Internet Draft
Expires: December 23, 2003
Document: draft-beck-opes-irml-03.txt
June 24, 2003
Category: Informational

A. Beck
M. Hofmann
Lucent Technologies

IRML: A Rule Specification Language for Intermediary Services

Status of this Memo

Abstract

The Intermediary Rule Markup Language (IRML) is an XML-based language that can be used to specify rules for the execution of OPES services.

OPES services are a new class of applications running on network intermediaries, such as caches, proxies, gateways, etc. or dedicated (callout) servers. They are invoked through intermediaries acting on behalf of application endpoints. IRML is designed to serve as a simple and efficient, but yet powerful language to express the service execution policies of application endpoints. IRML rules are typically processed by intermediaries that trigger the execution of OPES services according to these rules and policies.

Table of Contents

1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2].

DATA TRANSACTION

A message exchange between a data consumer and a data provider, typically consisting of a data request and a data response message.

DATA PATH

The path that data requests and responses take through the network. Typically, data requests and responses flow between a data consumer and a data provider.

DATA PATH PROTOCOL

The application-layer protocol used between the two endpoints of a data transaction, e.g. HTTP or RTSP.

Other terminology used in this document is consistent with that defined and used in [3].

2 Introduction

This document defines the Intermediary Rule Markup Language (IRML) in an attempt to create a standard representation of OPES service execution policies. It is designed to be simple, efficient, and easy to understand but yet powerful enough to cover complex scenarios with a large number of rules and rule conditions.

IRML can be used in particular, but not exclusively, in the context of the OPES framework as proposed in [3] and [4]. Since OPES services may only be executed on behalf of data providers or data consumers - the two endpoints of a data transaction - IRML-specified rules must reflect the intents of either of the two endpoints. A data provider like CNN, for example, may wish to specify the conditions under which CNN web pages may be adapted to fit the screen for users with small wireless devices. The customers of an ISP, on the other hand, may wish to specify the conditions for the execution of a privacy service that removes certain information from user requests before they are sent to an origin server (e.g. the "referer" header in HTTP requests).

In its basic form, IRML is not tied to any particular data path protocol or deployment environment. However, IRML allows for extensions that can be used to better accommodate the specifics of data path protocols or different deployment environments.

It is anticipated that IRML rules will either be authored directly by data providers or data consumers or indirectly through an authorized delegate such as an ISP.

IRML-specified rules are meant to be processed by a rule processor on an intermediary device (e.g. a caching proxy, an access router, a wireless gateway, or a switch) located in the path between data providers and consumers. IRML rules are matched by evaluating rule conditions against the properties of incoming and outgoing messages and possibly also system and environment variables.

IRML is designed to be easily created and edited by graphical tools. It is based on XML [5], so many publicly available parsers and editing tools can be used in this process. The structure of the language maps closely to its behavior, so that an editor can easily understand IRML rule modules, even ones written by hand. The language is also designed so that an IRML-enabled intermediary or admin server can easily confirm the validity of IRML rule modules.

Although this document does not define a secure and reliable mechanism for transferring IRML rule files to intermediary devices (or other OPES entities), it is expected that existing protocols (e.g. HTTPS) can be used for this purpose.

3 IRML Syntax and Grammar

IRML is an application of XML. Thus, its syntax is governed by the rules of the XML syntax as defined in [5], and its grammar is specified by a DTD, or Document Type Definition. The IRML DTD can be found in Appendix A.

An IRML rule module that appears as a top-level XML document is identified with the formal public identifier "-//IETF//DTD RFCxxxx IRML 1.0//EN".

An IRML rule module embedded as a fragment within another XML document is identified with the XML namespace identifier "http://www.rfc-editor.org/rfc/rfcxxxx.txt".

3.1 High-level Structure of IRML Documents

Valid and well-formed IRML documents consist of one or more rule modules. Each rule module has a section that describes the rule module author. IRML rule modules can be authored directly by data providers or consumers but also indirectly through authorized delegates. The IRML rule author section is followed by one or more IRML rule sets and information about the party that authorized each set of rules.

The rules contained in a rule set each consist of a number of IRML rule conditions and a number of consequent IRML rule actions that are to be performed if the rule conditions become true. Through IRML rule actions, endpoints can request the execution of services.

The conditions within a rule refer to message properties in the request or response message of a given data transaction. They are met if the property value matches the pattern(s) specified in the rule condition(s). Rule conditions may also refer to system or service-manipulated environment variables.

3.2 IRML Rule Modules

IRML rule modules represent the service execution policies of data providers and data consumers. The service execution policy of a specific data provider or consumer MAY be expressed through more than one IRML rule module.

3.2.1 The "rulemodule" Element

The "rulemodule" element is the root element for all IRML rule modules and MAY/MUST contain the following elements (see also IRML DTD in Appendix A).

