LIRICS

Deliverable D.5.1.C

API for morpho-syntactic annotations

Project reference number	e-Content-22236-LIRICS
Project acronym	LIRICS
Project full title	Linguistic Infrastructure for Interoperable Resource and Systems
Project contact point	Laurent Romary, INRIA-Loria 615, rue du jardin botanique BP101. 54602 Villers lès Nancy (France) romary@loria.fr
Project web site	http://lirics.loria.fr
EC project officer	Erwin Valentini
Document title	API for morpho-syntactic annotations
Deliverable ID	D.5.1.C
Document type	Report
Dissemination level	Public
Contractual date of delivery	M12
Actual date of delivery	21/12/2005
Status & version	1.0
Work package, task & deliverable responsible	WP5 - UFSD
Author(s) & affiliation(s)	Julien Nioche, Kalina Bontcheva (UFSD)
Additional contributor(s)	Niraj Aswani, Ian Roberts (UFSD)
Keywords	API MAF

Document evolution

version	date	version	date
0.1	17-10-05		
1.0	20-12-05		

Content

1 Introduction

This document specifies the Application Programming Interface (API) for the **Morpho-Syntactic Annotation Framework** (**MAF**). It is currently based on the ISO TC 37/SC 4 N119 Rev. 2 document.

This document follows the guidelines of the LIRICS deliverable D1.1 (“Guidelines and tools for producing standards, test-suites and API’s”).

2 Overview of the API

The LIRICS deliverable D1.1 (“Guidelines and tools for producing standards, test-suites and API’s”) distinguishes between two levels of API:

- **Service API** for interactions with a processor (i.e. web services, local program)

- **Linguistic Content API** for manipulating the linguistic content returned by a processor

2.1 API use cases

In this chapter we describe how the MAF API fits in different architectures.

2.1.1 Local access to a native MAF system

In this case the MAF system is directly accessible by the user application. Both programs are running on the same machine and are using the same implementation language. The MAF system runs as a library within the user application. The user application sends a representation of the document to be processed and gets in return an instance of the MAF **Linguistic API** objects in the implementation language of the user application. This approach is shown in Figure 1.

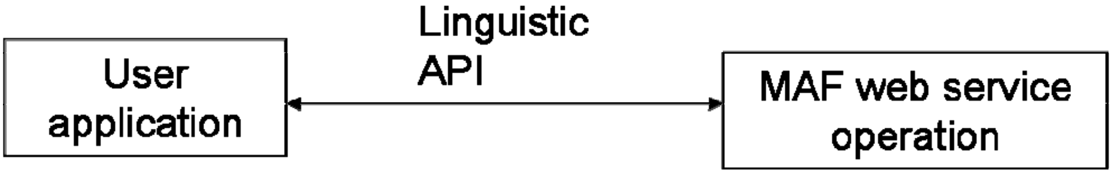

Figure 1 Direct access

2.1.2 Connection to a MAF Web Service

The MAF system runs on a remote server and is accessible through a Web Service. This approach is the one chosen in the context of the LIRICS project. The reference implementation will be remote-access-based.

A WSDL file describes the syntax and location of the annotation service. The user application sends a representation of a document to this service. The information returned can be of two types:

1. **Serialised objects:** the annotation service returns a set of serialized objects. These objects implement the **MAF Linguistic API**. The user application de-serialises the objects and uses them directly as in 3.3.1. This approach is not chosen in the LIRICS project, since it requires that the user application and service use the same implementation language, which is not a portable solution.

