



Australian Government
Australian Digital Health Agency

National Clinical Terminology Service Guide for Implementers

31 January 2021 v.1.2

Approved for external information

Document ID: DH-3405:2021



Acknowledgements

The Australian Digital Health Agency is jointly funded by the Australian Government and all state and territory governments.

Regenstrief Institute (LOINC)

This material contains content from [LOINC](#)TM. The LOINC table, LOINC codes, LOINC panels and forms file, and LOINC linguistic variants file are copyright © 1995–2015, Regenstrief Institute, Inc. and the Logical Observation Identifiers Names and Codes (LOINC) Committee and available at no cost under the license at the [LOINC Terms of Use](#). LOINC is a trademark of Regenstrief Institute, Inc., registered in the United States.

IHTSDO (SNOMED CT)

This material includes SNOMED Clinical TermsTM (SNOMED CT[®]) which is used by permission of the International Health Terminology Standards Development Organisation (IHTSDO). All rights reserved. SNOMED CT[®] was originally created by The College of American Pathologists. “SNOMED” and “SNOMED CT” are registered trademarks of the [IHTSDO](#).

HL7 International

This document includes excerpts of HL7TM International standards and other HL7 International material. HL7 International is the publisher and holder of copyright in the excerpts. The publication, reproduction and use of such excerpts is governed by the [HL7 IP Policy](#) and the HL7 International License Agreement. HL7 and CDA are trademarks of Health Level Seven International and are registered with the United States Patent and Trademark Office.

Disclaimer

The Australian Digital Health Agency (“the Agency”) makes the information and other material (“Information”) in this document available in good faith but without any representation or warranty as to its accuracy or completeness. The Agency cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

Document control

This document is maintained in electronic form and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is the latest revision.

Copyright © 2021 Australian Digital Health Agency

This document contains information which is protected by copyright. All Rights Reserved. No part of this work may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems – without the permission of the Australian Digital Health Agency. All copies of this document must include the copyright and other information contained on this page.

OFFICIAL

Document information

Key information

Owner	Michael Costello
Date of next review	30 June 2021
Contact for enquiries	Australian Digital Health Agency Help Centre
	Phone 1300 901 001
	Email help@digitalhealth.gov.au

Product or document version history

Product or document version	Date	Release comments
v1.0	31 Oct 2017	Initial release
v1.1	30 Jun 2020	Changes to reflect FHIR R4
v1.2	31 Jan 2021	Updated to reflect AEHRC Docker registry moved from Docker Hub to Quay.io

Table of contents

1	Introduction	6
1.1	Purpose	6
1.2	Intended audience	6
1.3	Scope.....	6
2	Overview.....	7
2.1	National Terminology.....	7
2.1.1	SNOMED CT-AU	7
2.1.2	Australian Medicines Terminology	7
2.1.3	RCPA Pathology Terminology and Information Models	8
2.2	National Clinical Terminology Service.....	8
2.2.1	Components	8
2.2.2	Supported Use Cases	10
3	Key Concepts.....	14
3.1	Fast Healthcare Interoperability Resources (FHIR).....	14
3.2	NCTS Content Types.....	14
3.3	Conformant Terminology Server Application	15
3.4	Conformant Syndication Server Application.....	17
3.5	Terminology Syndication	18
3.6	Ontoserver	20
3.7	Snapper	22
4	Common Tasks	23
4.1	Setting up a Terminology Server.....	23
4.1.1	Generating a System Credential.....	23
4.1.2	Requesting an Access Token.....	23
4.1.3	Accessing the FHIR API on the NTS.....	24
4.1.4	Accessing the Syndication API on the NSS.....	25
4.1.5	Downloading the Ontoserver distribution	25
4.1.6	Running Ontoserver Locally.....	26
4.1.7	Loading National Content into a Local Ontoserver Instance	26
4.1.8	Interacting with the Ontoserver FHIR API	27
4.2	Receiving Updates.....	27
4.2.1	Setting up a Scheduled Job to Retrieve New Releases.....	27
4.2.2	Updating the Ontoserver Docker container	28
4.3	Mapping between Local and National Terminologies	29
4.3.1	Using Snapper with the National Terminology Server	29
4.3.2	Using Snapper with a Local Ontoserver Instance	30
4.3.3	Generate a Self-Signed TLS Certificate for Ontoserver	30
5	Design Considerations	33
5.1.1	Security features and considerations	33
5.1.2	Designing for high performance.....	34

5.1.3	Designing for high availability.....	34
6	Deployment Scenarios	35
6.1	Design-time Applications Directly Accessing the NTS.....	35
6.2	Local Terminology Service for Clinical Information Systems	36
6.3	Local Terminology Services with Multi-Tiered Syndication	38
6.4	Syndication for Vendors.....	39
6.5	Terminology Validation for EMRs	40
6.6	Local Terminology Server for Authoring, with Authentication	41

1 Introduction

1.1 Purpose

The purpose of this document is to:

- Provide an introduction to the National Clinical Terminology Service (NCTS), including a description of the products and services that are available;
- Provide guidance on how to integrate with the NCTS within the scope of common use cases, such as setting up local terminology services and mapping to national terminologies;
- Inform integrators of relevant deployment considerations with respect to the use of conformant terminology servers, and;
- Refer users to other relevant documentation, which describes the technical aspects of the terminology products and NCTS interfaces.

1.2 Intended audience

This document is intended for use by those evaluating the use of and integration with NCTS, including:

- Implementers and integrators of clinical information systems
- Implementers and deployers of terminology servers
- Local terminology maintainers
- Public health researchers

1.3 Scope

This document is limited to guidance regarding use of and integration with NCTS products and services.

It includes information on how to deploy Ontoserver, use Snapper, and integrate with the NCTS Application Programming Interfaces (APIs).

This document does not include:

- Information on how to register for NCTS or use the NCTS portal;
- Information on the semantics of the terminology distributions themselves; or
- Design considerations related to the use of terminology content within clinical information systems and research initiatives.

2 Overview

This section provides a brief introduction to the National Clinical Terminology Service including the terminologies, tools and support services made available through the service.

2.1 National Terminology

This section describes the national terminologies that are currently published as part of the NCTS.

There are many other national terminologies in existence, and the plan is to publish as many of these as we can, over time.

2.1.1 SNOMED CT-AU

SNOMED CT is a standardised clinical terminology used to facilitate the structured representation of clinical terms within computer systems.

SNOMED CT includes a wide range of clinical concepts which are organised into a number of top-level hierarchies, for example “Body structure”, “Clinical finding”, “Procedure” and “Substance”.

SNOMED CT-AU is the Australian edition of SNOMED CT, and includes localised information for core SNOMED CT concepts, additional concepts specific to Australian use, and reference sets that group concepts relevant to specific Australian use cases.

For more information, see [SNOMED CT-AU Australian Technical Implementation Guide](#).

Updates to SNOMED CT-AU are published on a monthly basis, via the NCTS.

SNOMED CT-AU is published in two formats:

- [Release Format 2](#) (RF2).
- Ontoserver Binary.

2.1.2 Australian Medicines Terminology

The Australian Medicines Terminology (AMT) is a terminology designed to enable consistent and unambiguous description of medicines commonly used within Australia. The concepts defined within AMT form a subset of the concepts that comprise SNOMED CT-AU.

AMT is designed for use within a number of use cases, including:

- Prescribe – product or pack-based prescribing in a primary care setting.
- Order – inpatient medication ordering in an acute care setting.
- Dispense – dispensing of prescriptions within retail/community pharmacy and acute care settings.

For more information, see [AMT Concept Model and Business Use Cases](#).

AMT is distributed as part of the SNOMED CT-AU product, and is also published on a monthly basis. Monthly releases of AMT include updates from the Pharmaceutical Benefits Scheme (PBS) and Therapeutic Goods Administration (TGA).

2.1.3 RCPA Pathology Terminology and Information Models

The [Pathology Terminology and Information Models](#) from the Royal College of Pathologists Australasia (RCPA) provide a standard information model and terminology for use within pathology requests and reports within Australia.

The code systems within this product are based upon the SNOMED CT-AU and LOINC terminologies. LOINC is an international standard for tests, observations and measurements that is commonly used within pathology workflows.

2.2 National Clinical Terminology Service

The National Clinical Terminology Service (NCTS) is a suite of software and services designed to distribute the national terminology sets to users, and ease terminology adoption.

2.2.1 Components

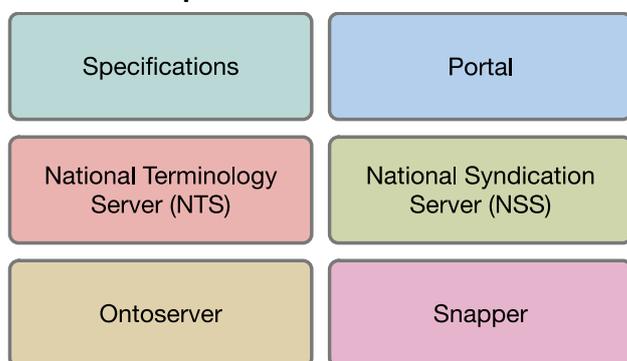


Figure 1 - High-level components of NCTS

The NCTS consists of the following components:

- **Specifications**, describing:
 - Representation of terminology sets and mappings in standard computer-readable formats;
 - Standard interfaces for exposing terminology held within computer systems; and
 - Interfaces exposed as part of the NCTS national infrastructure.
- **Portal** – a web site which can be used to:
 - Register the participation (and acceptance of licensing) of individuals and organisations;
 - Download national terminology content;
 - Download documentation;
 - Manage API credentials and access of users within organisations.
- **National Terminology Server** – a national terminology server, providing FHIR-based access to monthly releases of SNOMED CT-AU.
- **National Syndication Server** – a server that exposes an Atom-based feed of national terminology release files.

- **Ontoserver** – a terminology server implementation that conforms to the NCTS specifications, provided free of charge for use in local deployments.
- **Snapper** – a web-based tool for authoring and maintaining mappings between local and national terminologies, capable of reading from and writing to terminology servers that implement the FHIR interface defined within the NCTS specifications. This tool is also provided free of charge.

