ICT Project Guidance

Discovery:   
Technical - Service Domain Naming

Version:

0.3

Author:

Sky Sigal, Solution Architect

## Description

This document describes aspects to consider when developing a naming strategy for services offered via different channels, supported by different resources.

## Synopsis

A Service is provided via multiple channels, in different formats, supported by different resources. These elements must be discoverable with the least friction by following an common organisation pattern.

## Contents

[Description 1](#_Toc150609639)

[Synopsis 1](#_Toc150609640)

[Contents 2](#_Toc150609641)

[Introduction 4](#_Toc150609642)

[Background 4](#_Toc150609643)

[Issues 4](#_Toc150609644)

[Resolution 5](#_Toc150609645)

[Considerations 5](#_Toc150609646)

[Top Level Domain 5](#_Toc150609647)

[Considerations 5](#_Toc150609648)

[Recommendations 5](#_Toc150609649)

[Organisation Identifier 5](#_Toc150609650)

[Considerations 5](#_Toc150609651)

[Recommendations 5](#_Toc150609652)

[Service Identifier 5](#_Toc150609653)

[Considerations 6](#_Toc150609654)

[Recommendations 6](#_Toc150609655)

[Service Tenancy Identifiers 6](#_Toc150609656)

[Advantages: 6](#_Toc150609657)

[Considerations: 6](#_Toc150609658)

[Recommendation: 7](#_Toc150609659)

[Service Components 7](#_Toc150609660)

[Considerations 7](#_Toc150609661)

[Recommendation: 7](#_Toc150609662)

[API Channel 8](#_Toc150609663)

[Considerations: 8](#_Toc150609664)

[Recommendation 8](#_Toc150609665)

[API Channel Type & SubType 8](#_Toc150609666)

[Considerations 8](#_Toc150609667)

[Recommendation 8](#_Toc150609668)

[API Channel Versions 9](#_Toc150609669)

[Considerations 9](#_Toc150609670)

[Recommendations 9](#_Toc150609671)

[GUI Channel 9](#_Toc150609672)

[Considerations 9](#_Toc150609673)

[Recommendations 10](#_Toc150609674)

[Self Help Channel 10](#_Toc150609675)

[Considerations 10](#_Toc150609676)

[Recommendation 10](#_Toc150609677)

[Support Channel 10](#_Toc150609678)

[Considerations 11](#_Toc150609679)

[Recommendation 11](#_Toc150609680)

[Default Sub Domains 11](#_Toc150609681)

[Recommendations 11](#_Toc150609682)

[Default Base URL 11](#_Toc150609683)

[Considerations 11](#_Toc150609684)

[Recommendation 11](#_Toc150609685)

[Option 1: Ad hoc 12](#_Toc150609686)

[Option 2: Organised 12](#_Toc150609687)

[Appendices 14](#_Toc150609688)

[Appendix A - Document Information 14](#_Toc150609689)

[Images 14](#_Toc150609690)

[Tables 14](#_Toc150609691)

[References 14](#_Toc150609692)

[Review Distribution 14](#_Toc150609693)

[Audience 14](#_Toc150609694)

[Structure 14](#_Toc150609695)

[Diagrams 14](#_Toc150609696)

[Terms 15](#_Toc150609697)

## Background

When deploying services, one has to consider how to provide and improve discoverability, learnability and usability qualities of the service, its channels, and supporting services.

The following is an example list of some of the above aspects that require consideration:

* A service specific area within a publicly accessible enterprise wide website (e.g. <https://myorg.tld/ourservice>)
* An optional service specific Brochure Website (e.g., <https://info.myservice.myorg.tld>)
* The Service Website itself
* Publicly accessible Resources:
  + Cacheable Static resources
* Sub services, including:
  + Service Self Help documentation
  + A Support service
* commodity services providing supporting services
  + privately accessed (notification, caching, searching, etc.?)
* Other interfaces endpoints:
  + API
* Documentation of other interfaces.

Beyond the above categories there are also expected conventions to consider:

* [Someservice.tld](http://www.Someservice.tld) redirects to [www.someservice.tld](http://www.someservice.tld)

## Issues

It is unfortunately common that services are delivered in an ad hoc manner, with the base service released first, and no forethought as to where other channels (APIs) or documentation or supporting services will go – and often even less thought as to providing different versions of these aspects in parallel.

## Risks

The lack of forethought may lead to irreconcilable logical clashes that may require changes to sort out.

Changes to endpoints leads to a decrease in discoverability, in search engines and/or breaking service clients, and learnability of how to interact with the service.

