

DATALINE SOFTWARE LIMITED ©

---

# Feature set for SnAPI™

---



Suite 6 • Clarence House • 30-31 North Street • Brighton • East Sussex

Phone 01273 324939 • Fax 01273 205576 • [www.dataline.co.uk](http://www.dataline.co.uk)

## Table of Contents

Prologue .....	4
Feature set for SnAPI™ .....	5
Introduction .....	5
About this document .....	6
GUIDS.....	6
Return status and Error Messages.....	7
Using wm_GetConcept .....	7
Lexical Matching.....	8
Whole Word Matching.....	9
Prefix Matching .....	9
Suffix Matching .....	9
NHS Extension .....	9
SubSets.....	10
Favourites.....	12
Accessing the SnAPI web service .....	13
Authentication (GUID) .....	13
The SnAPI web service URL.....	13
Acronyms .....	13
Example SnAPI application using C# .....	14
Required Software .....	14
Source Code Example Syntax.....	14
Create a Visual Studio project.....	14
Add SnAPI™ as a service reference .....	15
Add the SnAPI using directive .....	16
Error handling (try-catch) .....	16
Creating an instance of the SnAPI web service.....	16
Calling the GetSuffix SnAPI web method.....	17
Testing the program .....	17
Troubleshooting.....	18
Invalid GUID .....	18
Testing connectivity .....	18
Compilation Errors .....	18
Further support.....	18
Using other SnAPI Web Methods .....	19
Get SubSets .....	19
Get Favourites .....	19
Get Description .....	19
GetConcept .....	19
GetNeighbours .....	19
SnAPIService_ClientExample - Program.cs source code.....	20
Example SnAPI application using Java.....	21

Required Software .....	21
Source Code Example Syntax .....	21
Install the SOAP Web Services plugin in NetBeans .....	21
Create a NetBeans project .....	21
Add a Web Service reference to the project .....	21
Add the SnAPI import directive .....	22
Error handling (try-catch) .....	22
Creating an instance of the SnAPI web service .....	23
Calling the GetSuffix SnAPI web method .....	23
Testing the program .....	24
Troubleshooting.....	25
Invalid GUID .....	25
Testing connectivity .....	25
Compilation Errors .....	25
Further support.....	25
SnAPIService – main.java source code.....	26
Support.....	27
SNOMED Forum .....	27
Email .....	27
Dataline Software Ltd .....	27
Appendix 2: Features deferred until Beta 2 .....	28
Cross Map Support .....	28
MapSetId .....	28
TargetCodes .....	28
Ancestor Matching.....	29
Transitive Closure .....	30
AncestorId .....	31
DecendentId .....	31
The difference between DecendentId and AncestorId .....	32

## Prologue

*"The application of what we know already will have a bigger impact on health and disease than any drug or technology likely to be introduced in the next decade"*

Dr Muir Gray, Director of clinical knowledge for the NHS.

In translational research, the ability to retrospectively apply structure to free text through coding is important. Medical notes tend to defy the principals of natural language and using coded sentences to act as "meaningful snippets of text", aid in the process of parsing electronic records. That said, if the notes were indeed coded at the point of entry, translational research and its implications for public health, would become a great deal easier.

To suggest the appropriate text at the point of entry is the goal for the future and for this we need a fast, reliable and "helpful" interface to SNOMED. SNOMED is a large data set of which only a small proportion is relevant to any one practitioner. Subsets reduce SNOMED into smaller meaningful data sets, meaningful and helpful to specific groups of practitioner.

It is useful to develop this idea further and allow individual SubSets (*FavouriteSets*) to be created for each individual practitioner. It is likely that a practitioner will require more than a single *FavouriteSet*, each one created being tailored to a specific working practice. Moreover, the "most used" Concepts within a particular *FavouriteSet* could be given a concise shortform or synonym, allowing them to be called up with minimum key strokes.

To be useful in suggesting appropriate text, we need to maximise the signal and minimise the noise. Good textual matching, fast enough for type ahead, that presents Concepts preferentially from a personalised *FavouriteSet* would be a step towards this goal. If the *FavouriteSet* evolved through usage, so the system "learns" the practitioner's individual preference we reach a point where SNOMED coding is actually helpful to the practitioner, rather than an impediment.

## Feature set for SnAPI™

### Introduction

SNAPI Beta 1 has a particular design goal: To support all the functionality of the existing SnoFlake portal in a simple web service API. SnAPI Beta 2 seeks to retain the simplicity of Beta 1 whilst moving SnAPI beyond a “look up” API and enhance it with the requisite functionality to support cross maps, subsumption testing, and ancestor matching through transitive closure.

In Beta 1, new search features have been added and ALL of the various matching techniques described below can be used in conjunction with each other in any combination.

In Beta 2, we will introduce the negation of such things as Subsets and FavouriteSets. Where, for example, in Beta 1 this document states:

*“If you want those Concepts that appear in more than one Subset, say “NHS dm+d Gluten Free” combined with “NHS dm+d Preservative Free” then pass in 68201000001134, 68301000001139. Any number of Subsets can be combined in this manner. “*

In Beta 2 we will support the following idea:

*“If you want those Concepts that appear in one SubSet BUT NOT in another, say “NHS dm+d Gluten Free” but NOT “NHS dm+d Preservative Free” then pass in 68201000001134, -68301000001139. (note the minus ‘-’ sign). Any number of SubSets can be combined in this manner. “*

NOTE: There is no expectation that the user will actually provide these keys (e.g. SubSet Ids ) without assistance from the SnAPI interface. A service is provided to list all SubSets which could be used to populate a multi-select pick list.

## About this document

This document sets out to describe the Functional API supported by SnAPI in terms of six web services:

wm_GetConcept	The main function which returns eligible Concepts in keeping with the filter criteria provided.
wm_GetDescription	Returns the various descriptions pertaining to a particular Concept
wm_GetFavourite	Returns a list of Favourites – owned by the User. These can be passed to <b>wm_GetConcept</b> as filter criteria.
wm_GetNeighbour	Returns a list of immediate ancestors and immediate children (i.e. to one level only) as described by the Relationship table, each being marked as either 'P' or 'C' indicating whether they are a Parent or Child of the Concept specified.
wm_GetSubSet	Returns a list of SubSets. These can be passed to <b>wm_GetConcept</b> as filter criteria.
wm_GetSuffix	Returns a list of Suffixes. These can be passed to <b>wm_GetConcept</b> as filter criteria.

The main interface is controlled by **wm\_GetConcepts** which returns eligible Concepts from the Concept table. **wm\_GetDescription** and **wm\_GetNeighbour** each take a ConceptId as argument. **wm\_GetDescription** return the fuller description(s) from the description table - **wm\_GetNeighbour** returns the immediate children and parents from the Relationship table. In this context, the term “immediate” means “immediate proximity” i.e. parent or child NOT grandparent or grandchild. Each record is marked with either a 'P' or 'C' to indicate Parent or Child of the specified Concept. We use this in SnoFlake to populate the “snow flake” relationship object in the right hand pane of the browser. **wm\_GetSubSet**, **wm\_GetFavourite** and **wm\_GetSuffix** take no arguments (save the GUID which is described next) and are used to implement the usual pick lists for SubSets, Favourites and Suffixes. These pick lists assist the user in choosing SubSet or favourite keys or suffixes for subsequent use as arguments to **wm\_GetConcept** .

