag – used for caching and concurrency control in HTTP |
| HTTP | Hyper Text Transfer Protocol |
| JSON | JavaScript Object Notation |
| TLS | Transport level Security, successor of SSL, HTTPS connections are encrypted using TLS. Always involves a server side certificate to validate the identity of the server |
| TLS Mutual Authentication | When both parties in a HTTPS connection are verified using certificates and connections are only allowed with trusted parties on both sides then mutual TLS trust is established. A client side TLS certificate is used in addition to the server side one any TLS connection always requires. |
| OAuth 2.0 | An authorisation delegation framework that allows an API consumer to transact on behalf of a user without having to know the credentials or identity of the user. The OAuth2.0 framework is described by IETF internet draft: <https://tools.ietf.org/html/draft-ietf-oauth-v2-31> |
| REST | Representative State Transfer |
| SOAP | Simple Object Access Protocol |
| WSDL | Web Service Definition Language |
| XML | eXtensible Markup Language |
| XSD | XML Schema Definition |
| Open Application Interface Specification v 2.0  “Swagger” | API specification format.  Called Open Application Interface v 2.0, initially called swagger  Can be auto converted to other format like JSON schema and RAML as required |

# Getting started

## REST primer

RESTful services are a lightweight HTTP based API using existing HTTP verbs and structure. The application of HTTP verbs on REST services is as follows:

|  |  |  |
| --- | --- | --- |
| **Verb** | **Usage** | **Additional Information** |
| GET | Read | Will retrieve one or more (collection) of requested items |
| POST | Create | In REST creates the given record and often idempotent in REST (repetition will have no side effects and same result)  In this API this operation is not idempotent at submission level, since it starts a workflow process.  At more detailed RSI element level a generic PUT is used that will replace elements or create the as needed, to remove the need to differentiate between POST and CREATE  In REST custom commands are often created using POST. This API uses POST for custom commands. |
| PUT | Update | Will update the entire resource, not partial. |
| DELETE | Delete | Will only succeed once to remove the resource by ID. |
| PATCH | Update | Will to a partial update in REST APIs. Not as widely consistently used as the other REST verbs.  For this API other verbs are used to achieve the same result. |

The URL format used to execute a RESTful service conforms to the following syntax:

/services/{version}/{namespace}/{resource}/{resource-id}/{sub-resource}/{sub-resource-id}

All payloads (request and response) will be in JSON format. If an API Consumer requires another payload format / structure (e.g. XML) then it is up to the consumer application to convert from JSON as appropriate.

The API schema is described using OpenAPI Specification 2.0 (Swagger).

The final NZRIS API Domain name and domain URLs will be advised once the test environments have been set up and made available for use.

## Introduction to the NZRIS data provider API Structure

(For a more detailed list of all the functionality and operations see section **6.1 Data Provider** API Inventory)

The API allows submission of data to NZRIS and provides operations to:

* Check data provider setup and connectivity and security credentials.
* Create and update submissions of NZRIS data to the NZRIS sensitive data collection and NZRIS public collection and website (critical API content).
* Read related reference data such as codesets – (delivery might need to be delayed in favour of critical API content).
* Review suppression of records.

The API allows batch submission of data records for NZRIS usage. Such a batched collection of records is called a submission and is accompanied with an approval workflow. In the REST API the collection is also called a submission. All data records are provided or updated in the context of a submission which has its own processing workflow.