2. **XML serialisation:** the annotation service returns a XML representation of the MAF document. This approach is illustrated by the Figure 2 **.**

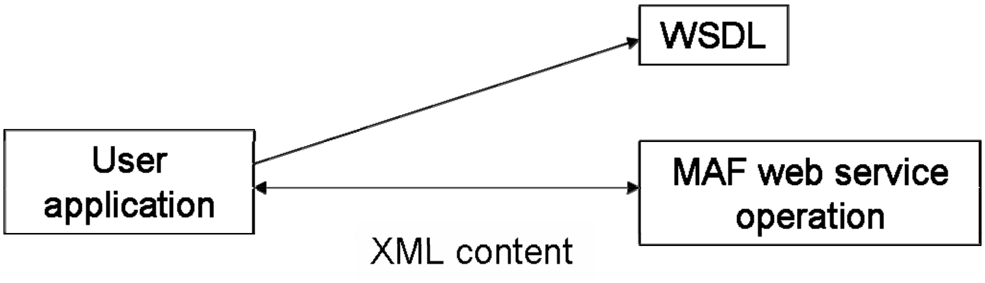

Figure 2 Remote access based on XML

The user application developer can choose whether to use this XML representation directly or to convert it into a MAF Linguistic API implementation. The latter requires a conversion library. After the conversion the MAF information can be accessed directly as in 3.3.1. This approach is illustrated by the Figure 3.

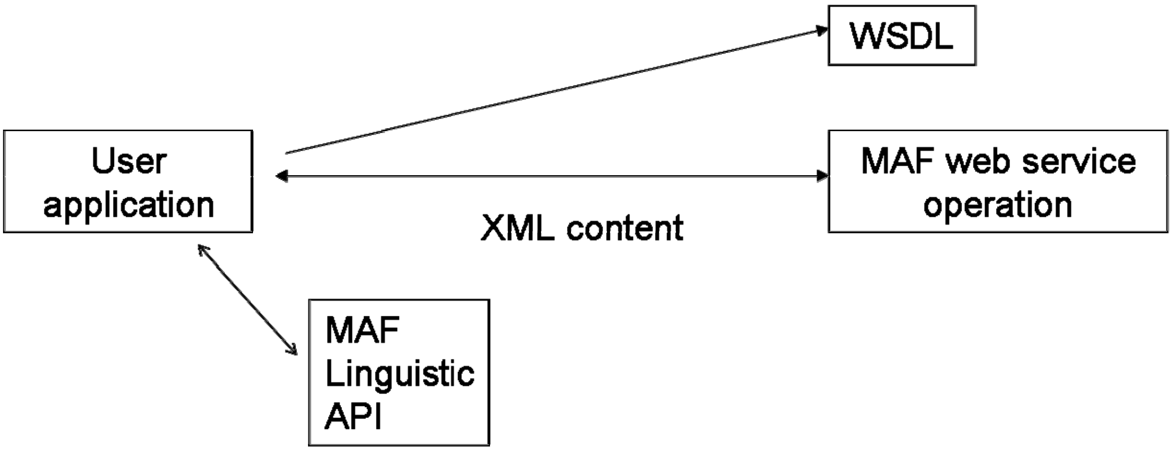

Figure 3 Remote access with conversion to a linguistic API

2.2 API strategy in LIRICS

The MAF linguistic API can be defined on the basis of the MAF model and implemented in a given language. As described in the sections above, instances of a linguistic API can be obtained:

1. directly from a MAF compliant system, provided that the system is loaded by the user code (see 3.3.1)

2. from a web-service returning serialized objects in the same programming language as the user code (see 3.4.2 - 1)

3. after converting the XML information returned by a web service (see 3.4.2 - 2)

In the context of the LIRICS project the MAF API is specified as a **service API**. Thus it will be language-independent and will be defined on the basis of web services in order to support distributed NLP resources. Dealing with XML content also leaves more choice to the client application in handling the content.

3 References:

- **Morpho-Syntactic Annotation Framework** (**MAF**) ISO TC 37/SC 4 N119 Rev. 2 document

- LIRICS deliverable D1.1 **Guidelines and tools for producing standards, test-suites and API's**

- **Web Service Definition Language** (**WSDL**) *http://www.w3.org/TR/wsdl*

- **ISO 639-2** *http://www.id3.org/iso639-2.html*

4 MAF Service API

As specified in the LIRICS deliverable D1.1, the service API is described with a WSDL file. The WSDL for MAF Services is provided in Annex A.