3.3 IRML Rule Authors

IRML rule modules can be authored by data providers, data consumers, or authorized delegates who author rule modules on behalf of data providers or consumers.

3.3.1 The "author" Element

The "author" element specifies the author of a rule module. Each rule module MUST have exactly one author.

Attributes of "author"

Name	Values	Default
type	delegate|self	self

The "type" attribute assigns a rule module author to one of the two possible types of rule module authors. An attribute value of "delegate" indicates that the rule module author acts as a delegate for one or more endpoints. An attribute value of "self" indicates that a rule module is authored directly by a data provider or consumer.

The rule module author is described further by a "name", "id", and optionally a "contact" element which are described in the following sections.

3.3.2 The "name" Element

The "name" element MUST contain a descriptive name for the rule module author. The name does not have to be unique among rule module authors.

3.3.3 The "contact" Element

The "contact" element is an optional element that, if used, MUST contain a valid email address at which the rule author can be contacted.

3.3.4 The "id" Element

The "id" element MUST contain a globally unique identifier for the rule module author, for example a URI or an email address.

3.3.5 IRML Rule Author Examples

```
<author type="delegate">
  <name>Comcast</name>
  <contact>rule-management@comcast.com</contact>
  <id>www.comcast.com</id>
</author>

<author type="self">
  <name>Andre Beck</name>
  <contact>abeck@home.com</contact>
  <id>abeck@home.com</id>
</author>
```

3.4 IRML Rule Sets

IRML rule sets represent a collection of rules that apply to and have been authorized by the same data provider or data consumer.

3.4.1 The "ruleset" Element

The "ruleset" element MUST contain one or more "rule" elements. Rule modules authored directly by a data provider or consumer MUST contain exactly one "ruleset" element. Rule modules authored indirectly through an authorized delegate on the other hand MAY contain more than one set of rules where each rule set MUST be authorized by a different data provider/consumer.

3.4.2 The "authorized-by" Element

The "authorized-by" element specifies the authorizing data transaction endpoint for a set of rules. This MUST be either a data

provider or a data consumer. In self-authored rule modules, the
authorizing endpoint MUST be identical with the rule module author.

Name	Values	Default
class	data-provider|data-consumer	
type	individual|group	

The "class" attribute specifies whether the authorizing endpoint is a data provider or data consumer.

The "type" attribute specifies whether the corresponding rule set applies to and has been authorized by an individual data provider/consumer or a group of data providers/consumers. An attribute value of "group" means that the rules in the corresponding rule set apply to all members of the specified group. A value of "group" MAY only be used if the rule module author is a delegate and primarily serves as a means of simplifying cases where a large number of data providers/consumers have identical rules. This can be the case, for example, if an ISP manages rules on behalf of its customers or in an enterprise environment.

The "authorized-by" element MUST contain a "name" and "id" element as described in 3.3.2 and 3.3.4 and MAY contain a "contact" element as described in 3.3.3 to further identify the authorizing endpoint.

If the authorizing endpoint is of the type "individual", then these elements MUST contain the endpoint's name, globally unique id (such as a URI or email address), and optionally a contact email address.

If the authorizing endpoint is of the type "group", then the "name" and "id" elements MUST contain a descriptive name and a globally unique identifier for the corresponding group of data providers/consumers and the "contact" element SHOULD contain an email address at which the delegate managing the group can be contacted.

3.4.3 The "protocol" Element

The "protocol" element specifies the applicable data path protocol for a set of rules. It MUST contain the protocol acronym of the applicable data path protocol. For example, rule sets applying to HTTP messages MUST specify "HTTP" in the "protocol" element. Each rule set MUST apply to exactly one data path protocol.

3.5 IRML Rules

IRML rules make up the actual service execution policies of data providers and consumers. They are composed of rule conditions and rule actions.

3.5.1 The "rule" Element

The "rule" element typically contains one or more "property" elements which represent rule conditions in IRML. In the case of unconditional rules, "rule" elements MAY also contain elements representing rule actions ("execute" elements).

Attributes of "rule"

Name	Values
processing-point	1|2|3|4

The "processing-point" attribute specifies at which of the four points in figure 1 a rule MUST be processed by the rule engine on the intermediary device. The four common processing points of an OPES intermediary are further defined in [6]. Implementation architectures for other intermediary devices might define different or additional processing points.