|  |  |  |
| --- | --- | --- |
| Collection | Entity | Actions: |
| Submission | Refer to the [NZRIS conceptual framework](https://www.mbie.govt.nz/science-and-technology/science-and-innovation/research-and-data/nris/) and the [NZRIS data specification](https://www.mbie.govt.nz/science-and-technology/science-and-innovation/research-and-data/nris/) for detail on the entity types. | **Created**: POST[[1]](#footnote-1) /submissions/{submissionId} – provide ID  Or POST /submissions/ - ID will be assigned  **Read**: GET /submissions/{submissionId}  **Updated** in entirety: PUT /submissions/{submissionId}  **Updated** with merge: PUT with …/elementsreplace/ URL  **Abort** submission:  DELETE /submissions/{submissionId}  **Reclassify as deleted** all data in submission, even if previously released. DELETE /submissions/{submissionId}/deleteall  Note: PATCH formats not uniformly widely supported. |
|  | Any one of:  Application  Application Decision  Application Review  Asset Pool  (Asset Pool) Planned Distribution  Award Granted  Award Received  Resource Distributed  Resource Received  Project  Output  Organisation  Group  Person | **Updated** in entirety or **New Record** Created/added – added to collection:  PUT with …/elementsreplace/… URL  **Read** in bulk as above only, no single get due to multiplicity and OAS2 lack of support for polymorphism  **Removed** with:  DELETE /submissions/{submissionId}/element/{type}/ |

Any changes inside a submission can only be applied if that submission is not yet approved or closed, apart from reclassifying all data therein as deleted.

Subsequent data changes need to be part of a new submission.

Each entity resource is a unit record as described in the [NZRIS conceptual framework](https://www.mbie.govt.nz/info-services/science-innovation/research-and-data/nris/document-image-library/NRIS-conceptual-framework.pdf), which means it is the view of that entity from the perspective of the data provider, including the local ID used by that organisation to refer to that record in their system.

The data inside each of the entity resource types are described in the [NZRIS data specification](https://www.mbie.govt.nz/info-services/science-innovation/research-and-data/nris/document-image-library/nris-data-specification.pdf).

For a more detailed list of all the functionality and operations see section **6.1 Data Provider** API Inventory

## Introduction to IDs and namespaces

**Discussion on options to approach**

In keeping with JSON best practices as also applied by MBIE the resource structure is kept flat to two levels. There are logically five levels (data provider, submission, data model entity type, data model entity, subsection and protection patterns). These are combined so the first level handles data provider and submission, and the second level handles data model type and entity and everything below it.

This interface does not follow common REST practices of using long generated GUIDs to identify records, but relies on the central identifiers specified in the data specification.

|  |
| --- |
| Technical discussion |
| Some reasons for not using system generated GUIDs include:   * The system, to some extent, is not a pure REST resource collection, but reflects exchange transactions and workflow, and system state replication. * Using the data provider IDs is easy to understand in the context of the data specification which already has wide acceptance. Those will be used for diverse purposes including linking and feedback anyway, adding additional IDs add complexity. * Using system GUIDS will create a lot of additional overhead on both sides in translating, and add no value since the records are always in the context of a submission and the data provider, therefore record IDs don’t need to be globally unique in isolation. * Using GUIDs would shift the emphasis towards a central curated collection rather than a synchronised copy of the data provider record collections. * Inside the database storage unique internal Technical keys are used already, they are necessary but different from such GUIDs at API level |

Data providers are each provided a unique recognisable “DataProviderID” by the data supply agreements, and described in the data specification. These are easy and practical to use as IDs in this context.

A submission could have an ID assigned by the data provider or NZRIS will generate a short sequential one.

e.g.

provider-api.NZRIS.org.nz/services/v0.1/data-provider/submissions/MBIE.S5

The primary IDs used for entities are those from the data provider organisation’s local systems, referred to as Local IDs in the NZRIS data specification. Global persistent IDs are always typed. To use these mechanisms in a very flat JSON structure the data provider ID or global ID type is often used as a prefixed namespace. Due to REST APIs using URLs and colon being a reserved character with specific meaning a full stop/dot (“.”) is used between the namespace prefix and the ID e.g.

e.g. Example where MBIE as asset pool manager/public research funder is the data provider:

/submissions/MBIE.S12345 (points to submission)

## API publishing

During development of the solution, two versions of the new API will be published.

