Network Working Group
Internet-Draft
Expires: August 29, 2002

I. Cooper
Editor
D. Li
Cisco Systems, Inc.
M. Dahlin
University of Texas
M. Hamilton
JANET Web Cache Service

February 28, 2002

Requirements and Guidelines for A Resource Update Protocol
draft-ietf-webi-rup-reqs-03

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.

This Internet-Draft will expire on August 29, 2002.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document provides guidelines for the development of a Web Resource Update Protocol to facilitate cache coherence in Web intermediary systems such as caching proxies and surrogates. Such a protocol is useful to maintain consistency in an environment where periodic revalidation is unacceptable in terms of performance and/or cache consistency. This memo suggests invalidation methods as the

only required functionality of a candidate protocol, but outlines other functionality that should be supported (possibly at a later date).

Table of Contents

1. Introduction

This memo outlines guidelines for the speficification of a resource update protocol for the Web to enable cooperating content servers and web intermediaries (caching proxies and surrogates, for example) to offer a higher degree of cache coherence than is typically possible using per-request validation based on HTTP [6] conditional request directives (e.g. If-Modified-Since).

While HTTP cache control mechanisms provide a means for individual entities to be flagged with their own cacheability information, this is not always sufficient for content publishers requirements. Setting a long expiration time results in potentially correct but "old" material being served; setting a short expiration time results in newer content but requires more revalidation requests. Empirical evidence suggests the need for a dynamic content-server driven update mechanism.

Cache coherence and invalidation is discussed in detail in the caching literature, e.g. see [8] and [9] for background information. At the time of writing several examples of cache conherence/ invalidation protocols are known to the editors: WCIP [1], PSI [2] and DOCP [3].

[Ed Note: Should we give examples as above if we can't provide an exhaustive list?]

A carefully developed mechanism for the communication of information about changes to Internet resources offers the potential for other functions above and beyond invalidation of cached objects. Resource updates may also be an appropriate way of informing systems which generate content dynamically that the underlying data which they manipulate (e.g. to produce HTML pages) has changed. Such uses are currently outside the scope of consideration in this requirements document.

The goal of this document is to provide guidelines for developers of a Resource Update Protocol (RUP), the purpose of which is to improve cache coherence over HTTP's conditional request polling technique. Base level requirements for candidate protocols are given; it is expected that protocols will initially provide support for only invalidation but that they will be suitably designed to provide richer functionality in the future.

2. Terminology

This document uses terms defined in the Internet Web Replication Taxonomy [5], and the HTTP/1.1 specification [6]. The reader is

expected to be familiar with these documents. The term SURROGATE is used to refer to a demand driven surrogate origin web server unless explicitly stated otherwise. ORIGIN SERVER refers to the master origin server.

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 [4].

2.1 Definitions

RUP SERVER
The entity that knows the state of a content provider's resources and generates resource updates. Notionally co-located with the origin server.

RUP CLIENT
The entity that needs to know the state of resources and receives updates from the RUP SERVER. Notionally co-located with a SURROGATE.

INVALIDATION
The signal from a RUP SERVER to a RUP CLIENT to indicate that the master copy of an entity has changed.

CONSISTENCY GUARANTEE
To be defined...

[Ed note: Term used later in the text but not defined]

RESOURCE GROUP
To be defined...

[Ed note: Term used later in the text but not defined]

3. Design Guidelines

[Ed note: These are Ian's interpretations of the design guidelines in the -01 version and as such are up for discussion.]

Protocol designers are expected to use their best judgement in their design and implementations, however, the following list provides high-level guidance:

1. Extensibility: the protocol SHOULD be extensible to enable richer functionalities to be provided in subsequent versions.

2. The protocol SHOULD leverage existing technologies where possible (e.g. XML, HTTP, URIs).

3. Interoperability: it is anticipated that multiple protocols will exist in this area; protocols SHOULD interoperate or co-exist with each other.

4. Scaling: INVALIDATIONS are expected to be sent to tens of thousands of SURROGATES. Protocols SHOULD provide reasonable features to enable scaling to such numbers, e.g. by the use of relays or multicasting.