Figure 1 shows the typical HTTP data flow between a client, an IRML-enabled intermediary (in this case a caching proxy), and an origin server. The four processing points (1-4) represent locations in the round trip message flow where rules can be processed and service applications can be executed. A virus scanning service for instance could be executed at point 3 in figure 1 in order to scan web objects for viruses before they are stored in the cache. A URI-based request filtering service on the other hand could be executed at point 1 and a language translation service could be executed at point 4. Note that in a caching proxy the message flow may skip points 2 and 3 after point 1 if the requested object can be served from cache.

```
+--------+       +-----------+       +--------+
|        |<------|4         3|<------|        |
| Client |       |  Caching  |       | Origin |
|        |       |   Proxy   |       | Server |
|        |------>|1         2|------>|        |
+--------+       +-----------+       +--------+
```

Figure 1: Rule Processing/Service Execution Points

Depending on the service type, rules may be processed and services may be executed at any of the four points outlined in figure 1. However, the local policy of an intermediary MAY prevent endpoints from executing service applications at certain processing points. For example, data consumers may not be allowed to execute service applications at processing point 3 because a modified response message may subsequently be cached and served to other data consumers as well.

3.6 IRML Rule Conditions

IRML rule conditions specify a pattern and a message, system, or service property. Rule conditions are evaluated by matching the specified pattern against the value of the specified message, system, or service property at the time of the evaluation.

3.6.1 The "property" Element

The "property" element represents a rule condition and MAY contain one or more other "property" elements and/or one or more elements representing rule actions, i.e. "execute" elements. Nested "property" elements represent a hierarchical "AND" relationship. This means that an inner "property" condition can only be true if the outer "property" condition is true and so forth.

Attributes of "property"

Name	Values	Default
name	CDATA	
context	(req-msg|res-msg|system|service)	
matches	CDATA	
not-matches	CDATA	
case-sensitive	(yes|no)	"no"
sub-system	CDATA	"standard"

The "name" attribute specifies the name of the property that is to be matched.

The "context" attribute specifies the property type further. A property context value of "req-msg" or "res-msg" indicates that the corresponding "property" element refers to a request or a response message property, i.e. a protocol-specific request or response header. For HTTP messages, for example, the list of protocol-specific header names is defined in [7]. IRML, however, is not limited to the message properties defined in protocol specifications. User-defined message properties (e.g. user-defined protocol headers) MAY also be used.

Note that rule conditions at processing points 1 and 2 can typically only reference request message properties, whereas rule conditions at processing points 3 and 4 can reference request and response message properties.

If the "context" attribute is specified as "system", then the property name refers to system variables that are set by the intermediary.

IRML defines the following general system property variables which

MUST be supported by all IRML-compliant intermediaries:

Property Name	Value
"system-date"	a timestamp using the Internet date-time format as defined in [8]
"client-ip"	the IP address of the data consumer in the common dotted-decimal format

In addition to these general system properties, IRML also defines data path protocol-specific system properties.

Currently, IRML only defines protocol-specific system properties for HTTP which can be found in Appendix B.

If the property "context" attribute is specified as "service", then the property name refers to service-specific environment variables that can be set and modified by service applications. These can be used by service applications to maintain state information beyond a particular session. If these service variables are referenced in IRML rule conditions, then service applications can dynamically adapt the conditions that lead to specific rule actions without altering the actual rule.

Service-specific variables can also be used for the communication between different services, e.g. if one service applications sets a state variable that is subsequently read by another service application.

The "matches" attribute specifies the pattern against which the property value MUST be matched by the rule engine on the intermediary device. The "matches" pattern MUST be a regular expression compliant with the extended regular expression syntax as defined in [9].

A "property" rule condition is considered true if the pattern provided in the "matches" attribute matches the message, system, or service attribute value referred to by the "property" element through its "name" and "type" attribute values.

The "not-matches" attribute MAY be used instead of the "matches" attribute to express "property" rule conditions that are considered true if the provided pattern does NOT match the referenced attribute value.

The "case-sensitive" attribute specifies whether the matching of the pattern specified in the "matches" or "not-matches" attributes must be performed case sensitive or not. The default value for this attribute is "no" meaning that pattern matching is case insensitive unless otherwise specified. The matching of property names, regardless of their type, is always case insensitive.

The "sub-system" attribute can be used with a "context" attribute
value of "system" to specify rules for services that require the

evaluation of non-standard system properties which may not be supported by all intermediaries. For example, limited client bandwidth adaptation and streaming media adaptation services may require service execution rules that reference quality of service properties, such as the allocated bandwidth or the packet loss rate.

The default sub-system for IRML "property" elements is "standard" which means that only those system properties MAY be referenced that are defined in this document. If the "sub-sytem" attribute is used to specify the name of a sub-sytem other than "standard", then it is up to the referenced sub-sytem to define the supported system properties and how "property" rule conditions are to be matched.

Appendix C lists all currently known IRML sub-sytem specifications. Rule modules that use an IRML sub-system other than the "standard" sub-system MAY only be distributed to intermediaries that specifically offer support for the specified sub-system. An IRML rule processor that encounters a "property" rule condition with an unknown sub-system MUST consider the rule condition as false.