The Web Services Description Language (WSDL) is an XML format for describing network services as a set of operations on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete operations are combined into services.

The content of the messages is the lexical information returned by a MAF server and will be in XML format. The XML format of this information is described in the MAF norm and reproduced in Annex B. In the WSDL file, references to the MAF XML elements are done through their namespace only. This way, the independence between the API description and the XML representation of MAF is preserved.

The MAF service API is based on procedures and has 2 operations:

4.1 getSupportedLanguages

This operation returns a list of character strings indicating the languages supported by the MAF service. The codes returned are compliant with ISO 639-2 and are 3-character long and lowercase.

The code for English, for instance, is *'eng'*, French is *'fre'*, German is *'ger'*.

4.2 getAnnotation

This operation takes two arguments: a first character string representing the textual content document to be annotated, in a raw text form, without any linguistic annotations or file format data, and second, a string containing an ISO 639-2 code for the language of the document.

The character strings are represented in the SOAP XML message and thus must be in the same encoding as the SOAP message. The recommended encoding is UTF-8. However, most users are lickely to connect to a MAF Web Service using an existing toolkit (e.g. http://ws.apache.org/axis/ for Java) which will handle the encoding automatically.

The operation returns an XML representation of an annotated MAF document. This document will be compliant with the RELAX-NG schema in Annex B, which is specified in the MAF norm.

Annex A MAF Service WSDL description

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="urn:MAFService"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:MAFService">
<wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:MAFService">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
<complexType name="ArrayOf_xsd_string">
<complexContent>
<restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
  </restriction>
  </complexContent>
  </complexType>
 <!--
  A MAF Document is not defined here but referred to by its namespace
   only
  -->
<complexType name="MAFDocument">
 <sequence>
 <any namespace="http://www.iso.org/maf/1.0" processContents="lax" />
  </sequence>
  </complexType>
  </schema>
  </wsdl:types>

 <!--
 Returns a list of supported languages based on ISO 639-2 codes
  -->
<wsdl:message name="getSupportedLanguagesResponse">
  <wsdl:part name="getSupportedLanguagesReturn"
type="tns:ArrayOf_xsd_string" />
  </wsdl:message>

  <wsdl:message name="getSupportedLanguagesRequest" />
 <!--
  Message sent to the Annotation service. Contains an empty text
   document and the iso code for its language
  -->
<wsdl:message name="getAnnotationRequest">
  <wsdl:part name="document" type="xsd:string" />
  <wsdl:part name="language" type="xsd:string" />
  </wsdl:message>

 <!-- MAF Document returned by the service -->
<wsdl:message name="getAnnotationResponse">
```

```
  <wsdl:part name="getAnnotationReturn" type="tns: MAFDocument " />
  </wsdl:message>

 <!-- List of operations supported by a MAF Service -->
<wsdl:portType name="MAFService">
<wsdl:operation name="getSupportedLanguages">
  <wsdl:input name="getSupportedLanguagesRequest"
message="tns:getSupportedLanguagesRequest" />
  <wsdl:output name="getSupportedLanguagesResponse"
message="tns:getSupportedLanguagesResponse" />
  </wsdl:operation>
<wsdl:operation parameterOrder="document language"
name="getAnnotation">
  <wsdl:input name="getAnnotationRequest"
message="tns:getAnnotationRequest" />
  <wsdl:output name="getAnnotationResponse"
message="tns:getAnnotationResponse" />
  </wsdl:operation>
  </wsdl:portType>

