Network Working Group
Internet Draft
<draft-RUPObjs-01.txt>
20 April 1999
Expires 20 October 1999

R. Di Cosmo
ENS France
P.E. Martínez López
UNLP Argentina

The Remote Update Protocol (RUP). Part II: RUP V.1.0 Commands and Objects.

April 22, 1999

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Copyright Notice

About certain keywords

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.

1 Abstract

The Remote Update Protocol (RUP) is a public protocol, described in [DM99], whose intention is to help in the reduction of the amount of work Web Robots should do in order to perform the task of Web indexing. Web Servers providing RUP can send information about the updates produced in their Web spaces to certain Web Indexing Robots, and using this information, the Robots can drastically reduce the amount of work needed to have a good index of the Web.
The [DM99] document describe the main features of RUP, but left unspecified many details concerning commands used for the information interchanging between servers and robots, the exact format of the data to be used, the alternatives provided, etc. All these details should be provided in order to obtain usable implementations of the RUP protocol.
In this document we describe the format of command and data objects to be used in the RUP Version 1.0.

2 Introduction

Web indexing is the task of constructing an index of the information living in the WWW in a given moment. A Web Robot is an agent that traverses the Web's hypertext structure, performing mainly indexing tasks. Due to the widely variable lifetime of URLs and the fearful amount of information involved, this task is a daunting one. Without additional aid, the snapshot taken by a given Robot in a given moment is doomed to be inaccuratly, in the best case, because information can change immediately after being indexed, and the Robot will take some time before discovering that fact.
The Robot Update Protocol, as described in [DM99], can be used by Web Robots and Web Servers to collaborate in the task of Web indexing. A Web Server mantains a log of relevant changes in its own

Web Space, and using the RUP protocol, notifies Robots about them, helping in this way with the indexing task. Advantages are many, including lower server and robot loads – because unchanged information should not be reindexed –, and increased index liveness. The RUP protocol is composed by:

- a mechanism for a robot to find the alternatives offered by a given server,
- a mechanism for a robot to register with servers, informing them about its communication preferences, and
- a mechanism for further interaction between the robot and the server.

This document should be read within the context of [DM99]. The [DM99] document doesn't describe the exact way in which these mechanisms are realized, leaving it open for future documents to refine. In this paper we describe a set of commands and data objects implementing partly the mechanisms, and providing in this way a first usable version of the RUP protocol – called RUP V.1.0. Both commands and data objects are MIME objects, as specified in [DM99].

3 RUP Version 1.0

The original specification of RUP establishes that commands and objects in a RUP implementation must be MIME objects, but it doesn't establish the exact nature of those objects. It also mention that RUP can be viewed as a degenerate case of CIP [AM98a, AM98b], leaving it open the decision of defining RUP as an instantiation of CIP.
In this first version of RUP we decide to emulate the structure of the CIP type trees for commands and data objects, but keeping them independent of any existing hierarchy. The reason for this decision is to keep RUP as close as possible to CIP, yet mantaining independence from it. The decision can be changed in future versions of RUP, making this protocol an instance of CIP.
There are two main actors in RUP: the Web Server and the Web Robot. The former mantains a local Web Space that is indexed by the latter. The Robot sends *requests* to the server in order to agree on the

nature of communication of index information between them. The Server responds either with a response (indicating for example success or failure of the operation), or a result (with some information in appropriate format). Further results can be sent by the Server according to its agreement with the Robot. So, as in the CIP protocol, messages in the RUP protocol fall into three categories: requests, responses, and results.
Since these MIME objects can be sent over several different protocols, body termination is specified by the transfer protocol. Regarding the line-breaks contained within each body, they are defined by the encoding specified by the MIME object itself – we use `<linebreak>` to denote the line separator. Linebreaks in headers and separating the body from the headers follow existing standards.

3.1 The RUP type tree

In order to represent the three kinds of messages, we define a media type tree, called `application/rup`, including all messages as "facets" [FKP96]; this tree will be divided into the following branches:

```
application/rup.cmd._command_
application/rup.response
application/rup.obj._type_
```

where `_command_` is one of the commands specified in Sect.4, and `_type_` identifies the type of data object contained in the body – see Sect.6.
We present in App.A the registration templates for the `application/rup` tree that should be sent to IANA upon approval of this document by IESG.