## GUIDS

All of the services listed above have three things in common:

1. They all return an integer value to indicate success or failure
2. They all return an error message (string) to indicate the type of failure
3. They all take a GUID as argument, which is a unique 36 character string.

The GUID is unique to each User of the SnAPI interface and is used to validate Users. If you try to use any of the SnAPI services with a missing or invalid GUID the function will return Error 4: “Invalid GUID”. Your GUID is provided when you register on the SnoFlake web site:

<http://snomed.dataline.co.uk/>

The GUID can be found under the “Account & Favourites” menu option at the bottom of the left hand pane.

## Return status and Error Messages

Return status and error messages currently supported in this release (Beta 1) are as follows:

RetCode	Return String
0	Success
1	None numeric Key List
2	Malformed Suffix or TargetCodes String
3	Buffer overflow: SQL Syntax too large
4	Invalid GUID
5	Maximum Number of SubSets exceeded
6	Invalid SubSet
7	Maximum Number of Favourites exceeded
8	Invalid Favourite
ELSE	Unknown Error

## Using wm\_GetConcept

Most of the functions described above are relatively simple to use taking just the GUID as argument or GUID plus ConceptId. These functions are adequately explained in the next section: Using SnAPI. The exception is wm\_GetConcept which takes a rich set of arguments and is described in detail here:

wm\_GetConcept (

*Str* As varchar ,

*SubSetId* As varchar ,

*FavouriteId* As varchar ,

*Suffix* As varchar ,  
*NameSpace* As varchar ,  
*MaxResultSetSize* As int ,  
*MatchType* As int ,  
*GUID* As varchar,  
*RetStr* As ) As SqlResultStream

In this section we will describe each of the arguments to `wm_GetConcept` in detail.

Str	A sentence of words or the prefixes of words (depending upon the setting of MatchType). Only SNOMED FSNs (Fully Specified Names) which match ALL of the words in the sentence (in any order) will be returned. See Lexical Matching for more information
SubSetId	A comma delimited list of SubSet ids. Found by calling <code>wm_GetSubSet</code> . See SubSets (below) for more information.
FavouriteId	A list of comma delimited Favourite ids. Found by calling <code>wm_GetFavourite</code> . They are created within the SnoFlake web portal. See Favourites (below) for more information.
Suffix	A comma delimited list of Suffixes. These are the words in brackets appended to the end of each FSN. By including a Suffix list, only SNOMED FSNs with the specified suffix will be returned. See Suffix Matching (below) for more information.
NameSpace	Allows the use of extensions to the core SNOME data set. Currently only the Core dataset and the NHS extension are supported. See NHS Extension (below) for more information.
MaxResultSetSize	Determines the maximum number of records that will be returned. If this is less than 1 or greater than 1000 then we default to 1000.
MatchType	Specifies if we are using prefix matching or whole word matching. 0 = Whole word matching 1=Prefix matching See Lexical Matching (below) for more information.
GUID	The unique identifier that determines the User of the SnAPI interface. See GUID (above) for more information.
RetStr	The return string which provides the requisite error message. See Return status and Error Messages (above) for more information.

### Lexical Matching

Searching for SNOMED Concepts by entering words in any order has been enhanced to support prefix matching. Users can now choose either whole word matching (as per the current version) OR prefix matching.

### Whole Word Matching

*Whole Word Matching* is the default technique used in the current version of SnoFlake. Words which are not part of SNOMED are simply discarded and play no part in the search. SNOMED Concepts are returned only if they contain ALL of the “un-discarded” words and are ordered by “relevance” the most relevant being returned first.

“Relevance” is calculated using a simple “weighting” function which compares the text, specifically the *Fully Specified Name (FSN)*, of each SNOMED Concept with the words entered by the user and determines a ratio between words which match and those which do not. The subsequent “weight” is used to order the result set.

### Prefix Matching

*Prefix Matching* works in precisely the same way as *Whole Word Matching* except that it matches any words in the SNOMED Concept’s FSN that begin with the words entered by the user.

As a simple example, in prefix matching mode, “hear” will match “hearing” and “heart” whereas in whole word matching “hear” will only match “hear”. To extend this example a little “hear mur fun” will return “Functional Heart Murmur (finding)” in prefix matching mode.

### Suffix Matching

All active SNOMED Concepts have a FSN which ends in a suffix contained in parentheses - e.g. (finding) or (situation). The current version of SNOFlake supports *Suffix Matching* but is limited to a single suffix only. The next release of SNOFlake will fall in line with SnAPI and multiple suffix matching will be supported as it is in this Beta1. For example, to find all Concepts which have the suffix (finding) or (situation) we pass in “(finding),(situation)”. Passing in nothing disregards suffix matching and we return all Concepts irrespective of their suffix.

### NHS Extension

The *NHS Extension* to the Core SNOMED CT is included in this release and (to assist with this) we require the idea of a *NameSpace* - each *Extension* to the core SNOMED CT has a unique *NameSpace*. The *NameSpace* for the *NHS Extension* is 1000000. Although the SNOMED Core

CT has no *Namespace* we have nominally given it the *Namespace* of zero. To allow for future *Extensions* to SNOMED we support multiple *Extensions*.

Examples of *Extensions* in SnAPI Beta 1 are as follows:

<b>NameSpace</b>	<b>Extension</b>
0	SNOMED Core only
1000000	NHS Extension only
0,1000000	SNOMED Core + NHS Extension

Passing nothing as a *Namespace* defaults to the SNOMED Core only.

### SubSets

Just as *Extensions* extend SNOMED, *SubSets* reduce it. There are many *SubSets* released with SNOMED CT and in SnAPI Beta 1 our initial intention is to support those in the table below, although more can be added on request.

