ANGEWANDTE INFORMATIONSTECHNIK
Forschungsgesellschaft mbH

Technical decription

SOOM

AIT Vocabulary Service Description

Version 1.0

2009-05-04

SOOM

AIT Vocabulary Service Description

Technical Design

Project Number				
Project Title				
Document Reference Title	AIT Vocabulary Service Description		Date	2009-05-04
Document Name	AIT Vocabulary Service Description.doc		Version 1.0	draft ☐ final ☑
Restrictions	public ☐	internal ☐	restricted ☑ :	
Distribution				
Authors	Alexander Herzog, Odo Benda			
Abstract	Describes the AIT vocabulary service wsdl and its usage			
Keywords	WebServices, WSDL,			

Document Revisions			
Version	**Date**	**Author(s)**	**Description of Change**
1.0	Mai 4, 2009	Alexander Herzog	Initial final

Table of Contents

1 Introduction

In this paper, the AIT vocabulary service is discussed.

Notation

Some parameters are maked with *, ? or +. These markers are the short repeat information. None to infinite repeatations is equivalent to *, optional parameters are marked with an ? for zero or one repetations. If at least one repeatation is needed, then the + will be used.

2 Parameters

This chapter describes the paramers used by the Methodes described in chapter 4.

2.1 getAllThesauri

Used as request at 4.3. Contains a *session_ticket* string

2.2 getAllThesauriResponse

Used as response at 4.3. Contains a list of ThesaurusInfo (3.1) elements

2.3 getLanguages

Used as request at 4.4. Consists of a

- session_ticket
- thesaurus_name
- lang*: You will get the list of languages that includes
  - The international three digit country code (ISO 639-3)
  - the native language name
  - and the translation you specify here.

2.4 getLanguagesResponse

Used as response at 4.4. Consists of a

- message
- language* (3.3): A LanguageInfo type: A list of languages for which terms exist in the thesaurus.

2.5 query

Used as request at 4.5. Consists of a

- session_ticket
- thesaurus_name
- query_lang: The language of the query. Only terms in this language match. Default: any
- result_lang?: The languages of the result nodes. Other languages will be stripped from the result. Default: any
- Followed by one of

  - simpleQuery (3.11): Simple query expression
  - complexQuery (3.4): Complex (boolean) query expression

- origin? (3.4): Specify the ancestor nodes of result nodes. This means only nodes will be included in the result when they have an ancestor that is matches "origin". Default: The thesaurus root.
- max_depth?: For each node that maches add up to this many levels of children to the result. Default: 0
- max_nodes?: For each node that maches add up to this many levels of children to the result. Default: 10

2.6 queryResponse

Used as response at 4.5 and 4.7. Consists of

- message:
- result_id:
- node* (3.12): The resulting node list.

2.7 getNodes

Used as request at 4.7. Consists of a

- session_ticket
- thesaurus_name
- id: The list of ids of the nodes to return
- levels: Either: 0 (the specified ids only) or 1, 2, .. maximum depth for children to be added or -1, -2, .. maximum depth for parents to be added to the result.
- max_nodes?: Maximum size of the result. Default: 10

2.8 getMetaAttribute

Used as request at 4.8. Consists of a

- session_ticket
- thesaurus
- meta_att_name: Attribute name
- node_id*: A list of nodes to query. The meta attributes of these nodes will be returned

2.9 getMetaAttributeResponse

Used as reponse at 4.8. Consists of a

- message
- attributes* (3.2)

2.10 login

Used as request at 4.1. Consists of a

- username
- password
- timout: Inactivity duration in second. If no request is sent within this amount of time the session_ticket will become invalid (just like when calling logout). Default: 1800

2.11 loginResponse

Used as response at 4.1. Consists of a

- message
- session_ticket
- account_information (3.16)

2.12 logout

Used as request at 4.2. Consists of a

- session_ticket

2.13 logoutResponse

Used as response at 4.2. Consists of a

- message
- session_ticket: This is empty on a succuessful logout. Otherwise it contains the still valid session ticket.
- account_information (3.16)

2.14 createNode

Used as request at 4.9. Consists of a

- session_ticket: The authentication token from the login
- thesaurus_name All following nodes are added to this thesauru

- node* (3.12)

2.15 manipulatedNodeResponse

Used as response at 4.9 and 4.10. Consists of a

- message:
- node* (3.12) The nodes as they where really stored

2.16 updateNode

Used as request at 4.10. Consists of a

- session_ticket
- thesaurus_name
- node (3.12)

2.17 deleteNode

Used as request at 4.11. Consists of a

- session_ticket
- thesaurus_name
- node_id*: List of node ids to delete
- slice?: If true the children of the node will not be deleteted but inserted as children of the deleted node's parent.

2.18 deleteNodeResponse

Used as response at 4.11. Consists of a

- message

2.19 setMetaAttribute

Used as request at 4.12. Consists of a

- session_ticket
- node_id
- attribute_name
- attribute_value

2.20 setMetaAttributeResponse

Used as response at 4.12. Consists of a

- message

2.21 importNodes

Used as request at 4.13. Consists of a

- session_ticket
- thesaurus_name
- data_format Choose the encoding/data schema of the import data. This is one of:
  - SKOS/museumvok (SKOS schema encoded as museumvok XML)
  - CSV/ait (ait table schema, as comma separated table)
  - ZIP/xml
- data The data encoded as base64 binary blob. Any data has to be encoded that was, no matter if it's text or real binary content.

2.22 importNodesResponse

Used as response at 4.13. Consists of a

- message

2.23 exportNodes

Used as request at 4.14. Consists of a

- session_ticket
- thesaurus_name
- data_format Is one of the following:
  - SKOS/museumvok

2.24 exportNodesResponse

Used as response at 4.14. Consists of a

- message
- data The data encoded as base64 binary blob

2.25 scanIndex

Used as request at 4.6. Consists of a

- session_ticket
- thesaurus_name

- field
- term
- max_terms Number of returned terms

2.26 scanIndexResponse

Used as response at 4.6. Consists of a

- message
- result_id In the case when more hits exist than the maximum limit specified with the call this allow continuation. The result_id is either
  - 0 (Result complete)
  - -1 (continuation not possible)
  - >0 an identifier allowing the continuation.
- term* The resulting node list

2.27 getFieldsRequest

Used as request at 4.15. Consists of a

- session_ticket
- thesaurus_name

2.28 getFieldsResponse

Used as response at 4.15. Consists of a

- message
- fieldDescription (3.18)

3 Common types

Within the parameters, the types described in this chapter are used on various occations.