3.2 Information for Robots

In order for a Robot to find the alternatives offered by a given server, the [DM99] document establishes that every Web Server supporting RUP should have in its root a file called `/rupinfo.txt`. This file should contain, at least, information about the version, the valid registration procedure, and the implemented features of the protocol. The contents and syntax of the `/rupinfo.txt` file may vary from version to version – the only requirement is that the first line should

specify the version number. In this version, the first line of the file must contain the string "`RUP Version 1.0`".
In this version of RUP, lines indicating the various alternatives have the following format:

alternative	::=	attribute:<space>value<linebreak>
attribute	::=	1*[a-zA-Z-]
value	::=	{any character until the linebreak}

Multiple attributes with the same name are allowed.
the following lines should appear in the `/rupinfo.txt` file:

Supported transport protocols lines The attribute name in each of these lines should be "`TransportProtocol`". The value is the name of a known transport protocol, together with the contact address. For example

```
TransportProtocol:  SMTP (rup@enterprise.org)
```

or

```
TransportProtocol:  CGI (rup-cgi)
```

CGI es un protocolo de transporte?

Allowed latency times line The attribute of this line is "`AllowedLatencies`". The value is a comma separated list of time designators with obvious meanings – such that `hour`, `day`, `week`, `month`, etc. The latency a Robot can prefer is a multiple of one of these, expressed with the syntax integer – timedesignator.

Minimum latency time line The attribute of this line is "`MinimumLatency`". The value is a string indicating the minimum latency time as a multiple of a time designator; for example

```
MinimumLatency:  1-day
```

Query kinds lines ***Hablar sobre las posibles restricciones que el Robot puede solicitar. No tengo idea del formato que pueden tomar estas lineas. Alguna idea?***

Supported signature protocols lines ***Decir algo de protocolos de firma?***

Supported certificates lines ***Decir algo de certificados aceptados?***

FALTA ALGO?

4 RUP Commands

The commands of the version 1.0 are the following ones.

- `Register`: this command is used by a Robot to agree with the Server in the exact nature of further interaction between them.
- `Modify`: it is used to change the preferences specified in the registration.
- `Unregister`: it allows a Robot to cancel its registration with the Server.
- `GetInfo`: a Robot can ask the Server about the preferences registered by itself with this command.
- `GetIndex`: this command is used by any Robot (registered or not) to ask the server for certain portion of the log information; it allows non-registered flow of data.
- `Version`: the Server returns the Version number of its RUP implementation.

The first four involve the registering and communication of preferences mechanism, the fifth one is involved with the interaction between the Server and the Robot, and the last one is just for reference.
The definition for this MIME type tree is:

MIME type name: `application`
MIME subtype name: `index.cmd._command_`
Optional parameters: (none)
Security considerations: (see Sect.7)

The format of the body is defined by each command.

4.1 `Register` command

Command name: `application/rup.cmd.register`
Required parameters: (none)

A Robot can choose to register with the Server, in order to establish the nature of further interactions between the Server and himself. There are many possibilities for this interaction, represented in the body of the command as a list of `euppref` SOIF objects. – see Sect.6.1.
The `register` command object MUST be signed by the Robot, as required by [DM99]. Implementations can impose additional security conditions.

4.2 `Modify` command

Command name: `application/rup.cmd.modify`
Required parameters: (none)

Once registered, a Robot may want to modify its preferences. This command is similar to a register command, but it assumes that the Robot is already registered with the Server. The effect of this command is to overwrite completely the previous preferences of the Robot.
The body of this command is composed by a list of preferences, represented as `euppref` SOIF objects – see Sect.6.1.
The `modify` command object MUST be signed by the Robot, as required by [DM99]. Implementations can impose additional security conditions.

4.3 `Unregister` command

Command name: `application/rup.cmd.unregister`
Required parameters: (none)

A registered Robot may want to cancel its registration, and for that purpose it should use the unregister command.
The body of this command is empty.
The `unregister` command object MUST be signed by the Robot, as required by [DM99]. Implementations can impose additional security conditions.

4.4 `GetInfo` command

Command name: `application/rup.cmd.getinfo`
Required parameters: (none)

A registered Robot may want to know its current preferences with a given server. The `getinfo` command is used with that purpose.
The body of the command is empty. If the Robot that sent the command is registered within the Server, then the result is a `rup.obj.preferences` object, containing the preferences of that Robot in the form of a list of `euppref` SOIF objetcs; if it is not, an error response is answered.
The `getinfo` command object MUST be signed by the Robot, as required by [DM99]. Implementations can impose additional security conditions.