## Resolution

The resolution to the above issue is forethought to develop a pattern that accounts for known potential needs, taking into account considerations of the different aspects involved.

## Considerations

There are several aspects that require consideration before stabilising on a flexible domain naming strategy that will permit accommodation of change of service and components over the full service lifespan.

### Top Level Domain

The top-level domain (TLD) is there to indicate what type of service is being offered.

#### Considerations

* In the majority of cases TLDs are composed of a single element (e.g.: *.com*, *.io*, etc.) but in some cases, can be composed of two parts (e.g., govt.nz), while still being referred as the Top Level Domain(s).
* All services offered by New Zealand government agencies are obligated by regulation to be offered under the *.govt.nz* TLD[[1]](#footnote-2)[[2]](#footnote-3).

#### Recommendations

Use the most appropriate or obligated TLDs for the development for all services.

### Organisation Identifier

The second level domain provides information as to whom service consumers are transacting with.

#### Considerations

N/A.

#### Recommendations

Include the enterprise name that is the service provider (e.g.: <https://ourorg.tld>).

### Service Identifier

Next comes the name of the service.

#### Considerations

* Avoid dashes.
* Avoid reliance on diacritic symbols (e.g.: Māori macrons).

#### Recommendations

Example:

* <https://ourservice.ourorg.tld>

### Service Tenancy Identifiers

A common service may be made available for subscription by multiple organisations (e.g.: schools).

The highest quality solution is to provide each tenant with their own custom domain ([https://*ourservice.ourschool.tld*](https://ourservice.ourschool.tld)), but this has the highest implementation complexity, cost and risk of service interruptions due to not renewing and reinstalling certificates before their expiration.

A lower cost solution is to either provide the tenancy identifier prepended as a domain sub domain name ([https://*ourtenancy.myservice.ourorg.tld*](https://ourtenancy.myservice.ourorg.tld)), or appended as an url variable: [https://*www.ourservice.ourorg.tld?t=ourtenancy&*](https://www.ourservice.ourorg.tld?t=ourtenancy&)*...*

While putting the tenancy on the right is technically easier to implement, there is a slightly higher risk that the variable is omitted or mistyped, leading to unexpected outcomes.

#### Advantages:

* Provides a low cost means of increasing trustability via the visible URL’s domain name.

#### Considerations:

* Putting the tenancy id to the left requires:
  + a permissive https certificate.
  + This requires firewalls and WAFs to be configured to allow in \*.myservice.myorg.tld (after DNS entries for www., api., etc.)
  + The service itself should have logic to not permit the registration of tenancies that clash with subcomponent or channel information “www.”, “api.”, etc.
  + Increases trustability as the browser displays the tenancy name in the domain name.
* Putting the tenancy id on the right:
  + Only requires changes to the service, not firewalls or WAFs.
  + But introduces an optional element in the services URLs, complicating the development of predictable routes, requiring more testing
  + Higher risk of omitting when copying and pasting, the tenancy name as a variable -- resulting in unexpected outcomes.

#### Recommendation:

* Include the tenancy id on the left as a subdomain (e.g.: *[*https://*mytenancy.]myservice.myorg.tld)*:
  + https://myservice.myorg.tld: default entry point, redirecting to user profiles preferred tenant.
  + https://*mytenancyA*.myservice.myorg.tld
  + [https://*mytenancyB*.myservice.myorg.tld](https://mytenancyB.myservice.myorg.tld)
* Making the system able to also accept tenancy information as a variable on the right, *but only if none is provided to left*.

Note:  
Using a subdomain for providing tenancy information should only be applicable to the main gui service component (any system subcomponent that is generally communicated with by systems and not end users – eg cache, storage, etc. – should receive context via the variables.

### Service Components

Services benefit by offloading services to supporting services or devices. For example, static media can be offloaded to a local caching service or a distributed cloud-based caching service.

#### Considerations

* Where possible, CDNs are recommended to decrease the latency of imagery, scripts, etc.
* Whereas performance is important it’s uncommon that projects start off with this objective in mind, as they chase other outcomes.

#### Recommendation:

Even if not yet focusing on performance, we recommend providing as early as possible a designated subdomain for where static resources could be sourced from when it becomes required:

* <https://static.myservice.myorg.tld>

Note:  
While probably overkill, the option of including a tenancy identifier is still possible:

<https://static.mytenancy.myservice.myorg.tld>

### API Channel

The technologies used for APIs change over time. For example, SOAP was a strong preference in the 2000’s that has since lost support, replaced by REST.

gRPC and a couple of other case specific API protocols are currently emergent, but gaining popularity quickly.

