National Agricultural Library
Thesaurus Web Services

Programmer's Reference Manual

V2.0

March 15, 2004

Author: Tim Lynch
National Agricultural Library
10301 Baltimore Avenue
Beltsville, Maryland 20705
tlynch@nal.usda.gov

Table of Contents

Section I- Overview 3

Section II- Organization of Thesaurus Content 4

Concept – The Fundamental Unit of the Thesaurus 5

Name and Scope 6

Section III– TWS Functions 8

Some Generalizations 9

searchForConcepts 13

fetchConcepts 14

fetchHier 15

fetchCluster 16

fetchSiblings 17

Appendices

Appendix A – Concept Attributes 18

Appendix B – Failure Codes 19

Appendix C – Access to the Thesaurus Web Services 20

Appendix D – Revision History 21

Appendix E - WSDL 23

Appendix F - Examples 27

Appendix G - Pre-production web services URLs 28

Section I - Overview

This manual describes the National Agricultural Library's Thesaurus Web Service implementation. The intended audience is those who will implement the software that will consume the web service.

The manual is divided into three sections: 1) this brief overview, 2) an explanation of the organization of thesaurus content from the web service perspective, and 3) a detailed description of the SOAP-RPC functions available to the programmer.

The National Agricultural Library will continue to refine the Thesaurus Web Service. The Library's web site will clearly reflect when new editions of the service, and accompanying documentation are available. Please check the NAL web site for announcements on the availability of the latest versions of the software and this manual.

Section II - Organization of Thesaurus Content

The NAL Thesaurus Web Services facility makes the National Agricultural Library's Thesaurus available as a Web Service using two access methods: XML-RPC and SOAP-RPC[1]. Before diving into the implementation details of these two protocols, it is helpful to first explain how the thesaurus content is organized. Understanding the organization of the Thesaurus will greatly simplify understating the data returned by the Web Services.

[1] It is anticipated that a future release of the Thesaurus Web Service will provide support for REST and message-style SOAP.

Concept – The Fundamental Unit of the Thesaurus

Regardless of what web service protocol you employ (XML-RPC, SOAP), the NAL Thesaurus Web Service (hereafter abbreviated TWS) will return to you a term or list of terms matching your query[2]. If, for example, you ask the TWS to return to you information regarding the term '`livestock`', you might get back something like:

```
Name: livestock
Scope:
Use for:
    livestock and poultry *
Scope Note: Includes poultry.
Narrower:
    buffaloes
    camels
    cattle
    monogastric livestock
    range livestock
    small ruminants
    yaks
Broader:
    farmed animal species
Use for and Type:
    transport of livestock *
```

Example 1 - concept: Livestock

We've reformatted the output here to something that's nice to read, the actual TWS output is of course in XML.

What you see above in Example 1 is what throughout this manual we will refer to as a ***concept***. We will use the word ***concept*** to refer to the entire structure that you see above (everything from 'Name' down through 'Use for and Type'). We'll try to stick to using ***name*** to refer the ***attribute*** 'Name'. In other words, the ***name*** of this example ***concept*** is 'livestock.' Appendix A contains a list of all possible TWS attributes.

[2] You'll also get a statusCode indicating success or the reason why your call failed.

Name and Scope

The two attributes ***name*** and ***scope*** together uniquely identify a concept in the TWS. Either one by itself does not uniquely identify a concept. Let's look at an example.

Our previous *livestock* example specifies a ***scope*** but the value is empty (a null string). Here are two ***concepts*** that do specify the ***scope*** attribute:

```
Name: Agathis
Scope: Araucariaceae
  Narrower:
      Agathis australis
  Broader:
      Araucariaceae



Name: Agathis
Scope: Hymenoptera
  Narrower:
      Agathis pumila
  Broader:
      Braconidae
```

Example 2 - homographs

The point here is that there are two ***concepts*** in the Thesaurus with the same ***name*** attribute. Hence, the ***name*** attribute alone does not uniquely identify a ***concept***. The word *agathis* is what is more commonly known as a homonym[3]. The TWS uses the ***scope*** attribute to differentiate homonyms. As we see in Example 2, *Agathis* has two scopes: **Araucariaceae** and **Hymenoptera**. Alternatively, we could say there are two ***concepts*** in the Thesaurus: *Agathis/Araucariaceae* and *Agathis/Hymenoptera*.