4.5 `GetIndex` command

Command name: `application/rup.cmd.getindex`
Required parameters: (none)

This command is used for asking the server about updating information. It can be used in an unregistered as well as registered ways.
The body of a `getindex` command is a list of `euppref` SOIF objects, stating the preferences and requested data for this question. If the Server accepts to answer the question, the result is a `rup.obj.info` object containing the requested information; if it does not, it is an appropriate error respone.
The `getindex` command object MAY be signed by the Robot, as required by [DM99]. Servers can decide whether or not to accept unsigned requests of this kind. Implementations can impose additional security conditions.

4.6 `Version` command

Command name: `application/rup.cmd.version`
Required parameters: (none)

This command is redundant with the first line of `/rupinfo.txt`.

The body of this command is empty. The result is the string "`RUP Version 1.0`" in this version; subsequent version must modify this response.
The `version` command object MAY be signed by the Robot, as required by [DM99]. Servers can decide whether or not to accept unsigned requests of this kind. Implementations can impose additional security conditions.

5 Responses

Every request should be answered with a response or a result, as specified in the request definition, except when no return path is available. For this purpose, we define a MIME media type for responses, in the same way that the CIP protocol does.

> MIME type name: `application`
> MIME subtype name: `rup.response`
> Required parameters: `code`
> Optional parameters: `charset`
> Security considerations: { see Section 7 }

The code parameter is a 3 digit return code that denotes the status of the last command. The meaning of various digits in the response code is discussed in RFC 821 [Pos82]. The charset parameter has its normal MIME meaning. The body has a first line with a human-readable comment about the nature of the error, and the format and meaning of the rest of it (if it exists) are meant to be defined by each response. The responses described in this document do not use the rest of the lines in the body, but implementors must not disallow the posibility for the future.

6 RUP Objects

RUP objects are instances of the `application/rup.obj` media type tree. We will define two kinds of objects in this document: preferences objects, and update information objects. The first kind will be used by the Server to reply a `getinfo` command, and the second one will be used to send information about the updates occurred in the local Web Space. The body of these objects will be lists of corresponding

SOIF objects `euppref` and `rupinfo` – both of them are defined in the corresponding subsection.

6.1 Preference objects

Object type: `application/rup.obj.preferences`
Required parameters: (none)

This media type is a wrapper for SOIF objects describing the communication preferences of Robots. The body of this object is just a list of `ruppref` SOIF objects, defined below. Each `ruppref` object establishes the preferences for a subspace of the Web Space mantained by the Server.
The `preferences` object MAY be encripted and signed by the Server for security concerns.

6.1.1 `ruppref` SOIF objects

In order to represent the Robot's preferences for communication with a Server, a new SOIF template type is defined. This new type is called `ruppref`. The URL is used to indicate the Web subspace to which the preferences should apply. The attributes in the attribute list are one of the following:

`Transport-Protocol:` this is a mandatory attribute indicating the transport protocol that the Robot wants to use in order to obtain the update information. It should also contain the necesary contact information; for example, if the protocol is SMTP, an email address should appear, and if it is HTTP, then a server name and a port number should be specified.

`Action:` this attribute indicates if the Robot prefers to receive the update information directly ("SendIndex"), or if it prefers to be notified in order to later recover the information using a `getindex` command ("RingMe"). The default for this attribute is "RingMe".

`Query-Restriction:` this attribute specifies a number of restrictions to be applied to the update information. Tyipical restrictions are excluding some type of files from the update information, or including only some type of files in it. Depending on the version and on the implemented features, the types of files can be determined by a pattern or can be more complex (for example, free files, small files, home page files, etc.). The default value for this attribute is to apply no restrictions.

`Latency:` it is the frequency of notifications of update information. The minimum time is determined by how often the Server performs the update of information, but any multiple of this time is allowed. The default for this attribute is the minimum time.

`Seq-Number:` this is an optional attribute indicating the initial number to be used in the sequential numbering of communications between Server and Robot. The purpose of this numbering is to identify if some message was missing. The default value is the number 1.