#### Considerations:

* Once published, there is no certainty of being able to move all service clients using a protocol to a replacement protocol.   
  Hence the need to serve information via multiple APIs in parallel.

#### Recommendation

Prepare for the eventuality that multiple APIs will be developed for the service over its service lifespan, by preparing a specific subdomain group (“.api”) to host them under:

* https://soap.api.myservice.myorg.tld
* https://rest.api.myservice.myorg.tld

Additionally, whereas GUI URLs benefit from having tenancy id’s prepended as subdomains, API endpoints are solely accessed by other computers who once they instructed correctly execute the same action repeatedly without error.   
Therefore, for API endpoints, we recommend tenancy identifiers are provided as arguments on the right.

### API Channel Type & SubType

Plain vanilla REST has been extended to add common operations (Filtering, Ordering, Paging, etc.) as well as complex Queryability over joined models. ODATA is one standards based extension, therefore subtype, of REST.

Note:  
GraphQL is often discussed as a subtype of REST, while not strictly RESTful -- but most developers would still classify it as being under the REST umbrella rather than give it a separate category entirely.

#### Considerations

* Versioning still occurs for subtypes.

#### Recommendation

* GraphQL and OData should be in parallel, as they fulfil the same logical task.
* Noting that either way has logical flaws (GraphQL is not a subtype of Rest, and OData is not separate for Rest), we recommend putting OData in the URL in parallel to Rest rather than nested, as nesting adds no logical or functional benefit.
* <https://odata.rest.api.myservice.myorg.tld>
* <https://v1.odata.rest.api.myservice.myorg.tld>
* <https://odata.api.myservice.myorg.tld>
* <https://v1.odata.api.myservice.myorg.tld>
* <https://v3.graphql.api.myservice.myorg.tld>

### API Channel Versions

APIs are consumed by service clients, the maintenance of which are outside the service provider’s control (i.e., one can’t ensure that all communication between parties moves to a new version or protocol in a synchronised manner). Hence different Version of APIs are required to be supported.

#### Considerations

* There is ongoing debate as to whether to consider a versioned set of API endpoints as a different service channel (<https://v2.rest.api.myservice.myorg.tld>) or resources (<https://v2.rest.api.myservice.myorg.tld>?v=2).
* Whereas plain vanilla REST APIs are individual and only logical sets, OData APIs are physical sets describing a common model.
* Multiple versions of OData sets may be delivered.

#### Recommendations

* Based on the behaviour of OData sets, it is our recommendation to prepare for that eventuality and include the major element of the version as an URL subdomain prefix:
  + <https://v1.rest.api.myservice.myorg.tld>
  + <https://v2.rest.api.myservice.myorg.tld>
* Do *not* publish a “default” API URL (without version). It’s technically necessary therefore important to specify which version one is expecting to connect to.

### GUI Channel

Services are used by systems (via APIs) and Users (via User Interfaces).

Note:  
It is incorrect to consider the G/UI as being more or less important than an API. It’s just another, parallel, channel.

#### Considerations

* It is relatively rare, but it does happen that different GUIs are provided to different user cohorts. An example of this is when an Assessment tool is offered, where one interface is aimed towards adults, for administration purposes, and another is aimed towards a younger age group, with a simpler and friendlier interface.
* There are other cases where different versions of interfaces are presented to users (A/B testing). But this *may* be better served by system User Preferences rather than URLs.

#### Recommendations

The following options work:

* <https://gui.myservice.myorg.tld>
* <https://ui.myservice.myorg.tld>
* <https://parents.ui.myservice.myorg.tld>
* <https://a.parents.ui.myservice.myorg.tld>
* <https://b.parents.ui.myservice.myorg.tld>
* <https://teachers.ui.myservice.myorg.tld>
* <https://learners.ui.myservice.myorg.tld>

### Self Help Channel

Documentation of the service should be provided for users to self-help, reducing support costs.

#### Considerations

* The URL should be predictable to improve discoverability.
* There is ambiguity in the use of the term ‘help’ as to whether it is self-help, versus assisted help. We suggest it be reserved for assisted help (see *Support Channel*, covered next).

#### Recommendation

* <https://self-help.myservice.myorg.tld>
* <https://documentation.myservice.myorg.tld>
* <https://docs.myservice.myorg.tld>

### Support Channel

When self-help is not sufficient, a support service should be provided where issues can be logged for response by support specialists.

#### Considerations

* Support costs in terms of subscription, users, resource for the full duration of the service lifespan. Therefore, it is highly recommended to effort into both the service to improve its learnability and self-help channel.