Hence, the ***name*** and ***scope*** attributes together uniquely identify a ***concept***.

As you'll see in later examples, a search of the Thesaurus on the term *Agathis* will return more than one answer whereas a search on the ***name/scope*** pair {name: *Agathis*, scope:*Hymenoptera*} returns a unique result.

Name/scope pairs uniquely identify **concepts**. For the *Agathis*/*Hymenoptera* **concept** listed above, the *narrower* attribute's value is actually 'Agathis pumila'/' '. That is, the *name* is 'Agathis pumila'; the empty string indicates no *scope* is specified for this concept. The standard convention used on the NAL Thesaurus web site[4] is to include the *scope* in parentheses when specified; no term in parentheses indicates there is no *scope* attribute

[3] A word spelt like another but of different meaning.
[4] http://agclass.nal.usda.gov/

for the concept in question. Understand, though, that the concept still has a field *scope*, the value, however, in this instance is the empty (null) string.

It is recommended you familiarize yourself with the NAL Thesaurus structure by reviewing the introduction to the NAL Thesaurus at:
http://agclass.nal.usda.gov/agt/intro.htm

Section III– TWS Functions

Some Generalizations

For the function descriptions in Section II, the following definitions apply. We gather the definitions here rather than repeat them in each function description. You might want to skip over the generalizations until after you've read the function descriptions as they do not make sense without the context of the function descriptions.

`termList`

For functions that accept a `termList` variable:

`termList` is an **array** of terms that might or might not be in the thesaurus. For example, a Python call to searchForConcepts might look like:

```
termList = ['pigs', 'cattle', 'snowbunnies']
results = searchForConcepts(termList, 0)
```

An equivalent Perl invocation would be:

```
$termList = ('pigs', 'cattle', 'snowbunnies');
$results = &searchForConcepts($termList, 0);
```

(For now, ignore the '0' you see in the sample invocations; it will be explained in the searchForConcepts section.)

`nameScopeList`

For functions that accept a `nameScopeList` variable:

`nameScopeList` is an **array** of **structs** consisting of *name/scope* pairs that are expected to be in the thesaurus.

For example, a Python call to fetchConcepts might look like:

```
nameScopeList = [ {'name':'agathis','scope':'hymenoptera'},
                  {'name':'pigs', 'scope':''},
                  {'name':'cattle', 'scope':''} ]

results = fetchConcepts(nameScopeList, attributes)
```

(`attributes` is explained below)

An equivalent Perl invocation might be:

```
$nameScopeList = ( {name => 'agathis',
                    scope => 'hymenoptera'},
                   {name => 'pigs',
                    scope => ''},
                   {name => cattle',
                    scope => ''},);

$results = &fetchConcepts($nameScopeList, attributes);
```

`attributes`

For functions that accept an attributes variable:

`attributes` is an array of valid attribute abbreviations.

For example, a Python call to fetchConcepts might look like:

```
attributes = ['bt', 'nt', 'rt', 'use']
nameScopeList = [ {'name':'agathis','scope':'hymenoptera'},
             {'name':'pigs', 'scope':''} ]
results = fetchConcepts(nameScopeList, attributes)
```

An equivalent Perl invocation would be:

```
$attributes = ('bt', 'nt', 'rt', 'use');
$nameScopeList = ( {name => 'agathis',
                    scope => 'hymenoptera'},
                   {name => 'pigs',
                    scope => ''} );
$results = &fetchConcepts($nameScopeList, attributes);
```

The special keyword `'ALL'` (either upper or lowercase) can be used to indicate “return all attributes”:

```
attributes = ['ALL']
nameScopeList = [ {'name':'agathis','scope':'hymenoptera'},
                  {'name':'pigs', 'scope':''} ]
results = fetchConcepts(nameScopeList, attributes)
```

Note: the special keyword 'ALL' is still passed as a single-element array.

Function Return Values

All Thesaurus Web Services functions return a **struct** containing two elements: the actual results object and a numeric code indicating success or reason for not succeeding. Here's the struct format returned by *searchForConcepts*:

```
{ 'results':ARRAY, 'statusCode':X }
```

searchForConcepts happens to return an array of matching concepts. Other functions return either arrays of concepts or a single concept.

