ESTRELLA

IST-2004-027655

European project for Standardized Transparent Representations in order to Extend LegaL Accessibility
Specific Targeted Research or Innovation Project

Specific Targeted Research Project
Information Society Technologies

Deliverable N°: 4.4

A refined specification of APIs for interacting with and using legal knowledge systems

Version:	FINAL 1.0
Due date of Deliverable:	
Actual submission date:	
Start date of Project:	1 January 2006
Duration:	30 months
Project Coordinator:	Universiteit van Amsterdam (NL)
Lead contractor deliverable:	Corvinus University Budapest
Participating contractors:	Alma Mater Studiorum - Universita di Bologna (IT), University of Liverpool (UK), Fraunhofer Gesellschaft zur foerderung der angewandten forschung e.v. (DE), RuleWise b.v. (NL), RuleBurst (EUROPE) Ltd. (UK), knowledgeTools International GmbH (DE), Interaction Design Ltd. (UK), SOGEI - Societa Generale d'Informatica S.P.A. (IT), Ministro per le Riforme e le Innivazioni nella Publica Amministrazione (IT), Hungarian Tax and Financial Control Administration (HU), Budapesti Corvinus Egyetem (HU), Ministero dell'Economia e delle Finanze (IT), Consorzio Pisa Ricerche SCARL (IT)

Project funded by the European Community under the 6th Framework Programme

	Dissemination Level	
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

EXECUTIVE SUMMARY

Currently, the only inference engine which is capable of handling the most widely used LKIF format among Estrella partners (namely the LKIF XML format) is the Carneades engine. As it is a reference engine implemented in Scheme and provides an API solely in this language, integration with other systems is problematic.

Two goals which follow from the above facts are hence that (1) a standard programming interface has to be defined according to which vendors can develop tools either offering or deploying the services of an LKIF-compliant inference engine, and (2) as a first implementation of a web-service oriented LKIF inference engine a wrapper translating web-service calls to those of Carneades has to be implemented.

In this paper we analyze possible use-cases such an inference engine can offer, and based on that we propose a web-service oriented API for such engines.

Deliverable 4.4

A refined specification of APIs for interacting with and using legal knowledge systems

András Millinghoffer
Corvinus University Budapest

Contents

Introduction

Currently, the only inference engine which is capable of handling the most widely used LKIF format among Estrella partners (namely the LKIF XML format) is the Carneades engine. As it is a reference engine implemented in Scheme and provides an API solely in this language, integration with other systems is problematic.

Two goals which follow from the above facts are hence that (1) a standard programming interface has to be defined according to which vendors can develop tools either offering or deploying the services of an LKIF-compliant inference engine, and (2) as a first implementation of a web-service oriented LKIF inference engine a wrapper translating web-service calls to those of Carneades has to be implemented.

In this paper we analyze possible use-cases such an inference engine can offer, and based on that we propose a web-service oriented API for such engines.

1. Services of an ESTRELLA inference engine

In the section we shortly overview the LKIF representation format, and examine some typical sessions how one may interact with the engine.

1.1. The supported language

Currently the LKIF language consists of several sublanguages: this architecture depicts the building and application of a logical knowledge base. Besides OWL, which is also stated to be part of the language, the two main (native) layers are those of rules and arguments. These can be characterized as follows.

Rules are used for describing the "static" part of the overall environment, i.e. the background knowledge which consists of the description of the domain, like the roles present in it, their possible relationships and the rules governing the area.

The initial/starting information about the current/examined case(s) (i.e. the actors and their properties) is described by so-called axiom, which can be regarded as zero-condition rules, the conclusions of which are always true[1].

Argumentation structures (sole arguments and graphs built from them) are used for representing the results of inference about the queries about the domain. An argument graph contains interrelated propositions about the domain, which together provide a proof for a given conclusion (i.e. the queried statement). Arguments can be regarded as rules with their variables instantiated by some concrete values/objects of the domain.

Currently, one XML-based serialization format is supported by the Carneades engine for any kind of LKIF knowledge base elements, and hence this format will be used in the SOAP messages as well.

1.2. Use-cases

A typical scenario how one might use and deploy the services of an ESTRELLA inference engine is as follows.

- Create a knowledge base using an independent editor tool. Such a knowledge base may consist of several different kinds of knowledge base elements: (1) concept definitions, (2) general rules describing the domain itself, (3) facts and arguments about the case at hand, or even (4) argumentation elements.

- Before the knowledge base could be processed for inference, its syntactic validity has to be checked. The complete system (i.e. the inference engine and

[1] In the previous version of LKIF, axioms (formerly called facts) actually were represented by rules with no body parts, although this method is obsolete now it still can be used.

its wrapper together) must be capable of this; since Carneades currently provides only some basic functionality about this, a full-fledged XML validator has to be incorporated in the wrapper.