<b>SubSet Id</b>	<b>SubSet Name</b>
33511000000130	Administration Medication
33501000000133	Administrative Procedure
33711000000137	Adverse Reaction Event
33311000000136	Adverse Reaction Propensity
33541000000131	Alcohol
33601000000132	Allergies and Adverse Reaction Groups
33631000000135	Allergy Event
33301000000138	Allergy Propensity
33731000000131	Blood Pressure
33361000000139	CDA Care Setting Type
33371000000132	CDA Document Type
33401000000134	CDA Encounter Type
33641000000130	Child Health Behavioural & Special Needs
33651000000133	Child Health Community Referrals
33521000000138	Device Type
33531000000136	Diagnosis
33741000000136	Discharge
33331000000130	Document type
33721000000134	Drug - allergy and adverse reaction constraint
33431000000137	Encounter disposition
33591000000135	Encounter type
33681000000136	Endoscopy Diagnosis
33691000000139	Endoscopy Procedures
33461000000133	Family History
33421000000139	Finding
33291000000137	Food - allergy and adverse reaction constraint

33441000000132	Food allergens
33411000000131	Height
33981000000133	Incidental Location Type
34161000000139	Instantiable CRE (care record element) types
33321000000133	Investigations
33551000000134	Lifestyle
33351000000137	Link Assertion MIM
34211000000130	Navigable or instantiable CRE (care record element) types
34111000000136	NHS Laboratory Collection Method SubSet
34101000000138	NHS Laboratory Investigation Method SubSet
34091000000132	NHS Laboratory Investigation SubSet
34131000000130	NHS Laboratory Isolate SubSet
34081000000130	NHS Laboratory Laterality SubSet
34071000000133	NHS Laboratory Morphology SubSet
34121000000133	NHS Laboratory Pre-condition SubSet
34051000000138	NHS Laboratory Specimen SubSet
34061000000135	NHS Laboratory Topography SubSet
34181000000134	NHS Pathology Bounded Code List Concepts
33471000000135	Non-food substance allergens
34191000000131	Non-Human
33281000000139	Occupational Therapy Assessment Scales
33661000000131	Paeds ICU Neurological Jan 2008
33671000000138	Paeds ICU Respiratory Jan 2008
33701000000139	Personal Preferences
33451000000130	Provision of Advice
34041000000136	Public Health Language
33571000000139	Risk to Person
33341000000135	Services, care professionals and carers
33621000000137	Smoking
33561000000132	Social and Personal Circumstances
33481000000138	Specimen Material Type
33381000000134	Supply
33391000000131	Supply Request
33611000000134	Temperature
33491000000136	Treatments
34201000000133	UK Admin SubSet
33581000000137	Weight

Users will not be limited to a single *SubSet*. For example, if a user wants only those Concepts in the “Diagnosis” *SubSet* then pass in 33531000000136.

If you want those Concepts that appear in more than one *SubSet*, say “Diagnosis” combined with “Discharge” then pass in “33531000000136,33741000000136” and those concepts occurring in BOTH *SubSets* will be returned. Any number of *SubSets* can be combined in this manner.

If no *SubSet Id*'s are passed in then the entire SNOMED CT dataset is used by default.

### Favourites

User Favourites (*FavouriteSets*) can be thought of as private SubSets. They are created and maintained in the SnoFlake web portal. You can create as many *FavouriteSet* as you like. Once created *FavouriteSet* can be used within SnAPI in exactly the same way as SubSets are used. Consider an application that needs to provide SNOMED codes at different points along a patient pathway. (e.g. referral, investigation, diagnosis, outcome etc.). Clearly a different set of SNOMED codes will be applicable at each point in the pathway. The idea of *FavouriteSets* is to allow User defined SubSets to be created, each containing SNOMED codes applicable to the specific point along the patient pathway. The application can call the SnAPI web service specifying a *FavouriteSet* (*or FavouriteSets*) which is used to filter the SNOMED dataset into a manageable and User defined set of appropriate codes. All other matching criteria can be applied in conjunction with the specified *FavouriteSet*, so the User can still type in appropriate key words (for example) BUT the *FavouriteSet* will establish the first level filter and will reduce the SNOMED data set in exactly the same way as a standard SubSet.

## Accessing the SnAPI web service

### Authentication (GUID)

To access the SnAPI web service you will require a GUID, you can obtain a GUID by registering for a Dataline Snoflake Browser account at the following URL: <http://snomed.dataline.co.uk/>

*Note: In this release of the SnAPI a GUID is the only form of authentication required*

### The SnAPI web service URL

The web service URL can be accessed programmatically (via a SOAP request) by any programming language that has a SOAP library.

The WSDL URL for SnAPI is: <http://snapi.dataline.co.uk/SnAPIService.svc?wsdl>

### Acronyms

The table below lists various acronyms that are used throughout this document.

Abbreviation	Meaning
API	Application Program Interface
GUID	Globally Unique Identifier
IDE	Integrated Development Environment
SnAPI	Snomed API
SOAP	Simple Object Access Protocol
URL	Universal Resource Locator
WSDL	Web Services Description Language

## Example SnAPI application using C#

In this example we will create a Microsoft Visual Studio C# console application; you can use any version of Visual Studio from 2005 onwards. Our example console application will connect to the SnAPI web service and invoke the GetSuffix web method.

### Required Software

To follow the example SnAPI application in this document you will require Microsoft's Visual C# software. You can download a free copy of Visual c# 2010 Express (registration required) from the following URL - <http://www.microsoft.com/express/Downloads/#2010-Visual-CS>.

### Source Code Example Syntax

Within this document the following formatting conventions are used:

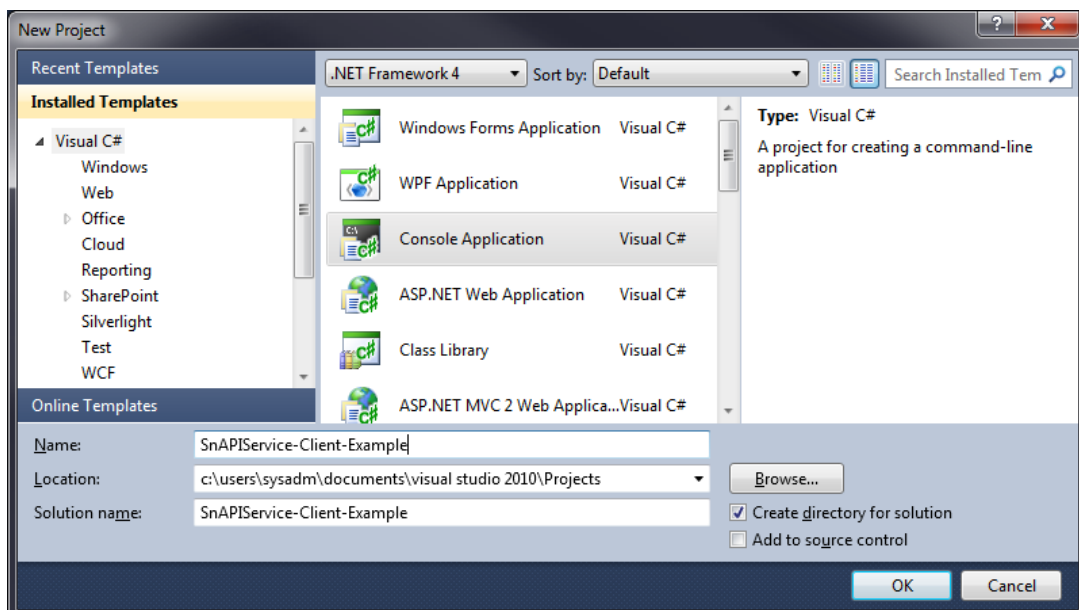
- Source code is displayed within grey shaded boxes
- New lines of code to be added to a program are highlighted in bold
- Menu and dialogue buttons are highlighted bold

### Create a Visual Studio project

The first step is to create a new Visual Studio project:

1. Within Visual Studio click **File > New > Project**
2. Select **Visual C# > Console Application**, enter a name of **SnAPIService-Client-Example** and click **OK**