### Prototype API

A prototypal full definition of some planned go-live operations exists to match those attached to this document.

<https://NRIS-dev.portal.azure-api.net/> (note no Z for name change yet)

Please navigate to the API documentation link, and the on the top left sort by tags (icon looks like staggered tenga blocks) to get logical groupings. Limited capacity for registrations and calls is currently in place.

Sandbox API

“Sandbox” is an alternate term for the EAT environment (external acceptance test).

A working version called “sandbox” with a “0.x” version numbering scheme is planned and will be made available mid-2019.

Once available it will provide a documentation tab for details on the scope of operations currently supported, version implemented and how to subscribe and test.

### Production Website

The production website with ability to upload data will support the same JSON format described in this document as well as spreadsheet, using multi factor login. It will focus on allowing the submission POST (create) and submission PUT (update) by supplying the entire submission dataset as a file.

Production API

The production system API will be secured at machine to machine level.

## API versioning scheme

### Version updates and management

The major version number will be incremented where breaking changes are introduced.

Up to the go live candidate release 0.n is considered a major version, thereafter n.0.

Major versions will be updated alongside the relevant Data Specification version in order to keep these in Sync – i.e. all v2.x versions will relate to the latest v2.0 version of the released Data Specification.

## First tranche API Authentication

### Machine to machine trust

Initial data provider access will be based on machine-to-machine trust due to condensed timelines and de-scoping of federated single sign on from the initial release.

Mutual TLS certificate trust and firewalling will be applied for the four public funder / asset pool manager organisations initially providing data to use via the API.

### Website authentication

Manual upload of spreadsheet will be supported on a secure website. This will initially rely on multi factor authentication and local logins managed by the central NZRIS custodians on behalf of the data providers.

### Federated identity

As the system must be able to carry sensitive/restricted information, and consumers will have access to such data, the preference is to use identities that expire when people leave job roles.

Users will be added as guests to the Azure Tenancy used by NZRIS from their organisational Microsoft cloud accounts where possible.

In this sector that is likely to translate into the [Tuakiri federation](https://www.parliament.nz/resource/0000196166) (for research collaboration) for most research organisations, Active Directory Federation services for most public funders, and possible RealMe or Tuakiri virtual homes for the remainder. However, this is out of scope for the initial project.

(To complete section Ian might want to change this to reduce it since the Part A doesn’t cover it yet)

Potential API Authentication Via OAuth2.0

Future versions of NZRIS might utilise OAuth 2.0 as part of a federated identity strategy

## How to sign up to the NZRIS data provider API

When NZRIS provides the integration testing & production versions of the API, a signup process will be made available.

The NZRIS getting started webpage will provide the basic steps in the process for subscribing to and then using the APIs

Contact NZRIS for organisational considerations.

# Technical features of the NZRIS Data provider API

This section outlines technical features of the API

**Note: This has not had implementation vendor review and might change, initial feedback on compatibility and feature concerns from funders are meanwhile appreciated.**

### HTTP headers

The following headers are supported in the header for requests or responses (as indicated). Where a header is supported on a particular operation, this is indicated as mandatory (M) or optional (O) or mandatory on a set of conditions (C - refer purpose for explanation). N/A value means not supported (and will be ignored if provided). Examples per API will include these where applicable:

| **HTTP Request or Response** | **HTTP Header** | **Data type** | **Purpose** | **Example** | **HTTP Verbs** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GET** | **POST** | **PUT** | **DELETE** |
| Request | NZRIS-org-nz-Request-Id | String | API Consumer generated unique transaction ID  Provides NZRIS and the API consumer with a “tracking identifier”, which is useful to trace and troubleshoot transactions that have failed  This is also used on a POST operation to prevent a transaction from accidentally being replayed | NZRIS-org-nz-Request-Id: C65A1C5A-8F74-4E0B-BA06-16CB47358D85 | O | O | O | O |
| Request | Authorization | String | Contains the OAuth token in the format “Authorization: Bearer <token>” | Authorization: Bearer 456789ab-fedc-2345-6543-abcfeda543987 | M | M | M | M |
| Request | Accept | String | The NZRIS data provider API will only return responses in JSON format  If no Accept header is provided, JSON will be returned anyway  If any other format is requested, the PPSR API will return a “406: Not acceptable” response | Accept: application/json | O | O | O | O |
| Request | Content-Type | String | The New PPSR API will only accept JSON requests and will return responses in JSON format  If no Content-type header is requested, JSON will be returned anyway  If any other format is requested, the PPSR API will return a “406: Not acceptable” response | Content-type: application/json | N/A | M | M | N/A |
| Request | Host | String | Domain name of the NZRIS API gateway hosting the NZRIS data provider API | Host: api.NZRIS.org,nz | M | M | M | M |
| Response | Date | Date | Date/time the response was generated (in Greenwich mean time) | Date: Wed, 18 Nov 2015 21:14:52 GMT | M | M | M | M |
| Response | Service-Version | Version number | Confirm which version (major number only) of the API the response is compliant with | Service-Version: v1 | M | M | M | M |
| Response | Last-Modified | Date | Date/time the returned resource was last updated (in Greenwich mean time) | Last-Modified: Wed, 18 Nov 2015 21:14:52 GMT | M | M | M | M |
| Response | Content-Location | String | Contains a URI of the resource (not provided when a collection is returned) | Content-Location: /services/v1/companies-office/ppsr/financing-statements/FH04R729A8ND9224 | C | M | M | M |
| Response | Content-Type | String constant | The structure of the response payload (where one is returned). The New API will always return JSON | Content-Type: application/json | M | M | M | M |
| Response | NZRIS-org-nz-Request-Id | String | As described in the Request section above. Only if supplied by consumer in the request. The same ID is returned to the consumer so that the response can be paired with the request sent (confirmation) | NZRIS-org-nz-Request-Id: C65A1C5A-8F74-4E0B-BA06-16CB47358D85 | C | C | C | C |
| Response | NZRIS-org-nz-Correlation-Id | String | NZRIS generated unique ID for the response. | NZRIS-org-nz-Correlation-Id: 514dbf50-4475-4e4d-86ee-e2001f158026 | O | M | M | M |
| Response | Cache-Control | String | For reference data API only. Indicates how long the response can be used / relied upon | Cache-Control: private,max-age=14400,must-revalidate | C | N/A | N/A | N/A |
| Response | Cache-Control | String constant | For all transactional APIs (excluding reference data), the response should not be cached. Consumers should always GET a fresh copy of the expected resource (confirming ETag) before any PUT operation | Cache-Control: No-cache | C | M | M | N/A |

### HTTP status codes

The following table defines the status codes returned and mapping to methods:

| **Status Code** | **Success / Error** | **Meaning** | HTTP response example | **HTTP Verbs** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GET** | **POST** | **PUT** | **DELETE** |
| 200 Ok | Success | Retrieval operation succeeded. Note: no results may be found / returned and this is still deemed successful |  | X |  |  |  |
| 201 Created | Success | Resource / sub-resource successfully created.  Also used on status change / “action” operations (such as discharging a financing statement) |  |  | X |  |  |
| 204 No Content | Success | Resource / sub-resource successfully updated or removed. |  |  |  | X | X |
| 304 Not modified | Success | The requesting client has already received the latest version of the requested resource. Thus, the body of the response message must be empty. This status code is returned as a result of a conditional GETwhere the the specified conditions (i.e. If-Non-Match, IfModified-Since) are not met |  | X |  |  |  |
| 400 Bad Request | Error | Any request that fails validation due to:   * an invalid JSON representation (ie. badly formed) * wrong data type / maxlength exceeded etc. * properties supplied that are not defined in the swagger |  | X | X | X | X |
| 401 Unauthorised | Error | When authentication is required and has failed or has not yet been provided | HTTP/1.1 401 Unauthorized  Content-Type: application/json  {  "errorMessage": "Not authenticated",  "errorDescription": "The requester is not authenticated",  "errorCode": "TBC"  } | X | X | X | X |
| 403 Forbidden | Error | User has authenticated but does not have permission / authority to access or update resource | HTTP/1.1 403 Forbidden  Content-Type: application/json  {  "errorMessage": "Not authorised",  "errorDescription": " The requester is not authorised for the specified <resource>: <resource ID>",  "errorCode": "TBC"  } | X | X | X | X |
| 404 Not Found | Error | When the specified resource does not exist. E.g. invalid identifier | HTTP/1.1 404 Not Found  Content-Type: application/json  {  "errorMessage": "<resource> not found",  "errorDescription": "The specified <resource>: <resource ID> does not exist",  "errorCode": "TBC"  } | X |  | X | X |
| 406 Not Acceptable | Error | The requested media type is not supported. For example, a GET request wants to retrieve an entity in a media type (specified as value of the Accept header of the request) not supported by the server | HTTP/1.1 406 Not Acceptable  Content-Type: application/json  {  "errorMessage": "<media type> not supported",  "errorDescription": "The specified <media type> cannot be returned. Only application/json is supported",  "errorCode": "TBC"  } |  |  |  |  |
| 412 Precondition Failed | Error | Concurrency control failure. Either the If-Match parameter was expected (and missing) or provided and the value did not match the stored (ETag) against the resource being updated. Update failed | HTTP/1.1 412 Precondition Failed  Content-Type: application/json  {  "errorMessage": "<resource> could not be updated / deleted",  "errorDescription": "The specified <resource>: <resource ID> ETag differs from the If-Match provided (or If-Match expected and missing)",  "errorCode": "TBC"  } |  | X | X | X |

### API characteristics

The following table outlines characteristics / behaviours common to all New APIs:

| **API Characteristic** | **Value** | **Comments** |
| --- | --- | --- |
| *Mediation style* | Synchronous | Immediate response. Indicates whether the request has been processed (retrieve, create or update operation) |
| *Message protocol* | RESTful | RESTful method definitions are used as follows for HTTP verbs:   * POST = create * PUT = update   GET = retrieve / query |
| *Message protection* | No additional payload encryption and Signing | There is no additional message payload security protection applied |
| *Message Format* | JSON | Internal structure of request / response payload – schema |
| *Transport* | HTTPS via TLS1.2 | To protect OAuth2.0 tokens and payload data in transit, all API calls will be made over HTTPS |
| *Compensation / Rollback* | None | There is no compensation (rollback) capability. An API either succeeds or fails (in its entirety) as indicated by the HTTP status code and an appropriate error code and message, with URI to the element(s) in error |
| *Retry (duplicate submission) safe* | Yes | Query / retrieve operations are by definition retry safe  For POST operations, If the same consumer provided request ID has already been processed, the transaction will be rejected with an appropriate error response.  PUT operations do not require a request ID since all PUT operations replace / update the entire resource (partial updates are not permitted), thus ensuring integrity |
| *Guaranteed delivery* | No | Since all APIs are synchronous, if an API returns an HTTP/201 or HTTP/ 204 that confirms any POST or PUT has succeeded and been persisted in the NZRIS database as a submission to be processed. |
| *Message sequencing (guaranteed delivery order)* | No | Transactions sent in quick succession may be processed in any order |
| *Data classification* | Sensitive/Restricted | NZRIS treats all data provided by the API consumer in both request, response and error payloads as SENSITIVE/RESTRICTED data classification |
| *Callback interface* | None | No callbacks. All operations are atomic and synchronous |
| *Special characters* | Backslash escape | To submit data in requests (or interpret responses) with special characters, these will need to be escaped as per JSON standards (leading backslash \ character). Examples are:   * \" * \\ * \/ * \b * \f * \n * \r * \t   \u followed by four-hex-digits |
| *Character Case Sensitivity* | None | All search operations ignore case and can be provided upper, lower or mixed case.  Note: Exact matches are performed for searches, unless the property supports wildcard. No phonetic / fuzzy matching is performed. |
| *Character Case Preservation* | As supplied | Unless a reference data item (where a code is supplied and description is obtained from NZRIS reference data) data supplied by the API consumer will be preserved as supplied |
| *Data usage* | Read (GET)  Update (POST / PUT / DELETE) | As per RESTful standards:   * GET and DELETE operations are parameterised via PATH and/or QueryString   PUT and POST operations are parameterised via PATH and request body |
| *Correlation* | No | None of the APIs return a correlation ID that is required to be supplied on subsequent requests.  A submission ID and version will be returned that can be used for queries.  The submission ID will be used as input if viable otherwise allocated.  The submission version will start at 1 and increment, if a higher number is input in a POST/PUT it will be used. |
| *Message Splitting* | No | All API requests are self-contained in a single API call and do not require multi-part invocation. Any limitations on response sizes are in part controlled by the page size requested and also server side limits imposed |
| *Paging* | GET operations that return a *collection* resource - NA | Collection resources are not retrievable but can be viewed via the BI reporting interface provided.  Submissions will be retrieved in entirety without pagination |

