



D16.1v0.2 BNF grammar for WSMO language

WSMO Working Draft 17 June 2004

This version

<http://www.wsmo.org/2004/d16/d16.1/v0.2/20040617>

Latest version

<http://www.wsmo.org/2004/d16/d16.1/v0.2>

Previous version

<http://www.wsmo.org/2004/d16/d16.1/v0.2/20040612>

Editor:

Eyal Oren

Authors:

Eyal Oren
Michael Kifer
Holger Lausen
Dumitru Roman
Michael Felderer

Reviewer:

Jos de Bruijn

For printing and off-line reading, this document is also available in non-normative [PDF](#) version.

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Table of Contents

- 1 [Introduction](#)
 - 2 [Example](#)
 - 3 [BNF grammar for WSML](#)
 - 4 [Acknowledgment](#)
-

1 Introduction

This document presents the grammar for the Web Services Modelling Language (WSML). This language is meant for representing the concepts introduced in the Web Service Modelling Ontology (WSMO) in a human-readable way. WSMO is a meta-ontology for describing various aspects related to Semantic Web Services. The four main elements of WSMO are: ontologies, web services, mediators and goals. These (and other) concepts and their relations form the meta-ontology WSMO, which is described in ([Roman et al. 2004](#)).

Using WSMO one can describe different web services, that each have a certain capability, which are expressed in a certain ontology, that use certain mediators to overcome heterogeneity; using WSMO one can also describe goals that one may have, expressed in a certain ontology, which could be achieved using one or more capabilities which a web service offers. One of the languages to express this all is WSML. All concepts from WSMO can be expressed in a certain way using WSML: language constructs exist to express ontologies, web services, goals, etc.

This document specifies unambiguously the syntax for WSML. The semantics of WSML will be described in other documents, yet to be written. The language to write down this syntax is a variant of Extended Backus Nauer Form; a variant readable by ANTLR ([Parr and Quong 1995](#)). ANTLR is a tool that can construct recognisers, parsers and compilers from a grammar specification, its functionality is (among others) similar to the well-known lex/yacc tools. Visit www.antlr.org for more information.

As this is a draft version, certain issues have not yet been resolved. Some of these issues come from the fact that WSMO is also work in progress, so not all concepts have been defined completely yet; some of these issues are not related to WSMO as such, but are modelling decisions regarding only the language syntax. For completeness, we will name these outstanding issues explicitly:

- this version only describes WSML-Standard, as complement to WSMO-Standard. A version for WSML-Lite and WSML-Full will have to be defined also.
- this version does not describe the concepts of choreography and orchestration, as they are not defined in WSMO-Standard yet. A grounding of a Web Service to a URI is therefore not described yet, as this will be part of the choreography.
- this version does not prescribe a syntax for logical expressions; in the grammar, a logical expression is represented by an arbitrary string. This is done for two reasons: first, the grammar becomes language-independent, it is up to a reasoner to decide whether the logical expression encoded in the string is valid. This allows using different logical languages (with different reasoners) while using WSML. The second reason is pragmatic: since the language evolves continuously, and decisions on the logical syntax change rapidly, we postpone defining the syntax for the logical expressions until some stable developments are reached.

2 Example

Examples on the use of WSML can be found in ([Stollberg et al. 2004](#)), which extensively models a real-world scenario, using WSMO and WSML. It shows both how to use WSMO to achieve semantic web services, and how to practically annotate web services, goals, ontologies and mediators using WSML.

We refrain from putting a small toy example in this document, and refer the interested reader to ([Stollberg et al. 2004](#)) for a real-world example.

3 BNF grammar for WSML

```

header {
package deri.wsmo.bnf.validator;
}

class WSMLParser extends Parser;

wsmoDefinition:
    (ontologyDefinition | webServiceDefinition | goalDefinition | mediatorDefinition)*
;

namespaceDefinition:
    NAMESPACE (uri)?
    (COMMA namespaceBlock)*
    (TARGETNAMESPACE uri)?
;

namespaceBlock:
    uri EQUAL uri;

goalDefinition:
    GOAL uri
        (namespaceDefinition)?
        (nonFunctionalDefinition)?
        (USEMEDIATOR uri (COMMA uri)*)?
        ((POSTCONDITION | EFFECT) axiomDefinition)+
;

mediatorDefinition:
    ooMediatorDefinition | ggMediatorDefinition | wgMediatorDefinition | wwMediatorDefinition
;

ooMediatorDefinition:
    OOMEDIATOR uri
        (namespaceDefinition)?
        (nonFunctionalDefinition)?
        (SOURCE uri)+ // ontology or ooMediator
        (TARGET uri)+ // ontology or goal or webservice or mediator
        (USESERVICE uri)? // goal or wwMediator
;

ggMediatorDefinition:
    GGMEDIATOR uri
        (namespaceDefinition)?
        (nonFunctionalDefinition)?
        SOURCE uri // goal or ggMediator
        TARGET uri // goal or ggMediator
        (USEMEDIATOR uri (COMMA uri)*)? //ooMediator
        (REDUCTION axiomDefinition)?
;

wgMediatorDefinition:
    WGMEDIATOR uri
        (namespaceDefinition)?
        (nonFunctionalDefinition)?
        SOURCE uri // webservice or wgMediator
        TARGET uri // goal or wgMediator
        (USEMEDIATOR uri (COMMA uri)*)? // ooMediator
        (REDUCTION axiomDefinition)?
;