3.1 ThesaurusInfo

Used as type in 2.2. Consists of a

- thesaurus_name: A unique short name identifying the thesaurus.
- description: A link to a human readable description of the thesaurus' content.
- rights: Copyright information
- genericLanguage: The main language of the thesaurus. Every entry must exist in this language. This is also the default when no language is selected and the fallback when a certain entry is not available in some chosen language.
- availableLanguage*: A list of all languages used by the terms in the thesaurus. (Iso codes)

3.2 MetaAttribute

Used as type in 2.9. Consists of a

- node_id
- attribute_name
- attribute_value

3.3 LanguageInfo

Used as type at 2.4. A list of languages for which terms exist in the thesaurus. Consists of a

- iso639_3 The three letter international language name code (ISO 639-3). Do not confuse this with ISO 639-2 which is a three letter language name code based on the english names. (E.g. Germany: ISO 639-3: "deu", ISO 639-2 "ger")
- native The native name of the language.
- translated? An optional translation to another language.

3.4 QueryExpr

Used as type in 2.5, 3.6, 3.7 and 3.10. Consists of one of the following elements

- compare (3.5) RelationExpr
- and (3.6) AndExpr

- or (3.7) OrExpr
- not (3.10) NotExpr

3.5 RelationExpr

Used as type in 3.4 and 3.11. Consists of a

- field (3.8) QueryableField
- op (3.9) Operator
- value

3.6 AndExpr

Used as type in 3.4. Consists of a

- left (3.4) QueryExpr
- right (3.4) QueryExpr

3.7 OrExpr

Used as type in 3.4. Consists of a

- left (3.4) QueryExpr
- right (3.4) QueryExpr

3.8 QueryableField

Used as type in 3.5. A String with one of the following values

- TRM
- Scope
- TRM[@preferred]
- TRM[not(@preferred)]

3.9 Operator

Used as type in 3.5.

Can be one of the following strings

- matches Used to match the word index
- equals
- less_than
- less_or_equal

- greater_than
- greater_or_equal

3.10 NotExpr

Used as type in 3.4. Consists of a

- arg (3.4) QueryExpr

3.11 SimpleQuery

Used as type in 2.5. Consists of a

- fulltext
- term
- more_fields* (3.5) RelationExpr
- combine_by_or Boolean value. By default all terms must apply. With this options nodes where only some terms apply will be included in the result. Set this to true to connect the terms with an "or" statement

3.12 ThesaurusNode

Used as type in 2.6, 2.14, 2.15 and 2.16. Consists of a

- TRM+ There has to be exactly one the in the generic language of the thesaurus and optionally translations into other languages. Each TRM should have a xml:lang attribute. If it is missing the generic will be used
- term
- BT* (3.15) ThesaurusNodeLink
- NT* (3.15) ThesaurusNodeLink
- RT* (3.15) ThesaurusNodeLink
- UF* (3.13) A list of used for terms
- SCOPE* (3.14) Scope

The following attributes apply to this element

- @id
- schema?
- gs?

3.13 ThesaurusNodeNP

Used as type in 3.12. Consists of a

- TRM+ There has to be exactly one the in the generic language of the thesaurus and optionally translations into other languages. Each TRM should have a xml:lang attribute. If it is missing the generic will be used
- RT* (3.15) ThesaurusNodeLink
- USE* (3.15) ThesaurusNodeLink
- SCOPE* (3.14) Scope

The following attributes apply to this element where you can use either

- @id?
- schema?
- gs?

or

- @href?

3.14 Scope

Used as type in 3.12, 3.13 and 3.15. This element can contain any element in any namespace

3.15 ThesaurusNodeLink

Used as type in 3.12 and 3.13. This is the connection between two thesaurus nodes. Each link must provide an @href attribute to the other ThesaurusNode/@id attribute.

Related terms may have a @role attribute to classify the relation.

Each realtion may cache the TRM and SCOPE of the target node. If it is cached the node can be found by a query to this field.

Consists of a

- TRM* There has to be exactly one the in the generic language of the thesaurus and optionally translations into other languages. Each TRM should have a xml:lang attribute. If it is missing the generic will be used
- SCOPE* (3.14) Scope

The following attributes apply to this element

- @href
- @role?

3.16 AccountInformation

Used as type in 2.11 and 2.13. Consists of a

- fullname? Account owner.
- email? Email address of the account owner.
- additional? Additional information
- e_points? E-Points for this service account all together. E-Points are consumed be any operation that creates, updates, retrieves or delete nodes.
- e_points_left? Remaining E-Points. E-Points are consumed be any operation that creates, updates, retrieves or delete nodes.

3.17 Costs

Consists of a

- duration? Number
- bandwith? Number
- hits? Number
- license? Number
- e_points? Number
- e_points_left? Number

3.18 FieldDescription

Consists of a

- name The name of the field
- type is one of WORD|PHRASE|SORT|GENERIC
- operation* a list of operations that can be used in query methods

4 Methods

This chapter describes the methods of the AIT vocabulary service are reference in chapter 2.

4.1 login

Authenticate yourself to the system and get a sessionticket to execute operations on the thesaurus.

Input Message: loginRequest (2.10)

Output Message: loginResponse (2.11)

4.2 logout

Invalidate a session ticket immediately.

Input Message: logoutRequest (2.12)

Output Message: logoutResponse (2.13)

4.3 getAllThesauri

Retrieve a list of all thesauri accessable by this service.

Input Message: getAllThesauriRequest (2.1)

Output Message: getAllThesauriResponse (2.2)

4.4 getLanguages

Get a list of all languages available in the thesaurus. This will retrieve a list of languages for which entries do exist. However the thesaurus will need not be completely available in each language.

Input Message: getLanguagesRequest (2.3)

Output Message: getLanguagesResponse (2.4)

4.5 query

Query the thesaurus. This operation combines simple and complex query functionality.

Input Message: queryRequest (2.5)

Output Message: queryResponse (2.6)

4.6 scanIndex

Lookup terms from an index. This works with phrase indices only and will report all matching terms and their frequency in the thesaurus.