*NOTE: if required, change the Project's Target Framework to the appropriate version of .NET.*

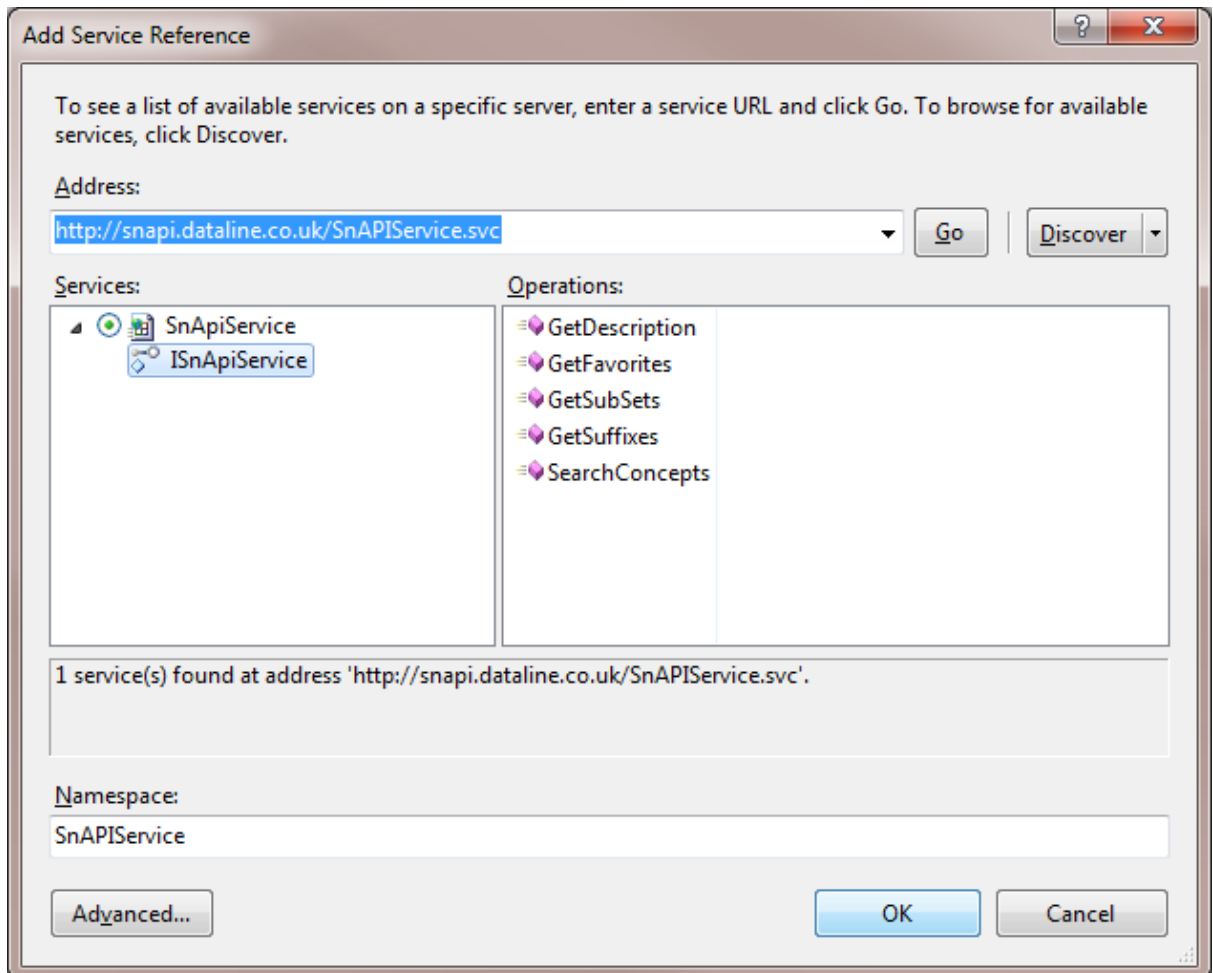


*Visual Studio 2010 Express – Creating a new Project dialogue*

## Add SnAPI™ as a service reference

The next step is to add a project reference to the SnAPI™ web service:

1. To add the SnAPI™ web service reference click **Project > Add Service Reference**
2. In the URL field enter <http://snapi.dataline.co.uk/SnAPIService.svc> and click the **Go** button
3. Under **Namespace:** enter **SnAPIService**, and click **OK**



*Visual Studio 2010 Express – Adding a Web or Service Reference dialogue*

## Add the SnAPI using directive

1. Add the below namespace directive to **Program.cs** after the default namespaces:

```
using SnAPIService_Client_Example.SnAPIService; //Used for SnAPI Methods
```

2. The list of directives should then read as follows:

```
using System; //default directives
using System.Collections.Generic; //default directives
using System.Linq; //default directives
using System.Text; //default directives
using SnAPIService_ClientExample.SnAPIService; //used for SnAPI Methods
```

## Error handling (try-catch)

Now that we've added a reference to the SnAPI™ web service, we can write some code to run the Get Suffix web method. After the opening bracket of the `static void Main(string[] args)` section add a [try-catch](#) block, this will assist in troubleshooting and handling any Exceptions:

```
static void Main(string[] args)
{
    try
    {
    }
    catch(Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

## Creating an instance of the SnAPI web service

After the opening bracket of the try-catch statement add the following code to create an instance of the SnAPI web service called 'client', we use the a [using](#) statement as this automatically closes our instance of the SnAPI web service when we are finished with it:

```
static void Main(string[] args)
{
    try
    {
        using (SnApiServiceClient client = new SnApiServiceClient())
        {
        }
    }
    catch(Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

## Calling the GetSuffix SnAPI web method

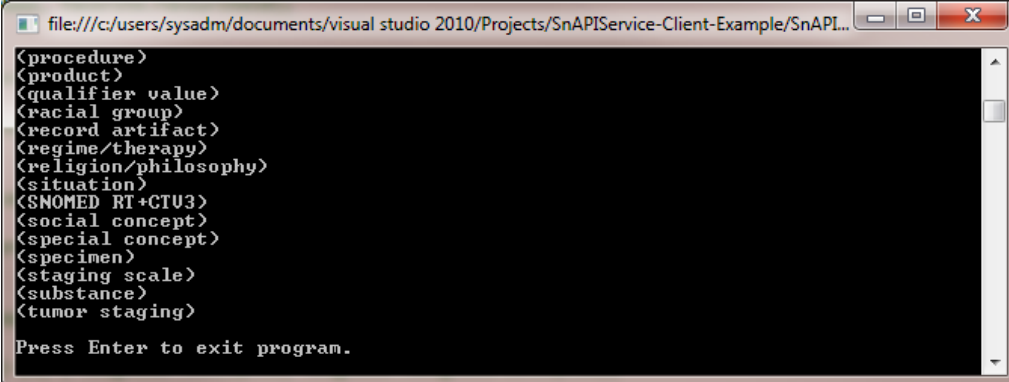
- 1) We can now add code to do the following:
  - a. run the GetSuffix web method
  - b. loop through the results
  - c. print the results to the screen
- 2) We'll also add a confirmation message when the program has finished to wait for user input before exiting, otherwise the program will exit before we've had a chance to read the output:

```
static void Main(string[] args)
{
    try
    {
        using (SnApiServiceClient client = new SnApiServiceClient())
        {
            foreach (var suffix in client.GetSuffixes("GUID"))
            {
                Console.WriteLine(suffix.suffix);
            }
        }
    }
    catch(Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    Console.WriteLine("\nPress Enter to exit program.");
    Console.ReadLine();
}
```

*Note: Remember to replace the text "GUID" with a valid GUID.*

## Testing the program

- 1) We can now build and test our application by clicking the menu option, **Debug > Start Debugging**, or by pressing the **F5** key
- 2) If the program executed successfully you should see a list of suffixes as in the below screenshot. If you don't see the output below see the section below (2.9) on troubleshooting.