```

```

wwMediatorDefinition:
    WWMEDIATOR uri
        (nameSpaceDefinition)?
        (nonFunctionalDefinition)?
        SOURCE uri // webservice or wwMediator
        TARGET uri // webservice or wwMediator
        (USEMEDIATOR uri (COMMA uri)*)? // ooMediator
;

webServiceDefinition:
    WEBSERVICE uri
        (nameSpaceDefinition)?
        (nonFunctionalWSDefinition)?
        (USEMEDIATOR uri (COMMA uri)*)?
        (capabilityDefinition)?
        (interfaceDefinition)*
;

nonFunctionalWSDefinition:
    NFP
        nonFunctionalProperties
        (PERFORMANCE (String | uri))?
        (RELIABILITY (String | uri))?
        (SECURITY (String | uri))?
        (SCALABILITY (String | uri))?
        (ROBUSTNESS (String | uri))?
        (ACCURACY (String | uri))?
        (TRANSACTIONAL (String | uri))?
        (TRUST (String | uri))?
        (FINANCIAL (String | uri))?
        (NETWORKQOS (String | uri))?
;

capabilityDefinition:
    (USECAPABILITY uri) |
    (CAPABILITY uri
        (nonFunctionalDefinition)?
        (USEMEDIATOR uri (COMMA uri)*)?
        ( (PRECONDITION | POSTCONDITION | ASSUMPTION | EFFECT) axiomDefinition)+
    )
;

interfaceDefinition:
    (USEINTERFACE uri) |
    (INTERFACE uri
        (nonFunctionalDefinition)?
        (USEMEDIATOR uri (COMMA uri)*)?
        choreographyDefinition
        (orchestrationDefinition)?
    )
;

choreographyDefinition:
    CHOREOGRAPHY PLACEHOLDER
;

orchestrationDefinition:
    ORCHESTRATION PLACEHOLDER
;

ontologyDefinition:
    ONTOLOGY uri
        (nameSpaceDefinition)?
        (nonFunctionalDefinition)?
        (USEMEDIATOR uri (COMMA uri)*)?
        (conceptDefinition | axiomDefinition | instanceDefinition | relationDefinition |
        variableDefinition | functionDefinition)*
;

conceptDefinition:
    CONCEPT uri
        (nonFunctionalDefinition)?
        (SUBCONCEPT uri )?
        (attributeDefinition)*
        (methodDefinition)*
;

methodDefinition:
    METHOD
        (nonFunctionalDefinition)?
        (parameterDefinition)*
        (rangeDefinition)?
;

rangeDefinition:
    RANGE uri OFTYPE uri
;

variableDefinition:
    VARIABLE uri (COMMA uri)* MEMBEROF uri
;

```

```

functionDefinition:
    FUNCTION uri
        (nonFunctionalDefinition)?
        (parameterDefinition)*
        (rangeDefinition)?
    ;

attributeDefinition:
    uri OFTYPE (SET)? uri
    ;

instanceDefinition:
    INSTANCE uri MEMBEROF uri
        (uri valueDefinition)*
    ;

valueDefinition:
    (HASVALUE value) |
    (HASVALUES CURLYLEFT value (COMMA value)* CURLYRIGHT)
    ;

value: String | uri;

relationDefinition:
    RELATION uri
        (nonFunctionalDefinition)?
        parameterDefinition
    ;

parameterDefinition:
    PARAMETER uri OFTYPE uri
    ;

axiomDefinition:
    AXIOM uri
        (nonFunctionalDefinition)?
        logicalExpression
    ;

logicalExpression:
    BEGINLOGIC
        String
    ENDLOGIC
    ;

nonFunctionalDefinition:
    NFP nonFunctionalProperties
    ;

nonFunctionalProperties:
    (TITLE (uri | a:String))?
    (CREATOR (oneOreMoreuriOrString))?
    (SUBJECT (oneOreMoreuriOrString))?
    (DESCRIPTION (oneOreMoreuriOrString))?
    (PUBLISHER (oneOreMoreuriOrString))?
    (CONTRIBUTOR (oneOreMoreuriOrString))?
    (DATE (uri | String))?
    (TYPE (oneOreMoreuriOrString))?
    (FORMAT (oneOreMoreuriOrString))?
    (DCSOURCE (oneOreMoreuriOrString))?
    (LANGUAGE (oneOreMoreuriOrString))?
    (DCRELATION (oneOreMoreuriOrString))?
    (COVERAGE (oneOreMoreuriOrString))?
    (RIGHTS (oneOreMoreuriOrString))?
    (VERSION (uri | String))?
    ;

oneOreMoreuriOrString: (uri | String) (COMMA (uri | String))* ;
uri: (NCName ":" => NCName ":" NCName | NCName);

class WSMLLexer extends Lexer;
options {
    k=10;
    charVocabulary = '\u0003'..' \uFFFF';
    testLiterals = false;
}

protected NCBegin : (LowerCaseLetter | Underscore | UpperCaseLetter);
protected NCRest: (NCBegin | POINT | Dash | Hash | Slash | Amp | Question | Digit | Colon | Tilde);

NCName options { testLiterals = true ; } :
    NCBegin (NCRest)*;

// white and other skippers
WS: (' ' | '\n' | '\t' | '\r' {newline();})+ {$setType(Token.SKIP);} ;
SLComment : "comment:" (~('\n'|\r'))* ('\n'|\r') {newline();} {$setType(Token.SKIP);} ;

// internal lexer tokens
protected Dash: '-';