2.2.2 Supported Use Cases

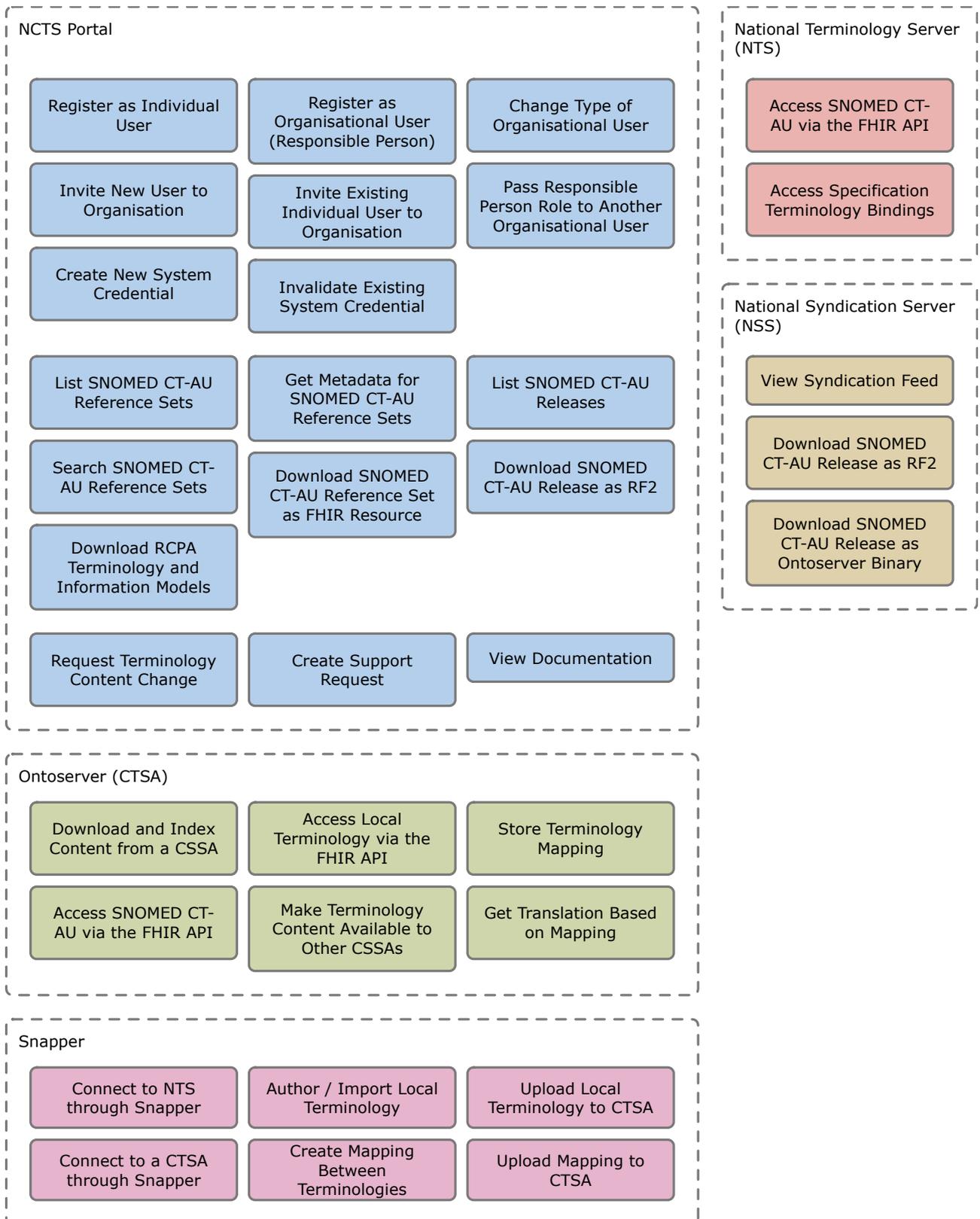


Figure 2 - Use cases supported by NCTS

Component	Use Case	Description
Portal	Register as Individual User	Register for an account to access NCTS content and APIs as an individual.
Portal	Register as Organisational User (Responsible Person)	Register a new organisation, and apply to be authorised to manage users and credentials for that organisation.
Portal	Change Type of Organisational User	Change the type of a user within an organisation to either Standard, or Responsible Person.
Portal	Invite New User to Organisation	Send an email to an email address allowing the recipient to register as a user associated with a particular organisation.
Portal	Invite Existing Individual User to Organisation	Where a user is already registered for NCTS as an individual, invite them to associate themselves with a particular organisation.
Portal	Pass Responsible Person Role to Another Organisational User	Change the type of one Responsible Person to Standard and change the type of another user to Responsible Person in their place.
Portal	Create New System Credential	Create a Client ID and Client Secret for use in authenticating a connecting application to the NCTS APIs.
Portal	Invalidate Existing System Credential	Prevent an existing system credential from being able to be used to authenticate with the NCTS APIs in future.
Portal	List SNOMED CT-AU Reference Sets	Get a list of all published SNOMED CT-AU reference sets.
Portal	Get Metadata for SNOMED CT-AU Reference Sets	Get information about a specified reference set, including description and intended use.
Portal	List SNOMED CT-AU Releases	Get a list of all published releases of SNOMED CT-AU.
Portal	Search SNOMED CT-AU Reference Sets	Get a list of published reference sets that contain specified text within their metadata.
Portal	Download SNOMED CT-AU Reference Set as FHIR Resource	Query the National Terminology Server for a ValueSet expansion of the specified reference set.
Portal	Download SNOMED CT-AU Release as RF2	Download the RF2 release files for a specified SNOMED CT-AU release.
Portal	Download RCPA Terminology and Information Models	Download the latest release of the RCPA Terminology and Information Models.
Portal	Request Terminology Content Change	Request an addition or update to a national terminology product.
Portal	Create Support Request	Ask for help with NCTS.

Component	Use Case	Description
Portal	View Documentation	Read business and technical documentation regarding the national terminology products and the NCTS itself.
Ontoserver (Conformant Terminology Server Application (CTSA))	Download and Index Content from a Conformant Syndication Server Application (CSSA)	Download a specified Content Item from the configured syndication endpoint and make that content available via the FHIR API.
Ontoserver (CTSA)	Access Local Terminology via the FHIR API	Access FHIR resources that have been authored to represent local terminologies required by users of this Ontoserver instance.
Ontoserver (CTSA)	Store Terminology Mapping	Host a ConceptMap resource authored to represent a mapping between two different terminology sets.
Ontoserver (CTSA)	Access SNOMED CT-AU via the FHIR API	Query SNOMED CT-AU releases through the FHIR API.
Ontoserver (CTSA)	Make Terminology Content Available to other CSSAs	Expose a syndication interface that allows syndication clients to discover what is on this server, and download it.
Ontoserver (CTSA)	Get Translation based on Mapping	Use a ConceptMap hosted within this Ontoserver instance to translate a code from one code system to another.
Snapper	Connect to NTS through Snapper	Authenticate with and access the NTS for the purpose of bringing national terminology content into Snapper.
Snapper	Author / Import Local Terminology	Connect to a locally deployed read / write Ontoserver, and be able to both import existing terminology content and publish new content.
Snapper	Upload Local Terminology to CTSA	Create FHIR resources to represent local terminology content on a conformant terminology server.
Snapper	Connect to a CTSA through Snapper	Connect and authenticate to a conformant terminology server.
Snapper	Create Mapping Between Terminologies	Author a mapping between two terminologies.
Snapper	Upload Mapping to a CTSA	Publish the result of a mapping as a ConceptMap to a conformant terminology server.
National Terminology Server (NTS)	Access SNOMED CT-AU via the FHIR API	Provide query services for SNOMED CT-AU via the NTS.
National Terminology Server (NTS)	Access Specification Terminology Bindings	Access FHIR resources that represent allowable values in clinical document specifications.
National Syndication Server (NSS)	View Syndication Feed	Retrieve a feed with entries for published terminology products, along with links to download them.

Component	Use Case	Description
National Syndication Server (NSS)	Download SNOMED CT-AU Release as RF2	Download a specified RF2 release file for a particular release of SNOMED CT-AU.
National Syndication Server (NSS)	Download SNOMED CT-AU Release as Ontoserver Binary	Download the Ontoserver binary file for a particular release of SNOMED CT-AU.

3 Key Concepts

3.1 Fast Healthcare Interoperability Resources (FHIR)

FHIR is a standards framework for the interchange of healthcare information, published by HL7 International. While the FHIR standard is still relatively new and continues to evolve rapidly, it has already achieved considerable support within the community of healthcare software implementers.

FHIR is based around a set of modular components known as “resources”. A number of [standard resources](#) have been defined within FHIR to support a range of different use cases, including those relating to clinical terminology. FHIR resources represent a domain model and serialisation format for healthcare integration solutions.

FHIR also defines an API specification that defines the operations and behaviour of a FHIR server. This includes search, create, read, update and delete operations for interacting with any type of resource, in the [REST architectural style](#).

FHIR also defines a [Terminology Service API](#), which is a standard API for interacting with FHIR terminology servers. This API includes operations specific to servers hosting [SNOMED CT](#), [LOINC](#) and [RxNorm](#) terminology content.

3.2 NCTS Content Types

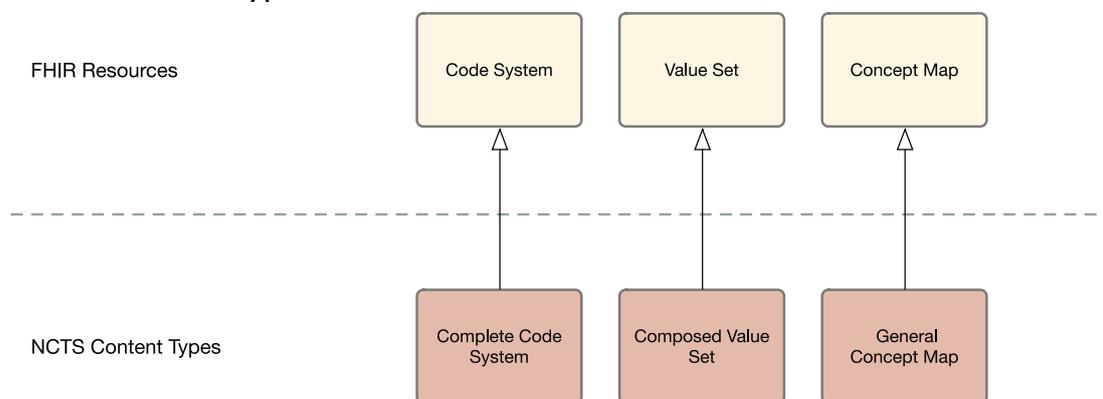


Figure 3 - Relationship between FHIR resources and NCTS Content Types

The NCTS has selected three terminology-related resources from the FHIR standard to form the basis of our specifications, with the goal of promoting standardisation of the representation of terminology sets and mappings.

[Code System](#) – a set of codes with meanings (also known as a terminology).

[Value Set](#) – a set of codes defined in one or more Code Systems for use within a particular context.

[Concept Map](#) – a statement of relationships between the codes within two Value Sets.

NCTS specifications define a profile for each of these resources. Each profile defines some extra constraints on the resource which are seen to have value in terms of interoperability within the

NCTS ecosystem. Each of the NCTS profiles have been given a more specific name, and an associated [StructureDefinition](#) resource which helps define them:

Complete Code System

<https://healthterminologies.gov.au/fhir/StructureDefinition/complete-code-system-3>

Composed Value Set

<https://healthterminologies.gov.au/fhir/StructureDefinition/composed-value-set-3>

General Concept Map

<https://healthterminologies.gov.au/fhir/StructureDefinition/general-concept-map-3>

NCTS specifications also define content types for SNOMED CT and LOINC releases, for the purposes of supporting their inclusion within syndication feeds.