You can glean all the details by looking at the TWS WSDL file:

http://<TEMP-URL>/Thesaurus/soap/thesaurus.wsdl[5]

Response objects

Typically, you don't need to know about or refer to the response object; your client code library will deal with extracting the response object's content for you. Our example immediately above is represents a typical response object's content. To provide some potentially helpful debugging info, however, we detail here the full response object.

For all functions, the key used to identify the returned **struct** is of the form:

```
functionnameResponse
```

Thus, the function *fetchHier* uses **fetchHierResponse** to identify the **struct** it returns, whereas the function *fetchConcepts* uses **fetchConceptsResponse**. Working with our example above, the full object returned by *searchForConcepts* is:

```
{'searchForConceptsResponse':{ 'results':ARRAY, 'statusCode':X } }
```

statusCodes

A status code value of zero indicates successful completion of the Web Services function call. Failure is indicated by a positive integer. Appendix B lists the failure codes and their interpretation.

Note that TWS functions can return success even when no concepts are returned. For example, a call to *searchForConcepts* for the term `'foobar'` will result in no matches but *searchForConcepts* will still return a statusCode of zero.

[5] See Appendix G for pre-production URL.

searchForConcepts

SYNOPSIS

```
searchForConcepts(termList, level)
```

DESCRIPTION

For each term in the list *termList*, `searchForConcepts` searches the NAL Thesaurus looking for matches. *Level* indicates the type of term expansion to try when looking for a match.

Level is an integer in the range 0 – 5 inclusive:

0 – no wildcarding of any type
1 – add/remove trailing s and es ('pigs' will match 'pig' and, 'cloth' will match 'clothes')
2 – suffix regex ('animal' becomes `'animal.+$'`)[6]
3 – prefix regex ('animal' becomes `'^.+animal'`)
4 – prefix/suffix combined ('animal' becomes `'^.+animal.+$'`)
5 – auto search: each of the previous four expansions is tried in ascending order until a match is found.

RETURNS

searchForConcepts returns the a two element struct of `(results, statusCode)`.

Results is an array of 0 or more concepts.

PROTOCOL SPECIFICS

None

WARNINGS

None

[6] Usual regular expression matching for suffix and prefix

`fetchConcepts`

SYNOPSIS

```
fetchConcepts(nameScopeList, attributes)
```

DESCRIPTION

For each name/scope pair in *nameScopeList*, return the matching concept with attributes matching those specified in *attributes*.

By default, `'name'` and `'scope'` are always returned, in addition to whatever else *attributes* specifies.

RETURNS

fetchConcepts returns a two element struct of `(results, statusCode)`.

Results is an array of 0 or more concepts.

PROTOCOL SPECIFICS

None

WARNINGS

None

`fetchHier`

SYNOPSIS

```
fetchHier(nameScope, level, direction)
```

DESCRIPTION

For the specified *name/scope*, `fetchHier` returns array of concepts in the *direction* specified and to the *level* specified or to the end of the hierarchy. For example, given an invocation such as:

```
nameScope = {'name':'agathis','scope':'hymenoptera'}
level=2
direction='parents'

fetchHier( nameScope, level, direction)
```

`fetchHier` will return all parents of `'Agathis (Hymenoptera)'` and all parents of those parents.

If *level* is not specified, it defaults to 1.

A *level* of 0 is a request to return all concepts to the end of the hierarchy.

RETURNS

`fetchHier` returns a two element struct of `(results, statusCode)`.

Results is an array of 0 or more concepts.

PROTOCOL SPECIFICS

This function returns a one-dimensional array of concepts, which is sufficient to allow for the reconstruction of the Broader/Narrower hierarchy. Earlier versions of the TWS fetchHier function attempted to return a recursive structure that represented the two-dimensional nature of the Broader/Narrower hierarchy. Unfortunately, those early versions proved to be incompatible with some Web Services implementations. Hence, this version of the TWS returns a single-dimensional array; the code invoking the fetchHier call must reconstruct the two-dimensional structure after the fact.

None

WARNINGS

Specifying a level > 2 (or 0) can result in the return of very large arrays.