Input Message: scanIndexRequest (2.25)

Output Message: scanIndexResponse (2.26)

4.7 getNodes

Starting from a certain nodeid go up or down in the generic structure and retrieve the nodes.

Input Message: getNodesRequest (2.7)

Output Message: queryResponse (2.6)

4.8 getMetaAttribute

Get a certain meta data attribute of a thesaurus node.

Input Message: getMetaAttributeRequest (2.8)

Output Message: getMetaAttributeResponse (2.9)

4.9 createNode

Add one or more nodes to the thesaurus that did not exist before.

Detailed operation:

- If "@id" is given it must be a number and must not yet exist, if it is empty it will be auto assigned.
- "@gs" will be automatically generated, overriding any value given
- "BT" an existing parent node id. The created node will be added as the last child of the parent unless you specify it like {parent_id} '#' {previous_sibling_id}. An empty previous_sibling_id created the node as the first child of its parent. The first "BT2 is always the broader term in the generic structure
- "NT" if these nodes already exist in the thesaurus they will be moved to be children of the created node.

Input Message: createNodeRequest (2.14)

Output Message: createNodeResponse (2.15)

4.10 updateNode

Changes the nodes content or moves the node in the hierachy.

The generic structure (@gs) will be created automatically and override any value you set there. To move a node in the hierachy modify the broader (BT) and narrower (NT) terms fields of the node.

The operation will fail if the node id does not yet exist.

Note: It is not possible to change the nodes id. In that case you have to delete the node and create a new one.

Input Message: updateNodeRequest (2.16)

Output Message: updateNodeResponse (2.15)

4.11 deleteNode

Delete or slice one or more nodes from the theaurus.

The operation will fail if the id does not exist.

Input Message: deleteNodeRequest (2.17)

Output Message: deleteNodeResponse (2.18)

4.12 setMetaAttribute

Set a meta attribute of a thesaurus node to a specific value.

Input Message: setMetaAttributeRequest (2.19)

Output Message: setMetaAttributeResponse (2.20)

4.13 importNodes

Batch import nodes into the thesaurus.

Input Message: importNodesRequest (2.21)

Output Message: importNodesResponse (2.22)

4.14 exportNodes

Batch export all nodes from the thesaurus.

Input Message: exportNodesRequest (2.23)

Output Message: exportNodesResponse (2.24)

4.15 getFields

Batch export all nodes from the thesaurus.

Input Message: getFieldsRequest (2.27)

Output Message: getFieldsResponse (2.28)

I. Glossar

WSDL	WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information http://www.w3.org/TR/wsdl [Urldate: 03/03/2009]

II. WSDL

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:mus="urn:museumvok"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://www.aitbiz.com/ns/Thesauri"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Thesauri"
        targetNamespace="http://www.aitbiz.com/ns/Thesauri"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <wsdl:types>
   <xsd:schema targetNamespace="http://www.aitbiz.com/ns/Thesauri"
        xmlns:tns="http://www.aitbiz.com/ns/Thesauri">
    <xsd:element name="getAllThesauri">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="session_ticket" type="xsd:string"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="getAllThesauriResponse">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="thesauri" type="tns:ThesaurusInfo" minOccurs="0"
       maxOccurs="unbounded"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="ThesaurusInfo">
        <xsd:sequence>
                <xsd:element name="thesaurus_name" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>
                                        A unique short name identifying the thesaurus.
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="description" type="xsd:anyURI">
                        <xsd:annotation>
                                <xsd:documentation>
                                        A link to a human readable description of the
                                        thesaurus' content.
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="rights" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>
                                        Copyright information
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
               <xsd:element name="genericLanguage" type="xsd:string">
                 <xsd:annotation>
                        <xsd:documentation>
The main language of the thesaurus. Every entry must exist in this language.
This is also the default when no language is selected and the fallback when a
 certain entry is not available in some chosen language.
</xsd:documentation>
                 </xsd:annotation></xsd:element>
```

```
            <xsd:element name="availableLanguage" type="xsd:string" minOccurs="0"
        maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>
 A list of all languages used by the terms in the thesaurus.
</xsd:documentation>
                </xsd:annotation></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="getLanguages">
        <xsd:complexType>
                <xsd:sequence>
                 <xsd:element name="session_ticket" type="xsd:string"></xsd:element>
                 <xsd:element name="thesaurus_name"
                            type="xsd:string">
                    </xsd:element>
                    <xsd:element name="lang" type="xsd:string"
                            minOccurs="0">
                      <xsd:annotation>
                        <xsd:documentation>
You will get the list of languages that includes:
  * The international three digit country code (ISO 639-3)
  * the native language name
  * and the translation you specify here.
</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="getLanguagesResponse">
        <xsd:complexType>
                <xsd:sequence>
                 <xsd:element name="message" type="xsd:string"></xsd:element>
                 <xsd:element name="language" type="tns:LanguageInfo"
                            minOccurs="0" maxOccurs="unbounded">
                      <xsd:annotation>
                        <xsd:documentation>
A list of languages for which terms exist in the thesaurus.
</xsd:documentation>
                      </xsd:annotation>
                 </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="query">
        <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="session_ticket"
                            type="xsd:string">
                    </xsd:element>
                    <xsd:element name="thesaurus_name"
                            type="xsd:string">
                    </xsd:element>
                    <xsd:element name="query_lang" type="xsd:string"
                            maxOccurs="1" minOccurs="0">
                            <xsd:annotation>
                                <xsd:documentation>
                                    The language of the query. Only terms in
                                    this language match.

                                    Default: any
```

```
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="result_lang" type="xsd:string"
                    maxOccurs="unbounded" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>
                            The languages of the result nodes. Other
                            languages will be stripped from the
                            result.

                            Default: any
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:choice>
                    <xsd:element name="simpleQuery"
                        type="tns:SimpleQuery">
                        <xsd:annotation>
                            <xsd:documentation>
                                Simple query expression
                            </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="complexQuery"
                        type="tns:QueryExpr">
                        <xsd:annotation>
                            <xsd:documentation>
                                Complex (boolean) query expression
                            </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                </xsd:choice>

                <xsd:element name="origin" type="tns:QueryExpr"
                    maxOccurs="1" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>
                            Specify the ancestor nodes of result
                            nodes. This means only nodes will be
                            included in the result when they have an
                            ancestor that is matches "origin".

