INTERNET-DRAFT
Document: draft-ng-opes-irmlsubsys-00.txt
Expires: January 2002

C. W. Ng
P. Y. Tan
H. Cheng
Panasonic Singapore Labs

July 2001

Sub-System Extension to IRML

Status of this Memo

Abstract

The Intermediary Rule Markup Language (IRML) [2] is an XML-based language that can be used to describe service-specific execution rules for network edge intermediaries under the Open Pluggable Edge Services (OPES) framework, as described in [3] and [4]. This memo discusses the need for OPES framework to have different sub-systems in different deployment scenario, and proposes additions to IRML for a more flexible approach to supporting different sub-systems.

Table of Contents

1. Introduction

The Intermediary Rule Markup Language (IRML) [2] is an XML-based language that can be used to describe service-specific execution rules for network edge intermediaries under the Open Pluggable Edge Services (OPES) framework, as described in [3] and [4]. This memo discusses the need for OPES framework to have different sub-systems in different deployment scenario, and proposes additions to IRML for a more flexible approach to supporting different sub-systems.

This memo begins in Section 2 by presenting the motivation behind having sub-systems support in IRML. Section 3 proposed a set of QoS extension to the "property" element defined in the IRML, and Section 4 presents some examples illustrating possible use of these extensions.

2. Motivation for Different Sub-Systems

In [4], various different examples services that the OPES intermediary can provide are presented. These services cover a wide application range, including data insertion into HTML pages, web or AV content adaptation, and user profiles creation. These different services would have different set of requirements. The current set of IRML properties, in its initial drafting, has been focused to the Hyper Text Transfer Protocol (HTTP). These lead to difficulty in constructing rules for other applications. For instance, limited client bandwidth adaptation and streaming media adaptation requires a whole set of quality of services properties, such as bandwidth allocated and the packet lost rate, which is absent from the IRML framework. Creation of user profiles needs user specific parameters, such as the user identification, current IP address of the user, etc.

Since the required set of property parameters is different for different services, it would be much more manageable to classify these parameters into different sub-systems. Furthermore, this allows a specific implementation of the OPES intermediary to incorporate only the parameters in the sub-systems that it needs for the services it provides, and not the entire range of properties that is defined.

In addition, since the development of the OPES framework is still in its infancy stage, the sub-systems concept in IRML allows researchers to create new sub-systems to experiment with new properties, and still maintain conformance to the standard OPES framework. For example, some implementations may desire the ômatchesö attributes of the ôpropertyö element to have arithmetic support, instead of restricting to regular expression. They can implement such support using new sub-systems.

3. Proposal to IRML Sub-Systems

This memo proposed two new attributes to the ôpropertyö element of the current IRML specifications: ôsub-systemö and ôno-sub-systemö.

3.1. The ôsub-systemö Attribute

The ôsub-systemö attribute is used to specify the sub-system where the value of the property specified by the ônameö attribute can be derived.

In order to maintain compatibility with the current IRML specification, the ôsub-systemö attribute is optional. When it is omitted, the default value of ôStandardö is assumed, which implies that the property belongs to the set of parameters currently defined in [2].

3.2. The ôno-sub-systemö Attribute

The ôno-sub-systemö attribute complements the ôsub-systemö attribute, and is used to specify the default matching result when the sub-system required (as specified by the ôsub-systemö attribute)

is not supported by the IRML engine. It can have a value of ômatchö or ôno-matchö.

A value of ômatchö implies that if the required sub-system is not supported, the IRML engine should treat it as if the ôpropertyö condition is met. Conversely, a value of ôno-matchö implies that if the requires sub-system is not supported, the rule engine should treat it as if the ôpropertyö condition is not met.

In order to maintain compatibility with the current IRML specification, the ôno-sub-systemö attribute is optional. When it is omitted, the default value of ôno-matchö is assumed.

3.3. Proposed IRML DTD

The proposed IRML DTD (Document Type Definition) with the two proposed attributes for the ôpropertyö element is shown below.

```
<!ELEMENT rulemodule (owner, protocol, rule+)>

<!ELEMENT owner      (name, id)>
<!ATTLIST owner      class (content provider|access provider|
                            client) #REQUIRED>

<!ELEMENT name       (#PCDATA)>

<!ELEMENT id         (#PCDATA)>

<!ELEMENT protocol   (#PCDATA)>

<!ELEMENT rule       (property|action)+>
<!ATTLIST rule       processing-point (1|2|3|4)      #REQUIRED>

<!ELEMENT property   (property|action)+>
<!ATTLIST property   name            CDATA           #REQUIRED>
<!ATTLIST property   matches         CDATA           #REQUIRED>
<!ATTLIST property   case-sensitive  (yes|no)        ônoö>
<!ATTLIST property   sub-system      CDATA           ôStandardö>
<!ATTLIST property   no-sub-system   (match|no-match) ôno-matchö>
```

```
<!ELEMENT action     (#PCDATA)>
```

3.4. Proposed Architecture

The proposed architecture with sub-systems is shown in Figure 1 below. Here the entire IRML rule engine consists of three parts: the rule parser, the ôStandardö sub-system, and any other additional sub-systems.