`fetchCluster`

NOT IMPLEMENTED IN VERSION 2

SYNOPSIS

```
fetchCluster(nameScopeList, attributes)
```

DESCRIPTION

`fetchCluster` returns an array of concepts.

For each concept in `nameScopeList`, `fetchCluster` will return the concept with all name/scope pairs associated with all attributes of that concept replaced by the equivalent full concept. This replacement is done not only for *bt* (Broader) and *nt* (Narrower) attributes but for all other attributes, such as *rt* (Related), as well.

It is not possible to specify a level as with `fetchHier`.

RETURNS

`fetchCluster` returns the standard two element **array** of `(results, successIndicator)` where result is the cluster **struct**.

PROTOCOL SPECIFICS

None

WARNINGS

None

`fetchSiblings`

NOT IMPLEMENTED IN VERSION 2

SYNOPSIS

```
fetchSiblings(nameScopeList, attributes)
```

DESCRIPTION

`fetchSiblings` returns all the children of all the parents of the terms specified in `nameScopeList`.

RETURNS

`fetchSiblings` returns the standard two element array of `(results, successIndicator)` where `results` is an array of concepts.

PROTOCOL SPECIFICS

None

WARNINGS

None

Appendix A

Concept Attributes

The complete list of attributes known to the Thesaurus Web Services application programming interface.

This appendix provides brief descriptions of the attributes available through the TWS API. For an up-to-date and complete description of these attributes please consult the National Agricultural Library's Thesaurus project web site: http://agclass.nal.usda.gov/agt/agt.htm

Standard Abbreviation	Full Name	Description
`name`	Name	Term typically used to identify the concept
`scope`	Scope	Provides context for homonyms. Name/Scope pairs uniquely identify concepts.
`df`	Definition	The dictionary-like description of the concept.
`sn`	Scope Note	Serves to clarify meaning of concept in relation to other concepts in the Thesaurus.
`ct`	Category	A 'top-level' concept. Category concepts are non-preferred terms. The dozen or so categories serve as a starting point for browsing the Thesaurus vocabulary.
cn	Contains	Inverse of Category. Category concepts will enumerate the preferred terms they contain. Those contained terms will enumerate the categories in which they are contained.
bt	Broader	Along with Narrower, defines the hierarchical structure of of the Thesaurus vocabulary.
nt	Narrower	Inverse of Broader.
rt	Related	Defines an associative relationship between preferred terms that is neither hierarchical or equivalent in nature.
use	Use	Identifies the preferred term to use in place of non-preferred term
uf	Use For	Inverse of Use.
usa	Use and Type	Combines notion of scope with non-preferred term
ufa	Use For and Type	Inverse of Use and Type

Appendix B

Failure Codes

Code	Interpretation
201	Invalid IP address Your IP address is not recognized as having been registered with the National Agricultural Library as a legitimate consumer for the NAL Thesaurus Web Services. See Appendix C, *Access to the Thesaurus Web Services* for information on registering your IP address.
202	Invalid search level The search *level* specified for the searchForConcepts web service function is not a legitimate value. See the searchForConcepts page for a description of legal values.
203	No terms specified The searchForConcepts web service function could not identify a term or list of terms to search for. Make sure you are passing your search terms as an array even if only one term is specified.
204	No attributes specified No list of concept attributes to be returned could be identified in your function invocation. Make sure you are passing the list of attributes to be returned as an array and that you are using the standard attribute abbreviations as listed in Appendix A.
205	Invalid name/scope An invalid format for a name/scope pair was detected. Make sure you are passing name/scope pairs as structs (i.e., associative arrays). Also see the Thesaurus Web Services WSDL file, Appendix E.
206	Invalid direction An invalid value for the *direction* parameter was passed to the fetchHier function. See the page for fetchHier for valid values for the *direction* parameter.
207	Invalid depth An invalid value for the *depth* parameter was passed to the fetchHier function. See the page for fetchHier for valid values for the *depth* parameter.

Appendix C

Access to the Thesaurus Web Services

Access to the National Agricultural Library Thesaurus Web Services interface is regulated. If you would like access, please write to or email Lori Finch, Thesaurus Specialist:

Lori Finch
National Agricultural Library
10301 Baltimore Avenue
Beltsville, Maryland 20705

lfinch@nal.usda.gov

Please include the following information:

Your name
Your email address
Your organization and your organization's address
The IP address of the computer that will consume the NAL Thesaurus Web Services
A brief description of the application in which you intend to use the NAL Thesaurus

Once this information has been received, you will be informed by email when access has been granted.