                            Default: The thesaurus root.
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            <xsd:element name="sort" type="xsd:string" maxOccurs="1"
minOccurs="0"></xsd:element>
            <xsd:element name="max_depth" type="xsd:int"
                    maxOccurs="1" minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>
                            For each node that maches add up to this
                            many levels of children to the result.

                            Default: 0
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="max_nodes" type="xsd:int"
                    maxOccurs="1" minOccurs="0">
                    <xsd:annotation>
```

```
                                        <xsd:documentation>
                                                Maximum size of the result.

                                                Default: 10
                                        </xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="queryResponse">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="message" type="xsd:string"></xsd:element>
                 <xsd:element name="result_id" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>
</xsd:documentation></xsd:annotation></xsd:element>
                 <xsd:element name="node" type="tns:ThesaurusNode"
                                minOccurs="0" maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>
The resulting node list.
</xsd:documentation>
                    </xsd:annotation>
                 </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="getNodes">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="session_ticket"
                                type="xsd:string">
                        </xsd:element>
                        <xsd:element name="thesaurus_name"
                                type="xsd:string">
                        </xsd:element>
                        <xsd:element name="id" type="xsd:string"
                                maxOccurs="unbounded" minOccurs="1">
                                <xsd:annotation>
                                        <xsd:documentation>
                                                The list of ids of the nodes to return.
                                        </xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="levels" type="xsd:int">
                                <xsd:annotation>
                                        <xsd:documentation>
                                                Either: 0 (the specified ids only) or 1,
                                                2, .. maximum depth for children to be
                                                added or -1, -2, .. maximum depth for
                                                parents to be added to the result.
                                        </xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="max_nodes" type="xsd:int">
                                <xsd:annotation>
                                        <xsd:documentation>
Maximum size of the result.

Default: 10
</xsd:documentation>
```

```
                        </xsd:annotation></xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

        <xsd:element name="getMetaAttribute">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="session_ticket"
                        type="xsd:string">
                    </xsd:element>
                    <xsd:element name="thesaurus" type="xsd:string"></xsd:element>
                    <xsd:element name="meta_att_name" type="xsd:string">
                        <xsd:annotation>
                            <xsd:documentation>
Attribute name.
</xsd:documentation>
                        </xsd:annotation></xsd:element>
                    <xsd:element name="node_id" type="xsd:string" maxOccurs="unbounded"
            minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>
A list of nodes to query. The meta attributes of these nodes will be returned.
</xsd:documentation>
                        </xsd:annotation></xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="getMetaAttributeResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="message" type="xsd:string"></xsd:element>
                    <xsd:element name="attributes"
                        type="tns:MetaAttribute" maxOccurs="unbounded"
                        minOccurs="0">
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="login">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="username" type="xsd:string"></xsd:element>
                    <xsd:element name="password" type="xsd:string"></xsd:element>
                    <xsd:element name="timeout" type="xsd:int" minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>
Inactivity duration in second. If no request is sent within this amount of
time the session_ticket will become invalid (just like when calling logout).

Default: unlimited
</xsd:documentation>
                        </xsd:annotation></xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="loginResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="message" type="xsd:string"></xsd:element>
                    <xsd:element name="session_ticket"
                        type="xsd:string">
                    </xsd:element>
```

```
                <xsd:element name="account_information"
    type="tns:AccountInformation"></xsd:element>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="logout">
    <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="session_ticket" type="xsd:string"></xsd:element>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="logoutResponse">
    <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="message" type="xsd:string"></xsd:element>
                <xsd:element name="session_ticket"
                        type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>
                                        This is empty on a succuessful logout.
                                        Otherwise it contains the still valid
                                        session ticket.
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="account_information"
                        type="tns:AccountInformation">
                </xsd:element>

            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="thesaurusFault">
    <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="fault_message" type="xsd:string"></xsd:element>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="query_fault1">
    <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="query_fault1" type="xsd:string"></xsd:element>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="getLanguages_fault">
    <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="getLanguages_fault"
                        type="xsd:string">
                </xsd:element>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="LanguageInfo">
    <xsd:sequence>
            <xsd:element name="iso639_3" type="xsd:string">
                <xsd:annotation>
                        <xsd:documentation>The three letter international language name
    code (ISO 639-3). Do not confuse this with
```

```
ISO 639-2 which is a three letter language name code based on the english names.
(E.g. Germany: ISO 639-3: "deu", ISO 639-2 "ger")</xsd:documentation>
                        </xsd:annotation></xsd:element>
                <xsd:element name="native" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>The native name of the
        language.</xsd:documentation>
                        </xsd:annotation></xsd:element>
                <xsd:element name="translated" type="xsd:string" minOccurs="0">
                        <xsd:annotation>
                                <xsd:documentation>An optional translation to another
        language.</xsd:documentation>
                        </xsd:annotation></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="QueryExpr">
        <xsd:choice>
                <xsd:element name="compare" type="tns:RelationExpr"></xsd:element>
                <xsd:element name="and" type="tns:AndExpr"></xsd:element>
                <xsd:element name="or" type="tns:OrExpr"></xsd:element>
                <xsd:element name="not" type="tns:NotExpr"></xsd:element>
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="RelationExpr">
        <xsd:sequence>
                <xsd:element name="field" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>A field from the getFields
        method</xsd:documentation>
                        </xsd:annotation></xsd:element>
                <xsd:element name="op" type="tns:Operator"></xsd:element>
                <xsd:element name="value" type="xsd:string"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="AndExpr">
        <xsd:sequence>
                <xsd:element name="left" type="tns:QueryExpr"></xsd:element>
                <xsd:element name="right" type="tns:QueryExpr"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="OrExpr">
        <xsd:sequence>
                <xsd:element name="left" type="tns:QueryExpr"></xsd:element>
                <xsd:element name="right" type="tns:QueryExpr"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="QueryableField">
      <xsd:restriction base="xsd:string">
                <xsd:enumeration value="TRM"></xsd:enumeration>
                <xsd:enumeration value="SCOPE"></xsd:enumeration>
                <xsd:enumeration value="TRM[@preferred]"></xsd:enumeration>
                <xsd:enumeration value="TRM[not(@preferred)]"></xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Operator">
          <xsd:annotation>
                <xsd:documentation>matches -> Word index
```