For the formal definition of all the NCTS Content Types, see the [NCTS Specifications](#).

3.3 Conformant Terminology Server Application

The NCTS provides a profile of the FHIR Terminology Service API, adding some extra conformance points that improve interoperability within the context of the NCTS ecosystem. This profile is referred to as the [NCTS FHIR API](#), and is defined within the NCTS Specifications.

A [Conformant Terminology Server Application \(CTSA\)](#) is a software application that implements the NCTS Integration API, and complies with all conformance points within the specification. This includes the following operations from the FHIR API and FHIR Terminology Service API:

Table 1 - Integration API operations

FHIR operation	FHIR specification URL	Use case
RESTful API Operations		
capabilities	http://hl7.org/fhir/R4/http.html#capabilities	Get capabilities of a CTSA
search	http://hl7.org/fhir/R4/http.html#search	Search <i>Content Items</i>
create	http://hl7.org/fhir/R4/http.html#create	Create a new <i>Content Item</i>
read	http://hl7.org/fhir/R4/http.html#read	Get current version of a <i>Content Item</i>
vread	http://hl7.org/fhir/R4/http.html#vread	Get a specific version of a <i>Content Item</i>
update	http://hl7.org/fhir/R4/http.html#update	Update a <i>Content Item</i>
delete	http://hl7.org/fhir/R4/http.html#delete	Remove an existing <i>Content Item</i>
batch	http://hl7.org/fhir/R4/http.html#transaction	Submit a set of operations within a single request
Resource Operations		
validate	http://hl7.org/fhir/R4/resource-operations.html#validate	Validate a <i>Content Item</i>

FHIR operation	FHIR specification URL	Use case
CodeSystem Operations		
lookup	http://hl7.org/fhir/R4/codesystem-operations.html#lookup	Get details of a concept within a CodeSystem
subsumes	http://hl7.org/fhir/R4/codesystem-operations.html#subsumes	Test subsumption relationship between two codes
ValueSet Operations		
expand	http://hl7.org/fhir/R4/valueset-operations.html#expand	Expand: <i>Composed Value Set</i>
validate-code	http://hl7.org/fhir/R4/valueset-operations.html#validate-code	Validate that a coded value is in the set of codes allowed by a ValueSet
ConceptMap Operations		
translate	http://hl7.org/fhir/R4/conceptmap-operations.html#translate	Translate a code from one ValueSet to another
closure	http://hl7.org/fhir/R4/conceptmap-operations.html#closure	Maintain a client-side transitive closure
Terminology Service Operations		
expand (Implicit)	http://hl7.org/fhir/R4/snomedct.html#implicit http://hl7.org/fhir/R4/loinc.html#implicit	Expand: <i>SNOMED CT</i> implicit ValueSet <i>LOINC</i> implicit ValueSet <i>Complete Code System</i> implicit ValueSet
translate (Implicit)	https://www.hl7.org/fhir/R4/snomedct.html#implicit-cm	Translate a code from one SNOMED CT implicit ValueSet to another

For the formal definition of the FHIR API and Conformant Terminology Server Application, see the [NCTS Specifications](#).

3.4 Conformant Syndication Server Application

NCTS provides a profile of the [Atom Publishing Protocol](#) (AtomPub), defining a standard structure for representing syndicated NCTS Content Types. This profile is referred to as the [NCTS Syndication API](#), and is defined within the NCTS Conformant Server Applications Technical Service Specification.

A [Conformant Syndication Server Application](#) (CSSA) is a software application that implements the NCTS Syndication API, and complies with all conformance points within the specification. This includes the following operations from the AtomPub specification:

Table 2 - Syndication API operations

AtomPub operation	AtomPub specification URL	Use case
Listing Collection Members	https://tools.ietf.org/html/rfc5023#section-5.2	List all entries within a syndication feed
Retrieving a Resource	https://tools.ietf.org/html/rfc5023#section-5.4.1	Download the <i>Content Item</i> referenced by an entry in the syndication feed

For the formal definition of the NCTS Syndication API and Conformant Syndication Server Application, see the [NCTS Specifications](#).

3.5 Terminology Syndication

Syndication is term that is used to describe the transfer of terminology data between terminology servers. The NCTS specifications define a standard interface contract for exposing this functionality within conformant terminology server implementations.

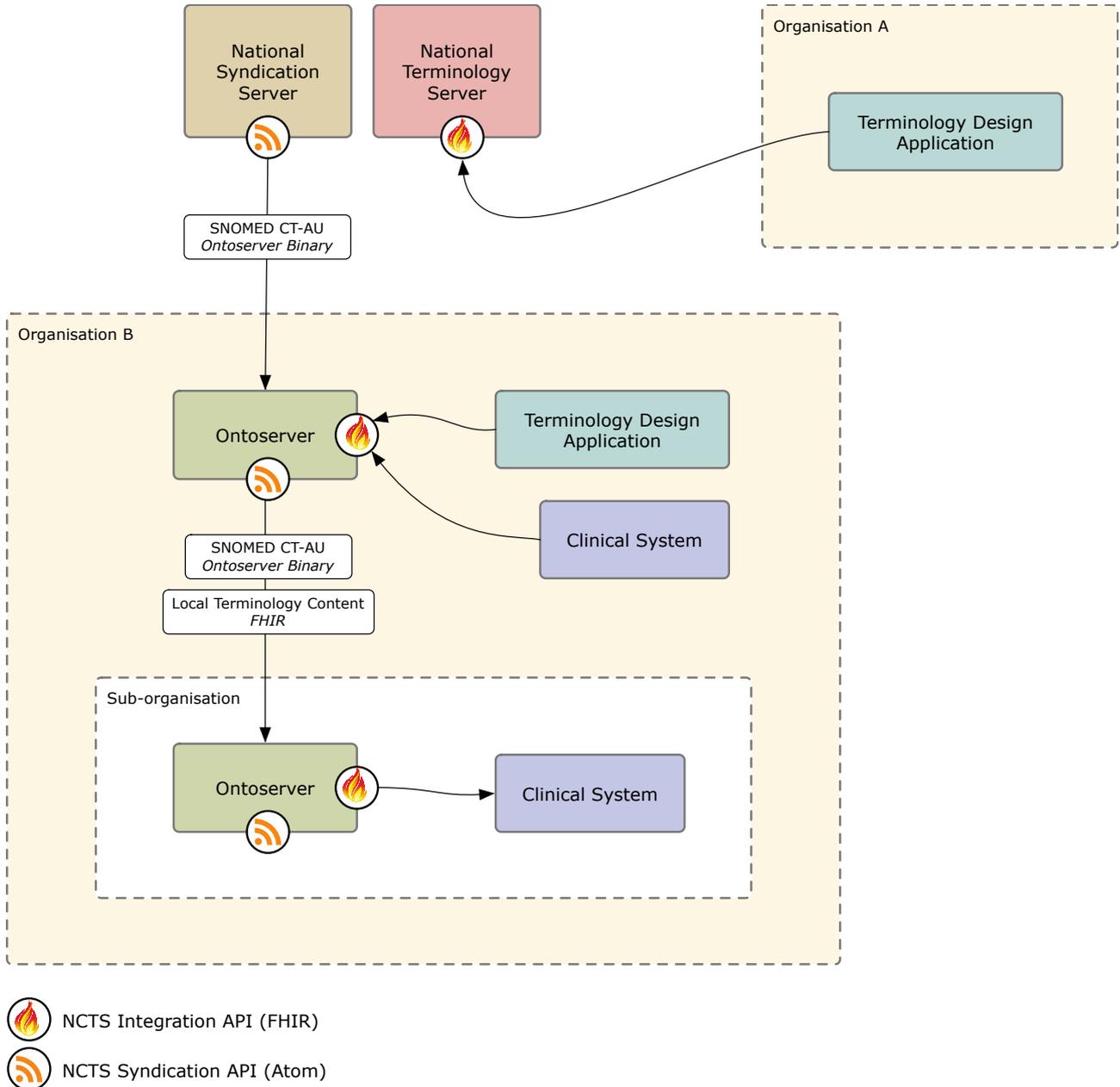


Figure 4 - Example of multi-tiered terminology syndication using Ontoserver

The standard syndication interface allows implementing software to programmatically interrogate a catalogue of available terminology artefacts, which can then be downloaded. This can be used, for example, to enable the automatic discovery and download of monthly release content for SNOMED CT-AU.

This model enables some decentralisation of the responsibility of distributing terminology within the greater ecosystem. For national terminology, NCTS can serve the role of seeding top-level

terminology servers, which can then expose their own syndication feeds to terminology consumers within their own areas of jurisdiction. Syndication feeds can also be augmented with localised, context-specific terminology sets at lower levels of the tree.

This supports the intent that operators of clinical systems make use of local terminology servers for run-time queries from clinical systems, rather than querying the National Terminology Server.