Appendix D

Thesaurus Web Services Revision History

This document reflects the revision history of the Thesaurus Web Services, not the revision history of this reference manual.

Version/revision naming conventions

Minor version numbers follow standard convention of appending to major number with dot notation, i.e., minor revision 7 of major release 3 would be indicated as 3.7.
Even minor numbers indicate a bug fix release with no new features added nor changes to existing features except to make them work as documented :)
Odd minor numbers indicate new features or changes to existing features.

This appendix documents the revision history of the Thesaurus Web Service itself and this document.

March 15, 2004

Version 2 of Thesaurus Web Services released.

This version changes what fetchHier returns from a recursive structure to a flat array. This change was made to accommodate ColdFusion and other clients which appear to have problems with recursive structures.

A number of other bugs with the WSDL file have been fixed as well.

August 18, 2003

WSDL file v1 now available.

July 24, 2003 – v1.2 – SOAP services

SOAP services now supported and documented. SOAPRPC style interface now available.

Format of the structure of the returned concepts has been simplified. Reference to concept *index* has been eliminated. All references to concepts are now based on *name/scope* rather than *index*.

July 14, 2003 – v1.1 – no public release

SOAP fully supported, but not documented

June 27, 2003 - v1.0 – initial release

XML-RPC fully supported
SOAP available in demonstration/test mode
REST not supported
WSDL not supported

Appendix E

WSDL file for NAL Thesaurus Web Service

You can view this file online at:
http://laurel.nal.usda.gov/Thesaurus/soap/thesaurus.wsdl[7]

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  name="Thesaurus"
  targetNamespace="http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
    <xsd:schema
      targetNamespace="http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:tns="http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="termsArray">
        <xsd:complexContent>
          <xsd:restriction base="SOAP-ENC:Array">
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" minOccurs="0" name="item"
type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="xsd:string[]"/>
          </xsd:restriction>
        </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="baseConcept">
        <xsd:all>
          <xsd:element maxOccurs="1" minOccurs="1" name="name" type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="1" name="scope" type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="df" type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="sn" type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="ct" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="cn" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="bt" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="nt" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="rt" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="use" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="uf" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="usa" type="tns:baseConceptArray"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="ufa" type="tns:baseConceptArray"/>
        </xsd:all>
      </xsd:complexType>
      <xsd:complexType name="baseConceptArray">
        <xsd:sequence>
          <xsd:element
            maxOccurs="unbounded"
            minOccurs="1"
```

[7] If the above address does not resolve to the WSDL file, try:
http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl

```
      name="oneConcept"
      type="tns:baseConcept"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="conceptArrayResponse">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="statusCode" type="xsd:integer"/>
    <xsd:element
      maxOccurs="1"
      minOccurs="1"
      name="concepts"
      type="tns:baseConceptArray"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="conceptSingleResponse">
  <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="statusCode" type="xsd:integer"/>
    <xsd:element maxOccurs="1" minOccurs="1" name="concept" type="tns:baseConcept"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="nameScope">
  <xsd:all>
    <xsd:element maxOccurs="1" minOccurs="1" name="name" type="xsd:string"/>
    <xsd:element maxOccurs="1" minOccurs="1" name="scope" type="xsd:string"/>
  </xsd:all>
</xsd:complexType>
<xsd:complexType name="nameScopeArray">
  <xsd:choice>
    <xsd:element
      maxOccurs="unbounded"
      minOccurs="1"
      name="aNameScope"
      type="tns:nameScope"/>
  </xsd:choice>
</xsd:complexType>
<xsd:simpleType name="conceptAttribute">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="all"/>
    <xsd:enumeration value="name"/>
    <xsd:enumeration value="scope"/>
    <xsd:enumeration value="df"/>
    <xsd:enumeration value="sn"/>
    <xsd:enumeration value="ct"/>
    <xsd:enumeration value="cn"/>
    <xsd:enumeration value="bt"/>
    <xsd:enumeration value="nt"/>
    <xsd:enumeration value="rt"/>
    <xsd:enumeration value="use"/>
    <xsd:enumeration value="uf"/>
    <xsd:enumeration value="usa"/>
    <xsd:enumeration value="ufa"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="hierDirection">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="parents"/>
    <xsd:enumeration value="children"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="attrsArray">
  <xsd:sequence>
    <xsd:element
      maxOccurs="13"
```