```

```
protected Hash: '#';
protected Slash: '/';
protected Amp: '&';
protected Question: '?';
protected Underscore: '_';
protected Colon: ':';
protected LowerCaseLetter: 'a'..'z';
protected UpperCaseLetter: 'A'..'Z';
protected Digit: '0'..'9';
protected Tilde: '~';
//protected URICharacter: (LowerCaseLetter | UpperCaseLetter | Digit | StrangeCharacter | Colon );

// single characters
COMMA: ',';
EQUAL: '=';
protected POINT: '.';

String: '"' (~'"')* '"';
NAMESPACE : "namespace";
TARGETNAMESPACE: "target-namespace";

USEMEDIATOR : "use-mediator";
OOMEDIATOR : "oo-mediator";
GGMEDIATOR : "gg-mediator";
WGMEDIATOR : "wg-mediator";
WWMEDIATOR : "ww-mediator";
SOURCE : "source";
TARGET : "target";
USESERVICE : "use-service";
REDUCTION : "reduction";

GOAL : "goal";
WEBSERVICE : "webservice";

USECAPABILITY: "use-capability";
CAPABILITY: "capability";
PRECONDITION: "precondition";
POSTCONDITION: "postcondition";
ASSUMPTION: "assumption";
EFFECT: "effect";

USEINTERFACE: "use-interface";
INTERFACE: "interface";

CHOREOGRAPHY: "choreography";
ORCHESTRATION: "orchestration";
PLACEHOLDER: "****";

ONTOLOGY: "ontology";
CONCEPT: "concept";
SUBCONCEPT: "subconceptOf";
OFTYPE: "oftype";
SET: "set";
INSTANCE: "instance";
MEMBEROF: "memberOf";
HASVALUE: "hasvalue";
HASVALUES: "hasvalues";

RELATION: "relation";
PARAMETER: "parameter";
FUNCTION: "function";
VARIABLE: "variable";
METHOD: "method";
RANGE: "range";

AXIOM: "axiom";
BEGINLOGIC: "logical-expression";
ENDLOGIC: "end-logical-expression";

TITLE: "dc:title";
CREATOR: "dc:creator";
SUBJECT: "dc:subject";
DESCRIPTION: "dc:description";
PUBLISHER: "dc:publisher";
CONTRIBUTOR: "dc:contributor";
DATE: "dc:date";
TYPE: "dc:type";
FORMAT: "dc:format";
DCSOURCE: "dc:source";
LANGUAGE: "dc:language";
DCRELATION: "dc:relation";
COVERAGE: "dc:coverage";
RIGHTS: "dc:rights";
VERSION: "version";

NFP : "non-functional-properties";
PERFORMANCE : "performance";
RELIABILITY : "reliability";
SECURITY : "security";
SCALABILITY : "scalability";
ROBUSTNESS : "robustness";
ACCURACY : "accuracy";
```

```
TRANSACTIONAL : "transactional";  
TRUST : "trust";  
FINANCIAL : "financial";  
NETWORKQOS : "network-related-qos";
```

References

PARR, T.J. and R.W. QUONG, Jul 1995: *ANTLR: A predicated-LL(k) parser generator*. Software, Practice and Experience, **25**(7), pp. 789-810.

ROMAN, D., H. LAUSEN, and U. KELLER, 2004: *Web Service Modeling Ontology Standard*. Tech. rep., WSMO Working Draft. Available from <http://www.wsmo.org/2004/d2/v02/20040306/>.

STOLLBERG, M., H. LAUSEN, A. POLLERES, *et al.*, 2004: *Wsmo use case modeling and testing*. Tech. rep., WSMO Working Draft.
<http://www.wsmo.org/2004/d3/d3.2/v0.1/20040607/>

4 Acknowledgment

The work is funded by the European Commission under the projects [DIP](#), [Knowledge Web](#), [Ontoweb](#), [SEKT](#), [SWWS](#), [Esperanto](#) and [h-TechSight](#); by Science Foundation Ireland under the [DERI-Lion](#) project; and by the Vienna city government under the [CoOperate](#) program.

The editors would like to thank all the [members of the WSMO working group](#) for their advice and input into this document.