The rule parser and the ôStandardö sub-system are mandatory. Together, they implements all the standard IRML rule engine functionality specified in [2]. Any other additional sub-systems are optional. These additional sub-systems either provide enhance functionality, or is for experimental purposes.

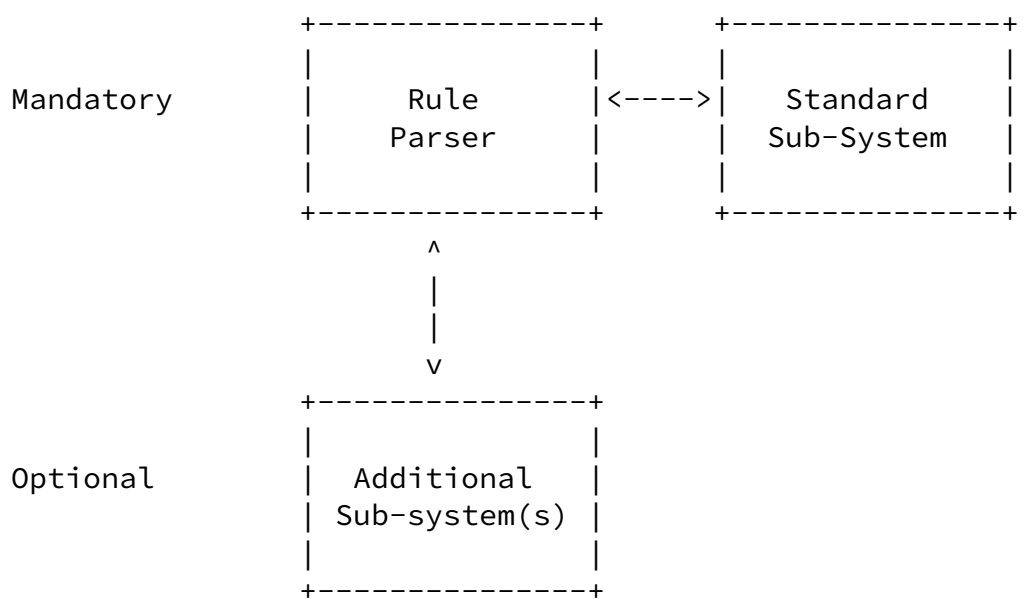


Figure 1 Architecture of Rule Engine

With such an implementation, the rule parser will parse each property and see what kind of sub-system the property uses. If the required sub-system is supported, the property is then passed to the corresponding sub-system for evaluation (i.e. check if the condition specified is met). In the event that the required sub-system is absent, the rule parser will then assume the condition to be met or not according to the ôno-sub-systemö attribute of the property.

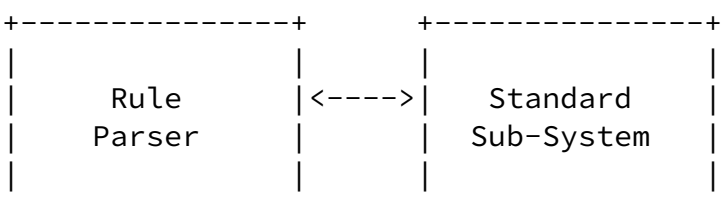
In this modular approach, implementation becomes easier. In addition, because conditions are evaluated by the sub-systems, each sub-system can choose to support arithmetic comparison, bolean expressions, etc, instead of being limited to regex, which may be sufficient for matching HTTP headers, but are at best awkward for evaluating conditions which involve QoS or System parameters.

4. Example

4.1. Scenario Overview

Figure 2 below depicts a scenario, which illustrates the concept of sub-system. In this figure, three sub-systems are shown interfacing with the OPES rule engine. These are: ôStandardö sub-system, ôQoSö sub-system, and the ôSystemö sub-system. The ôStandardö sub-system uses the standard HTTP properties as defined by [2]. The ôQoSö sub-system provides the rule engine an interface with the QoS control

and monitoring modules, such as Traffic Engineering Module. This enables rules to construct condition involving QoS parameters. The ôSystemö sub-system provides the rule engine an interface to the operating system. This enables rules to be constructed using conditions involving system parameters, such as the system load, the number of TCP/UDP connections the system is currently handling, etc.