5. Compliance: protocols MUST define levels of compliance and document any side effects that might arise should lesser levels of compliance be used.

6. Expense: protocols SHOULD make expensive operations OPTIONAL.

4. Scoping

It is anticipated that RUP will be deployed in origin servers (and their delegates) and Web intermediaries, including surrogates and caching proxies. Other deployments are possible (e.g. within search engine crawlers), though some carry scaling and denial of service implications if implemented in HTTP user agents (see [6]). Specific warnings relating to implementations in user agents MUST be included in RUP protocol candidate specifications; this is most likely text to be included in the "Security Considerations" section of those documents. Also see Security Considerations (Section Section 8).

If conflicts arise between the needs of SURROGATEs and caching proxies, the needs of the SURROGATEs SHOULD take precendence. However, designers SHOULD make those parts of the protocol OPTIONAL.

5. Use Cases

The following use cases provide examples of how RUP protocols are likely to be deployed. While it is noted that these scenarios are only slightly different in places it is felt beneficial to identify them for the benefit of potential protocol developers.

Particular attention is brought to the fact that RUP may be used to to interface between systems running alternative protocols beyond the boundaries defined here. For example, individual content delivery networks (CDNs) may choose to operate proprietary protocols internally while speaking a RUP protocol externally.

5.1 Intra-CDN

A Content Distribution/Delivery Network (CDN) may choose to use RUP to pass INVALIDATIONS within its own delivery network, e.g. from a control node to intermediaries under its control.

CDNs must typically have control protocols to ensure they serve the correct content as requested by their customers (the content providers). Such protocols are currently proprietary in nature. Standardization within IETF will enable CDN operators to share their knowledge of their needs in a beneficial way, and will also gain from input from a wider technical community.

5.2 Inter-CDN

Content Networks may internetwork, sharing the responsibility of serving content requests to end users (see [18][19]). As such, INVALIDATIONS will also have to be shared among the cooperating networks.

As an open standard, RUP would enable multiple Content Networks to share INVALIDATIONS, while allowing each Network to operate an invalidation protocol of its own choosing internally.

5.3 Content provider to CDN

When using the services of a CDN, a content provider must use protocols or tools provided by that CDN in order to pass invalidation or update messages in order for the CDN to propagate that information to its own surrogates.

A standardize protocol interface between content provider and CDN would ensure that content providers are not "locked" into the use of a single CDN through use of propriety protocols and the pain of migration. Further, content management system developers would benefit by being able to publish into a variety of CDNs, not just those from which they license the interface technology.

5.4 Content provider to arbitrary Web intermediary

Caching proxies provide "best effort" content delivery capabilities on behalf of the network operators running them (e.g. they are not under direct or indirect control of the content provider, unlike SURROGATEs). In an effort to ensure that end users see the content they wish, at all times, content providers will frequently make material appear uncacheable.

Providing a protocol that enables content providers a tighter binding

between their publication systems and caching proxies would enable those caching proxies to operate as SURROGATEs for that content. Such content could be treated as cacheable irrespective of indications within HTTP headers and would therefore offload the requests from the origin server. In the absense of the control protocol, other caching proxies would treat the content as uncacheable as before.

This use case is similar to Section 5.3 above.

6. Operations

The following sections outline the various operations that make up a Resource Update Protocol.

The RUP protocol SHOULD transfer multiple INVALIDATIONS in one request/response transaction when possible.

6.1 RUP SERVER-driven invalidation

A RUP SERVER sends entity or resource group INVALIDATIONS to RUP CLIENTS.

The RUP SERVER (and its administrator) controls the activity according to the RUP SERVER's load, scheduling, and configured preference. The connection between RUP CLIENT and RUP SERVER MAY be established by either party; connections SHOULD be persistent. It SHOULD be possible to monitor the update guarantee.

RUP SERVER-driven invalidations MUST be supported by candidate protocols.

6.2 RUP CLIENT-driven invalidation

A RUP CLIENT queries (polls) the RUP SERVER for freshness status of an entity or group of objects referenced by URI.

The RUP CLIENT controls RUP activity accorting to its own load, failure recover needs, and configured preferences; connections are established by the RUP CLIENT and SHOULD be persistent. The RUP SERVER responds to requests by indicating which entities or resource groups have been updated and must be invalidated.