```
          minOccurs="1"
          name="conceptAttribute"
          type="tns:conceptAttribute"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
</types>
<message name="fetchConceptsResponse">
  <part name="results" type="tns:conceptArrayResponse"/>
</message>
<message name="searchForConceptsResponse">
  <part name="results" type="tns:conceptArrayResponse"/>
</message>
<message name="fetchHierResponse">
  <part name="results" type="tns:conceptArrayResponse"/>
</message>
<message name="searchForConceptsRequest">
  <part name="terms" type="tns:termsArray"/>
  <part name="level" type="xsd:integer"/>
</message>
<message name="echoResponse">
  <part name="answer" type="xsd:string"/>
</message>
<message name="echoRequest">
  <part name="question" type="xsd:string"/>
</message>
<message name="fetchConceptsRequest">
  <part name="nameScopeList" type="tns:nameScopeArray"/>
  <part name="attributes" type="tns:attrsArray"/>
</message>
<message name="fetchHierRequest">
  <part name="nameScope" type="tns:nameScope"/>
  <part name="level" type="xsd:integer"/>
  <part name="direction" type="tns:hierDirection"/>
</message>
<portType name="ThesaurusPortType">
  <operation name="echo" parameterOrder="question arg1">
    <input message="tns:echoRequest"/>
    <output message="tns:echoResponse"/>
  </operation>
  <operation name="searchForConcepts">
    <input message="tns:searchForConceptsRequest"/>
    <output message="tns:searchForConceptsResponse"/>
  </operation>
  <operation name="fetchConcepts">
    <input message="tns:fetchConceptsRequest"/>
    <output message="tns:fetchConceptsResponse"/>
  </operation>
  <operation name="fetchHier">
    <input message="tns:fetchHierRequest"/>
    <output message="tns:fetchHierResponse"/>
  </operation>
</portType>
<binding name="ThesaurusBinding" type="tns:ThesaurusPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="echo">
    <soap:operation soapAction="capeconnect:Thesaurus:ThesaurusPortType#echo"/>
    <input>
      <soap:body encodingStyle="" use="literal"/>
    </input>
    <output>
      <soap:body encodingStyle="" use="literal"/>
    </output>
```

```
    </operation>
    <operation name="searchForConcepts">
      <soap:operation
        soapAction="capeconnect:Thesaurus:ThesaurusPortType#searchForConcepts"/>
      <input>
        <soap:body encodingStyle="" use="literal"/>
      </input>
      <output>
        <soap:body encodingStyle="" use="literal"/>
      </output>
    </operation>
    <operation name="fetchConcepts">
      <soap:operation
        soapAction="capeconnect:Thesaurus:ThesaurusPortType#fetchConcepts"/>
      <input>
        <soap:body encodingStyle="" use="literal"/>
      </input>
      <output>
        <soap:body encodingStyle="" use="literal"/>
      </output>
    </operation>
    <operation name="fetchHier">
      <soap:operation soapAction="capeconnect:Thesaurus:ThesaurusPortType#fetchHier"/>
      <input>
        <soap:body encodingStyle="" use="literal"/>
      </input>
      <output>
        <soap:body encodingStyle="" use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="Thesaurus">
    <port binding="tns:ThesaurusBinding" name="ThesaurusPort">
      <soap:address location="http://laurel.nal.usda.gov:9090/Thesaurus/thesaurus"/>
    </port>
  </service>
</definitions>
```

Appendix F

Examples

To be completed

Appendix G

Pre-production URLs

As of March 15, 2004, the following URL is available for use by registered users of the National Agricultural Library Thesaurus Web Services:

http://laurel.nal.usda.gov:9090/Thesaurus/soap/thesaurus.wsdl

Please note the port number is 9090. An older and now outdated version of the TWS was running on port 8080 -- you should not attempt to access that version.

As the TWS moves into full production, a more logical URL will be established for the service. All registered users of the service will be notified in time to allow for updating client code.