<wsdl:binding name="MAFServiceSoapBinding" type="tns:MAFService">
  <wsdlsoap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc" />
<wsdl:operation name="getSupportedLanguages">
  <wsdlsoap:operation soapAction=" xxxxx " />
<wsdl:input name="getSupportedLanguagesRequest">
  <wsdlsoap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MAFService" />
  </wsdl:input>
<wsdl:output name="getSupportedLanguagesResponse">
  <wsdlsoap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MAFService" />
  </wsdl:output>
  </wsdl:operation>
<wsdl:operation name="getAnnotation">
  <wsdlsoap:operation soapAction="xxxxx" />
<wsdl:input name="getAnnotationRequest">
  <wsdlsoap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MAFService" />
  </wsdl:input>
<wsdl:output name="getAnnotationResponse">
  <wsdlsoap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MAFService" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
  </wsdl:definitions>
```

Annex B Relax NG compact schema for MAF

```
# $Id: maf.rnc,v 1.1 2005/09/06 08:38:02 clerger Exp $

default namespace = "http://www.iso.org/maf/1.0"
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

## Preliminary Relax NG schema for MAF -- Morpho-syntactic Annotation
Framework
## Eric de la Clergerie <Eric.De_La_Clergerie@inria.fr>


## The following  is for Feature Structures
include "iso-fs-standalone.rnc"

start =
  element maf {
    ( maf.document,
      maf.addressing )? ,
    tagset ?,
    maf.metadata ?,
    maf.flow
  }
maf.document = attribute document { xsd:anyURI }
## To be defined in LAF
maf.addressing = attribute addressing { xsd:NMTOKEN }
## Global Metadata: to be completed
maf.metadata  |= notAllowed # to be imported from OLAC
\token =
  element token {
    attribute id { xsd:ID }?,
    token.information,
    (
      (
       attribute from { DocumentLocation },
       attribute to { DocumentLocation }
      )
     | ## DTD => ,
      (
      [ a:defaultValue = "no" ]
      attribute join { "no" | "left" | "right" | "both" | "overlap" }?,
      text
      )
    )
  }
token.information &=  attribute form { string }?
token.information &=  attribute phonetic { string }?
token.information &=  attribute transcription { string }?
token.information &=  attribute transliteration { string }?
wordForm =
  element wordForm {
    wordForm.identification,
    wordForm.tokens,
    wordForm*,
    wordForm.content ?
  }
wordForm.tokens =
    ( attribute tokens  { xsd:IDREFS }
    | ## DTD => ,
     \token*
    )
wordForm.identification &= attribute entry { xsd:anyURI } ?
wordForm.identification &= attribute lemma { string } ?
wordForm.identification &= attribute form { string } ?
maf.flow = (\token | wordForm | wordForm.alt | fsm )+
fsm =
  element fsm {
    ( attribute init { fsm.state },
      attribute final { fsm.state } ) ?,
    ( attribute tinit { fsm.state },
      attribute tfinal { fsm.state } )?,
    transition+
  }
fsm.state = xsd:Name
transition =
  element transition {
    attribute source { fsm.state },
    attribute target { fsm.state },
    (\token | wordForm | wordForm.alt)
  }
wordForm.alt =
  element wfAlt { wordForm+ }
wordForm.content =
  ( attribute tag { xsd:IDREFS }
  | ## DTD => ,
    fs
  )
fs |= notAllowed          # defined in iso-fs-standalone.rnc
tagset =
  element tagset {
    ( attribute ref { xsd:anyURI }
    |  ## DTD => ,
      (dcs* & fsd* & tagset.lib*)
    )
  }
dcs =
  element dcs {
    attribute local { xsd:NCName },
    ( attribute registered { xsd:anyURI },
       attribute rel { "eq" | "subs" | "gen" } )?,
    element description { text }*
  }
fsd |= notAllowed       # defined in future iso-fsd.rnc
tagset.lib |= fvLib
tagset.lib |= fLib
fLib |= notAllowed           # defined in iso-fs-standalone.rnc
fvLib |= notAllowed          # defined in iso-fs-standalone.rnc
DocumentLocation = xsd:NMTOKEN # defined in LAF
```