6.2 Update information objects

Object type: `application/rup.obj.info`
Required parameters: (none)

This media type is a wrapper for SOIF objects describing the update information generated by Servers. The body of this object is just a list of `rupinfo` SOIF objects, defined below. Each `rupinfo` object refer to a subspace of the Web Space mantained by the Server.
The `info` object MAY be encripted and signed by the Server for security concerns.

6.2.1 `ruppinfo` SOIF objects

In order to represent the update information that the Server should send to Robots, a new SOIF template type is defined. This new type is called `rupinfo`. The URL is used to indicate the Web subspace informed by this object. The attributes in the attribute list are one of the following:

`Info:` this attribute is a `ruppref` SOIF object.

No me acuerdo para que lo pusimos!

`URLS-Changed:` this attribute is a `multiURL` indicating the files that has changed since the last report.

`URLS-New:` this attribute is a `multiURL` indicating the files that has been created since the last report.

`URLS-Deleted:` this attribute is a `multiURL` indicating the files that has been deleted since the last report.

A `multiURL` field is defined by the following syntax:

multiURL	::=	∗(dateid<linebreak> 1∗(1∗whitespace URL <linebreak>))
dateid	::=	[seconds,date]
seconds	::=	{an integer representing the number of non-leap seconds since 00:00:00 UTC, January 1, 1970.}
date	::=	{a date in the format of 'date' command on Unix}
space	::=	<space>{decimal 32}/ <tab>{decimal 9}
URL	::=	{a Uniform Resource Locator as specified in RFC 1738 [BLMM94]}

where <`linebreak`> is a separation between lines corresponding to the encoding used by the MIME object containing this field.

7 Security Concerns

Security considerations should take into account the transmission of valuable, propietary, or highly sensitive data, the possibility for false information to be generated, and also the exposition of unwanted information.
The first two issues (transmission of valuable or false data), can be resolved using the MIME nature of update information objects. For example, valuable data may be encrypted without additional effort, and also signatures can be used to avoid false information to be introduced. Several commands specified in this documents impose signatures to be used. In the case of results generated by a server, they can be encrypted and/or signed in order to assure security and privacy in the information transmission.
The second issue is a bit more delicate. Usually Web Robots index the web spaces following links from a given 'root document' – in this

way they have a look of the web similar to a connected graph. But if RUP technology is added to Web Servers, then all the local web space can be indexed, and not only the connected components. The lacking of references to certain information can be used as a cheap way to provide security – if nobody knows the right address, nobody can access it.
The [DM99] document propose that servers should mantain a list of addresses that are supposed not to be indexed. In this way, the information behind them is never exposed to Web Robots. If the Web Server has a centralized publication policy – i.e. the information should be published only by the administrator – then it is easy to mantain this list. But when every user of a system can have his own local web space (tipically as a subdirectory of his home directory), this is not so easy. For this case, it is REQUIRED that every user should keep a `.ruprc` file in his home directory, indicating what addresses should be indexed and what of them should not. The absence of this file is enough reason for the indexing program to refuse indexing the local web space of the user.

8 Conclusions

Author's Address

Roberto Di Cosmo
DMI-LIENS (CNRS URA 1347)
Ecole Normale Supérieure
45, Rue d'Ulm
75230 Paris France.
E-mail:`Roberto.DiCosmo@ens.fr`

Pablo E. Martínez López
LIFIA, Universidad de La Plata,
CC.11, Correo Central,
La Plata, Argentina.
E-mail:`fidel@info.unlp.edu.ar`

A Media Type Registration Templates

The following templates will be sent to the IANA for registration as soon as this document is accepted by the IESG.

RUP tree

```
To: ietf-types@iana.org
Subject: Registration of MIME media type tree application/rup
```

```
MIME media type name: application
MIME subtype name: rup
Required parameters: none
Optional parameters: none
Encoding considerations: none
Security considerations:
   Section 7 of this document should be
   copied here.
Interoperability considerations:
Published specification:
   This draft should be referenced here once it has been
   approved by the IESG.
Applications which use this media type:
   This media type is used to contain information about
   updates in Web spaces, and is intended for
   communication between a Web Robot and a Web Server.
Additional information:
   This media type is not a standalone type. It is the
   top level of a tree similar to the vnd or prs trees
   specified in Section 2.1 of RFC2048. There are three
   specified branches for this tree:
      application/rup.cmd
      application/rup.response
      application/rup.obj
   Each of these branches is a tree in its own right with
   types registered below them. See those registrations
   for more information on the types allowed below those
   branches.
Person & email address to contact for further information:
Intended useage: LIMITED USE
Author/Change controller:
```