# API overview

## Data Provider API Inventory

These are NOT registered for separately, when registering during on-boarding as an NZRIS data provider, access to all the parts of the API(s) will be granted at the same time.

The following table lists all parts of this API by function.

All API operations can be invoked using an identity associated with the organisation. This can be

* machine-to-machine trust using mutual TLS and firewalling for the API
* and for the website NZRIS specific logins with multi factor authentication.

To create or update a submission with a data provider ID, or any record inside it, the identity needs to link back to the record for that data provider

Proposed de-scoping and future scoping is shown as deprecated (greyed out) below.

Details of operations are available in the environments and the swagger definition (import into editor.swagger.io for a n expandable version of the below)

|  |  |
| --- | --- |
| **Category** | Applicable outside an open submission workflow |
|  | See sections below |
|  | Yes |
| **Exlcuded:** | Only POST and GET and /deleteall |
| Excluded: | No |
|  | Yes |
|  | Yes |

## API Authorisation

The following table shows which groups of APIs (from the API table above) have authority restrictions by the role of the user

| **API Group** | **Authorised for users with “Custodian” role?** | **Authorised for users with “user” role?** | **Machine to machine trust with mutual TLS and firewalling** |
| --- | --- | --- | --- |
| All APIs and operations that create modify submissions or suppressions | Y | N | Y |
| All APIs and operations that view reference data or organisation data or submissions or suppressions | Y | Y | Y |

The user and their specific relationship with a customer organisation is maintained by NZRIS custodians and reflected by the data provider record, see the /dataprovider/{dataproviderID}/ API path

## API – Detailed specification

Please see the OpenAPI specification / swagger definition in section 2

# Data Definition

This section provides an overview of the payload structures and attribute definitions used across all new APIs.

## Logical data model

Each entity resource is a unit record as described in the [**NZRIS conceptual framework**](https://www.mbie.govt.nz/science-and-technology/science-and-innovation/research-and-data/nris/), which means it is the view of that entity from the perspective of the data provider, including the local ID used by that organisation to refer to that record in their system.

The data inside each of the entity resource types are described in the [**NZRIS data specification**](https://www.mbie.govt.nz/science-and-technology/science-and-innovation/research-and-data/nris/).