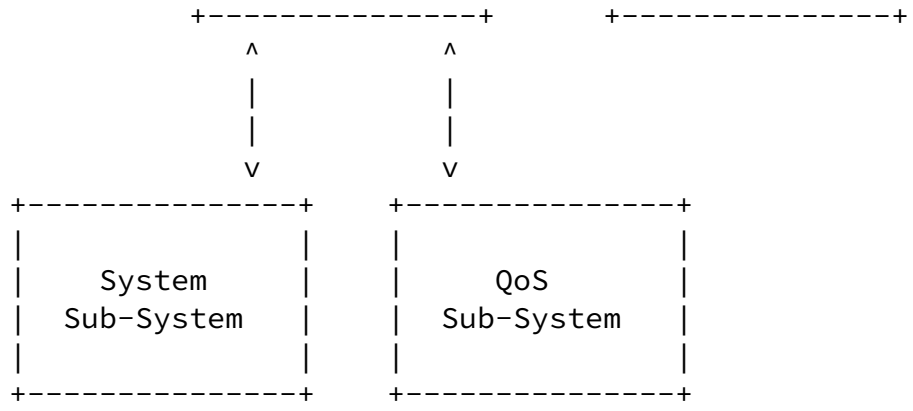


Figure 2 Sub-Systems interfaces with rule engine.

4.2. IRML Examples

The first example below illustrates the case where a HTML page is adapted to suit the allocated bandwidth of the client. Here we assume that there is a ôQoSö sub-system which defined the property name of ôallocated-bandwidthö to give the value of bandwidth allocated to the client in bits per second. In addition, the ôQoSö sub-system also overloads the ômatchesö attributes to support arithmetic comparison (i.e. greater than, smaller than).

In this example, the bandwidth of client is used to determine how the HTML page is translated to WML page. If the bandwidth allocated is large than 9.6 kbps, the translated WML page will contain some bitmaps. If it is smaller, bitmaps are replaced by alternate text.

When the ôQoSö sub-system is not supported, the rule-engine should assume that the client has a tight bandwidth.


```
<rule processing-point="4">
   <!-- check the allocated bandwidth and adapts accordingly -->
   <property  sub-system=öQoSö no-sub-system=ömatchö
              name="allocated-bandwidth" matches="<9600">
      <!-- Bandwidth is low, no image  -->
```

```
      <action>proxylet://localhost/html2wml?image=no</action>
    </property>
    <property  sub-system=öQoSö name="allocated-bandwidth"
              matches=">=9600">
      <!-- Bandwidth is high, can embed image  -->
      <action>proxylet://localhost/html2wml?image=yes</action>
    </property>
  </rule>
```

The second example illustrates the scenario where the access provider wishes to re-direct the client request periodically to a remote proxy for logging purposes. Here, we assume that there is a ôSystemö sub-system that provides support for the property name ôrequest-countö. This gives the accumulated number of requests the proxy has serviced. In addition, the ôSystemö sub-system also overloads the ômatchesö attribute to support arithmetic expressions. In this example, the ômatchesö attribute is ô($%400)==0ö. The ô$ö is a token to be replaced by the value of the parameter specified by the ônameö attribute.

```
  <rule processing-point="1">
    <!-- check the number of requests -->
    <property  sub-system=öSystemö no-sub-system=öno-matchö
              name="request-count" matches="($%400)==0">
      <!-- This is the 400th request, log it -->
      <action>icap://log.server/log</action>
    </property>
  </rule>
```

5. References

[1] Bradner, S., "The Internet Standard Process û Revision 3", BCP 9, RFC 2026, October 1996.

[2] Beck, A., Hoffman, M., "IRML: A Rule Specification Language for Intermediary Services", Work In Progress, draft-beck-opes-irml-00.txt, February 2001.

[3] Tomlinson, G., Orman, H., Condry, M., Kempf, J., "Extensible Proxy Services Framework", Work In Progress, draft-tomlinson-epsfw-00.txt, 2000.

[4] Beck, A., Hoffman, M., Condry, M., "Example Services for Network Edge Proxies", Work In Progress, draft-beck-opes-esfnep-01.txt, November 2000.

6. Author's Address

Chan-Wah Ng
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Send Ave #04-3530
Tai Seng Industrial Estate
Singapore 534415
Phone: (+65) 381 5420
Email: cwng@psl.com.sg

Pek-Yew Tan
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Send Ave #04-3530
Tai Seng Industrial Estate
Singapore 534415
Phone: (+65) 381 5470
Email: pytan@psl.com.sg

Cheng Hong
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Send Ave #04-3530
Tai Seng Industrial Estate
Singapore 534415
Phone: (+65) 381 5477
Email: hcheng@psl.com.sg