- Perform inference, i.e. query the engine for the truth value of some statements. The result of an inference is a truth value denoting whether the statement could be proven and in case of success, a stream of several argument graphs verifying the conclusion[2].

- In a realistic argumentation process, two parties might try to prove contradictory statements. In such a case, one of them provides an argumentation graph proving his truth (pro argument), but leaving some sub-statements (which have a default truth value) unproved. The opposing party then tries to undercut it by proving one of the above hanging sub-statements to have a different truth value than assumed previously (con argument).
  Hence an argumentation dialog consists of more and more developed argumentation trees sent from one party to the other. This implies the necessity of the capability of performing inference on given argumentation graphs, which might (and probably will) be augmented by further pieces of information about how arguments were created and hence how they could be rebutted.

1.3. Overview of services

Based on the above, the necessary services of a reference LKIF inference engine are the following.

- Loading and syntactic checking of a knowledge base.

  Input. The knowledege base in LKIF format or a web address pointing to a knowledge base.

  Output. True if the KB is syntactically correct, otherwise an error message describing the cause of the failure.

- Inference.

  Input. The goal statement to evaluate and the depth to which the "dialectical tree" of argumentation[3] has to be checked (additional, implementation-specific parameters may augment the call).

[2] Notice, that as in any logical inference system, a goal might have multiple supporting argumentation structures. However, in case of LKIF knowledge bases –since an argumentation can later be undercut through further examinations– it can be important to keep track of all of them in order to allow their later reuse.

[3] The inference engine searches a so-called "dialectical tree", where each level of the tree is an argument "pro" some proposition at issue or a counterargument "con" the statement at issue. Odd levels of the tree consist of pro arguments, while even levels consist of con arguments. The depth parameter controls how many levels in this dialectical tree to search.

Output. Notation of whether the statement could be proved; and in case of success the supporting argument graphs if demanded..

2. Overview of API calls

In the section we shortly overview what calls the SOAP API consists of and give a short description of each. Technical details are treated in Section 3.

2.1. Session handling

The communication with the server is organized along sessions, i.e. when a knowledge base is loaded the necessary resources are allocated for it and the corresponding inference tasks. These resources are freed when the client declares that the session can be closed or when it is detected to be broken (e.g. after the expiration of a specified timeout). Session management is necessary because of several reasons, these are the following.

- Efficiency: loading (retrieving) and processing of a knowledge base can be demanding tasks, and since a knowledge base is rarely used only for single inferences (often the results of previous inferences are reused in later ones), such optimizations are necessary.

- Usually multiple argument graphs are returned for a single query, however often only the existence of a positive answer is important, i.e. network traffic can be optimized if these answer graphs are transported only on demand. This also suggests the use of a session-based protocol.

The opening and closing of such a session can be done by the following two operations.

loadKnowledgebase. Loads an LKIF knowledge base from the specified URL (or alternatively from a string containing it) and assigns the specified ID to the allocated resources.
The returned value informs the client about the success of the operation or about the occurred error (e.g. syntactic errors in the knowledge base).

closeKnowledgebase. If the client has finished the experimenting with the knowledge base, it has to inform the server about it, so that it can free the corresponding resources. If the session breaks (i.e. a given timeout expires without any communication) the operation is preformed automatically.
In the response, only an affirmation is sent (of course the usual error messages, like referencing to an invalid knowledge base id can occur in this case as well).

2.2. Parameterization of inference engines

Beside the basic functionalities of inference, some auxiliary calls have to correspond to the implementation-specific properties of the inference engine[4]. The following calls support the querying and setting of such parameters.

getParametersList. The call has no input parameters, in the response, a list containing the names of possible parameters of the inference engine; a short description is provided with each of them as well.

getParameter. Returns the current value of a parameter. Parameters can be adjusted separately for each open knowledge base.

setParameter. Set the specified parameter to the selected value. Returned is the notification of success or an error message (e.g. invalid parameter name or value).

2.3. Inference

The most important part of the protocol handles corresponds to the inference about statements in the knowledge base. The task of inference is to determine the truth value of a query statement (i.e. whether it can be proven or not). The response for such a query is a series (stream) of different argument graphs which are valid proofs for the query statement. To handle this, these argument graphs are transmitted within smaller "sub-sessions": the argument graphs are stored locally, and are provided to the client on demand.

findArguments. Evaluates the query statement and stores the results, if any. Initially, only the notation of success (or failure) is returned, the argument graphs can be retrieved by calls to the following procedure.

getNextArgument. Return the next argument graph corresponding to a previous query. Streams of argument graphs are distinguished by "stream IDs".

[4] In case of Carneades such are the maximal number of expanded nodes, and the maximal number of "turns", i.e. the depth in which default-valued nodes are expanded.

