

FutureFarm WP4 – Rule evaluation interface

Authors: Raimo Nikkilä, Edward Nash

Version: 1.0

Last edit: 04.01.2010

Introduction

This document

This document specifies an interface for the automated evaluation of agricultural production standards. The representation specified by the FutureFarm project deliverable 4.1 for encoded agricultural standards is assumed and used in communication. Additionally, the methods for the discovery and distribution of these standards is assumed to be that, or equivalent of that specified in by the FutureFarm project deliverable 4.2.

The scope of this document is restricted only to the interface and communication of the evaluation service. The mechanical evaluation of the rules themselves is not discussed.

Purpose of the rule evaluator

The rule evaluator is a component which evaluates an encoded set of agricultural standards together with the farm data to determine whether the farming practices agree with those required by the standards. To achieve this, the evaluator provides two operations:

1. One operation to determine what data is required for the evaluation of a set of rules.
2. One operation to perform the actual evaluation.

The communication with the component is through the exchange of machine-readable encoded rules and data. The implementation of the component, and hence the communication with it, can be achieved through various technologies, e.g. web-services, COM component, software library, etc. This document specifies as one concrete example a RESTful web-service implementation using XML for rules and data transfer. Other implementations will require the use of other technologies as appropriate.

Technical description

Overview

This technical description is two-fold with part of it being abstract for the purpose of different technical implementations which may be a software component for Java or .Net platform or as a web service using REST or SOA. The detailed technical description in this document is focused on a REST-based implementation and provides the necessary interfaces and schemata for a REST-based implementation.

The evaluation service is stateless in the sense that it does not maintain sessions or involve itself in extended communication with any client software. Necessary information can be sent as a single package and reply received similarly as a single document. The service may allow existing information to be either in a database or obtained through another web service, however, this operation is transparent to the client. Additionally, it is possible for the client messages to contain external references that the service is expected to resolve automatically.

Results of evaluation

The constant values for the result of the evaluation are those of:

True	The rule evaluates to true with the given data
False	The rule evaluates to false with the given data
Indeterminate due to data	The rule can not be evaluated because the data is insufficient to properly determine a result for the rule. Note that this result is different from a service error reply indicating the lack of data for a rule.
Indeterminate due to rule	The rule can not be evaluated at all. This is because it is either flawed (uncomputable) or the implementation can not reliably evaluate the rule to any result (e.g. due to missing models or other computational components)..

The component provides two basic functionalities, first for identifying the data necessary for the evaluation of a collection of rules and second for evaluating a collection of rules with data. Since both of the operations are stateless they need not be used in order, although it is expected that they usually will be. An additional minor operation is also provided to allow the client to retrieve the list of data formats which are known to the Rules App.

Identifying the required data

Query: set of rules, acceptable data format(s)

Reply: description of required data

Possible error situations:

Syntactic error	The query contains a syntactic error
Semantic error	The query does not adhere to the expected format
Format error	The requested data format is not known to the Rules App
Bad reference	The query contains a reference to an invalid or non-existent resource
Internal errors	Unable to determine the required data
	General internal error

Submitting data and rules for evaluation

Query: set of rules and a set of data

Reply: result of evaluation for each rule

Possible error situations:

Syntactic error	The query contains a syntactic error
Semantic error	The query does not adhere to the schema (TBA)
Bad reference	The query contains a reference to an invalid or non-existent resource
Insufficient data	Lack of required data, rule evaluation can still result in "Indeterminate" even with the necessary data identified by the service
Bad data	Data contains errors
Erroneous rule	Rule that can not be evaluated or can not be evaluated in finite time

Internal errors	Timeout (can be caused by an erroneous rule)
	General internal error

Retrieving list of known data formats

Query: (no parameters)

Reply: list of known data formats

Possible error situations:

(none)

Detailed technical description of RESTful binding

General communication

The service is loosely based on REST and shares certain aspects with REST but can not be considered a true REST service unlike the catalogue and rule services. Since the interface is based entirely on the XML communication, the REST operations and resources are mere guidelines and should be ignored in an alternative implementation scheme.

All communication occurs in XML with four message formats specified. The XML documents are intended to contain all necessary information so that there is no dialogue between the client and the service and no additional files need to be uploaded. External references in the XML document are possible and the service can resolve these automatically.

The expected workflow of the system is:

1. Client requests list of supported data formats
2. Client composes a RuleSet message
3. RuleSet is sent to /requiredData resource
4. RuleSetData message is received in reply
5. Client interprets the RuleSetData message and composes an EvaluationSet message
6. EvaluationSet message is sent to /evaluate resource
7. EvaluationSetResults message is received in reply
8. Client interprets and utilises the EvaluationSetResults message

Since the error messages, defined in appendix C of this document, are almost all possible for both main resources, they are not explicitly stated in the description of the individual resources.

Values and identifiers

The only values specific to the rule evaluation interface are the constant values indicating the result of the evaluation for an ff:Rule or an ff:RulesSubSet. All other values and identifiers are those of existing schemata and are thus defined elsewhere.

The communication involves certain values which are somewhat complicated, such as the Concept element which represents and identifies the required data or the actual data itself. For these values, there is no predefined format, but rather any valid XML is considered as a legal value.

While not the most elegant solution, having the XPath queries as part of the required data is a very pragmatic approach and provides a convenient way to both identify and acquire the necessary information. These data elements only assume XML and the specific XML schema used is

described by the XPathRoot.

Identifying the required data

This operation returns the data required to evaluate a set of rules. For rules that can not be evaluated due to a reason that would result in "Indeterminate due to rule" no data is required.