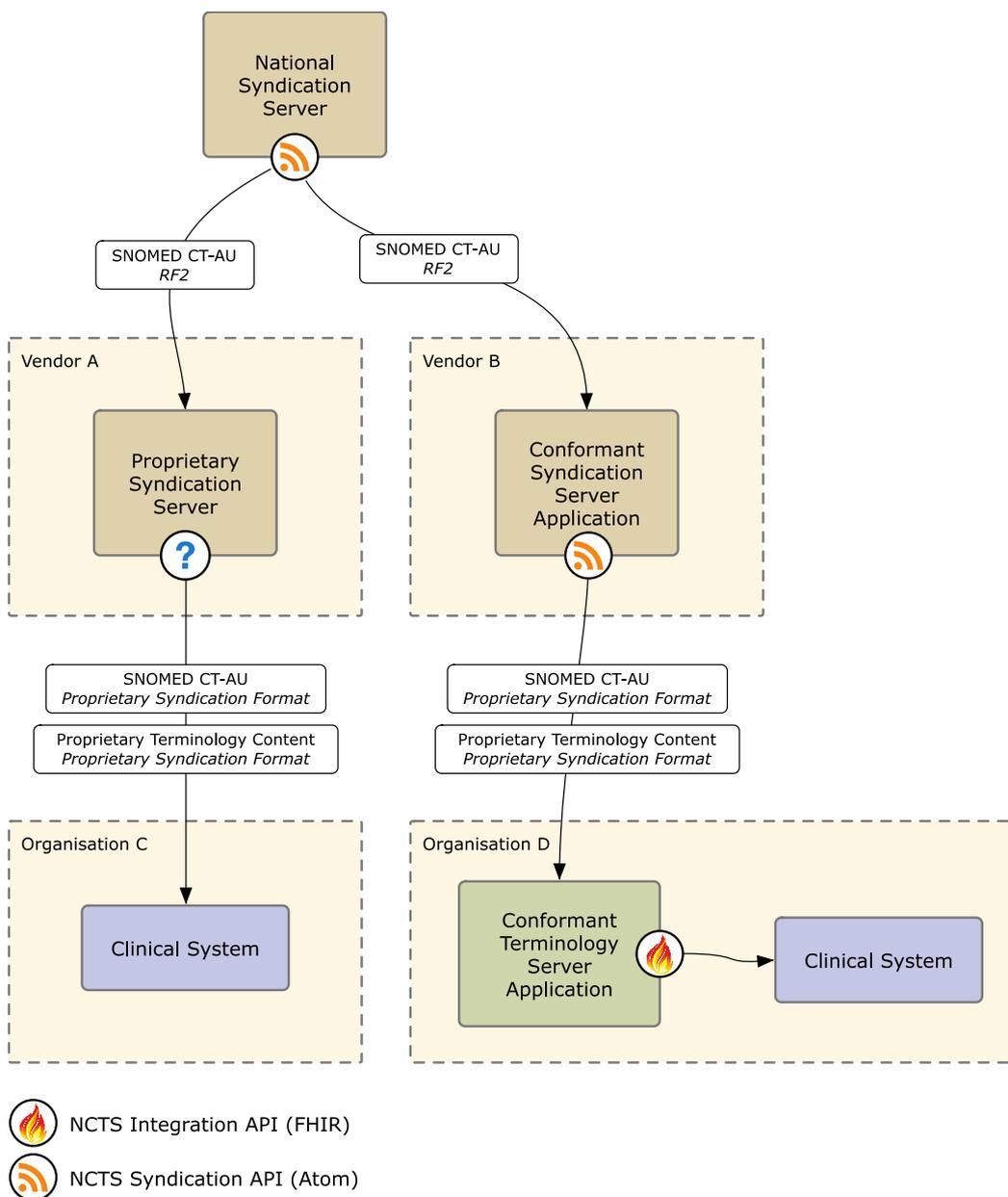


Figure 5 - Example of syndication to vendor CIS

Vendors can use the syndication service to retrieve updates to the national terminologies automatically, which they can then distribute to their users in a manner which suits their own software architecture. Vendors can optionally implement the standard interfaces for syndication and terminology services, adding interoperability benefits for their users.

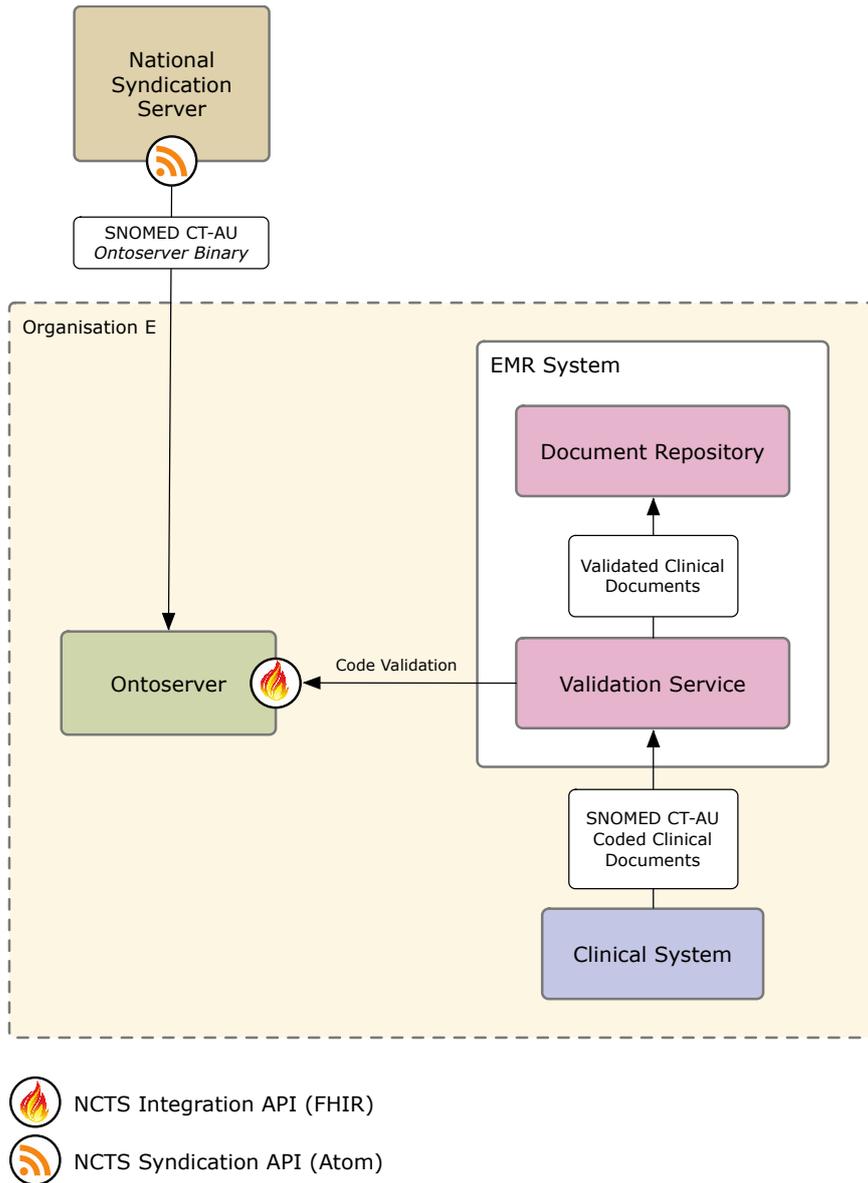


Figure 6 - Example of integration with EMR system

Terminology can also be syndicated to a local terminology server for the purposes of validating coded documents sent to an Electronic Medical Records (EMR) system. Use of a FHIR REST API call to Ontoserver eliminates the need to develop the terminology validation functionality within the EMR itself, or maintain transformations from the national terminologies to proprietary data loads.

3.6 Ontoserver

Ontoserver is a terminology server application developed by the CSIRO.

The purpose of providing Ontoserver as a part of NCTS is to make it easier to deploy terminology services to support clinical information systems.

Ontoserver supports the import of SNOMED CT-AU release bundles (RF2), and exposes a set of APIs for interacting with and syndicating terminology content.

Using the Ontoserver FHIR API, applications can interact with SNOMED CT-AU in the following ways:

- search for SNOMED CT codes using full-text search;
- filter searches by release, reference set and subsumption (hierarchical ancestor);
- look up the details of a given code;
- validate that a given code is within a specified subset of SNOMED CT-AU;
- traverse historical associations between SNOMED CT concepts; and
- perform batch searches and look ups.

Ontoserver can also be used to store local terminology sets and concept maps, which represent the mappings between different terminology sets. A concept map can be used to perform translate operations to convert codes between terminologies at run-time, or for batch-processing use cases.

The Ontoserver FHIR API provides a simplified interface to interact with complex terminology products such as SNOMED CT-AU, removing the need to process raw terminology release formats (such as RF2) for most use cases. Ontoserver can also be set up to automatically syndicate new releases from the National Terminology Server, making it easy to provide the latest terminology data available to client applications.

Ontoserver is both a [Conformant Terminology Server Application](#), and a [Conformant Syndication Server Application](#). It is interoperable with other applications participating in the NCTS ecosystem, such as the National Syndication Server and Snapper, and other applications implementing the terminology-related aspects of the FHIR specification.

Ontoserver is also able to publish terminology artefacts to its own syndication feed, which can then be consumed by other CTSA's. This allows for multi-tiered syndication architectures.

For more information about Ontoserver, see the documentation: <https://ontoserver.csiro.au/docs>

3.7 Snapper

Snapper is a web-based tool which can assist with the task of maintaining mappings between different terminology sets, optionally using a CTSA as the repository for these mappings.

As an example, a local or proprietary terminology set can be imported into Snapper, as the subject of a mapping to SNOMED CT-AU. An automap operation can then be executed, which automatically finds SNOMED CT-AU concepts that are likely matches for each code within the source code system.

For maintenance purposes, Snapper can indicate mappings to concepts which have become inactive in new releases of national content. Using historical and replacement associations between concepts, Snapper can also suggest replacements, alternatives or equivalent concepts to update the mapping.

The product includes several “tours” designed to familiarise users with its capabilities. You can access Snapper at <https://ontoserver.csiro.au/snapper2/>.

4 Common Tasks

4.1 Setting up a Terminology Server

This section describes the tasks involved in getting up and running with a copy of Ontoserver that can syndicate national terminology content from the NCTS.

4.1.1 Generating a System Credential

Connecting systems use a Client ID and Client Secret to authenticate to NCTS APIs.

To generate a Client ID and Client Secret, log in to the NCTS Portal and navigate to My Profile. The Client Credentials panel can be accessed through a tab on the lower portion of the form.

The screenshot shows the 'CLIENT CREDENTIALS' section of the NCTS Portal. At the top, there are input fields for 'State' (a dropdown menu), 'Postcode', and 'Country' (a dropdown menu set to 'Australia'). Below these are two tabs: 'USER DETAILS' and 'CLIENT CREDENTIALS'. The 'CLIENT CREDENTIALS' tab is active. A text box explains: 'A Client Credential is required for a connecting system to authenticate with the NCTS Integration or Syndication API. Once generated, your Client Credential can be used to request an access token from our OAuth endpoint. See the NCTS National Services Technical Specification for further details.' Below this is a table with three columns: 'System Name', 'Client ID', and 'Client Secret'. The table contains one entry: 'Ontoserver Test' with Client ID '3c06da97-b5b5-43b8-8e00-65d8491d6d84' and Client Secret 'f7502d3f-b0c4-4855-9334-59ec2d2320f4'. To the left of the table are two buttons: 'Add' and 'Delete'.

Figure 7 - System Credential generation in the NCTS Portal

Click the Add button to generate a new System Credential – you will be prompted for a System Name and System Purpose.

4.1.2 Requesting an Access Token

The Integration and Syndication APIs exposed by the NTS and NSS both implement the [Client Credentials grant type](#) within [OAuth 2.0](#).

To request an access token, simply send a POST request to the following endpoint:

<https://api.healthterminologies.gov.au/oauth2/token>

The POST body of the request must contain the following fields:

Attribute	Data type	Description
grant_type	string	Must be set to "client_credentials"
client_id	string	The Client ID generated by the NCTS Portal
client_secret	string	The Client Secret generated by the NCTS Portal

The request should have a Content-Type header with the value `application/x-www-form-urlencoded`.

An example request follows:

```
POST /oauth2/token HTTP/1.1
Host: www.healthterminologies.gov.au
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=yourclientid&client_secret=yourclientsecret
```

The NCTS publishes a [Postman](#) collection that describes how to authenticate to the NTS, along with other examples.