RUP Command tree

```
To: ietf-types@iana.org
Subject: Registration of MIME media type tree application/rup.cmd

MIME media type name: application
MIME subtype name: rup.cmd
```

```
Required parameters: none
Optional parameters: none
Encoding considerations: none
Security considerations:
   Section 7 of this document should be
   copied here.
Interoperability considerations:
   Implementors should handle unknown commands gracefully
Published specification:
   This draft should be referenced here once it has been
   approved by the IESG.
Applications which use this media type:
   This media type is the top of a tree of media types
   that express commands issued to a Web Server in order
   to communicate with it via the RUP protocol.
Additional information:
   This media type is not a standalone type. It is the
   top level of a tree similar to the vnd or prs trees
   specified in Section 2.1 of RFC2048. Types registered
   within this tree are limited to being commands as
   specified in the document(s) referenced in the
   "Published specifications" section.
Person & email address to contact for further information:
Intended useage: LIMITED USE
Author/Change controller:
```

RUP Response tree

```
To: ietf-types@iana.org
Subject: Registration of MIME media type application/rup.response

MIME media type name: application
MIME subtype name: rup.response
Required parameters: code
Optional parameters: none
Encoding considerations: none
Security considerations:
   Section 7 of this document should be
   copied here.
Interoperability considerations:
```

```
   Implementors should handle unknown responses gracefully
Published specification:
   This draft should be referenced here once it has been
   approved by the IESG.
Applications which use this media type:
   This media type is used to encode responses to RUP
   commands passed to Web Servers.
Additional information:
   This media type _is_ a standalone type. The code
   parameter contains the specific response code as
   specified by Appendix B of the specification document.
Person & email address to contact for further information:
Intended useage: LIMITED USE
Author/Change controller:
```

RUP Object tree

```
To: ietf-types@iana.org
Subject: Registration of MIME media type tree application/rup.obj

MIME media type name: application
MIME subtype name: rup.obj
Required parameters: none
Optional parameters: none
Encoding considerations: none
Security considerations:
   Section 7 of this document should be
   copied here.
Interoperability considerations:
   Implementors should handle unknown objects according
   to rules specified in the published specification.
Published specification:
   This draft should be referenced here once it has been
   approved by the IESG.
Applications which use this media type:
   This media type is the top of a tree media types
   that express information sent between Web Servers
   and Web Robots using RUP protocol.
Additional information:
   This media type is not a standalone type. It is the
```

```
   top level of a tree similar to the vnd or prs trees
   specified in Section 2.1 of RFC2048. Types registered
   within this tree are limited to being representation
   of information exchanged between Web Servers and Web
   Robots when using RUP protocol.
Person & email address to contact for further information:
Intended useage: LIMITED USE
Author/Change controller:
```

References

[AM98a] J. Allen and Michael Mealling. The architecture of the Common Indexing Protocol (CIP), November 1998. Internet-draft.
URL: `http://search.ietf.org/internet-drafts/`, filename: `draft-ietf-find-cip-arch-02.txt`.

[AM98b] J. Allen and Michael Mealling. MIME objects definitions for the Common Indexing Protocol (CIP), November 1998. Internet-draft.
URL: `http://search.ietf.org/internet-drafts/`, filename: `draft-ietf-find-cip-mime-03.txt`.

[BLMM94] T. Berners-Lee, L. MAsinter, and M. McCahill. Uniform Resource Locatord (URL). RFC 1738, December 1994.

[DM99] Roberto Di Cosmo and Pablo E. Martínez López. Remote Update Protocol. Part I: RUP Architecture, April 1999. Submitted as Internet-Draft.
URL: `http://search.ietf.org/internet-drafts/`, filename: `draft-RUP-01.txt`.

[FKP96] N. Freed, J. Klensin, and J. Postel. Multipurpose Internet Mail Extensions (MIME) Part Four: MIME Registration Procedures. RFC 2048, January 1996.

[Pos82] J. Postel. Simple Mail Transfer Protocol, RFC 821, August 1982.