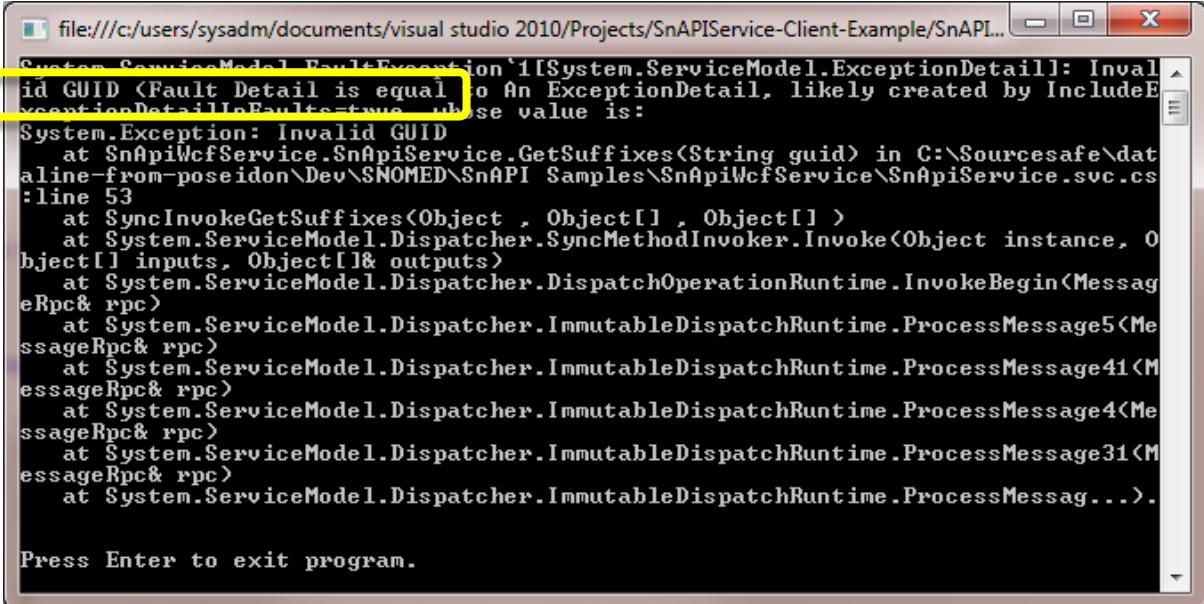
```
file:///c:/users/sysadm/documents/visual studio 2010/Projects/SnAPIService-Client-Example/SnAPI...
<procedure>
<product>
<qualifier value>
<racial group>
<record artifact>
<regime/therapy>
<religion/philosophy>
<situation>
<SNOMED RT+CTU3>
<social concept>
<special concept>
<specimen>
<staging scale>
<substance>
<tumor staging>
Press Enter to exit program.
```

*SnAPIService Client Example – Expected Output*

## Troubleshooting

### Invalid GUID

If a SnAPI web method is called with an invalid GUID the Exception, “Invalid GUID” will be returned, as per the screenshot below. To resolve this error confirm your GUID by logging on to the Dataline Snoflake Browser and make sure it’s entered correctly in your program.



```
file:///c:/users/sysadm/documents/visual studio 2010/Projects/SnAPIService-Client-Example/SnAPI...
System.ServiceModel.FaultException`1[System.ServiceModel.ExceptionDetail]: Invalid GUID (Fault Detail is equal to An ExceptionDetail, likely created by IncludeExceptionDetailInFaults=true) whose value is:
System.Exception: Invalid GUID
   at SnApiWcfService.SnApiService.GetSuffixes(String guid) in C:\Sourcesafe\dataline-from-poseidon\Dev\SNOMED\SnAPI Samples\SnApiWcfService\SnApiService.svc.cs:line 53
   at SyncInvokeGetSuffixes(Object , Object[] , Object[] )
   at System.ServiceModel.Dispatcher.SyncMethodInvoker.Invoke(Object instance, Object[] inputs, Object[]& outputs)
   at System.ServiceModel.Dispatcher.DispatchOperationRuntime.InvokeBegin(MessageRpc& rpc)
   at System.ServiceModel.Dispatcher.ImmutableDispatchRuntime.ProcessMessage5(MessageRpc& rpc)
   at System.ServiceModel.Dispatcher.ImmutableDispatchRuntime.ProcessMessage41(MessageRpc& rpc)
   at System.ServiceModel.Dispatcher.ImmutableDispatchRuntime.ProcessMessage4(MessageRpc& rpc)
   at System.ServiceModel.Dispatcher.ImmutableDispatchRuntime.ProcessMessage31(MessageRpc& rpc)
   at System.ServiceModel.Dispatcher.ImmutableDispatchRuntime.ProcessMessage...
Press Enter to exit program.
```

*SnAPI example console application – Invalid GUID error message*

### Testing connectivity

To test for any connectivity problems enter the following URLs into a web browser, if you can see pages below then you have successfully connected to the SnAPI web service:

- SnAPIService - <http://snapi.dataline.co.uk/SnApiService.svc>
  - This page should return an example how to create the .NET service client class
- SnAPIService WSDL - <http://snapi.dataline.co.uk/SnApiService.svc?wsdl>
  - This page should return a WSDL SOAP request which is an XML page

### Compilation Errors

If you are receiving errors when compiling the example program please refer to the full source code sample and compare your program for any possible errors.

### Further support

If you are having trouble connecting to the SnAPI web service, are unable to compile the example console application, or any other SnAPI problems please contact Dataline Software Ltd for assistance. Our contact details can be found in the support section of this document.

## Using other SnAPI Web Methods

You can interchange the Get Suffix web method used in this example with any of the available SnAPI web methods. The only code change required is within the [foreach](#) statement by specifying the appropriate web method, appropriate arguments and then printing out the desired results. Below are example foreach statements for each of the SnAPI web methods.