```
_other_ -> phrase, date or number index</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="matches"></xsd:enumeration>
                <xsd:enumeration value="equals"></xsd:enumeration>
                <xsd:enumeration value="less_than"></xsd:enumeration>
                <xsd:enumeration value="less_or_equal"></xsd:enumeration>
                <xsd:enumeration value="greater_than"></xsd:enumeration>
                <xsd:enumeration value="greater_or_equal"></xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>

        <xsd:complexType name="NotExpr">
                <xsd:sequence>
                        <xsd:element name="arg" type="tns:QueryExpr"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="SimpleQuery">
                <xsd:sequence>
                        <xsd:element name="fulltext" type="xsd:string"
        minOccurs="0"></xsd:element>
                        <xsd:element name="term" type="xsd:string"
        minOccurs="0"></xsd:element>
                        <xsd:element name="more_fields"
                                type="tns:RelationExpr" minOccurs="0"
                                maxOccurs="unbounded">
                        </xsd:element>
                        <xsd:element name="combine_by_or" type="xsd:boolean">
                                <xsd:annotation>
                                        <xsd:documentation>By default all terms must apply. With
        this options nodes where only some terms apply
will be included in the result.</xsd:documentation>
                                </xsd:annotation></xsd:element>
                </xsd:sequence>
        </xsd:complexType>


        <xsd:complexType name="ThesaurusNode">
                <xsd:sequence>
                        <xsd:element name="TRM" type="xsd:string"
                                minOccurs="1" maxOccurs="unbounded">
                                <xsd:annotation>
                                        <xsd:documentation>
                                                There has to be exactly one the in the
                                                generic language of the thesaurus and
                                                optionally translations into other
                                                languages. Each TRM should have a
                                                xml:lang attribute. If it is missing the
                                                generic
                                        </xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="BT" type="tns:ThesaurusNodeLink"
                                minOccurs="0" maxOccurs="unbounded">
                        </xsd:element>
                        <xsd:element name="NT" type="tns:ThesaurusNodeLink"
                                minOccurs="0" maxOccurs="unbounded">
                        </xsd:element>
                        <xsd:element name="RT" type="tns:ThesaurusNodeLink"
                                minOccurs="0" maxOccurs="unbounded">
                        </xsd:element>
                                <xsd:element name="UF"
```

```
                                type="tns:ThesaurusNodeNP" maxOccurs="unbounded"
        minOccurs="0">
                        </xsd:element>
            <xsd:element name="SCOPE" type="tns:Scope" minOccurs="0"
        maxOccurs="unbounded">
                    </xsd:element>
        </xsd:sequence>
            <xsd:attribute name="id" type="xsd:ID" use="required"></xsd:attribute>
            <xsd:attribute name="schema" type="xsd:string"
                    use="optional">
            </xsd:attribute>
            <xsd:attribute name="gs" type="xsd:string"></xsd:attribute>
        </xsd:complexType>


        <xsd:complexType name="ThesaurusNodeNP">
            <xsd:sequence>
                    <xsd:element name="TRM" type="xsd:string"
                            minOccurs="1" maxOccurs="unbounded">
                    </xsd:element>
                    <xsd:element name="RT" type="tns:ThesaurusNodeLink"
                            minOccurs="0" maxOccurs="unbounded">
                    </xsd:element>
                    <xsd:element name="USE" type="tns:ThesaurusNodeLink"
                            minOccurs="0">
                    </xsd:element>
                    <xsd:element name="SCOPE" type="tns:Scope"
                            minOccurs="0" maxOccurs="unbounded">
                    </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:ID"></xsd:attribute>
            <xsd:attribute name="schema" type="xsd:string"></xsd:attribute>
            <xsd:attribute name="gs" type="xsd:string"></xsd:attribute>
            <xsd:attribute name="href" type="xsd:IDREF"></xsd:attribute>
        </xsd:complexType>

        <xsd:complexType name="Scope">
            <xsd:sequence>
                    <xsd:any namespace="##other"></xsd:any>
            </xsd:sequence></xsd:complexType>

        <xsd:complexType name="ThesaurusNodeLink">
            <xsd:annotation>
                <xsd:documentation>This is the connection between two thesaurus nodes. Each
        link
must provide an @href attribute to the other ThesaurusNode/@id attribute.

Related terms may have a @role attribute to classify the relation.

Each realtion may cache the TRM and SCOPE of the target node. If it is cached
the node can be found by a query to this field.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                    <xsd:element name="TRM" type="xsd:string"
                            minOccurs="0" maxOccurs="unbounded">
                    </xsd:element>
                    <xsd:element name="SCOPE" type="tns:Scope"
                            minOccurs="0" maxOccurs="unbounded">
                    </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="href" type="xsd:IDREF" use="required"></xsd:attribute>
            <xsd:attribute name="role" type="xsd:string" use="optional"></xsd:attribute>
        </xsd:complexType>
```

```
<xsd:element name="createNode">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="session_ticket"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="thesaurus_name"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="node"
				type="tns:ThesaurusNode" maxOccurs="unbounded"
minOccurs="1">
			</xsd:element>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<xsd:element name="manipulatedNodeResponse">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="message"
type="xsd:string"></xsd:element>
			<xsd:element name="nodes"
				type="tns:ThesaurusNode" maxOccurs="unbounded"
minOccurs="0">
		<xsd:annotation>
			<xsd:documentation>
The nodes as they where really stored.
</xsd:documentation>
		</xsd:annotation>
			</xsd:element>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<xsd:element name="updateNode">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="session_ticket"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="thesaurus_name"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="node"
				type="tns:ThesaurusNode">
			</xsd:element>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<xsd:element name="updateNodeResponse">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="out" type="xsd:string"></xsd:element>
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<xsd:element name="deleteNode">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="session_ticket"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="thesaurus_name"
				type="xsd:string">
```