RUP CLIENT-driven invalidations MUST be supported by candidate protocols.

6.3 Content location update

A RUP SERVER designates a new location as the source for an entity. Location may be, for example, from a new URI, parent caching proxy, CDN peer, multicast object distribution channel.

By updating the location from which a given entity may be retrieved temporarily the RUP SERVER is able to indicate a deligate of the origin server that is better suited to respond to future queries. The need for Content Location Update was identified in section A.1.1 of Known HTTP/Caching Problems [20].

[Ed note: Should we draw an analogy with HTTP 30x responses?]

Support of content location update is OPTIONAL.

6.4 Content prefetch hint

A RUP SERVER identifies entities or RESOURCE GROUPS that are suitable for prefetching (pre-positioning). RUP CLIENTS MAY prefetch the content and pin it in their corresponding caches.

Prefetch hints enable content to be pre-positioned without the complexities involved in "push" operations (also see Section Section 6.5).

Support of content prefetch hint is OPTIONAL.

6.5 Content updates

A RUP SERVER sends either the full content of entities (overwrite) or delta updates to RUP CLIENTS in order to update or push content into the corresponding content caches.

At the time of writing there is insufficient understanding of the kinds of content updates that might need to be supported. In particular, those working in this area are aware that mixing signalling with data leads to problems of scaling, entity consistency, and security issues, among others. Existing mechanisms addressing content retrieval, e.g. HTTP and Delta Encoding [7] demonstrate the high complexity of such functionality.

Content prefetch hints (see Section Section 6.4) enable participating RUP CLIENTS to invalidate and fetch content with somewhat similar results to content updates.

Content updates SHOULD NOT be supported.

6.6 Metadata updates

[Ed note: place holder; we lost a comment about RUP not being used to "update content or HTTP meta information of cached entities." This is still up for discussion, so this section is just to make sure it doesn't get lost.]

7. Functional Requirements

7.1 Coherence Model

7.1.1 Confirmation of actions

A RUP SERVER MUST be able to specify whether actions it requests of a RUP CLIENT are acknowledged or not. In a unicast environment a RUP SERVER SHOULD request acknowledgements.

Acknowledgements from RUP CLIENTS SHOULD be sent as a result of carrying out an INVALIDATION.

Where relays are used in a single control domain the relay MUST preserve the semantics of acknowledgement. E.g. a relay MUST wait for every child's acknowledgement before acknowledging to the RUP SERVER. Semantics MUST be preserved irrespective of the number of levels of relays.

7.1.2 Loose consistency

[Ed note: should this section be here?]

To support applications that do not require tight coupling (e.g. batch mirroring), RUP SHOULD make loose consistency available. In particular RUP SHOULD provide delta consistency guarantees such that a RUP SERVER MAY specify a maximum acceptable staleness, defined in seconds.

If a cached entity is updated then within the period covered by the delta consistency guarantee the RUP CLIENT will either (a) be notified of the update, or (b) detect that its cache is no longer synchronized with the server. [Ed note: what does that latter part actually mean?]

7.1.3 HTTP Warnings

A cache MUST NOT return RUP-controlled entities that would be stale in HTTP without attaching an appropriate HTTP/1.1 Warning [6].

[Ed note: Some description of the warning code chosen, especially if

an existing code is used in this context]

7.1.4 Transitioning into/out of RUP control

To transition from HTTP cache control to RUP controlled coherence a RUP CLIENT MUST first consider the resource stale and revalidate via RUP. Protocol designers should carefully consider cases where a RUP CLIENT has become unsynchronized.

To transition from RUP controlled coherence to HTTP cache control a RUP CLIENT MUST first consider the resource stale and revalidate via HTTP.

7.2 Resource Grouping

It is common for multiple resources that are related to be updated or removed at the same time. By grouping such resources into RESOUCE GROUPS (by examining content management systems or server logs, for example) that are addressed in their own right it is possible to invalidate many resourced with a single INVALIDATION.

RUP protocols MUST provide a mechanism to enable RUP CLIENTs to update multiple entities as a result of receiving an INVALIDATION for a RESOURCE GROUP.