3.6.2 Unconditional Rules

If a "rule" element contains an element representing a rule action outside of any "property" elements, then the specified rule action MUST be performed for all data path protocol messages that originate from or are intended for the authorizing endpoint and pass through the specified processing point. Services with logging functionality, for example, may have to be triggered for all user requests that pass through the intermediary device.

3.7 IRML Rule Actions

Through IRML rule actions, endpoints can request the execution of services. This type of rule action is expressed through the following IRML elements.

3.7.1 The "execute" Element

The "execute" element is a rule action element that can be used to express the intent of the authorizing endpoint to execute a specific service application if the surrounding "property" rule conditions are true. The service that the rule author wants to be executed is further specified by the contained "service" element(s).

3.7.2 The "service" Element

Rule action elements apply to a service application which is further

specified by the "service" element. The "service" element contains
either a "uri" to identify a specific service application or an

"any" element if a rule action applies to any service application. It MAY also contain any number of "parameter" elements.

Attributes of "service"

Name	Values	Default
name	CDATA	
failure	(abort|ignore|try-alternate)	"abort"
type	(primary|alternate)	"primary"

The "name" attribute contains a descriptive name of the service application, e.g. "Norton Virus Scanning".

The "failure" attribute specifies how the intermediary MUST react to a service execution failure of the specified service application.

An attribute value of "abort" (which is also the default value) indicates that the intermediary MUST abort the current data transaction and inform the data consumer and possibly also the data provider of the service execution failure.

An attribute value of "ignore" indicates that the intermediary MUST ignore the service execution failure and continue the rule processing as though it never attempted to execute the specified service.

An attribute value of "try-alternate" indicates that the intermediary MUST attempt to execute an alternate service application instead of the failed service application. If multiple alternate services are given, then the intermediary MUST attempt to execute them in the order they are specified.

The "type" attribute specifies whether the "service" element specifies a primary or an alternate service application. Alternate service applications MUST only be executed if the execution of the primary service application fails. A rule action element MAY only contain one "service" element of the type "primary", but MAY contain multiple "service" elements of the type "alternate". If a "service" element has a "failure" attribute value of "try-alternate", then it MUST be followed directly by at least one "service" element of the type "alternate".

3.7.3 The "uri" Element

The "uri" element identifies the service application of the corresponding "service" element. The "uri" element does not, however, specify a specific instance of a service application, e.g. a specific installation on a specific server. Instead, the IRML-

enabled intermediary can map the identified service application to a
specific instance at run-time in order to accommodate for system or

network conditions, e.g. the current system load on a particular remote callout server.

The "uri" element MUST contain an absolute URI that follows the URI syntax as defined in [10] and uniquely identifies a service application including its version. Each "uri" element MAY contain one service identifier.

The service identifier, to serve its intended purpose, MUST have the characteristics of global uniqueness and persistence. It is not a goal that it conveys information about how to retrieve or locate a service. Because Uniform Resource Names [11] have been designed with these goals in mind, URNs SHOULD be used as service identifiers. However, other URIs can be managed in such a way as to achieve the same goals.

3.7.4 The "any" Element

The "any element is an empty element that can be used instead of the "uri" element to express that the surrounding rule action applies to any service application. For example, a rule author may wish to express that under certain rule conditions no service application should be executed. The "any" element MAY not be used in combination with the "execute" element.

3.7.5 The "parameter" Element

The "parameter" element is an optional element that can be used in combination with "execute" rule actions to specify one or more parameters that MUST be passed to the service application as it is being executed. It MAY contain either a "variable" or a "value" element which represent two different types of parameters. They are described in the following two sections.

Attributes of "parameter"

Name	Values	Default
name	CDATA	
type	(static|dynamic)	

The "name" attribute specifies the name of the parameter as it is passed to the service application.

The "type" attribute specifies whether the parameter value is static or dynamic. Static "parameter" elements MUST contain a "value" element and dynamic "parameter" elements MUST contain a "variable" element.

3.7.6 The "value" Element

The "value" element contains a static character value which MUST be passed to the service application along with the name of the surrounding "parameter" element.

3.7.7 The "variable" Element

The "variable" element specifies a message, system, or service property whose current value MUST be passed to the service application along with the name of the surrounding "parameter" element.

Attributes of "variable"

Name	Values	Default
name	CDATA	
context	(req-msg|res-msg|system|service)	
sub-system	CDATA	"standard"

The "name", "context", and "sub-system" attributes have the same semantics as the ones in the "property" element (section 3.6.1).