A Postman collection is a file that contains predefined API requests, and can be imported into Postman. A Postman environment defines some variables relating to a specific deployment of an API, and is used in conjunction with a Postman collection.

You can access the Postman collection, along with its environment file, at the following URL:

<https://www.healthterminologies.gov.au/tools/national-terminology-server/index.html>

For more information on authenticating to the national API services, see [API Security](#) within the NCTS Specifications.

4.1.3 Accessing the FHIR API on the NTS

Once you have an access token, you are ready to begin making requests to the operations supported by the NTS and NSS.

Authentication requires the client to present the access token within an Authorization header upon each request. The access token is preceded by the "Bearer" scheme label.

An example request follows. This performs a "lookup" operation on a concept, using a concept ID.

```
POST /integration/R4/fhir/CodeSystem/$lookup HTTP/1.1Host:
api.healthterminologies.gov.au
Authorization: Bearer [your access token]
Content-Type: application/json

{
  "parameter": [
    {
      "name": "coding",
      "valueCoding": {
        "code": "810311000168105",
        "system": "http://snomed.info/sct"
      }
    }
  ],
  "resourceType": "Parameters"
}
```

A Postman collection and environment file has been defined by CSIRO which you can download and import. This Postman collection covers Ontoserver more generally than the NCTS Postman collection, and includes examples of write operations and some of the more advanced functionality of the product.

<https://ontoserver.csiro.au/docs> (link in left sidebar)

For more information, see [FHIR API](#) within the NCTS Specifications.

4.1.4 Accessing the Syndication API on the NSS

The NSS syndication feed does not require authentication, and can be accessed by issuing a GET request to the following URL:

<https://api.healthterminologies.gov.au/syndication/v1/syndication.xml>

For more information, see [Syndication API](#) within the NCTS Specifications.

4.1.5 Downloading the Ontoserver distribution

Ontoserver is made available as a [Docker](#) image. If you make an enquiry to the Agency Help Centre (help@digitalhealth.gov.au), you will be granted access to a [Quay.io](#) Docker repository that will enable you to download the Ontoserver image, using the Docker client. You will be asked for a Quay.io user name.

You can download Docker at the following URL:

<https://www.docker.com/get-docker>

You can download the Ontoserver image using the following docker commands:

```
docker login quay.io
```

```
docker pull quay.io/aeherc/ontoserver:ctsa-6
```

4.1.6 Running Ontoserver Locally

The easiest way to get up and running with a working Ontoserver is to use [Docker Compose](#). A Docker Compose file describes a Docker application along with its configuration, and gives you some convenient commands for controlling the application.

The following Docker Compose file is a simple example of a configuration that is ready to get syndicated content from the NSS. Note that you will need to enter your Client ID and Client Secret into the environment section.

```
version: '3'
services:
  db:
    image: postgres
    container_name: db
    volumes:
      - pgdata:/var/lib/postgresql/data
  ontoserver:
    image: quay.io/aeherc/ontoserver:ctsa-6
    container_name: ontoserver
    ports:
      - '8443:8443'
    depends_on:
      - db
    environment:
      - authentication.oauth.endpoint.client_id.0=[your client id]
      - authentication.oauth.endpoint.client_secret.0=[your client secret]
    volumes:
      - ontodata:/var/onto
volumes:
  ontodata:
    driver: local
  pgdata:
    driver: local
```

With this file (named docker-compose.yml) in the current directory, Ontoserver can be started using the following command:

```
docker-compose up -d
```

Ontoserver is now running on port 8443, in the background. To view the logs, use the following command:

```
docker-compose logs -f
```

You can shut the Ontoserver stack down using the following command:

```
docker-compose down
```

4.1.7 Loading National Content into a Local Ontoserver Instance

Once Ontoserver is running locally, it can be instructed to download and index a SNOMED CT-AU release by issuing a request to its management API.

The following example would instruct Ontoserver to download and index the February 2017 release:

```
POST /api/indexCodeSystem HTTP/1.1
Host: localhost:8443
Content-Type: application/x-www-form-urlencoded

codeSystemId=http%3A%2F%2Fsnomed.info%2F%2Fsct&codeSystemVersion=http%3A%2F%2Fsnomed.info%2F%2Fsct%2F32506021000036107%2Fversion%2F20170228&validate=true
```

For further documentation on the Ontoserver management API, see <https://ontoserver.csiro.au/docs/5.0/api-swagger.html#Admin>.

4.1.8 Interacting with the Ontoserver FHIR API

You now have a fully functioning Ontoserver, running locally using Docker and loaded up with a release of SNOMED CT-AU from the National Terminology Server.

The CSIRO [Ontoserver Postman collection](#) is the best source of examples of requests to the Ontoserver FHIR API (see link in sidebar).

The Ontoserver FHIR API is an implementation of the NCTS FHIR API, which is formally defined within the [NCTS Specifications](#).

4.2 Receiving Updates

This section provides guidance on how to set up Ontoserver to automatically receive new national terminology releases, and how to update Ontoserver itself.

4.2.1 Setting up a Scheduled Job to Retrieve New Releases

In section 4.1.7, the method of instructing Ontoserver to retrieve a specified release of SNOMED CT-AU was described.

In order to keep an Ontoserver instance up-to-date with the latest releases as they arrive, this process needs to be repeated at a regular interval.

The following [Ruby](#) script provides an example of such a script:

```
require 'net/http'
require 'uri'
require 'rEXML/document'

# URL of the syndication feed to retrieve items from, this is the URL for the NTS.
SYND_URL = 'https://api.healthterminologies.gov.au/syndication/v1/syndication.xml'
# URL of the admin API of the Ontoserver instance that will index the content.
ONTO_ADMIN_URL = 'https://localhost:8443/api'

NCTS_URI = 'http://ns.electronichealth.net.au/ncts/syndication/asf/extensions/1.0.0'

if ARGV.length < 1
  puts "Usage: #{__FILE__} [Content Item Identifier]"
  exit 1
end

system = ARGV[0]
feed = Net::HTTP.get URI(SYND_URL)
doc = REXML::Document.new feed
# Query all entries with a `ncts:contentItemIdentifier` value that is equal to
# the command-line argument.
query = "/feed/entry[ncts:contentItemIdentifier[text() = '#{system}']]"
entries = REXML::XPath.match doc, query

# Extract the Content Item Version for each entry.
versions = entries.map do |entry|
```

```

    REXML::XPath.first(entry, 'ncts:contentItemVersion/text()')
  end

  # Send an index request to Ontoserver for each version found within the
  # syndication feed.
  versions.uniq.each do |version|
    print "Indexing item (system='#{system}' version='#{version}')... "
    # `indexCodeSystem` is documented here: http://ontoserver.csiro.au/docs/5.0/api-swagger.html#Admin
    query = {
      codeSystemId: system,
      codeSystemVersion: version,
      # Set this to true to download the file each time and validate that it
      # matches content already indexed within Ontoserver.
      validate: false
    }
    url = URI("#{ONTO_ADMIN_URL}/indexCodeSystem")
    Net::HTTP.start(url.host, url.port) do |http|
      # Set timeout to 20 minutes, to allow for download time on slower
      # connections.
      http.read_timeout = 1200
      request = Net::HTTP::Post.new(url)
      request.set_form_data(query)
      response = http.request(request)
      if response.is_a?(Net::HTTPSuccess)
        puts 'OK'
      else
        # If the request fails, output the response body so that we can work out
        # what the problem is.
        puts "#{response.code} #{response.message}"
        puts response.body
        puts
      end
    end
  end
end
end

```

4.2.2 Updating the Ontoserver Docker container

Ontoserver can be updated by instructing Docker to retrieve the latest image from Quay.io.

The image value controls which tagged version of the image will be used when Docker Compose refreshes the Docker containers.

As an example, to use version 6.2.0 of Ontoserver, the docker-compose.yml file should look like this:

```

services:
  ontoserver:
    image: quay.io/aeherc/ontoserver:ctsa-6.2.0
  ...

```

To use the latest published image supporting FHIR r4:

```

services:
  ontoserver:
    image: quay.io/aeherc/ontoserver:ctsa-R4
  ...

```

The full list of available tags can be viewed here (requires AEHRC Quay.io access):

<https://quay.io/repository/aeherc/ontoserver?tab=tags>

Once the docker-compose.yml has been updated, a running Ontoserver instance can be upgraded to the nominated version using the following command:

```
docker-compose up -d
```

4.3 Mapping between Local and National Terminologies

This section provides guidance on how to complete some common tasks associated with integrating with the NCTS.

4.3.1 Using Snapper with the National Terminology Server

Snapper can consume terminology from any FHIR conformant terminology server endpoint, including the National Terminology Server.

The main limitation with the NTS is that it is read-only. In order to save authored artefacts, you will need to point Snapper to a terminology server that allows write operations.

Snapper is a pure client-side web application. You can access Snapper at <https://ontoserver.csiro.au/snapper2/>

Snapper ships pre-configured to point to the NTS, and it is selected by default. If it is not selected, you can select the NTS configuration by clicking the gear icon in the top-right corner of the screen.

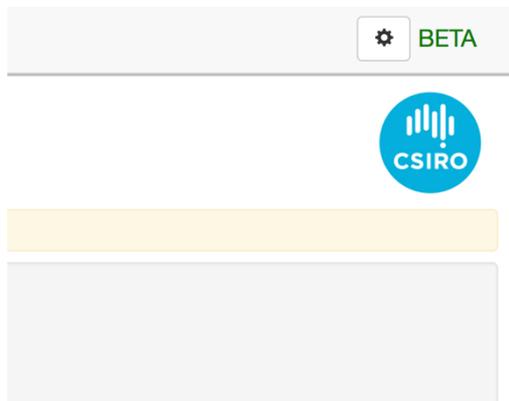


Figure 8 - Location of settings menu within Snapper

The NTS endpoint is <https://api.healthterminologies.gov.au/integration/R4/fhir>.