RUP SERVERs SHOULD send INVALIDATIONS for only those RESOURCE GROUPs to which a given RUP CLIENT has subscribed.

[Ed note: The above requirement serious limits the way in which a RUP SERVER operator may aid a RUP CLIENT, and the SHOULD clause seems inappropriate; a MAY clause seems meaningless. Is it even worth mentioning?]

7.2.1 Resource group identification

RESOURCE GROUPs are identified to the RUP CLIENT in one of two ways:

Out-of-band: individual entities within the group contain a URI reference of the group within an HTTP header (to be defined by the RUP protocol).

In-band: group identification and contents are transmitted from RUP SERVER to RUP CLIENT within the RUP protocol itself.

Since there is no assumption of a 1:1 relationship between RUP SERVER and origin server, RESOURCE GROUPs may identify entites on separate origin servers (see Security Considerations). Identification of a RESOURCE GROUP MUST indicate the method used to establish a

connection between RUP SERVER and RUP CLIENT (e.g. protocol:// example.com/channel7).

7.3 Relays

In order for RUP deployments to scale and to cross administrative boundaries it MUST be possible for the protocol to be relayed (single source, single destination) and/or broadcasted (single source, multiple destinations) by RUP proxies.

It SHOULD be possible for INVALIDATIONS to be cached on RUP proxies (and potentially caching proxies of other types). Such cached INVALIDATIONs MUST have the same effect as if the message had reached the destination directly from the source.

7.4 Extensibility

It is anticipated that additional operations will be required, and that options for specific implementations and uses may be desired. RUP protocols SHOULD contain a means of extending the base protocol for the future.

INVALIDATIONS MAY carry a set of options. Options MUST be atomic. RUP CLIENTS MAY ignore options but MUST indicate in acknowledgments which options were applied.

[Ed note: I believe that is impossible when using a relay.

7.5 Other issues

Intermediaries are connected to the Internet in many ways, an in order to scale invalidation services many network/service providers are likely to want to make use of tecnologies such as satellite transmission and IP multicast. Protocol designers should carefully consider the implications of running their protocol over such media (e.g. "broadcast" nature of satellites).

It is likely that either RUP SERVER or RUP CLIENT or both will be situated behind firewalls or Network Address Translators (NATs). Protocol desginers should consider implications of firewall traversal and the effects that NAT may have on their protocol.

8. Security Considerations

Intermediaries present security problems which do not exist in the classic end-to-end model of the Internet, by introducing a 'Man In The Middle' by design. As such, it is essential that protocol level work on intermediaries takes care to devise means by which the

integrity of the resources being updated can be preserved - or at least tested.

1. Authentication and authorization: the protocol SHOULD be able to authenticate the parties of a session before commencing the session, so that no un-authorized parties may participate the session.

2. Session integrity: the protocol SHOULD be able to ensure the integrity of the resource update messages, so that any tempering can be detected and/or recovered from.

3. Session secrecy: the protocol SHOULD be able to ensure the secrecy of the resource update messages, so that no un-authorized parties can eavesdrop the session.

4. Denial of service: the protocol SHOULD provide counter measures to denial-of-service attacks, such as distributed SYN flood or resource update storm.
[Ed note: Surely SYN flood operates at a level too far below anything that RUP would touch directly.]

5. In-band security: the protocol SHOULD be able to prevent trusted parties from flooding and/or disabling the RUP service, accidentally or intentionally.

The above major risks associated with the protocol MUST be quantified and specifically addressed by the protocol design.

9. Acknowledgements

Thanks to Mark Nottingham, Oskar Batuner, Mark Day, Phil Rzewski, Paul H. Gleichauf, Fred Douglis, Lisa Dusseault, Ted Hardie, Joe Touch, Brad Cain, Hilarie Orman, Lisa Amini, Joseph Hui, Alex Rousskov, Stephane Perret, Darren New, Christian Maciocco, Michael Condry, Renu Tewari, Tao Wu, and the rest of the WEBI working group for their contributions.

The JANET Web Cache Service is funded by the Joint Information Systems Committee of the UK Higher and Further Education Funding Councils (JISC).