### Get SubSets

```
foreach (var subset in client.GetSubSets("GUID"))
{
    Console.WriteLine(subset.SubSetName);
}
```

### Get Favourites

```
foreach (var favorite in client.GetFavorites("GUID"))
{
    Console.WriteLine(favorite.FavouriteName,
favorite.FavouriteId);
}
```

### Get Description

```
foreach (var description in client.GetDescription("ConceptID #",
"GUID"))
{
    Console.WriteLine("{0}:{1}, ",
description.ConceptId,
description.DescriptionId );
}
```

### GetConcept

```
foreach (var concept in client.SearchConcepts("test",
"",
"",
"",
"0,1000000",
50,
1,
"GUID"))
{
    Console.WriteLine("{0}:{1}-{2}, ", concept.ConceptId,
concept.CTV3ID,
concept.FullySpecifiedName);
}
```

### GetNeighbours

```
foreach (var neighbour in client.GetNeighbour(181000000101,
"0,1000000",
"GUID"))
{
```

```
        Console.WriteLine("GetNeighbours-{0}:{1}",
            neighbour.ConceptId,
            neighbour.NeighbourConceptId);
    }
}
```

## SnAPIService\_ClientExample - Program.cs source code

Below is the full source code for the example console application:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SnAPIService_Client_Example.SnAPIService;

namespace SnAPIService_Client_Example
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (SnApiServiceClient client = new SnApiServiceClient())
                {
                    foreach (var suffix in client.GetSuffixes("GUID"))
                    {
                        Console.WriteLine(suffix.suffix);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }
            Console.WriteLine("\nPress Enter to exit program.");
            Console.ReadLine();
        }
    }
}
```

## Example SnAPI application using Java

In this example we will create a Java console application using Oracle's Netbeans IDE. Our example console application will connect to the SnAPI web service and invoke the GetSuffix web method.

### Required Software

To follow the example SnAPI application in this document you will require Oracle's NetBeans (Java SE edition) software. You can download a free copy of NetBeans Java SE from the following URL - <http://netbeans.org/downloads/index.html>.

### Source Code Example Syntax

Within this document the following formatting conventions are used:

- Source code is displayed within grey shaded boxes
- New lines of code to be added to a program are highlighted in bold
- Menu and dialogue buttons are highlighted bold

### Install the SOAP Web Services plugin in NetBeans

The SOAP Web Services plugin is required to access SnAPI. You can check if the plugin is installed by going to the Installed plugins tab under Tools > Plugins. If the plugin isn't installed, it can be installed as follows:

1. Start NetBeans, click Tools > Plugins
2. Under the available Plugins tab find and select the SOAP Web Services entry, then click the Install button

### Create a NetBeans project

The next step is to create a new NetBeans project:

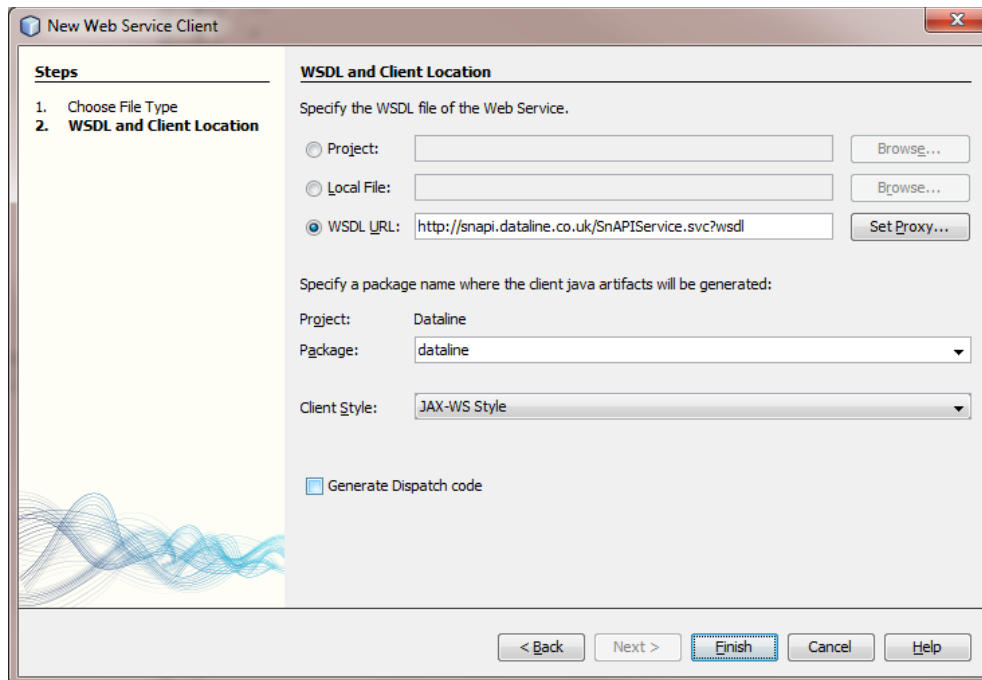
1. Click **File > New Project**
2. Select **Java** under **Categories**, and **Java Application** under **Projects**
3. Click the **Next** button
4. Under **Project Name:** enter **Dataline**
5. Tick **Create Main Class** (if not already ticked) and **Set as Main Project**
6. Click the **Finish** button

### Add a Web Service reference to the project

We can now add a reference to the SnAPI web service:

1. Click **File > New File**
2. Select **Web Services** under **Categories**, and **Web Service Client** under **File Types**
3. Click the **Next** button

4. Select the **WSDL URL:** radio box, and enter the following URL:
  - <http://snapi.dataline.co.uk/SnAPIService.svc?wsdl>
5. Under the **Package:** drop down select **Dataline**
6. Click the **Finish** button



*NetBeans IDE – Adding a Web Service reference dialogue – Step 2*

## Add the SnAPI import directive

Add the below import statements to Main.java, these should be added below immediately after the line `package dataline;`

```
import dataline.SnApiService;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
```

## Error handling (try-catch)

Now that we've added a reference to the SnAPI web service, we can write code to run the Get Suffix web method. After the opening bracket of the `public static void main(String[] args) {` section, add a [try-catch](#) block; this will assist in troubleshooting and handling any Exceptions:

```
public static void main(String[] args) {
    // TODO code application logic here
    try
    {
    }
}
```

```
catch (Exception e)
{
    System.out.println(e.getMessage());
}
}
```

## Creating an instance of the SnAPI web service

After the opening bracket of the try-catch statement, add the following code to create an instance of the SnAPI web service called 'service', and a binding to the web service called 'port'.