#### Recommendation

* <https://support.myservice.myorg.tld>
* <https://help.myservice.myorg.tld>

### Default Sub Domains

For humanly accessed endpoints (not automated service clients using APIs), it is convention to fill in default values:

* <https://myservice.com> redirects to https://[www.myservice.com](http://www.myservice.com)
* https://static.myservice.tld redirecting to <https://media.static.myservice.tld>

#### Recommendations

While default sub domains are expected for the default base UI service URL (discussed next), it is recommended to avoid providing default redirects for any service agent endpoint (APIs endpoints).

### Default Base URL

#### Considerations

The choice is either weighted towards first time *discovery*, or ongoing *use* thereafter*:*

* initial discovery of the service offering:
  + <https://myservice.org.tld> refers to the brochure website,   
    and a subdomain is used for the service itself (<https://info.myservice.org.tld>)
* ongoing use of the service thereafter:
  + <https://myservice.org.tld> is the service itself, and another site is used for the brochure website: <https://info.myservice.org.tld>

#### Recommendation

Although admittedly subjective, we are currently recommending leaning towards the option that facilitates discovery during use (not just the first time).

As such we recommend

* <https://myservice.myorganisation.tld> redirecting to
* <https://ui.myservice.myorganisation.tld> for the service itself,
  + But it must provide a link back to where one can find out about the service itself (it’s dedicated brochureware):
* <https://info.myservice.organisation.tld>

## Option 1: Ad hoc

Developing a service without an organising principle leads to a poor discoverability, learnability and usability outcomes.

## Option 2: Organised

Based on the above aspects and recommendations for each, the following is proposed:

[[[version.][type.][channel/component.] tenancy.]service.].org].tld  
  
The above pattern permits:

* <https://myorg.tld> for the enterprise website
* <https://myorg.tld/somearea> for general info on the above website
* <https://service.tld> if the service is to have a standalone URL that disassociates the service from the sponsoring organisation,
  + <https://myservice.org.tld> being a short alias to the UI of the service,
    - <https://ui.myservice.org.tld> for the User Interface of the service itself,
    - <https://info.myservice.org.tld> for brochureware on the service,
    - <https://help.myservice.myorg.tld> for static self-help documentation
    - <https://support.myservice.org.tld> for an issues/support service,
    - <https://static.myservice.org.tld> (cacheable publicly accessible static media)
      * <https://img.static.myservice.org.tld>
      * <https://sound.static.myservice.org.tld>
    - <https://api.myservice.tld>, maybe redirecting to a static website providing links to documentation of the various APIs.
      * <https://soap.api.myservice.tld> (do not use)
        + <https://v1.soap.api.myservice.tld> (do not use)
      * <https://rest.api.myservice.tld> (do not use)
        + <https://v1.rest.api.myservice.tld>
      * https://odata. api.myservice.tld
        + <https://v2.odata.api.myservice.tld>
      * <https://graphql.api.myservice.tld>
        + <https://v1.graphql.api.myservice.tld>
      * <https://gRPC.api.myservice.tld>
      * <https://websockets.myservice.tld>

Appendices

Appendix A - Document Information

Versions

* 1. Initial Draft
  2. Released for Review
  3. Corrections

### Images

### Tables

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Sandy Britain, Enterprise Architect |  |
| Matt Duguid, DevOps Engineer |  |
| Archana Sahani, Business Analyst |  |
| Dijana Sneath, Business Analyst |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project’s Glossary.

##### IT

: acronym for Information, using Technology to automate and facilitate its management.

##### ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

##### Second Level Domain

: the part to the left of a TLD within an URL. Generally used for the organisation’s identifier.

##### Third Level Domain

: the part to the left of a Second Level Domain within an URL. Generally used for the service identifier.

##### TLD

: acronym for Top Level Domain (.org, .com, .govt.nz, etc.)

##### URI:

Universal Resource Identifier, developed as follows: [[[{fourth-level-domain}.]{third-level-domain}.]{second-level-domain}].{top-level-domain}/{sub-directory}.

##### URL:

Universal Resource Location, developed as follows: {scheme}:://{URI}[?{variables}[#{fragment}]].

1. [Government domain names | NZ Digital government](https://www.digital.govt.nz/standards-and-guidance/technology-and-architecture/domain-names/) [↑](#footnote-ref-2)
2. [Domain name policy - Te Tahuhu - Sharing our knowledge (education.govt.nz)](https://intranet.education.govt.nz/organisation/policies/domain-name-policy/) [↑](#footnote-ref-3)