```
                                </xsd:element>
                                <xsd:element name="node_id" type="xsd:int"
                                        maxOccurs="unbounded" minOccurs="1">
                                        <xsd:annotation>
                                                <xsd:documentation>
                                                        List of node ids to delete.
                                                </xsd:documentation>
                                        </xsd:annotation>
                                </xsd:element>
                                <xsd:element name="slice" type="xsd:boolean" maxOccurs="1"
        minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>
If true the children of the node will not be deleteted but inserted
as children of the deleted node's parent.
</xsd:documentation>
                        </xsd:annotation>
                                </xsd:element>
                            </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="deleteNodeResponse">
                        <xsd:complexType>
                                <xsd:sequence>
                                        <xsd:element name="message"
        type="xsd:string"></xsd:element>
                                </xsd:sequence>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element name="setMetaAttribute">
                        <xsd:complexType>
                                <xsd:sequence>
                                        <xsd:element name="session_ticket"
                                                type="xsd:string">
                                        </xsd:element>
                                 <xsd:element name="thesaurus_name" type="xsd:string">
                                        </xsd:element>
                                 <xsd:element name="attribute_name"
                                                type="xsd:string">
                                        </xsd:element>
                                        <xsd:element name="attribute_value"
                                                type="xsd:string">
                                        </xsd:element>
                                 <xsd:element name="node_id" type="xsd:string">
                                        </xsd:element>
                              </xsd:sequence>
                                </xsd:complexType>
                </xsd:element>
                <xsd:element name="setMetaAttributeResponse">
                        <xsd:complexType>
                                <xsd:sequence>
                                        <xsd:element name="message"
        type="xsd:string"></xsd:element>
                                </xsd:sequence>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element name="importNodes">
                        <xsd:complexType>
                                <xsd:sequence>
                                        <xsd:element name="session_ticket"
                                                type="xsd:string">
                                        </xsd:element>
                                        <xsd:element name="thesaurus_name"
```

```
                                type="xsd:string">
                            </xsd:element>
                            <xsd:element name="data_format">
                    <xsd:annotation>
                        <xsd:documentation>
Choose the encoding/data schema of the import data.
This is one of:
  * SKOS/museumvok (SKOS schema encoded as museumvok XML)
  * CSV/ait (ait table schema, as comma separated table)
</xsd:documentation>
                    </xsd:annotation>
                    <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                            <xsd:enumeration
                                                value="SKOS/musemvok">
                                            </xsd:enumeration>
                                            <xsd:enumeration
        value="CSV/ait"></xsd:enumeration>
                                            <xsd:enumeration
        value="ZIP/xml"></xsd:enumeration>
                                        </xsd:restriction>
                                    </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="data" type="xsd:base64Binary">
                    <xsd:annotation>
                        <xsd:documentation>
The data encoded as base64 binary blob.
Any data has to be encoded that was, no matter
if it's text or real binary content.
</xsd:documentation>
                    </xsd:annotation>
                            </xsd:element>
                        </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="importNodesResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="out" type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="exportNodes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="session_ticket"
                            type="xsd:string">
                        </xsd:element>
                        <xsd:element name="thesaurus_name"
                            type="xsd:string">
                        </xsd:element>
                        <xsd:element name="data_format">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                    <xsd:enumeration
        value="SKOS/museumvok"></xsd:enumeration>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="exportNodesResponse">
```

```
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="message"
	type="xsd:string"></xsd:element>
				<xsd:element name="data"
					type="xsd:base64Binary">
				</xsd:element>
			</xsd:sequence>
		</xsd:complexType>
	</xsd:element>

	<xsd:complexType name="MetaAttribute">
		<xsd:sequence>
			<xsd:element name="node_id" type="xsd:string"></xsd:element>
			<xsd:element name="attribute_name"
				type="xsd:string">
			</xsd:element>
			<xsd:element name="attribute_value" type="xsd:string"></xsd:element>
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="AccountInformation">
				<xsd:sequence>
					<xsd:element name="fullname" type="xsd:string"
	minOccurs="0" maxOccurs="1">
						<xsd:annotation>
							<xsd:documentation>
Account owner.
</xsd:documentation>
						</xsd:annotation></xsd:element>
					<xsd:element name="email" type="xsd:string"
	minOccurs="0" maxOccurs="1">
						<xsd:annotation>
							<xsd:documentation>
Email address of the account owner.
</xsd:documentation>
						</xsd:annotation></xsd:element>
					<xsd:element name="additional" type="xsd:string"
	minOccurs="0" maxOccurs="1">
						<xsd:annotation>
							<xsd:documentation>
Additional information.
</xsd:documentation>
						</xsd:annotation></xsd:element>
					<xsd:element name="e_points" type="xsd:string"
	minOccurs="0" maxOccurs="1">
						<xsd:annotation>
							<xsd:documentation>
E-Points for this service account all together.
E-Points are consumed be any operation that creates, updates, retrieves or delete nodes.
</xsd:documentation>
						</xsd:annotation></xsd:element>
					<xsd:element name="e_points_left" type="xsd:string"
	minOccurs="0" maxOccurs="1">
						<xsd:annotation>
							<xsd:documentation>
Remaining E-Points.
E-Points are consumed be any operation that creates, updates, retrieves or delete nodes.
</xsd:documentation>
						</xsd:annotation></xsd:element>
				</xsd:sequence>
	</xsd:complexType>
```

```
                        <xsd:complexType name="Costs">
                                <xsd:sequence>
                                        <xsd:element name="duration" type="xsd:float"
        minOccurs="0" maxOccurs="1"/>
                                        <xsd:element name="bandwidth" type="xsd:float"
        minOccurs="0" maxOccurs="1"/>
                                        <xsd:element name="hits" type="xsd:float" minOccurs="0"
        maxOccurs="1"/>
                                        <xsd:element name="license" type="xsd:float"
        minOccurs="0" maxOccurs="1"/>
                                        <xsd:element name="e_points" type="xsd:float"
        minOccurs="0" maxOccurs="1"/>
                                        <xsd:element name="e_points_left" type="xsd:float"
        minOccurs="0" maxOccurs="1"/>
                                </xsd:sequence>
                        </xsd:complexType>
                        <xsd:element name="scanIndex">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="session_ticket"
                                                        type="xsd:string">
                                                </xsd:element>
                                                <xsd:element name="thesaurus_name"
                                                        type="xsd:string">
                                                </xsd:element>
                <xsd:element name="field" type="xsd:string">
                                                </xsd:element>
                <xsd:element name="term" type="xsd:string"></xsd:element>
                                                <xsd:element name="max_terms" type="xsd:int">
                                                </xsd:element>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="scanIndexResponse">
                                <xsd:complexType>
                                        <xsd:sequence>
                                        <xsd:element name="message"
        type="xsd:string"></xsd:element>
                <xsd:element name="result_id" type="xsd:string">
                        <xsd:annotation>
                                <xsd:documentation>
In the case when more hits exist than the maximum limit specified with the call this allow
        continuation.
The result_id is either
 * 0 (Result complete)
 * -1 (continuation not possible)
 * >0 an identifier  allowing the continuation.
</xsd:documentation>
                        </xsd:annotation></xsd:element>
                <xsd:element name="term" type="tns:ScannedTerm"
                                minOccurs="0" maxOccurs="unbounded">
                   <xsd:annotation>
                        <xsd:documentation>
The resulting node list.
</xsd:documentation>
                   </xsd:annotation>
                </xsd:element>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>