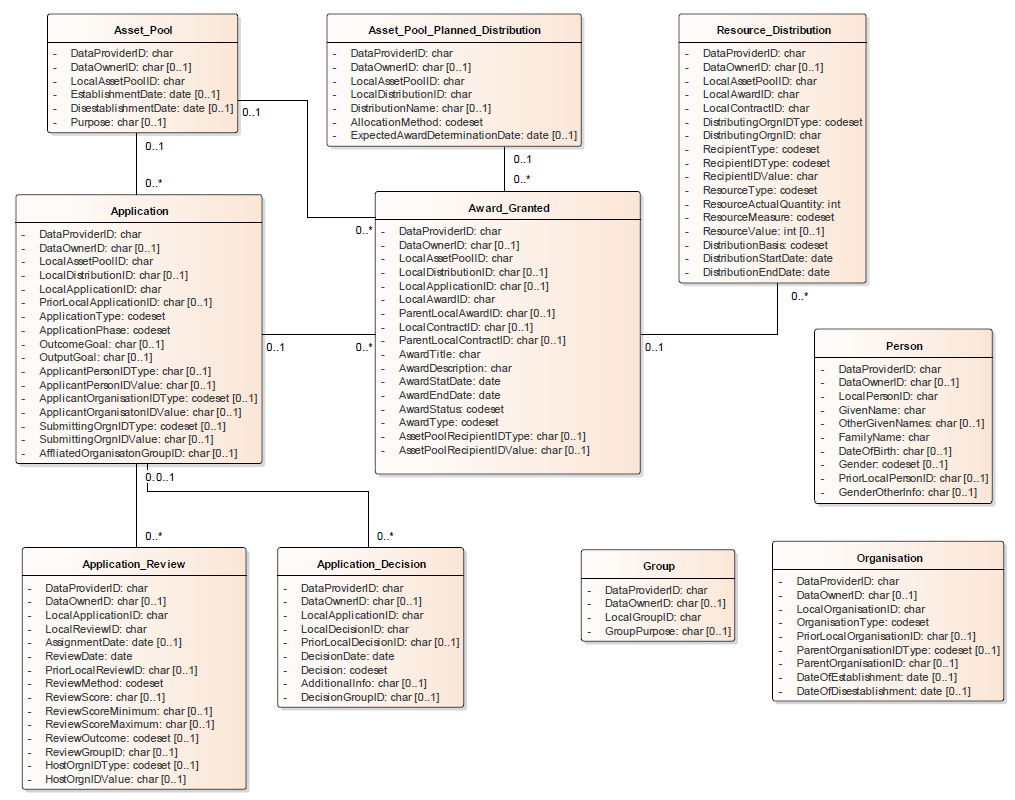
The logical data model for NZRIS is shown below. Also shown are the different authentication and authority entities. This model forms the basis of “resource” definition within the NZRIS data provider API.****

Figure 1: Simplified RSI data model

## API resource model

The following diagram shows the composition of each “top level” resource. A top level resource is one used in a request or response payload.



Figure 2: Permission Model

A data provider is registered and on-boarded and given an NZRIS ID.

All data provided to NZRIS is done via submissions, which batch the entity records

The entities block on the top right are any and all from the previous diagram, entity type is any one of

* Applications
* ApplicationDecisions
* ApplicationReviews
* AssetPools
* PlannedDistributions (Asset Pools Planned Distributions)
* AwardsGranted
* AwardsReceived
* ResourcesDistributed
* ResourcesReceived
* Projects
* Outputs
* Organisations
* Groups
* Persons

For any entity record supplied protection requirements need to be defined.

A data provider has authorised accounts that can access its submissions.

## Sample payloads

Samples can be generated in a Swagger UI. The below does not add much additional data to that. The preference will be to include meaningful sample data inside the swagger definition.

The following table contains sample payloads for the various resources included in API requests or responses documented above:

| **Resource** | **Sample payload** |
| --- | --- |
| Submission POST Request |  |

# Tools, Support and Resources

## Test harnesses

To be added when sandbox API is developed.