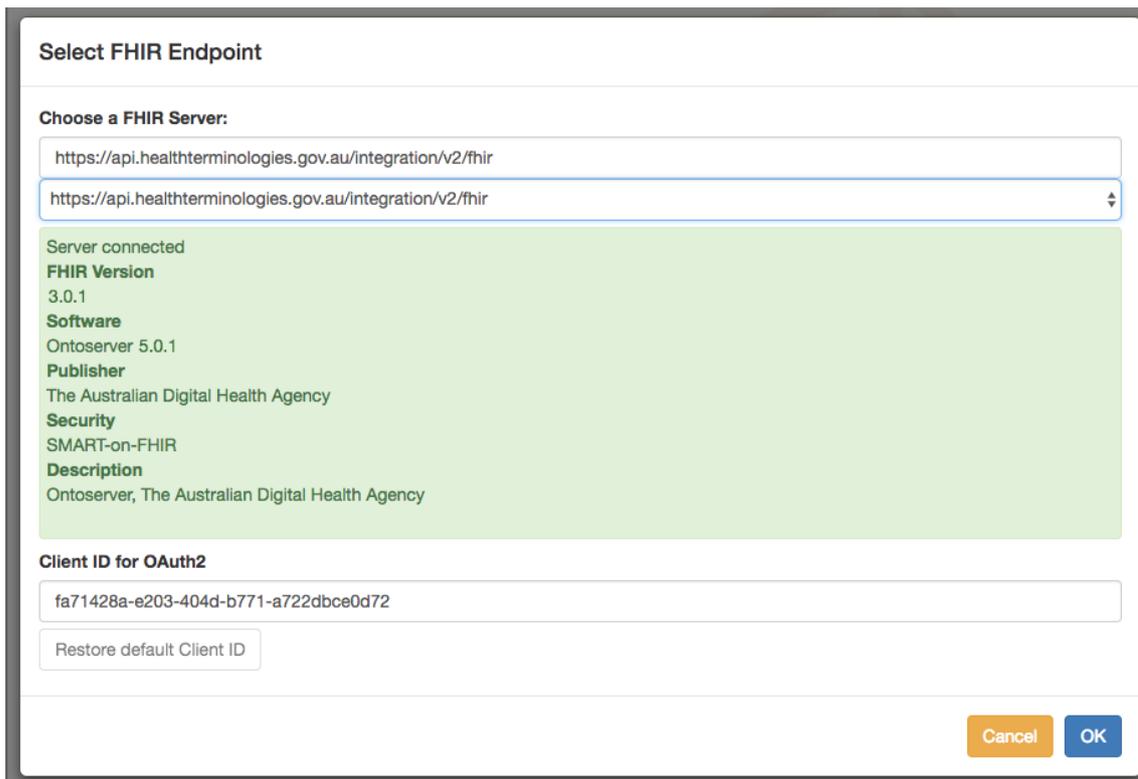


Figure 9 - NTS endpoint selection within Snapper

For more information on how to use Snapper, take the relevant tours available from the Snapper home screen.

4.3.2 Using Snapper with a Local Ontoserver Instance

Use of the copy of Snapper hosted by NCTS requires an Ontoserver instance that is secured by TLS. This is because Snapper is served via HTTPS, and will refuse to connect to a server which does not have a valid, trusted TLS certificate.

If you do not have an Ontoserver instance set up with a TLS certificate issued by a Certificate Authority (CA), you will need to generate a self-signed TLS certificate. Follow the instructions in section 4.3.3 to generate a certificate and configure Ontoserver to use it. You will then need to trust the root certificate at the operating system level.

To configure Snapper to point to your local Ontoserver, enter the URL of its FHIR API in “Select FHIR Endpoint” dialogue, for example:

`https://ontoserver.local:8443/fhir`

If the connection is successful, you should see a green box that reads “Server connected”, along with the details of the version of Ontoserver that you are running.

4.3.3 Generate a Self-Signed TLS Certificate for Ontoserver

If you want to serve Ontoserver using TLS for test purposes, you may need to generate a self-signed TLS certificate.

The following example shows the procedure to generate a self-signed certificate using OpenSSL, then package it up as a PKCS #12 archive. First create a directory for your certificates and keys:

```
mkdir certs && cd certs
```

Then you need to create the following file in the certs directory (ontoserver.local.ext):

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = ontoserver.local
```

The following commands create your certificate and key files:

```
openssl genrsa -out rootCA.key 2048
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout onto.key -out onto.crt
openssl req -new -newkey rsa:2048 -sha256 -nodes -keyout onto.key -out onto.csr
openssl x509 -req -in onto.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
onto.crt -days 365 -sha256 -extfile ontoserver.local.ext
```

You will be prompted to answer a number of questions. The only important one is the common name, which will be the local host name you will use for your Ontoserver instance. The example we will use is `ontoserver.local`.

```
openssl pkcs12 -export -in onto.crt -inkey onto.key -out keystore.p12 -name
"ontoserver"
```

You should now have a directory named `certs`, which contains three files: `onto.key`, `onto.crt` and `keystore.p12`. You will be asked for a password to secure the PKCS #12 file, take a note of this as you will need it for the Ontoserver configuration.

Add the following lines to the `ontoserver` service within the `docker-compose.yml` file:

- `server.ssl.key-store=/etc/ssl/certs/keystore.p12`
- `server.ssl.key-password=[your keystore password]`
- `server.ssl.key-store-password=[your keystore password]`

Add the following line to the `volumes` section of the `ontoserver` service:

- `./certs:/etc/ssl/certs`

Here is the updated `docker-compose.yml`:

```
version: '3'
services:
  db:
    image: postgres
    container_name: db
    volumes:
      - pgdata:/var/lib/postgresql/data
  ontoserver:
    image: quay.io/aeherc/ontoserver:ctsa-6
    container_name: ontoserver
    ports:
      - '8443:8443'
  depends_on:
    - db
  environment:
    - authentication.oauth.endpoint.client_id.0=[your client id]
    - authentication.oauth.endpoint.client_secret.0=[your client secret]
```

```
- server.ssl.key-store=/etc/ssl/certs/keystore.p12
- server.ssl.key-password=[your keystore password]
- server.ssl.key-store-password=[your keystore password]
volumes:
  - ontodata:/var/onto
  - ./certs:/etc/ssl/certs
volumes:
  ontodata:
    driver: local
  pgdata:
    driver: local
```

An entry will need to be added to your local hosts file (/etc/hosts on Linux/Mac, c:\Windows\System32\Drivers\etc\hosts on Windows):

```
127.0.0.1 ontoserver.local
```

Now you can re-run `docker-compose up -d`, which will update Ontoserver with the new configuration.

5 Design Considerations

5.1.1 Security features and considerations

While Ontoserver does not ship with its own authentication system, it does implement a role-based access control system and the ability to integrate with an [OAuth 2.0](#)-compatible authorisation server.

The authorisation server must have the ability to issue [JSON Web Tokens](#) (JWT). A JWT is simply a token that contains a payload of information, such as claims and expiry, which is signed with a shared secret. An application, such as Ontoserver, can verify the authenticity and integrity of the JWT without the need to interrogate an authorisation server directly.

The payload of a typical JWT used to authenticate with Ontoserver looks like this:

```
{
  "exp": 1506947837,
  "user_name": "somebody@somewhere.com",
  "authorities": [
    "ROLE_FHIR_READ",
    "ROLE_FHIR_WRITE"
  ],
  "jti": "d27686bd-d439-4368-b5db-16431740ee38",
  "client_id": "ac8fa7a3-cd05-4f7f-8756-095b4dcbe330"
}
```

This JWT grants read/write access to the FHIR API. The roles available within Ontoserver are as follows:

ROLE_FHIR_READ	Read access to the FHIR API
ROLE_FHIR_WRITE	Write access to the FHIR API
ROLE_API_READ	Read access to the admin API
ROLE_API_WRITE	Write access to the admin API
ROLE_SYND_READ	Read access to the syndication API
ROLE_SYND_WRITE	Write access to the syndication API

The `exp` parameter represents the expiry time of the token. Attempts to use the token after this time will be refused by Ontoserver.

The `client_id` parameter is a unique identifier for the client application, and can be used by the authorisation server within its authorisation logic, and to enforce allowable redirection URIs on a per-application basis.

The `jti` parameter is a unique identifier for the token. Its presence is generally used to mitigate against replay attacks.

5.1.2 Designing for high performance

Ontoserver has been designed to facilitate high performance read operations on large code systems. Terminology data is stored in an index for fast searches, lookups and code validation.

Response time varies greatly depending on the type of query Ontoserver is asked to fulfil. The FHIR API and FHIR Terminology Service API provide a very rich set of operations that can range in cost from very cheap to long-running. Performance is also dependent upon the size of the code system.

Ontoserver has a configuration parameter called `ontoserver.fhir.too.costly.threshold`. This specifically places a limit on the number of results returned from a ValueSet expansion, which can be a costly operation when the underlying CodeSystem has a large number of codes.

The simplest way to increase the capacity of a single Ontoserver instance is to install a HTTP cache in front of it. Ontoserver supports this well by setting the appropriate HTTP headers to inform any caches between it and the user of which requests are cacheable.

There are a number of HTTP cache implementations which can work well in front of Ontoserver with minimal configuration, such as [Nginx](#) and [Varnish](#).

If a deployment is using the terminology server in a read-only fashion, e.g. searches, lookups, code validation, then horizontal scaling is also an option. Multiple Ontoserver instances can be set up behind a load balancer which can share requests between the instances, reducing the load on any one instance.

5.1.3 Designing for high availability

High availability for a read-only Ontoserver cluster is usually best achieved by subscribing more than one instance to a load balancer, and hosting the instances in multiple geographically-diverse locations.

This provides redundancy if an instance fails for some reason, or needs to be taken down for maintenance. It also mitigates against a failure of the network at any one location due to a power outage, human error or disaster scenario.

Another advantage of having multiple instances is that maintenance and updates can be performed on individual instances without causing downtime for users.

Because Ontoserver is distributed as a Docker image, Docker-based architectures such as [Swarm](#), [Kubernetes](#) and [Amazon ECS](#) can also ease the task of designing redundancy into the system.

6 Deployment Scenarios

6.1 Design-time Applications Directly Accessing the NTS

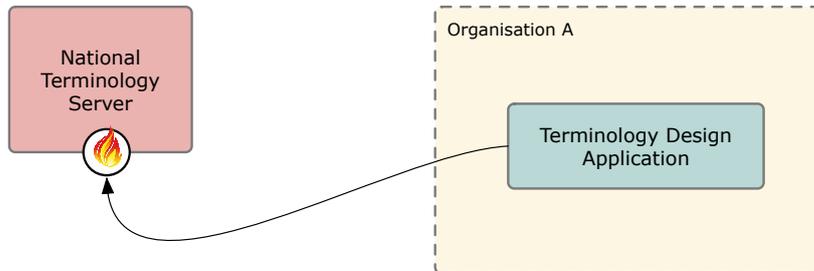


Figure 10 – Terminology design application accessing the NTS

One of the simplest ways that terminology users can integrate with the NCTS is to use the FHIR API of the NTS for read-only queries on the national terminologies.

We don't permit production clinical applications to interact directly with this API, but applications such as terminology authoring tools and research tools can. It is also valid for batch processes to query the NTS periodically for the purpose of retrieving terminology data using FHIR operations.