                        <xsd:complexType name="ScannedTerm">
                                <xsd:sequence>
```

```
                                        <xsd:element name="text"
        type="xsd:string"></xsd:element>
                                        <xsd:element name="frequency"
        type="xsd:int"></xsd:element>
                                </xsd:sequence>
                        </xsd:complexType>
                        <xsd:element name="getFields">
                                <xsd:complexType>
                                        <xsd:sequence>
                                <xsd:element name="session_ticket"
                                        type="xsd:string">
                                </xsd:element>
                                <xsd:element name="thesaurus_name"
                                        type="xsd:string">
                                </xsd:element>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="getFieldsResponse">
                                <xsd:complexType>
                <xsd:annotation>
                  <xsd:documentation>
A list of field descriptions. This lists contains all the queryable fields of the thesaurus, their type
        and available query operations.
</xsd:documentation>
                </xsd:annotation>
                <xsd:sequence>
                                <xsd:element name="message"
        type="xsd:string"></xsd:element>
                                                <xsd:element name="field"
                                                        type="tns:FieldDescription"
        maxOccurs="unbounded" minOccurs="0">
                                                </xsd:element>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>

                        <xsd:complexType name="FieldDescription">
                                <xsd:sequence>
                                        <xsd:element name="name"
        type="xsd:string"></xsd:element>
                                        <xsd:element name="type">
                                                <xsd:simpleType>
                                                        <xsd:restriction base="xsd:string">
                                                                <xsd:enumeration
        value="WORD"></xsd:enumeration>
                                                                <xsd:enumeration
        value="PHRASE"></xsd:enumeration>
                                                                <xsd:enumeration
        value="SORT"></xsd:enumeration>
                                                                <xsd:enumeration
        value="NUMERIC"></xsd:enumeration>
                                                        </xsd:restriction>
                                                </xsd:simpleType>
                                        </xsd:element>
                                        <xsd:element name="operation" type="xsd:string"
                                                maxOccurs="unbounded" minOccurs="0">
                                        </xsd:element>
                                </xsd:sequence>
                        </xsd:complexType>
  </xsd:schema>
 </wsdl:types>
 <wsdl:message name="getAllThesauriRequest">
```

```
  <wsdl:part element="tns:getAllThesauri" name="request"/>
</wsdl:message>
<wsdl:message name="getAllThesauriResponse">
  <wsdl:part element="tns:getAllThesauriResponse" name="response"/>
</wsdl:message>
<wsdl:message name="getLanguagesRequest">
      <wsdl:part name="request" element="tns:getLanguages"></wsdl:part>
</wsdl:message>
<wsdl:message name="getLanguagesResponse">
      <wsdl:part name="response" element="tns:getLanguagesResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="queryRequest">
      <wsdl:part name="request" element="tns:query"></wsdl:part>
</wsdl:message>
<wsdl:message name="queryResponse">
      <wsdl:part name="response" element="tns:queryResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getNodesRequest">
      <wsdl:part name="request" element="tns:getNodes"></wsdl:part>
</wsdl:message>
<wsdl:message name="getNodesResponse">
      <wsdl:part name="response" element="tns:queryResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getMetaAttributeRequest">
      <wsdl:part name="request" element="tns:getMetaAttribute"></wsdl:part>
</wsdl:message>
<wsdl:message name="getMetaAttributeResponse">
      <wsdl:part name="response" element="tns:getMetaAttributeResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="loginRequest">
      <wsdl:part name="request" element="tns:login"></wsdl:part>
</wsdl:message>
<wsdl:message name="loginResponse">
      <wsdl:part name="response" element="tns:loginResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="logoutRequest">
      <wsdl:part name="request" element="tns:logout"></wsdl:part>
</wsdl:message>
<wsdl:message name="logoutResponse">
      <wsdl:part name="response" element="tns:logoutResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="createNodeRequest">
      <wsdl:part name="request" element="tns:createNode"></wsdl:part>
</wsdl:message>
<wsdl:message name="createNodeResponse">
      <wsdl:part name="response" element="tns:manipulatedNodeResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="updateNodeRequest">
      <wsdl:part name="request" element="tns:updateNode"></wsdl:part>
</wsdl:message>
<wsdl:message name="updateNodeResponse">
      <wsdl:part name="response" element="tns:manipulatedNodeResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="deleteNodeRequest">
      <wsdl:part name="request" element="tns:deleteNode"></wsdl:part>
</wsdl:message>
<wsdl:message name="deleteNodeResponse">
      <wsdl:part name="response" element="tns:deleteNodeResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="setMetaAttributeRequest">
      <wsdl:part name="request" element="tns:setMetaAttribute"></wsdl:part>
</wsdl:message>
<wsdl:message name="setMetaAttributeResponse">
```

```
        <wsdl:part name="response" element="tns:setMetaAttributeResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="importNodesRequest">
        <wsdl:part name="request" element="tns:importNodes"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="importNodesResponse">
        <wsdl:part name="response" element="tns:importNodesResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="exportNodesRequest">
        <wsdl:part name="request" element="tns:exportNodes"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="exportNodesResponse">
        <wsdl:part name="response" element="tns:exportNodesResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="scanIndexRequest">
        <wsdl:part name="request" element="tns:scanIndex"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="scanIndexResponse">
        <wsdl:part name="response" element="tns:scanIndexResponse"></wsdl:part>
  </wsdl:message>
        <wsdl:message name="getFieldsRequest">
                <wsdl:part name="request" element="tns:getFields"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="getFieldsResponse">
                <wsdl:part name="response" element="tns:getFieldsResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:portType name="com.ait.service.thesaurus.Thesaurus">
    <wsdl:operation name="login">
        <wsdl:documentation>
Authenticate yourself to the system and get a sessionticket to execute operations on the
        thesaurus.</wsdl:documentation>
        <wsdl:input message="tns:loginRequest"></wsdl:input>
        <wsdl:output message="tns:loginResponse"></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="logout">
        <wsdl:documentation>
Invalidate a session ticket immediately.
</wsdl:documentation>
        <wsdl:input message="tns:logoutRequest"></wsdl:input>
        <wsdl:output message="tns:logoutResponse"></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getAllThesauri">
      <wsdl:documentation>Retrieve a list of all thesauri accessable by this
        service.</wsdl:documentation>
      <wsdl:input message="tns:getAllThesauriRequest"/>
      <wsdl:output message="tns:getAllThesauriResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getLanguages">
        <wsdl:documentation>
Get a list of all languages available in the thesaurus. This will retrieve a list of
languages for which entries do exist. However the thesaurus will need not be
completly available in each language.