3.8 IRML Rule Set Example

```
<ruleset>
  <authorized-by class="data-consumer" type="individual">
    <name>A. Beck</name>
    <contact>abeck@lucent.com</contact>
    <id>abeck@lucent.com</contact>
  </authorized-by>
  <protocol>HTTP</protocol>
  <rule processing-point=1>
    <!-- Log ALL my requests -->
    <execute>
      <service name="Request Log">
        <uri>opes://log.com/requestlog-v1.0</uri>
        <parameter name="timestamp" type="dynamic">
          <variable name="system-time" context="system"/>
        </parameter>
      </service>
    </execute>
  </rule>
  <rule processing-point=4>
    <!-- Is the requested web resource a HTML document? -->
    <property name="Content-Type" context="res-msg"
                                               matches="text/html">
      <!-- Is the user's preferred language supported? -->
      <property name="Accept-Languages" context="req-msg"
                                        matches="^de|^fr|^it|^es">
```

```
        <!-- Invoke translation service Babelfish -->
        <execute>
          <service name="BabelFish Translation">
            <uri>opes://altavista.com/babelfish</uri>
          </service>
        </execute>
      </property>
    </property>
  </rule>
</ruleset>
```

More complete and complex rule module examples can be found in Appendix D.

4 IRML Rule Processing

For each data transaction the rule processor on the intermediary located in the path between data consumers and data providers MUST process only those IRML rule sets that are relevant to the current transaction, i.e. all IRML rule modules that apply to the data path protocol of the transaction (as specified by the "protocol" element) and contain rule sets which have been authorized by either endpoint of the transaction. Within each rule set only those rules MUST be evaluated that apply to the current rule processing point.

The rule processor MUST also take into account that message and service property values may be modified by the execution of service applications. It MAY therefore be necessary for the rule processor to re-evaluate rule conditions after the execution of a service application. However, no service application MAY be executed more than once as a result of the re-evaluation of rule conditions.

4.1 Conflict Resolution for Endpoints

When processing rules, the rule processor MUST evaluate all rule conditions of relevant rules in order to determine the intents of both endpoints. Potential conflicts between the intentions of the two endpoints MUST then be resolved according to the local policy of the intermediary.

4.2 Order of Service Execution

The order in which service applications on the intermediary device are executed may influence the final result of a data transaction. For example, a content analyzer/filtering service executed against the result of a web page translation service may produce a different result than a reverse execution order.

Within sets of rules authorized by the same endpoint, it is reasonable that the authorizing endpoint should be able to determine the service execution order. Within any "ruleset" element, services

MUST therefore be executed in the order they are specified in the "rule", "property", and "execute" elements.

It is generally not possible, however, to automatically determine a correct order if both endpoints request the execution of different services at the same processing point. In these cases, the intermediary MUST determine the service execution order.

In many cases, it may make sense to base this decision on the request/response message flow of a data transaction, i.e. for incoming requests at processing points 1 and 2, the order of service execution would be:

1. Data consumer authorized rule actions
2. Data provider authorized rule actions

For outgoing responses at points 3 and 4, the service execution order would be reverse:

1. Data provider authorized rule actions
2. Data consumer authorized rule actions

5 IAB Considerations

This section quotes the IAB's architectural and policy considerations for the OPES framework [12] and describes how these are addressed by this document or why they are not relevant to IRML:

(1) "One-party consent: An OPES framework standardized in the IETF must require that the use of any OPES service be explicitly authorized by one of the application-layer end-hosts (that is, either the content provider or the client)."

As described in section 3.4.2 and throughout the document, this document REQUIRES that each set of rules in IRML rule modules be authorized by one of the endpoints of a data transaction.

(2) "IP-layer communications: For an OPES framework standardized in the IETF, the OPES intermediary must be explicitly addressed at the IP layer by the end user"

This document does not specify or impact the addressing of OPES intermediaries by the end user.

(3) "Notification: The overall OPES framework needs to assist content providers in detecting and responding to client-centric actions by OPES intermediaries that are deemed inappropriate by the content provider."

Both data providers and data consumers can use IRML to express their service execution policies. The processing of IRML rules from both endpoints will therefore help reveal conflicts between the service execution policies of both endpoints. However, it is beyond the scope of IRML to define a method of facilitating the detection of inappropriate service application behavior. Conflict resolution and notification of data providers/consumers is not defined by IRML and is done according to local policy.

(4) "Notification: The overall OPES framework should assist end users in detecting the behavior of OPES intermediaries, potentially allowing them to identify imperfect or compromised intermediaries."

It is beyond the scope of IRML to define a method of assisting data consumers and providers in detecting imperfect or compromised OPES intermediaries. However, IRML allows both endpoints to specify the behavior of the intermediary in the case of service execution failures (see section 3.7.4). The default behavior is to abort the data transaction and notify the application endpoint.

(5) "Non-blocking: If there exists a "non-OPES" version of content available from the content provider, the OPES architecture must not prevent users from retrieving this "non-OPES" version from the content provider."