Corresponding REST operation: POST

Resource: /requiredData

Query: <re:RuleSet>

Successful reply: <re:RuleSetData>-element

Submitting data and rules for evaluation

This operation evaluates the given rules and data producing a result for each rule.

Corresponding REST operation: POST

Resource: /evaluate

Query: <re:EvaluationSet>

Successful reply: <re:EvaluationSetResults>-element

Retrieving list of known data fomats

This operation returns the metadata of the service, including the list of available data formats .

Corresponding REST operation: GET

Resource: /metadata

Query: (none)

Successful reply: <re:Metadata>-element

Security considerations

A degree of security sufficient for the secure evaluation of rules requires encryption and basic authentication at least as an option. Several technologies exist that can provide this level of security and if the service is implemented over HTTP the technology to use would be HTTPS.

Discussion

This document does not cover the alternative implementations of this service as a software component or as other than a REST-based web service. These implementations can, however, benefit from the REST-based specification in this document.

Some design decision had to be made and as a result, an arbitrary set of rules is accepted instead of individual rules to prevent excessive redundancy of common data between rules in the evaluation. However, there is no evaluation above the rule level, i.e. there is no evaluation of ff:RulesSubSet elements but rather, these are left for the client to evaluate and leaves the evaluation component as a purely RIF-based evaluator.

Appendix A - Queries XSD schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ff="http://www.futurefarm.eu/standards"
```

```

xmlns:re="http://www.futurefarm.eu/evaluator" targetNamespace="http://www.futurefarm.eu/evaluator"
elementFormDefault="qualified">
  <xs:import namespace="http://www.futurefarm.eu/standards"
schemaLocation="http://schema.futurefarm.eu/agstandard/schema.xsd">
    <xs:annotation>
      <xs:documentation xml:lang="en">FutureFarm encoding of standards</xs:documentation>
    </xs:annotation>
  </xs:import>
  <xs:element name="EvaluatorRequest">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="re:RuleSet"/>
        <xs:element ref="re:EvaluationSet"/>
      </xs:choice>
      <xs:attribute name="version" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- re:RuleSet -->
  <xs:element name="RuleSet">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ff:RulesSubset" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!-- re:EvaluationSet -->
  <xs:element name="EvaluationSet">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ff:RulesSubset" maxOccurs="unbounded"/>
        <xs:element ref="re:Data" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Data">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="re:forRule" maxOccurs="unbounded"/>
        <xs:element ref="re:Concept"/>
        <xs:element ref="re:DataContent"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="forRule">
    <xs:complexType>
      <xs:attribute ref="ff:ruleId" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Concept" type="ff:ConceptType"/>
  <xs:element name="DataContent">
    <xs:complexType>
      <xs:sequence>
        <xs:any maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Appendix B - Replies XSD schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ff="http://www.futurefarm.eu/standards"
xmlns:re="http://www.futurefarm.eu/evaluator" targetNamespace="http://www.futurefarm.eu/evaluator"
elementFormDefault="qualified">
  <xs:import namespace="http://www.futurefarm.eu/standards"
schemaLocation="http://schema.futurefarm.eu/agstandard/schema.xsd">
    <xs:annotation>
      <xs:documentation xml:lang="en">FutureFarm encoding of standards</xs:documentation>
    </xs:annotation>
  </xs:import>
  <xs:element name="EvaluatorReply">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="re:RuleSetData"/>
        <xs:element ref="re:EvaluationSetResults"/>
        <xs:element ref="re:Metadata"/>
        <xs:element ref="re:Error"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

```

```

        <xs:attribute name="version" use="required"/>
    </xs:complexType>
</xs:element>
<!-- re:RuleSetData -->
<xs:element name="RuleSetData">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:RuleData" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="RuleData">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:forRule" maxOccurs="unbounded"/>
            <xs:element ref="ff:Concept"/>
            <xs:element ref="re:RuleDataXPath" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="forRule">
    <xs:complexType>
        <xs:attribute ref="ff:ruleId" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="RuleDataXPath">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:XPathRoot"/>
            <xs:element ref="re:XPath" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="XPathRoot" type="xs:string"/>
<xs:element name="XPath" type="xs:string"/>
<!-- re:EvaluationSetResults -->
<xs:element name="EvaluationSetResults">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:RulesSubSetResult" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="RulesSubSetResult">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:RuleResult" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="ff:ruleSetId" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="RuleResult">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="re:Result"/>
        </xs:sequence>
        <xs:attribute ref="ff:ruleId" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="Result">
    <xs:simpleType>
        <xs:restriction base="xs:token">
            <xs:enumeration value="True"/>
            <xs:enumeration value="False"/>
            <xs:enumeration value="DataFail"/>
            <xs:enumeration value="RuleFail"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<!-- re:Metadata -->
<xs:element name="Metadata">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="serviceId" type="xs:string" minOccurs="0"/>
            <xs:element name="serviceProvider" type="xs:string" minOccurs="0"/>
            <xs:element name="acceptsDataFormat" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<!-- re:Error -->

```

```

<xs:element name="Error">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="re:ErrorCode"/>
      <xs:element ref="re:Explanation" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ErrorCode" type="xs:string"/>
<xs:element name="Explanation" type="xs:string"/>
</xs:schema>

```

Appendix C - Error codes of a RESTful binding

100	Syntax error	HTTP Error code	Common for both operations
110	Semantic error	400	Common for both operations
120	Bad reference	400	Common for both operations
210	Insufficient data	400	
210	Bad data	400	
220	Bad rule	400	
300	Unable to determine the required data	500	
310	Timeout	500	Common for both operations
390	General internal error	500	Common for both operations
400	Authentication required	401	
410	Authentication error	403	