References

[1] Li, D., Cao, P. and M. Dahlin, "WCIP: Web Cache Invalidation Protocol", draft-danli-wrec-wcip-01.txt (work in progress), November 2000.

[2] Krishnamurthy, B. and C. Wills, "Piggyback server invalidation for proxy cache coherency", In Computer Networks and ISDN Systems, Volume 30 1998.

[3] Dilley, J., Arlitt, M., Perret, S. and T. Jin, "The Distributed Object Consistency Protocol", Technical Report HPL-1999-109, September 1999.

[4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[5] Cooper, I., Melve, I. and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, January 2001.

[6] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[7] Mogul, J., Krishnamurthy, B., Douglis, F., Feldmann, A., Goland, Y., van Hoff, A. and D. Hellerstein, "Delta encoding in HTTP", RFC 3229, January 2002.

[8] Belloum, A. and L. Hertzberger, "Maintaining Web cache coherency", In Information Research, Volume 6 No. 1, October 2000.

[9] Gwertzman, J. and M. Seltzer, "World-Wide Web Cache Consistency", In Proceedings 1996 USENIX Technical Conference, January 1996.

[10] Liu, C. and P. Cao, "Maintaining Strong Cache Consistency in the World-Wide Web", In Proceedings ICDCS97, May 1997.

[11] Yin, J., Alvisi, L., Dahlin, M. and C. Lin, "Using Leases to Support Server-Driven Consistency in Large-Scale Systems", In Proceedings ICDCS98, May 1998.

[12] Duuvvuri, V., Shenoy, P. and R. Tewari, "Adaptive Leases: A Strong Cache Consistency Mechanism for the World Wide Web", In Proceedings INFOCOM, 1999.

[13] Li, D. and D. Cheriton, "Scalable Web Caching of Frequently Updated Objects using Reliable Multicast", In Proceedings USITS, October 1999.

[14] Yin, J., Alvisi, L., Dahlin, M. and C. Lin, "Hierarchical Cache Consistency in a WAN", In Proceedings USITS, October 1999.

[15] Yu, H., Breslau, L. and S. Shenker, "A Scalable Web Cache Consistency Architecture", In Proceedings SIGCOMM, 1999.

[16] Yin, J., Alvisi, L., Dahlin, M. and A. Iyengar, "Engineering server-driven consistency for large scale dynamic web services", In Proceedings WWW10, 2001.

[17] Cohen, J. and S. Aggarwal, "General Event Notification Architecture Base", internet-draft http://www.alternic.org/drafts/drafts-c-d/draft-cohen-gena-p-base-01.pdf, July 1998.

[18] Day, M., "A Model for Content Internetworking (CDI)", draft-ietf-cdi-model-00 (work in progress), February 2002.

[19] Green, M., "Content Internetworking Architectural Overview", draft-ietf-cdi-architecture-00 (work in progress), February 2002.

[20] Cooper, I. and J. Dilley, "Known HTTP Proxy/Caching Problems", RFC 3143, June 2001.

Authors' Addresses

Ian Cooper
Editor
No address available

Phone: +44 7966 285145
EMail: ian@the-coopers.org

Dan Li
Cisco Systems, Inc.
170 W. Tasman Drive.
San Jose, CA 94043
USA

Phone: +1 650 823 2362
EMail: lidan@cisco.com

Mike Dahlin
University of Texas
Taylor Hall 2.124
Department of Computer Sciences
University of Texas
Austin, TX 78712-1188
USA

Phone: +1 512 327 7251
EMail: dahlin@cs.utexas.edu

Martin Hamilton
JANET Web Cache Service
Computing Services
Loughborough University
Loughborough, Leics LE11 3TU
UK

Phone: +44 1509 263171
EMail: martin@wwwcache.ja.net

Appendix A. Change Log

Please direct your comments to WEBI mailing list webi@lists.equinix.com. To subscribe, email webi-request@lists.equinix.com with body (un)subscribe. The mailing list archive is at http://www.wrec.org/webi-archive/.