IRML does not prevent data consumers from receiving a "non-OPES" version of content if the data provider offers a "non-OPES" version of its content. For example, data providers could offer a "non-OPES" version under a different URI and make sure that their IRML rules do not trigger any OPES services for the "non-OPES" URI.

(6) "URI resolution: OPES documentation must be clear in describing these services as being applied to the result of URI resolution, not as URI resolution itself."

This document does not define or describe OPES services.

(7) "Reference validity: All proposed services must define their impact on inter- and intra-document reference validity."

This document does not define or describe any specific OPES services.

(8) "Any services that cannot be achieved while respecting the above two considerations may be reviewed as potential requirements for Internet application addressing architecture extensions, but must not be undertaken as ad hoc fixes."

This document does not define or describe any specific OPES services.

(9) "Privacy: The overall OPES framework must provide for mechanisms for end users to determine the privacy policies of OPES intermediaries."

This part of the OPES framework is not defined by this document.

6 Security Considerations

Since data providers and data consumers can author IRML rule modules, it will be necessary to securely and reliably transfer rule modules from data providers/consumers to intermediary devices (or to other OPES entities in the same domain and from these entities to intermediary devices).

Because IRML rule modules can control the execution of services, the receiving intermediary must be able to authenticate the rule module author and verify the integrity of the received rule module.

IRML therefore REQUIRES that all IRML rule modules be signed by the rule module author using XML signatures as defined in [13].

Also, to protect the privacy of data providers and data consumers, application-layer or network-layer encryption SHOULD be applied to connections that are used for the transfer of IRML rule modules.

As IRML-enabled intermediaries could potentially be the target of security attacks, intermediaries SHOULD be able to reject submitted IRML rule modules if accepting them would compromise their ability to function properly. Intermediaries SHOULD also be able to reject IRML rule modules that are incompatible with their own policies.

7 Acknowledgements

The authors would like to thank all active participants in the OPES mailing list for their thought-provoking discussion, and many of the ideas and suggestions that have been incorporated into this document.

8 References

1 Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996
2 Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", Request for Comments 2119, Harvard University, March 1997
3 Barbir, A., et al., "An Architecture For Open Pluggable Edge Services (OPES)", Work in Progress Internet Draft: draft-ietf-opes-architecture-04.txt, December 2002.

4 Barbir, A., et al., "OPES Use Cases and Deployment Scenarios", Work in Progress Internet Draft: draft-ietf-opes-scenarios-01.txt, August 2002

5 Bray, T., et al., Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/2000/REC-xml-20001006, October 2000

6 Barbir, A., et al., "Policy, Authorization and Enforcement Requirements", Work in Progress, Internet Draft draft-ietf-opes-authorization-02.txt, February 2002

7 Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", Request for Comments 2616, June 1999

8 Klyne, G., et al., "Date and Time on the Internet: Timestamps", Work in Progress, Internet Draft "draft-ietf-impp-datetime-05.txt", November 2001

9 ISO/IEC DIS 9945-2:1992, Information technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (IEEE Std 1003.2-1992); X/Open CAE Specification, Commands and Utilities, Issue 4, 1992

10 Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", Request for Comments 2396, August 1998

11 Moats, R., "URN Syntax", Request for Comments 2141, May 1997

12 Floyd, S. et al., "IAB Architectural and Policy Considerations for OPES", RFC 3238, January 2002

13 Bartel, M. et al., "XML-Signature Syntax and Processing", W3C Recommendation, February 2002

Authors' Addresses

Andre Beck
Markus Hofmann
Bell Labs Research
Lucent Technologies
101 Crawfords Corner Rd.
Holmdel, NJ 07733
Phone: (732) 332-5983
Email: {abeck, hofmann}@bell-labs.com