</wsdl:documentation>
        <wsdl:input message="tns:getLanguagesRequest"></wsdl:input>
        <wsdl:output message="tns:getLanguagesResponse"></wsdl:output>
      </wsdl:operation>
    <wsdl:operation name="query">
        <wsdl:documentation>
Query the thesaurus. This operation combines simple and complex query
        functionality.</wsdl:documentation>
        <wsdl:input message="tns:queryRequest"></wsdl:input>
        <wsdl:output message="tns:queryResponse"></wsdl:output>
```

```
      </wsdl:operation>
   <wsdl:operation name="scanIndex">
         <wsdl:documentation>Lookup terms from an index.
This works with phrase indices only and will report all matching terms and their frequency in the
         thesaurus.</wsdl:documentation>
         <wsdl:input message="tns:scanIndexRequest"></wsdl:input>
         <wsdl:output message="tns:scanIndexResponse"></wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="getNodes">
         <wsdl:documentation>
Starting from a certain nodeid go up or down in the generic structure and retrieve the nodes.

</wsdl:documentation>
         <wsdl:input message="tns:getNodesRequest"></wsdl:input>
         <wsdl:output message="tns:getNodesResponse"></wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="getMetaAttribute">
         <wsdl:documentation>
Get a certain meta data attribute of a thesaurus node.</wsdl:documentation>
         <wsdl:input message="tns:getMetaAttributeRequest"></wsdl:input>
         <wsdl:output message="tns:getMetaAttributeResponse"></wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="createNode">
         <wsdl:documentation>
Add one or more nodes to the thesaurus that did not exist before.

@param session_ticket The authentication token from the login
@param thesaurus_name All following nodes are added to this thesauru
@param node (list of nodes)
Detailed operation:
 * If &quot;@id&quot; is given it must be a number and must not yet exist, if it is empty it will be
         auto assigned.
 * &quot;@gs&quot; will be automatically generated, overriding any value given
 * &quot;BT&quot; an existing parent node id. The created node will be added as the last child of
         the parent
   unless you specify it like {parent_id} '#' {previous_sibling_id}. An empty previous_sibling_id
   created the node as the first child of its parent. The first &quot;BT&quot; is always the broader
         term in the
   generic structure
 * &quot;NT&quot; if these nodes already exist in the thesaurus they will be moved to be children
         of the created node.
 </wsdl:documentation>
         <wsdl:input message="tns:createNodeRequest"></wsdl:input>
         <wsdl:output message="tns:createNodeResponse"></wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="updateNode">
         <wsdl:documentation>
Changes the nodes content or moves the node in the hierachy.

The generic structure (@gs) will be created automatically and override
any value you set there. To move a node in the hierachy modify the
broader (BT) and narrower (NT) terms fields of the node

The operation will fail if the node id does not yet exist.

Note: It is not possible to change the node's id. In that case you
have to delete the node and create a new one.
</wsdl:documentation>
         <wsdl:input message="tns:updateNodeRequest"></wsdl:input>
         <wsdl:output message="tns:updateNodeResponse"></wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="deleteNode">
         <wsdl:documentation>
```

Delete or slice one or more nodes from the theaurus.

The operation will fail if the id does not exist.</wsdl:documentation>
<wsdl:input message="tns:deleteNodeRequest"></wsdl:input>
<wsdl:output message="tns:deleteNodeResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="setMetaAttribute">
<wsdl:documentation>
Set a meta attribute of a thesaurus node to a specific value.

</wsdl:documentation>
<wsdl:input message="tns:setMetaAttributeRequest"></wsdl:input>
<wsdl:output message="tns:setMetaAttributeResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="importNodes">
<wsdl:documentation>
Batch import nodes into the thesaurus.
</wsdl:documentation>
<wsdl:input message="tns:importNodesRequest"></wsdl:input>
<wsdl:output message="tns:importNodesResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="exportNodes">
<wsdl:documentation>
Batch export all nodes from the thesaurus.
</wsdl:documentation>
<wsdl:input message="tns:exportNodesRequest"></wsdl:input>
<wsdl:output message="tns:exportNodesResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getFields">
<wsdl:input message="tns:getFieldsRequest"></wsdl:input>
<wsdl:output message="tns:getFieldsResponse"></wsdl:output>
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="ThesauriBinding"
type="tns:com.ait.service.thesaurus.Thesaurus">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="getAllThesauri">
<soap:operation
soapAction="http://www.aitbiz.com/ns/Thesauri/getAllThesauri" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getLanguages">
<soap:operation
soapAction="http://www.aitbiz.com/ns/Thesauri/getLanguages" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="query">
<soap:operation
soapAction="http://www.aitbiz.com/ns/Thesauri/query" />
<wsdl:input>
<soap:body use="literal" />

```
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getNodes">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/getNodes" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getMetaAttribute">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/getMetaAttribute" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="login">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/login" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="logout">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/logout" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="createNode">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/createNode" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
<wsdl:operation name="updateNode">
        <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/updateNode" />
        <wsdl:input>
                <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
                <soap:body use="literal" />
```

```
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="deleteNode">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/deleteNode" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="setMetaAttribute">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/setMetaAttribute" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="importNodes">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/importNodes" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="exportNodes">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/exportNodes" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="scanIndex">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/scanIndex" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getFields">
            <soap:operation
                soapAction="http://www.aitbiz.com/ns/Thesauri/getFields" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
```

```
  <wsdl:service name="Thesauri">
    <wsdl:port binding="tns:ThesauriBinding" name="ThesauriSOAP">
      <soap:address location="http://demo.ait.co.at/thesaurus/thesaurus" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```