Revision Log:

-02 to -03 [February 2002] / Edits by Ian Cooper

Note: Section numbers refer to those in draft-02 NOT this version

1. Changed title and publication date

2. Amended abstract:

   Rearranged text to flow more intuitively

   Better described how RUP candidates should consider invalidation a mandatory requirement but should consider other methods that are also described in the document

3. Re-worded introduction to try and identify motivation using better language.

4. Examples of cache coherence/invalidation protocols/techniques changed from a paragraph introducing them to references of some known work

5. Re-jigged the text in "design guidelines"

6. Scoping (section 4): placed requirement on protocol candidate specifications to carry warnings/security considerations on integration of the protocol in user agents (as defined in RFC2616)

7. Scoping (section 4): text indicating SURROGATES should be favored over caching proxies clarified

8. Cleaned up introduction to the Use Cases

9. Intra-CDN use case: tidied up wording to better explain why a CDN might want to use an IETF standardized protocol internally

10. Content provider to CDN: re-jigged to better explain why content providers and developers of content management systems benefit from a standardized protocol.

11. Attempted to more accurately describe Content provider to arbitrary web intermediary use case.

12. Moved 5.5, Operations, to a new major section (section 6)

13. Content location update could be seen as offering a means to address A.1.1 in RFC3143, text inserted to identify this fact.

14. RUP Client invalidation:

    was: "The RUP CLIENT controls RUP activity according to its load, failure recovery needs and configured preferences. The RUP SERVER replies to requests with latest changes since the last time the RUP CLIENT asked."

    now: "The RUP CLIENT controls RUP activity accorting to its own load, failure recover needs, and configured preferences; connections are established by the RUP CLIENT and SHOULD be persistent. The RUP SERVER responds to requests by indicating which entities or resource groups have been updated and must be invalidated."

15. RUP Client invalidation: removed seemingly redundant paragraph: "Whether and when the RUP CLIENT communicates with the RUP SERVER is determined by the consistency guarantee the RUP CLIENT

is committed to provide, and MUST [??] follow the semantic rules defined by the RUP protocol."

16. Content updates

was: "RUP proposals SHOULD NOT provide mechanisms for providing content updates. Any support of content updates is OPTIONAL.

Content updates refers to the scenario where a RUP SERVER sends either the full content of updates to entities, or deltas of those changes to RUP CLIENTS.

At the time of writing there is insufficient understanding of the kinds of content updates that RUP might need to support. In particular we are aware that mixing signalling with data leads to problems of scaling, object consistency, and security issues, among others.

Existing mechanisms addressing content retrieval, e.g., HTTP [6] and Delta Encoding [7] demonstrate the high complexity of such functionality. Content prefetch hints enable participating RUP CLIENTS to invalidate and fetch content with somewhat similar results to content updates."

now: "A RUP SERVER sends either the full content of entities (overwrite) or delta updates to RUP CLIENTS in order to update or push content into the corresponding content caches.

At the time of writing there is insufficient understanding of the kinds of content updates that might need to be supported. In particular, those working in this area are aware that mixing signalling with data leads to problems of scaling, entity consistency, and security issues, among others. Existing mechanisms addressing content retrieval, e.g. HTTP and Delta Encoding [7] demonstrate the high complexity of such functionality.

Content prefetch hints (see Section ...) enable participating RUP CLIENTS to invalidate and fetch content with somewhat similar results to content updates.

Content updates SHOULD NOT be supported."

17. HTTP Warnings: SHOULD changed to MUST

18. Resynchronization: the content included in this section seems more usefully included as a hint to designers in the prior section (transitioning into RUP control). "Express

resynchronization" was a special case of this situation.

19. Section 6.2 (Naming and Framing) onwards have serious edits as given below.

20. 6.2/1 This section describes the grouping of resources into a RESOURCE GROUP that has its own identifier for the purposes of scaling.

21. 6.2/2 This item not needed. Choice of grouping is up to the operator of the system, not even the protocol. Item removed.

22. 6.2/3 and 6.2/5 merged into a section on resource grouping.

23. 6.2/4 addresses guidence to protocol designers on aspects of scaling their protocols, but should be common sense. Item removed.