Appendix A - IRML DTD

The DTD in this section describes the XML syntax of IRML. Every rule module submitted to an IRML-enabled intermediary or admin server MUST comply with this DTD. Note that compliance with this DTD is not a sufficient condition for correctness of an IRML rule module, as some of the conditions described in this document are not expressible in the DTD syntax.

```
<!ELEMENT rulemodule    (author, ruleset+)>
<!ELEMENT author        (name, contact?, id)>
<!ELEMENT ruleset       (authorized-by, protocol, rule+)>
```

```
   <!ELEMENT authorized-by (name, contact?, id)>
   <!ELEMENT name          (#PCDATA)>
   <!ELEMENT contact       (#PCDATA)>
   <!ELEMENT id            (#PCDATA)>
   <!ELEMENT protocol      (#PCDATA)>
   <!ELEMENT rule          (property|execute)+>
   <!ELEMENT property      (property|execute)+>
   <!ELEMENT execute       (service+)>
   <!ELEMENT service       (any|uri, parameter*)>
   <!ELEMENT uri           (#PCDATA)>
   <!ELEMENT any           EMPTY>
   <!ELEMENT parameter     (value|variable)>
   <!ELEMENT value         (#PCDATA)>
   <!ELEMENT variable      (#PCDATA)>

   <!ATTLIST author        type        (delegate|self)          "self">
   <!ATTLIST authorized-by class       (data-provider|data-consumer)
                                                             #REQUIRED>
   <!ATTLIST authorized-by type        (individual|group) "individual">

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

   <!ATTLIST property      name        CDATA                 #REQUIRED>
   <!ATTLIST property      context     (req-msg|res-msg|system|service)
                                                             #REQUIRED>
   <!ATTLIST property      sub-system  CDATA                "standard">
   <!ATTLIST property      matches     CDATA                 #REQUIRED>
   <!ATTLIST property      not-matches CDATA                 #REQUIRED>
   <!ATTLIST property      case-sensitive    (yes|no)             "no">
   <!ATTLIST service       name        CDATA                  #IMPLIED>
   <!ATTLIST service       type        (primary|alternate)   "primary">
   <!ATTLIST service       failure     (abort|ignore|try-alternate)
                                                               "abort">
   <!ATTLIST parameter     name        CDATA                 #REQUIRED>
   <!ATTLIST parameter     type        (static|dynamic)      #REQUIRED>
```

Appendix B - IRML Extensions for HTTP

For HTTP messages, IRML defines the following system variables:

Property Name	Value
"request-line"	the first line of an HTTP request
"request-method"	the HTTP request method, e.g. GET or POST
"request-path"	the rel_path of the request URI as defined in [7]
"request-version"	the HTTP version number as it appears in the first HTTP request line, e.g. HTTP/1.1

```
"request-host"       the host name/IP address of the origin server
                     as it appears in the first HTTP request line
                     or HTTP "Host" request header
```

"request-uri"	the absolute request URI as defined in [7]
"response-line"	the first line of an HTTP response
"response-code"	the HTTP response code as it appears in the first HTTP response line, e.g. 200

All intermediaries that accept IRML rule modules with a protocol value of "HTTP" MUST support the above listed HTTP-specific system variables.

Appendix C - IRML Sub-Systems

The following is a list of currently known IRML sub-systems:

Sub-System Name	Description	Specification

Appendix D - Rule Module Examples

Data provider Rule Module Example

```
<?xml version="1.0"?>
<rulemodule xmlns="http://www.rfc-editor.org/rfc/rfcxxxx.txt">
  <author type="self">
    <name>Lucent Technologies</name>
    <contact>rule-info@lucent.com</contact>
    <id>www.lucent.com</id>
  </author>
  <ruleset>
    <authorized-by class="data-provider" type="individual">
      <name>Lucent Technologies</name>
      <contact>rule-info@lucent.com</contact>
      <id>www.lucent.com</id>
    </authorized-by>
    <protocol>HTTP</protocol>
    <rule processing-point="4">
      <!-- Is the requested web document our home page? -->
      <property name="request-path" context="system"
              matches="^/$|^/index.html$" case-sensitive="yes">
        <!-- Does the user send us a specific cookie? -->
        <property name="Cookie" matches="sew=23">
          <execute>
            <service name="Localized Content" failure="ignore">
              <uri>opes://local.net/insert-local-content</uri>
              <parameter name="clientip" type="dynamic">
                <variable name="client-ip" context="system"/>
              </parameter>
            </service>
          </execute>
        </property>
```

```
      </property>
    </rule>
  </ruleset>
</rulemodule>
<signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  ...
</signature>
```

Data consumer Rule Module Example

```
<?xml version="1.0"?>
<rulemodule xmlns="http://www.rfc-editor.org/rfc/rfcxxxx.txt">
  <author type="self">
    <name>Andre Beck</name>
    <contact>abeck@home.com</contact>
    <id>138.43.43.55</id>
  </author>
  <ruleset>
    <authorized-by class="data-consumer" type="individual">
      <name>Lucent Technologies</name>
      <contact>abeck@home.com</contact>
      <id>www.lucent.com</id>
    </authorized-by>
    <protocol>HTTP</protocol>
    <rule processing-point="1">
      <!-- Execute a privacy service that removes the HTTP "Referer"
           header from all my requests -->
      <execute>
        <service name="Privacy Service" failure="ignore">
          <uri>opes://privacy.net/priv-serv</uri>
          <parameter name="action" type="static">
            <value>remove-referer</value>
          </parameter>
        </service>
      <execute>
    </rule>
  </ruleset>
</rulemodule>
<signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  ...
</signature>
```