```
public static void main(String[] args)
{
    // TODO code application logic here
    try
    {
        dataline.SnApiService service =
            new dataline.SnApiService();
        dataline.ISnApiService port =
            service.getBasicHttpBindingISnApiService();
        ((BindingProvider) port).getRequestContext().put(
            BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
            "http://snapi.dataline.co.uk/SnAPIService.svc?wsdl");
    }
    catch (Exception e)
    {
        System.out.println(e.getMessage());
    }
}
```

## Calling the GetSuffix SnAPI web method

We can now add code to do the following:

- Add a string variable to store our GUID
- Call the SnAPI GetSuffix web method
- Loop through, and print the results to the screen

```
public static void main(String[] args)
{
    // TODO code application logic here
    try
    {
        dataline.SnApiService service =
            new dataline.SnApiService();
        dataline.ISnApiService port =
            service.getBasicHttpBindingISnApiService();
        ((BindingProvider) port).getRequestContext().put(
            BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
            "http://snapi.dataline.co.uk/SnAPIService.svc?wsdl");
    }
}
```

```
String guid = "12345678-ABCD-1234-ABCD-12345678ABCD";
ArrayOfSuffix as = port.getSuffixes(guid);

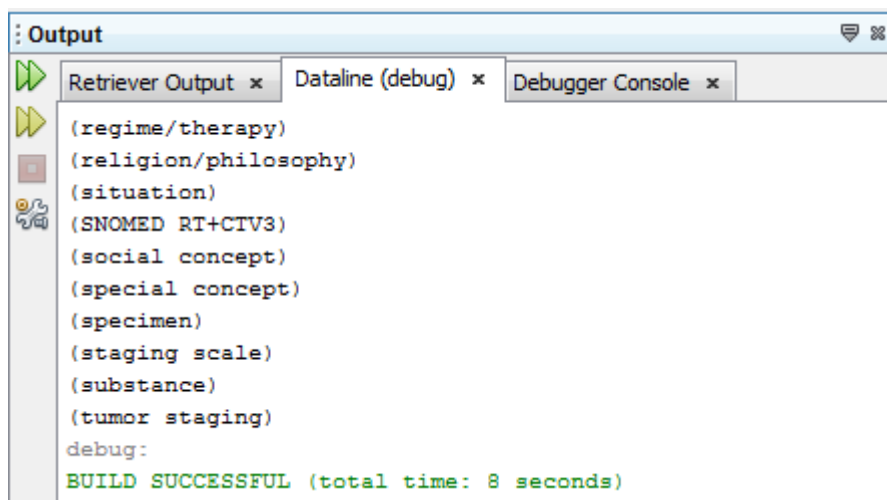
for (int i = 0; i < as.suffix.size(); i++)
{
    System.out.println(as.suffix.get(i).suffix.getValue());
}

}
catch (Exception e)
{
    System.out.println(e.getMessage());
}
}
```

**Note: Remember to replace the String guid with your guid!**

## Testing the program

- 3) We can now build and test our application by clicking the menu option, **Debug > Debug Main Project**, or by pressing the **Control + F5** key
- 4) If the program executed successfully you should see a list of suffixes in the **Output** window, as per the below screenshot. If you don't see the output below see the section below on troubleshooting.

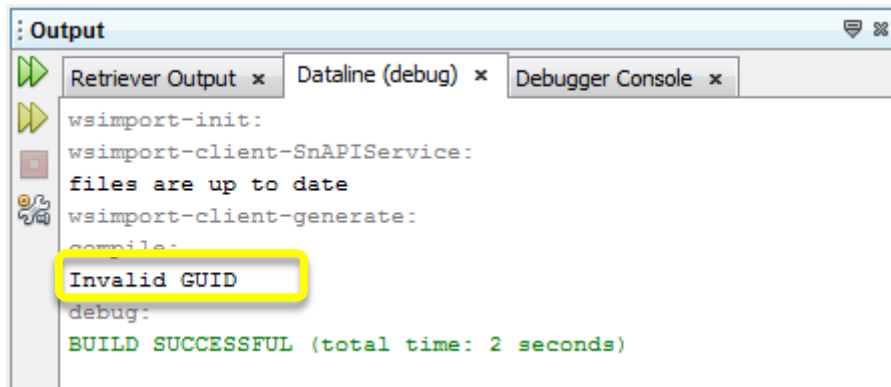


*SnAPIService example application – Expected Output*

## Troubleshooting

### Invalid GUID

If a SnAPI web method is called with an invalid GUID, the Exception “Invalid GUID” will be returned as per the screenshot below. To resolve this error confirm your GUID by logging on to the Dataline Snoflake Browser and make sure it’s entered correctly in your program.



*SnAPI example application – Invalid GUID error message*

### Testing connectivity

To test for any connectivity problems enter the following URLs into a web browser, if you can access the pages below then you have successfully connected to the SnAPI web service:

- SnAPIService - <http://snapi.dataline.co.uk/SnApiService.svc>
  - This page should return an example of how to create the web service client class
- SnAPIService WSDL - <http://snapi.dataline.co.uk/SnApiService.svc?wsdl>
  - This page should return a WSDL SOAP request which is an XML page

### Compilation Errors

If you are receiving errors when compiling the example program please refer to the full source code sample, and compare your program for any possible errors.

### Further support

If you are having trouble connecting to the SnAPI web service, are unable to compile the example console application, or any other SnAPI problems please contact Dataline Software Ltd for assistance. Our contact details can be found in the support section of this document.

## SnAPIService – main.java source code

Below is the full source code for the example console application:

```
package dataline;

import dataline.SnApiService;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;

public class Main
{

    public static void main(String[] args)
    {
        // TODO code application logic here
        try
        {
            dataline.SnApiService service =
                new dataline.SnApiService();
            dataline.ISnApiService port =
                service.getBasicHttpBindingISnApiService();
            ((BindingProvider) port).getRequestContext().put(
                BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
                "http://snapi.dataline.co.uk/SnAPIService.svc?wsdl");

            String guid = "12345678-ABCD-1234-ABCD-12345678ABCD";
            ArrayOfSuffix as = port.getSuffixes(guid);

            for (int i = 0; i < as.suffix.size(); i++)
            {
                System.out.println(as.suffix.get(i).suffix.getValue());
            }