3. SOAP interface of the web-service inference engine

In the following, the input and output parameters of each call are described in detail. The output is always a string containing the response to the request or an error message.

Each of the below sections end with a short XML fragment containing example requests and responses.

3.1. loadKnowledgebase

Loads a knowledge base and allocates the necessary resources corresponding to it.

Inputs

url. An URL pointing to the file containing the knowledge base, or a string containing it serialized in LKIF format.

knowledgebaseID. A string with the demanded ID for the knowledge base (i.e. for the associated session).

Output. Notification of success.

Exceptions.

- Invalid or already allocated knowledge base ID.
- URL inaccessible.
- Syntax error in knowledge base.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:loadKnowledgebase>
   <knowledgebaseID>myKB</knowledgebaseID>
   <url>file:///home/~milli/_projects/goods.xml</url>
  </ms:loadKnowledgebase>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:loadKnowledgebaseResponse>
   <response>true</response>
  </ms:loadKnowledgebaseResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.2. closeKnowledgebase

Deallocates the resources associated with the specified knowledge base.

Inputs.

knowledgebaseID.

Output. Notification of success.

Exceptions.

- Invalid (non-existent) knowledge base ID

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:closeKnowledgebase>
   <knowledgebaseID>myKB</knowledgebaseID>
  </ms:closeKnowledgebase>
 </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:closeKnowledgebaseResponse>
   <response>true</response>
  </ms:closeKnowledgebaseResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.3. getParametersList

Returns the names and descriptions of the adjustable parameters of the inference engine.

Inputs. None.

Output. A string containing a list of `(parameterID, parameterDescription)` pairs.

Exceptions. None.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:getParametersList>
  </ms:getParametersList>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:getParametersListResponse>
   <response>resources, turns</response>
  </ms:getParametersListResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4. getParameter

Returns the current value of one of the parameters of the inference engine associated with one of the sessions.

Inputs.

knowledgebaseID. ID of the knowledge base.

parameterID. ID of the parameter.

Output. The current value of the parameter in string format.

Exceptions.

- Invalid knowledge base ID.
- Invalid parameter ID.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <ms:getParameter>
   <knowledgebaseID>myKB</knowledgebaseID>
   <parameterID>turns</parameterID>
  </ms:getParameter>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:getParameterResponse>
   <response>1</response>
  </ms:getParameterResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.5. setParameter

Sets the parameter to the specified value.

Inputs.

knoweldgebaseID. ID of the knowledge base.

parameterID. ID of the parameter to adjust.

parameterValue. The new value of the parameter.

Output. Notification of success.

Exceptions.

- Invalid knowledge base ID.
- Invalid parameter ID.
- Invalid parameter value.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:setParameter>
   <knowledgebaseID>myKB</knowledgebaseID>
   <parameterID>turns</parameterID>
   <parameterValue>2</parameterValue>
  </ms:setParameter>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:setParameterResponse>
   <response>true</response>
  </ms:setParameterResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.6. findArguments

Evaluates a query statement and stores the proving argument graphs if any. The previously set (or default) values of the adjustment parameters are used.

Inputs.

knowledgebaseID. ID of the knowledge base.

streamID. The ID with which the stream of returned argument graphs will be referred to by subsequent `getNextArgument` calls.

queryStatement. The statement the truth value of which is examined.

initialArgument. The starting state of the argument graph which will be further expanded by the inference.

Output. Notification of whether any supporting graphs have been found.

Exceptions.

- Invalid knowledge base ID.
- Invalid (already existing) stream ID.
- Malformed query statement.
- Malformed initial argument graph.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:findArguments>
   <knowledgebaseID>myKB</knowledgebaseID>
   <streamID>myStream</streamID>
   <initialArgumentation></initialArgumentation>
   <query>’(goods g1)</query>
  </ms:findArguments>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:findArgumentsResponse>
   <response>true</response>
  </ms:findArgumentsResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.7. getNextArgument

Return the next element in a stream of argument graphs resulting from a previous `findArguments` call.

Inputs.

knowledgebaseID. ID of the referred knowledge base.

streamed. ID of the stream.

Output. A string containing the next argument graph (serialized LKIF format) in the stream if there is any more, otherwise an empty string.

Exceptions.

- Invalid knowledge base ID.
- Invalid stream ID.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:getNextArgument>
   <knowledgebaseID></knowledgebaseID>
   <streamID></streamID>
  </ms:getNextArgument>
 </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms">
 <SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ms:getNextArgumentResponse>
   <response></response>
  </ms:getNextArgumentResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. Architecture of a web-service oriented inference engine

A reference implementation of a web-service-based LKIF inference engine has been implemented based on the current (and only) reference implementation, namely Carneades. Hence the architecture of the whole system would consist of the reference inference engine Carneades itself, augmented by further modules responsible for (1) managing local resources of sessions, (2) communication through the web-service interface, including basic validation of incoming input and relaying it to the inference engine, and conversion between data formats used by the engine and by the web interface, and (3) parsing and validation of XML resources, since this ability is missing from Carneades.