24. 6.3 (Notification groups) defines a requirement for the protocol to transfer multiple INVALIDATIONs in the same request. Move that point to "Operations" section. 6.3/2 is really just a way of encapsulating group URIs together under yet another URI (a further level of indirection). This should be possible by recursive URI resolution of the groups in any case. Item removed.

25. 6.4 (extensibility) shortened

26. 6.5/1 Redundant; implicit; removed

27. 6.5/2 Mentioned in "Scoping", not necessary to repeat; presumtive; removed

28. 6.5/3 Already stated in design guidelines; removed

29. 6.5/4 Underlying media considerations reworded

30. 6.5/5 Duplication of an earlier functional requirement for acknowledgements

31. 6.5/6 Establishes a MUST requirement on acknowledgements; removed

32. 6.5/7 Description of operation based on non receipt of acknowledgements should be left to protocol designers, not included in this document; removed

33. 6.6/1 Already stated and unnecessary; removed

34. 6.6/2 Reworded under "Other Issues"

35. 6.6/3 This is an important point - promoted to a section directly under functional requirements. Note: There may be a requirement for RUP SERVERs to be able to understand acknowledgements that may be relayed from sources that are unknown to it

36. 6.7/1 If it should be possible to layer RUP on any underlying transport (e.g. from TCP to BEEP to SOAP) then it seems pointless saying that; removed.

37. 6.7/2 Duplication of a major security consideration; removed.

38. 6.7/3 RUP SERVER discovery is out of scope for this document; RESOURCE GROUP discovery is already covered; removed

20-Nov-2001/Edits by Ian Cooper

1. Moved revision log to appendix.

2. Changed draft to "compact" mode to save some space

3. Significant changes in second paragraph of introduction in an attempt to clarify motivation for this work

4. Cleared up terminology section
   Question: Does the term invalidation fit? Removed initial wording which said "RUP must clearly define the actions this signal implies or mandates and the semantics the actions accomplish, in relationship to the RUP coherence models. E.g., RUP may specify (among many things) that a RUP CLIENT must not serve an invalidated cached object without a conditional HTTP request.

5. Moved references to cache coherence literature to the introduction (sub-optimal, but at least it's not in the terminology section any more)

6. Design guideline 1, changed to just read it should be extensible

7. Ian's interpretation on the design guidelines.

8. Attempted some wording to try to stop RUP being put into user agents (or acting as a pointer to protocol developers to get them to try to design things that make doing this difficult). Need a security considerations section to go with this...

9. Moved performance requirement from "scoping" to "guidelines" (it can't be any more than that)

10. Attempted to clarify the differences of administrative domain use cases by splitting into sub-sections (more detail is probably needed).

11. Split operations into subsections. Attempted to clarify the language.

12. Content updates added as an operation, but wording added to state that it SHOULD NOT be provided.

13. Changed format of 6.1 (coherence model) to give each point a sub-sub- section

14. Coherence model, requirement 6.1/6 (resource update guarantees must propagate through scaling mechanisms) clarified and restated in 6.1.1 since this is already talking about relays. (Note: Eeek, state!!!)

15. Changed client/server to RUP CLIENT and RUP SERVER throughout

16. Changed object etc. to entity throughout (I hope)

Previous:/Edits by Dan Li

1. remove references to NAT.

2. describe the need to work with RUP session discovery mechanisms.

3. limit the doc to "requirements", and not "protocol design".

4. remove the terms "transaction" and "mirrors" since they convey strong meanings in other senses.

5. ensure that "confirmation of action" is a requirement.

6. clarify on the "guarantees" RUP needs to provides, and not claim to "support" or "require" strong consistency.

7. clarify the motivations for server-driven and client-driven modes.

8. clarify that general "meta data" is allowed, and accommodated as concrete payload types and arbitrary payload options, while invalidation is the primary payload at present.

9. clarify the terms "RUP client" and "RUP server", and remove ambiguous references to "client" or "server".

10. remove the distinction between "managed" and "unmanaged", remove any mention of them.

11. add use cases in terms of administrative roles, i.e., a list of entities that are allowed to participate within the RUP protocol realm.

12. include the reference to a microsoft's notification proposal.

13. add requirements on flexible content "selectors".

14. restructure the section on "naming and framing" to clarify three items

Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.