        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
}
```

## Support

For any problems or questions you have, please either visit our SNOMED forum, send us an email or call us using the contact details below.

### SNOMED Forum

<http://www.dataline.co.uk/forum/YaBB.pl?board=Snoflake>

### Email

[snomed@dataline.co.uk](mailto:snomed@dataline.co.uk)

### Dataline Software Ltd

Suite 6, Clarence House,

30/31 North St, Brighton,

BN1 1EB, UK.

Telephone: +44 (0)1273 324939.

Fax: +44 (0)1273 205576

## Appendix 2: Features deferred until Beta 2

The following API functions will be implemented in Beta 2:

### Cross Map Support

*Cross Maps* are provided for the *MapSets* in the following table.

More can be added on request. Two arguments are required to search using a *MapSet* - the *MapSetId* which identifies the map set the user wishes to use and the *TargetCode* (actually a list of target codes) that the user wants to match.

#### MapSetId

The *MapSetId* identifies the *MapSet* from the table shown below. The user can specify one or more *MapSets* although typically one map set will be used at a time. Currently, the five *Cross Map Sets* in the table below are supported:

MapSetId	MapSetName	MapSetSchemeName
100046	ICD-9-CM	International Classification of Diseases and Related Health Problems, 9th Revision, Clinical Modifications.
101045	ICD-10	International Statistical Classification of Diseases and Related Health Problems, Tenth Revision
102041	ICD-O-3	International Classification of Diseases for Oncology, 3rd Edition
11000000140	OPCS4-4	Office of Population Censuses and Surveys, Tabular List of the Classification of Surgical Operations and Procedures, Fourth Revision, Release 4
21000000148	OPCS4-5	Office of Population Censuses and Surveys, Tabular List of the Classification of Surgical Operations and Procedures, Fourth Revision, Release 5

#### TargetCodes

A comma delimited list of *TargetCode* keys that have meaning within the chosen *MapSet*.

These codes are specific to the selected *MapSetId*. For example, to find all the SNOMED codes that match the ICD-10 code Z834: pass 101045 as the *MapSetId* and Z834 as the *TargetCode*. If the user wants all the codes that map to both Z834 and Z835 combined, pass *MapSetId* = "101045" and *TargetCode* = "Z834, Z835".

## Ancestor Matching

One of the richest features of SNOMED is the relationships that exist between the various Concepts. SNOMED is organised into twenty three *Sub-Groups* (or trees) as listed in the table below. We refer to these as *Top Level Concepts*.

This is not a definitive list as groups can be added (and possibly removed) with each release. Unlike a family tree, SNOMED Concepts can have many *Parents* as well as many *Children*, however, there is a rule applied to SNOMED that insists that any Concept will exist in one, and only one, of the *Top Level Concept* trees.

Top Level Concepts in SNOMED CT October 2009 release	
ConceptId	Fully Specified Name
420881009	Allergic disorder by allergen type (disorder)
123037004	Body structure (body structure)
404684003	Clinical finding (finding)
240166008	CTV3 ROOT (special Concept)
308916002	Environment or geographical location (environment / location)
272379006	Event (event)
106237007	Linkage Concept (linkage Concept)
363787002	Observable entity (observable entity)
410607006	Organism (organism)
373873005	Pharmaceutical / biologic product (product)
78621006	Physical force (physical force)
260787004	Physical object (physical object)
71388002	Procedure (procedure)
362981000	Qualifier value (qualifier value)
419891008	Record artefact (record artefact)
243796009	Situation with explicit context (situation)
24221000000103	SNOMED CT UK administrative Concepts (administrative Concept)
100005	SNOMED RT Concept (special Concept)
48176007	Social context (social Concept)
370115009	Special Concept (special Concept)
123038009	Specimen (specimen)
254291000	Staging and scales (staging scale)
105590001	Substance (substance)

NOTE: Each of the *Top Level Concepts* shown in the above table share a common parent - the so called *SNOMED CT Concept (SNOMED RT+CTV3)* (ConceptId = 138875005) which is the ultimate root of all SNOMED Relationships (although this is going to change with the 2010 SNOMED release).

## Transitive Closure

This section is a little technical so feel free to skip it.

*Transitive Closure* sounds grand but is very simple. To expand a little on the previous statement that

“...there is a rule applied to SNOMED that insists that any Concept will exist in one and only one of the *Top Level Concept* trees....”

If the user begins with any SNOMED Concept and follows the “is a” relationship up the tree, they are bound to end up at one (and only one) of the penultimate root Concepts which are referred to in the table above as *Top Level Concepts*. Thus, even though SNOMED permits multiple *Parents* (and of course multiple *Children*) and the path up the tree might have multiple routes, the user tracing the pathway will eventually arrive at a single *Parent* node. (In truth there are a few exceptions to this rule and a handful of Concepts in SNOMED CT October 2010 edition actually have two *Top Level Concepts* as their ultimate ancestors, however, for the sake of simplicity I am ignoring these exceptions in this discussion.)

With this in mind, it is possible to write a recursive routine that does indeed transit the tree structure for each and every SNOMED Concept and travel along every possible pathway to the root and in so doing, create a table (the *Transitive Closure Table*) that has just two columns, namely the *ConceptId* and its *Ancestor(Id)*.

Suppose we have a *ConceptId* “xxx” and we know it has twelve *Ancestors* – then the *Transitive Closure table* will hold twelve records to describe this situation - the *ConceptId* will have the same value “xxx” and the *Ancestor* column will have the key to the *Ancestor* record. Once we have created the *Transitive Closure Table* it can be used to filter SNOMED Concepts in a variety of ways.

FYI the creation of the *Transitive Closure* view is described in the “SNOMED Clinical Terms Abstract Logical Models and Representational Forms” as follows:

### 2.2.4.2 Supertype aspects of concept definition views

#### 2.2.4.2.1 Comprehensive view of supertype ancestors (“transitive closure”)

*An inferred concept definition view may explicitly contain relationships to all supertypes ancestors of the defined concept.*

*This comprehensive view of supertypes is known in description logic as a "transitive closure". It involves traversing (transiting) the target of each "is a" relationship to look for and follow further "is a" relationships until all paths through the hierarchy reach the root concept (closure).*

*This is a highly redundant expression of the logical abstract model of a concept definition. Applied to the full content of SNOMED CT it results in tens of millions of relationships.*

In SNOMED CT Oct 2010 release the *Transitive Closure Table* contains approx seven million rows.

### **AncestorId**

If we wish to constrain SNOMED within a particular sub-tree an *AncestorId* can be passed which will limit the results to descendants of that particular *Ancestor*. For example, if the user is only interested in Concepts that belong to the top level Concept *Substance* then *AncestorId* = 105590001 is passed.

But we need not limit ourselves to *Top Level Concepts*. A frequently asked question regarding SNOMED is why *Disease* is not a *Top Level Concept*. In fact *Disease* sits one level below the *Top Level Concepts* having *Clinical Finding* as its immediate *Parent*. If we want to constrain SNOMED within the sub-tree *Disease* we simply look up the *ConceptId* for *Disease* (it is 64572001) and pass this in as the *AncestorId*.

Neither is the user constrained to a single *AncestorId*. If the user wants all the Concepts in two trees - *Disease* and *Substance* then *AncestorIds* = "105590001,64572001" are passed in.

### **DecendentId**

*DecendentId* is used in subsumption testing. Suppose we have a list of Concepts and we want to know if they share a set of common *Ancestors*. (We may in this case have arrived at this list of Concepts via a list of READ codes or ICD-10 codes and mapped them to a list of SNOMED Concepts.)

We pass these Concepts (as a comma delimited list) into the *DecendentId* and the result is all the common *Ancestors* shared by the Concept list. This is useful for determining whether a Concept (or list of Concepts) is "subsumed" by a *Parent* Concept higher in the tree, which is to say that the subsuming parent is an *Ancestor* of the subsumed Concept.

### The difference between *DescendentId* and *AncestorId*

It is worth just pointing out the difference between these two seemingly similar arguments. With *DescendentId* we constrain the search by insisting that all members of the result set be *Ancestors* of ALL of the Concepts passed in *DescendentId*. So, we are finding common *Ancestors*.

With *AncestorIds*, we constrain the result set to include only Concepts which have *Ancestors* matching ANY of the *ConceptIds* passed in *AncestorIds*. So, we are finding common *Descendents*.

These two arguments are NOT mutually exclusive. It is legitimate to constrain the search to include Concepts whose *Ancestors* exist in the *AncestorId* list BUT must themselves be the *Ancestors* of ALL *Descendents* passed in the *DescendentId* list.