See section 4.1 for the details of setting up a system credential to allow for programmatic access to the NTS FHIR API.

6.2 Local Terminology Service for Clinical Information Systems

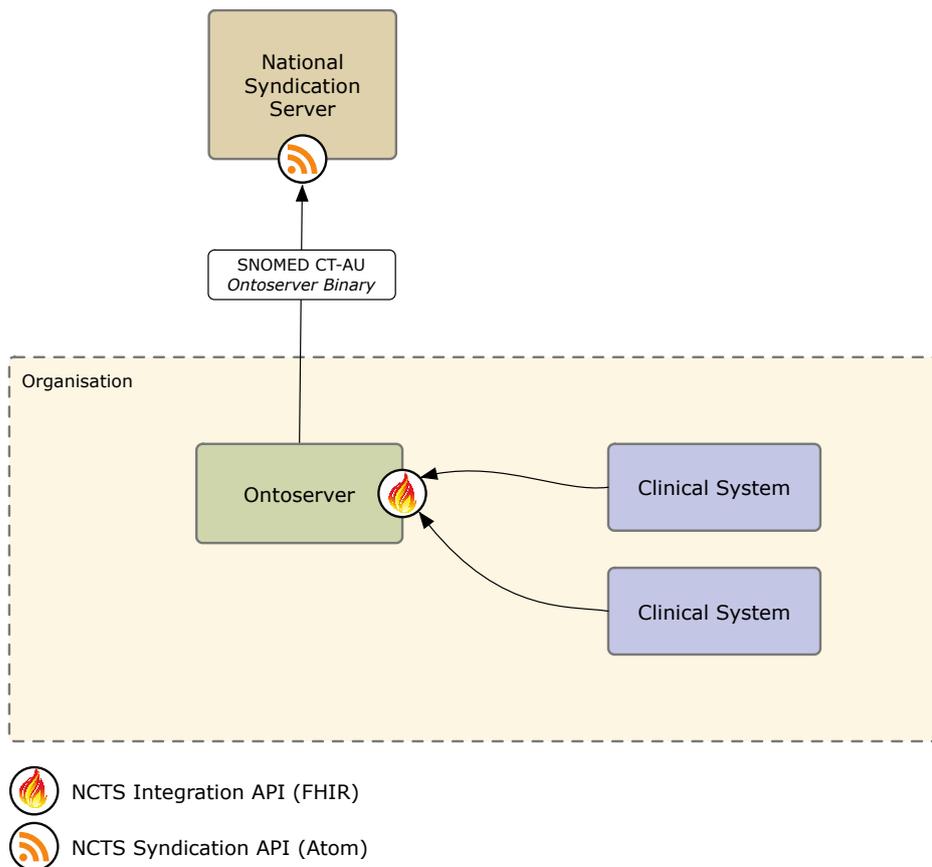


Figure 11 - Basic local terminology service being accessed by CIS

Local terminology services for production clinical applications can be enabled by deploying an Ontoserver instance. National terminology can be syndicated to the local Ontoserver instance through the National Syndication Server.

This is a simple deployment scenario, involving only one Ontoserver instance and providing services for only SNOMED CT-AU. The next diagram depicts a more complex scenario.

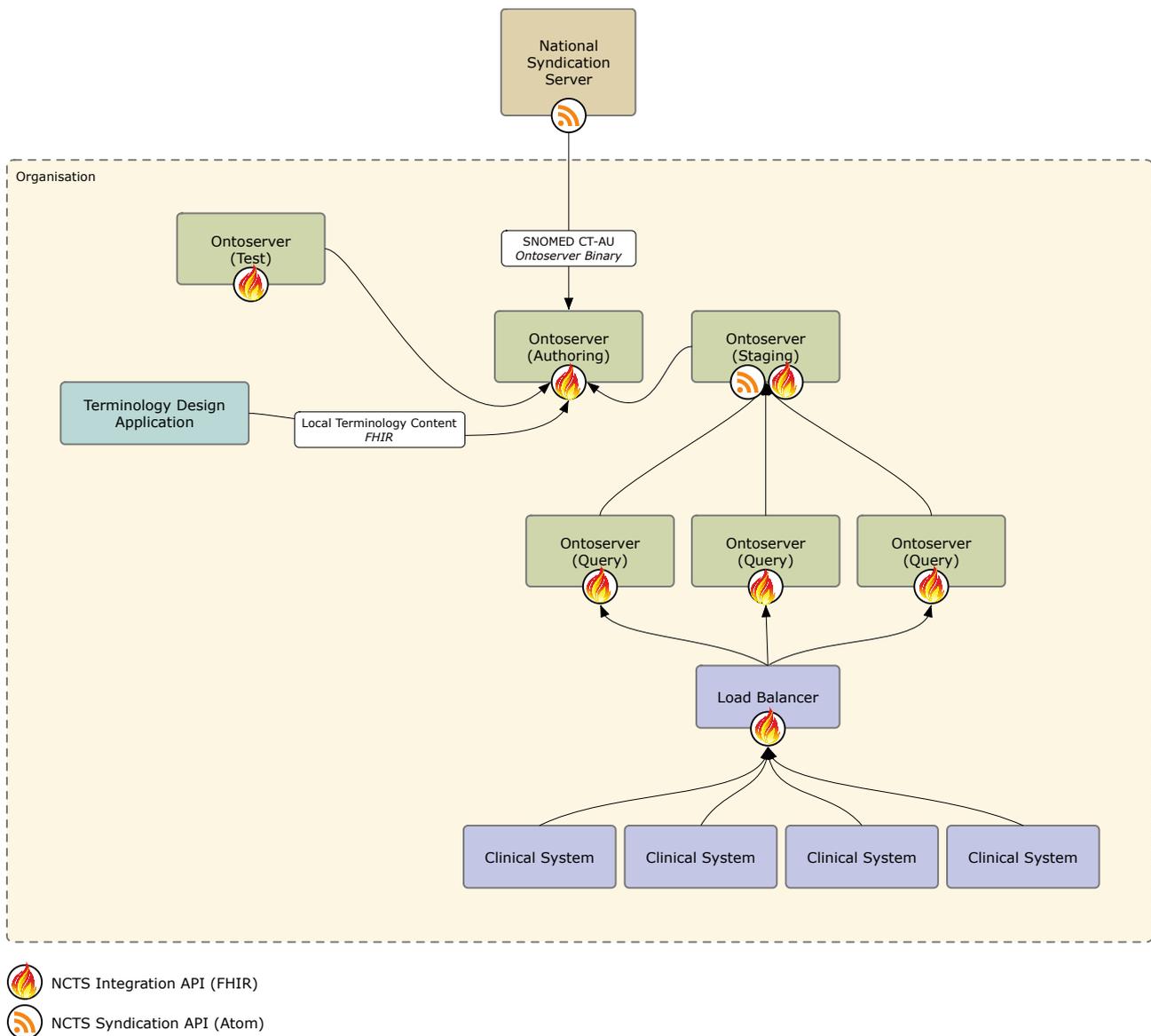


Figure 12 - Complex local terminology deployment, with local authoring

In this scenario, national terminologies are syndicated to an Ontoserver instance which is also used for authoring local terminologies and mappings. Terminology resources are then transferred to a test server where quality assurance processes are carried out ahead of an internal release.

Once a set of terminology resources is ready for release, it is transferred to a staging terminology server. This server has its syndication interface enabled.

A number of query-optimised Ontoservers have been set up behind a load balancer, for performance and availability reasons. These Ontoserver in `ClinicalTerminology_SNOMEDCT-AU_ReleaseNote_v20210204stances` can be instructed to retrieve the syndication feed of the staging server and download any new content. SNOMED CT-AU binaries are downloaded using the Syndication API, while FHIR resources are retrieved using links from the syndication feed that point to the FHIR API.

6.3 Local Terminology Services with Multi-Tiered Syndication

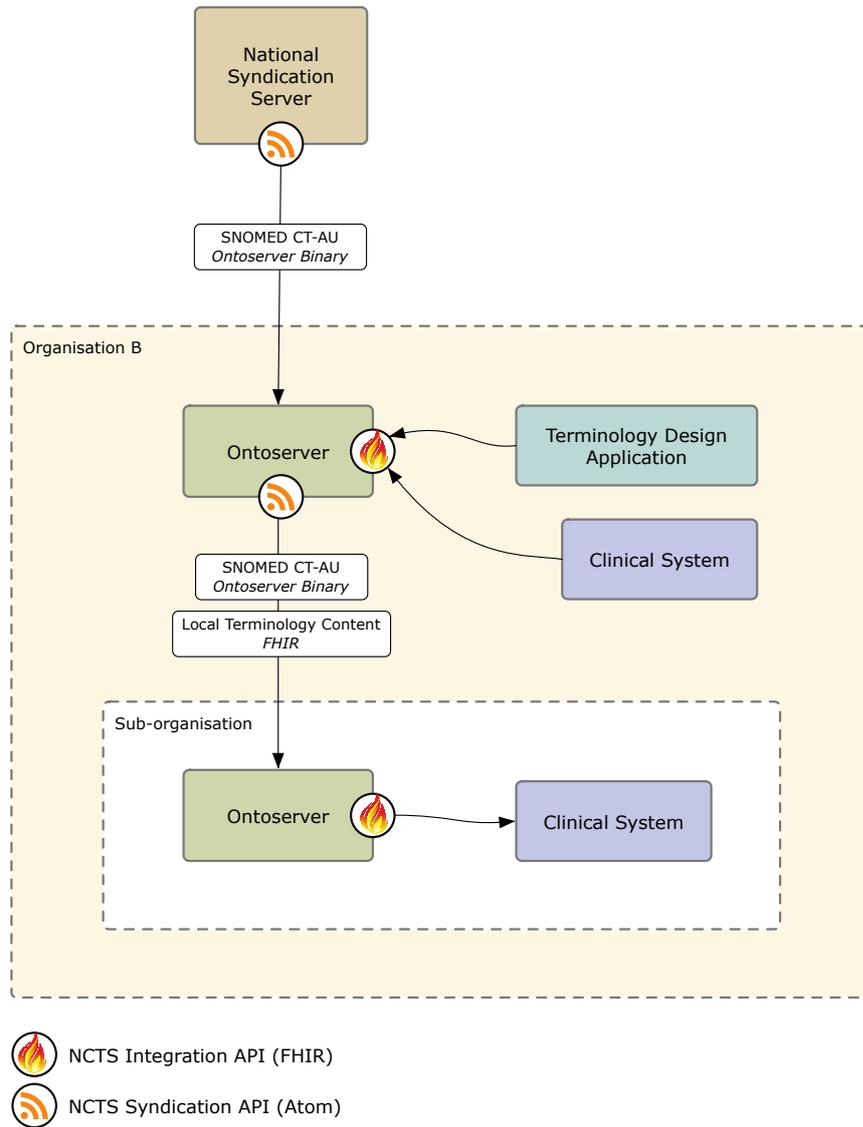


Figure 13 - Local terminology services with multi-tiered syndication

Ontoserver can expose its own syndication feed, which can be consumed by other downstream Ontoserver instances.

This enables the creation of multi-tiered syndication architectures, which can be useful for large organisations which have central functions, but also sub-organisations with their own unique local requirements. Local content can be injected into the syndication hierarchy at any level.

6.4 Syndication for Vendors

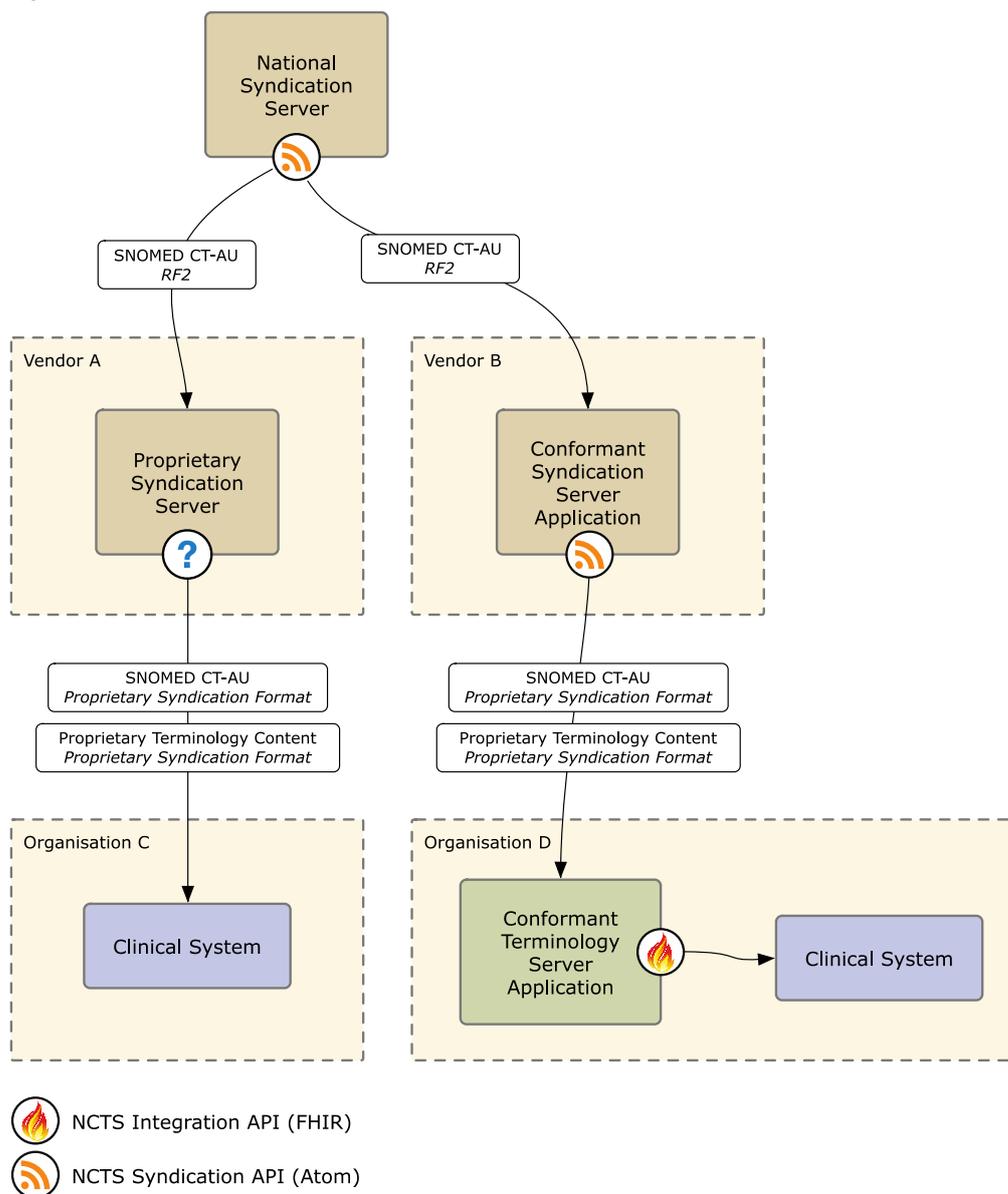


Figure 14 - Syndication option for vendors

The open and distributed architecture of NCTS presents numerous options for vendors looking to retrieve and incorporate national terminology into their products in more efficient and automated ways.

In this example, Vendor A has integrated with the NSS for the sole purpose of retrieving the RF2 files for SNOMED CT-AU on a monthly basis.

Vendor B has taken the decision to integrate more deeply, implementing the NCTS interface specifications for syndication and the FHIR API. This delivers interoperability benefits for users. FHIR and NCTS compatible tooling now works with their servers, making them better candidates for central terminology infrastructure.

6.5 Terminology Validation for EMRs

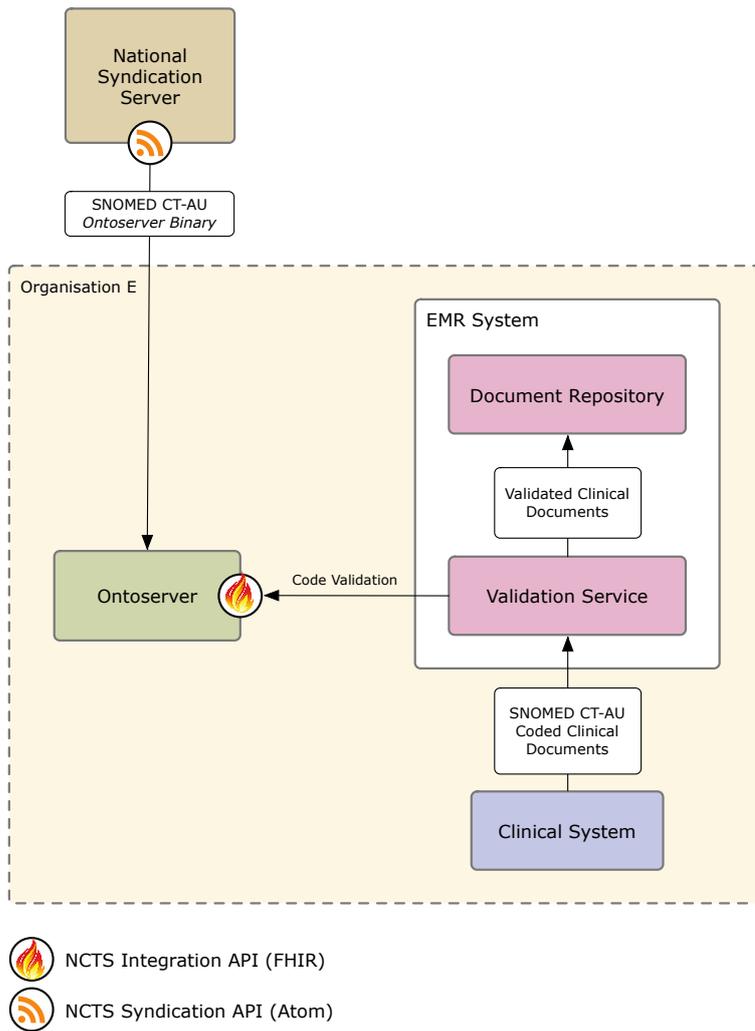


Figure 15 - Terminology validation for EMRs

Validation of codes within structured clinical documents can be achieved through a locally deployed read-only Ontoserver instance. The Ontoserver instance can be kept up-to-date with the latest national terminologies by subscribing to the national syndication feed.

6.6 Local Terminology Server for Authoring, with Authentication

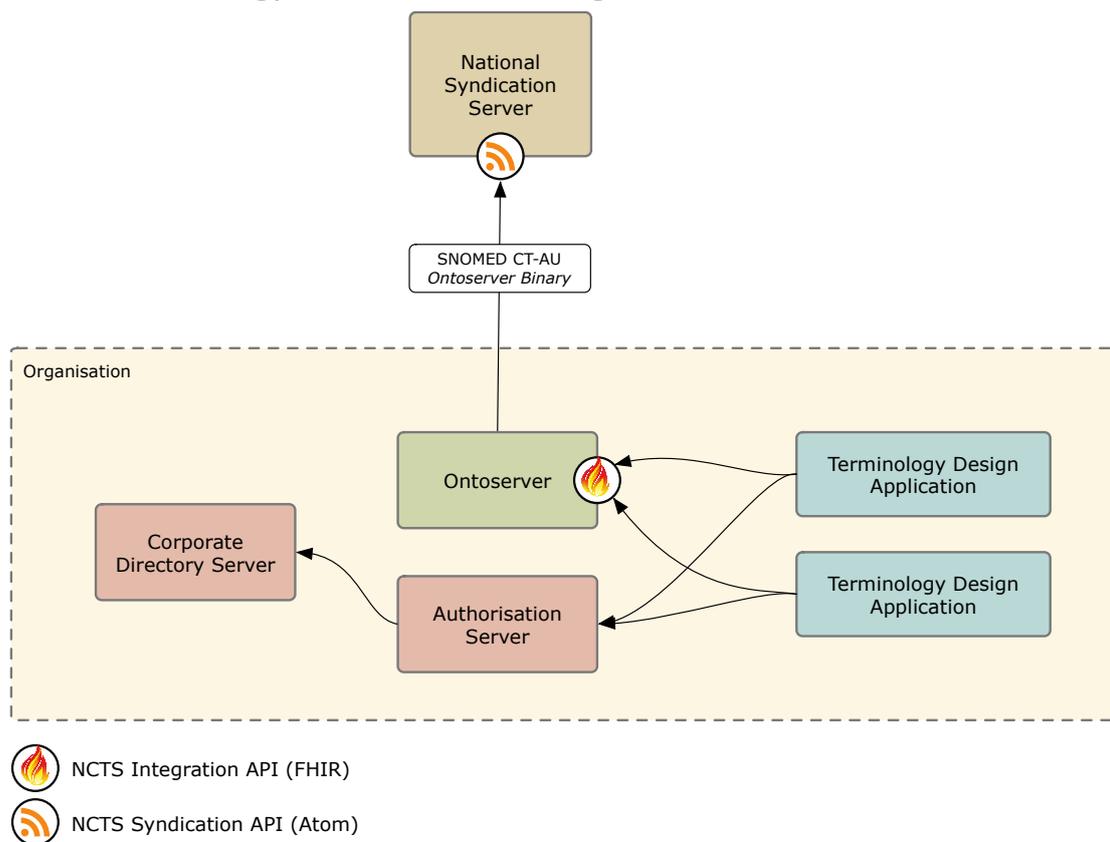


Figure 16 - Local authoring server with authentication

This scenario shows an Ontoserver instance that has set up read/write mode for local authoring.

Authentication was required, so an OAuth 2.0 authorisation server is set up to issue JSON Web Tokens to client applications for authentication to Ontoserver.

The organisation's existing corporate directory server is queried by the authorisation server when making decisions on whether to authorise requests.