## URLs

The following table lists all URLs referred to in this document in a single convenient location

|  |  |  |
| --- | --- | --- |
| **Summary** | **Description** | **Link** |
| Prototype API | Interactive location to browse structure of API | <https://nris-dev.portal.azure-api.net/> |
| Swagger editor | An on-line Swagger definition editor. Can be used to view the attached prototype | <http://editor.swagger.io/> |

# Appendix A – Detailed API swagger definition Change history

The changes are grouped into breaking change, non-breaking changes and description only (clarification) updates.

## V0.3 change log

**Status**

| **Aspect** | **Completeness** |
| --- | --- |
| Approach | A full proposal on the way to document |
| Structure | A full proposal on the supported operations, use of IDs, the structure of the data in and out  Future operations not supported due to lack of urgency are removed  Future operation and data formats not supported due to compatibility challenges are mostly removed |
| Data Model | All RSI entities defined in data specification are fully fleshed out (Facility and Infrastructure not defined there and not included here). |
| Security | An overview of the initial security approach and future direction of thinking |
| Environments | A high level overview of possible direction |
| URLs | Indicative only, prototype environment URL added |
| Error handling and response types | Indicative only, not complete or consistent yet. |
| Technical viability confirmation | Has not been fully vendor reviewed, will lead to some changes downstream during construction |

(Drive to conclusion for public release and either update or remove)

### Breaking changes

The following changes were made in v0.3.n that are not backwardly compatible with v0.2 (published on 21/12/18). Fields have either been renamed, moved or data types changed:

| **Operation ID or definition (object name)** | **Change # and description of breaking change** |
| --- | --- |
| All operations carrying submission or RSI data | Full data specification added, all attributes |
| NA/path to all | Name change to NZRIS |
| Suppression operations l | Suppression operations reduced |
| Definitions of Submission and Protection… | Updated |
| All operations carrying submission or RSI data | 3.4) Mandatory protection requirements on ApplicationReview and Person  Asset pool was missing Supplier Organisation Type and Supplier Organisation Industry  In ApplicationRequestedResource made optional maximumResourceQuantity and resourceValue |

### Non-breaking changes

The following changes are backwardly compatible with v0.2 of the API.

| **Operation ID or definition (object name)** | **Change # and description of non-breaking change** |
| --- | --- |
| All | Sample values added |
| All operations carrying submission or RSI data | 3.4)  Fixed an incorrect single digit sample NZBN value |
|  |  |

### Description only updates (no operation or field change)

These changes only update the annotation / description elements / metadata in the swagger specification and do not change any fields.

|  |  |
| --- | --- |
| **Operation ID or definition (object name)** | **Change log # and description of annotation / clarification** |
|  | 3.4) Spell checked |
|  | 3.4) Person Academic Record Qualification description |

## V0.2 change log

First shared draft, summary of completeness and maturity of proposal:

| **Aspect** | **Completeness** |
| --- | --- |
| Approach | A full proposal on the way to document  Still contains many notes on design decisions and sections with options summarized on some decisions. |
| Structure | A full proposal on the supported operations, use of IDs, the structure of the data in and out  Future operations not supported due to lack of urgency are shown as deprecated  Future operation and data formats not supported due to compatibility challenges are shown as deprecated |
| Data Model | Person is almost fully fleshed out (only some subtypes missing).  All entities are created, pending feedback, they will be fleshed out equivalently from the data specification |
| Security | An overview proposal on the initial security approach and future direction of thinking |
| Environments | A high level overview of possible direction |
| URLs | Indicative only, prototype environment URL added |
| Error handling and response types | Indicative only, not complete or consistent yet. |
| Technical viability confirmation | Has not been fully vendor reviewed, will lead to some changes downstream during construction |

1. Note, since it is transactional in starting off a workflow it is therefore not fully idempotent, even though the data provider provides the ID [↑](#footnote-ref-1)