Delegate Rule Module Example

```
<?xml version="1.0"?>
<rulemodule xmlns="http://www.rfc-editor.org/rfc/rfcxxxx.txt">
  <author type="delegate">
    <name>Comcast ISP</name>
    <contact>rule-info@comcast.com</contact>
    <id>www.comcast.com</id>
  </author>
  <ruleset>
```

```
    <!-- This set of rules applies to a group of ISP customers -->
    <authorized-by class="data-consumer" type="group">
      <name>Virus Scanning Subscribers</name>
      <contact>vs-subscribers@comcast.com</contact>
      <id>www.comcast.com/irml-groups/vs-subscribers</id>
    </authorized-by>
    <protocol>HTTP</protocol>
    <rule processing-point="4">
      <!-- Can the requested web object contain a virus? -->
      <property name="Content-Type" context="res-hdr"
                                              matches="application/">
        <execute>
          <service name="McAfee Virus Scanning Service"
                             type="primary" failure="try-alternate">
            <uri>opes://mcaffee.com/mscan</uri>
          </service>
          <service name="Norton Virus Scanning Service"
                                                    type="alternate">
            <uri>opes://norton.com/nscan</uri>
          </service>
        </execute>
      </property>
    </rule>
  </ruleset>
  <ruleset>
    <!-- This set of rules applies to a specific data provider -->
    <authorized-by class="data-provider" type="individual">
      <name>CNN</name>
      <contact>rule-info@cnn.com</contact>
      <id>www.cnn.com</id>
    </authorized-by>
    <protocol>HTTP</protocol>
    <rule processing-point=4>
      <property name="request-uri" context="system"
                               matches="http://www.cnn.com/file.exe>
        <!-- CNN wants to allow only the McAffee virus scanning
             service on this file because there are known issues with
             Norton's Virus scanning service and particular files.-->
        <execute>
          <service name="McAffee Virus Scanning Service">
            <uri>opes://mcaffee.com/mscan</uri>
          </service>
        </execute>
      </property>
    </rule>
  </ruleset>
</rulemodule>
<signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  ...
```

```
</signature>
```

Appendix E - IRML Change Log

Changes from draft-beck-opes-irml-02.txt

- aligned terminology with OPES WG documents
- removed references to Web services
- removed "may-execute" and "do-not-execute" elements
- updated references section

Changes from draft-beck-opes-irml-01.txt

- replaced the <action> element with three different rule action elements: <execute> (to request the execution of a service), <do-not-execute> (to disallow the execution of services), and <may-execute> (to permit the execution of services)
- introduced "delegate" as rule module author and separation between rule module author and authorizing endpoint
- introduced IRML rule sets as a collection of rules authorized by the same endpoint
- introduced separation between service identification through URI and service execution parameters
- added recommendation to use URNs as service identifiers
- introduced support for dynamic parameters (i.e. the ability to pass run-time message, system, and service property values to service applications)
- introduced "failure" attribute to allow endpoints to control the intermediary behavior in the case of service execution failures
- introduced the concept of executing alternate services in the case of service execution failures
- rewrote introduction
- added "Rule Pocessing" section
- added "IAB Considerations" section to address the recent IAB draft with considerations for OPES
- changed mandated order of service execution
- rewrote security considerations and introduced the requirement to use XML signatures in all rule modules
- fixed syntax of comments in examples, clarified the meaning of the "request-path" system property and the regular expression syntax to be used in rule condition patterns (as suggested by M. Cinquini)
- added more HTTP-specific system properties to simplify the rule matching for HTTP messages (as suggested by Lily Yang)
- introduced "not-matches" attribute in property element (as suggested by M. Cinquini)
- renamed "type" attribute in "property" elements to "context"
- added "context" attribute values "req-msg" and "res-msg" to better differentiate between request message and response message properties (as suggested by R. Rahman)
- introduced IRML sub-system attribute in "property" elements and

  appendix with list of known sub-systems (as suggested by C.W. Ng)
- moved pre-defined HTTP-specific system properties to HTTP-specific
  appendix

- added IRML document identifiers for XML
- rewrote all rule module examples to match the new IRML DTD
- added this change log

Changes from draft-beck-opes-irml-00.txt

- updated references to include the models and policy requirements drafts
- removed terminology section and referenced the taxonomy in the models draft instead
- revised use of terminology to be conform with models draft terminology
- removed access provider from list of rule authors (as suggested by new OPES charter)
- late binding in <action> field through OPES URI (as suggested by Lee Rafalow and others)
- added "request-host" variable (as suggested by Francis Zane)
- added type attribute to property element to accomodate three different types of properties: message, system, and service
- added "client-ip" system variable to support services that depend on the client IP, e.g. URL rewriting in a CDN scenario
- added support for arbitrary service state variables which also allows rule matching against members of dynamic groups (as suggested by Lee Rafalow and others)
- new date format in <system-date> system property (as suggested by Ian Cooper)
- removed DEFAULT keyword in IRML DTD (as suggested by Ting)

Full Copyright Statement