In short, the functionalities of the whole system can be classified into the following subgroups (which are implemented by respective modules).

Inference. The inference engine is responsible only for the basic operation of inference about query statements and the returning of the supporting argumentations.

Session management. The wrapper handles all session-related data, i.e. the parametrization of the engine, the cached streams of argument graphs, etc.

Auxiliaries. Retrieval of knowledge bases and their validation can be done by a third module, basically oriented towards XML processing.

Appendix A. WSDL of the SOAP interface

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="lkifInference"
 targetNamespace="ms"
 xmlns:tns="ms"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ms="ms"
 xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:MIME="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:DIME="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
 xmlns:WSDL="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

 <schema targetNamespace="ms"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ms="ms"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <import
namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
 </schema>

</types>

<message name="loadKnowledgebaseRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
 <part name="url" type="xsd:string"/>
</message>
```

```
<message name="loadKnowledgebaseResponse">
 <part name="response" type="xsd:string"/>
</message>

<message name="getParametersListRequest">
</message>

<message name="getParametersListResponse">
 <part name="response" type="xsd:string"/>
</message>

<message name="getParameterRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
 <part name="parameterID" type="xsd:string"/>
</message>

<message name="getParameterResponse">
 <part name="response" type="xsd:string"/>
</message>

<message name="setParameterRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
 <part name="parameterID" type="xsd:string"/>
 <part name="parameterValue" type="xsd:string"/>
</message>

<message name="setParameterResponse">
 <part name="response" type="xsd:string"/>
</message>

<message name="findArgumentsRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
 <part name="streamID" type="xsd:string"/>
 <part name="initialArgumentation" type="xsd:string"/>
 <part name="query" type="xsd:string"/>
</message>

<message name="findArgumentsResponse">
```

```
 <part name="response" type="xsd:string"/>
</message>

<message name="getNextArgumentRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
 <part name="streamID" type="xsd:string"/>
</message>

<message name="getNextArgumentResponse">
 <part name="response" type="xsd:string"/>
</message>

<message name="closeKnowledgebaseRequest">
 <part name="knowledgebaseID" type="xsd:string"/>
</message>

<message name="closeKnowledgebaseResponse">
 <part name="response" type="xsd:string"/>
</message>

<portType name="lkifInferencePortType">
 <operation name="loadKnowledgebase">
  <documentation>Service definition of function
ms__loadKnowledgebase
  </documentation>
  <input message="tns:loadKnowledgebaseRequest"/>
  <output message="tns:loadKnowledgebaseResponse"/>
 </operation>
 <operation name="getParametersList">
  <documentation>Service definition of function
ms__getParametersList
  </documentation>
  <input message="tns:getParametersListRequest"/>
  <output message="tns:getParametersListResponse"/>
 </operation>
 <operation name="getParameter">
  <documentation>Service definition of function
ms__getParameter
```

```
  </documentation>
  <input message="tns:getParameterRequest"/>
  <output message="tns:getParameterResponse"/>
 </operation>
 <operation name="setParameter">
  <documentation>Service definition of function
ms__setParameter
  </documentation>
  <input message="tns:setParameterRequest"/>
  <output message="tns:setParameterResponse"/>
 </operation>
 <operation name="findArguments">
  <documentation>Service definition of function
ms__findArguments
  </documentation>
  <input message="tns:findArgumentsRequest"/>
  <output message="tns:findArgumentsResponse"/>
 </operation>
 <operation name="getNextArgument">
  <documentation>Service definition of function
ms__getNextArgument
  </documentation>
  <input message="tns:getNextArgumentRequest"/>
  <output message="tns:getNextArgumentResponse"/>
 </operation>
 <operation name="closeKnowledgebase">
  <documentation>Service definition of function
ms__closeKnowledgebase
  </documentation>
  <input message="tns:closeKnowledgebaseRequest"/>
  <output message="tns:closeKnowledgebaseResponse"/>
 </operation>
</portType>

<binding name="lkifInference"
type="tns:lkifInferencePortType">
 <SOAP:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
 <operation name="loadKnowledgebase">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="getParametersList">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="getParameter">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="setParameter">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="findArguments">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="getNextArgument">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="closeKnowledgebase">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
     <SOAP:body use="encoded" namespace="ms"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
```

```
 </operation>
</binding>

<service name="lkifInference">
 <documentation>gSOAP 2.7.9k generated service
definition</documentation>
 <port name="lkifInference" binding="tns:lkifInference">
  <SOAP:address
location="http://localhost/~milli/cgi/myService.cgi"/>
 </port>
</service>

</definitions>
```