Internet-Draft
Expires: Mar 8, 2020
Intended status: Proposed Standard

M. Toomim
Invisible College
M. Milutinovic
UC Berkeley
B. Bellomy
Invisible College
Nov 18, 2019

Merge Types
draft-toomim-httpbis-merge-types-00

Abstract

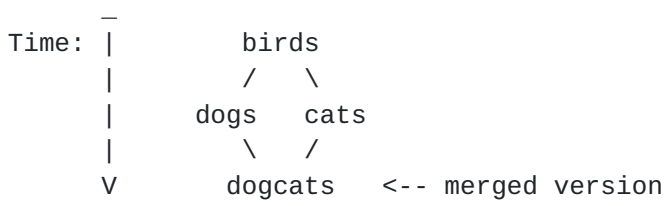
Merge Types specify how to merge multiple simultaneous edits to a resource. This happens when two computers edit the same state over a network with latency. A Merge Type specifies how to merge conflicting changes. If the computers implement and agree upon the same Merge Type, then they can guarantee to reach a consistent state, after arbitrary edits, eventually.

You can define new Merge Types. This document defines Merge Types, the structure of the Merge Type system, and IANA registration procedures for Merge Types.

Table of Contents

1. Introduction

In a version history, a merge type defines the output of a version that is derived from two or more parent versions:



This can be ambiguous if two or more edits overlap, and edit the same thing in different ways. It is desireable if all computers resolve ambiguities in the same way, because then they will achieve consistent results. In the example above, one person replaced "bird" with "dog", while the other replaced it with "cat", creating an ambiguity, which the Merge Type resolved by merging these edits together into "dogcats". Some other Merge Type might resolve this ambiguity differently.

Currently popular approaches are using conflict-free replicated data types [CRDT] or operational transformation [OT]. Different algorithms that merge in different ways. Different applications need different parts of their data to merge in different ways. The Merge Type specifies the outcome of a merge. Different algorithms can implement the same Merge Type to interoperate.

If a Merge Type is specified for a region of data, and all computers implement that Merge Type, then they can guarantee to obtain the same

resulting state, no matter the order that events arrive over the network, or the algorithms they use to implement the Merge Types.

This document defines how to specify a Merge Type within a standardized naming scheme.

1.1. Merge Types vary across applications

Different resources merge in different ways, and can use different merge types. For example, if a string represents a collaborative text document, then parallel edits should be merged together. But if a string represents a database entry ID and it is modified by two requests simultaneously, the appropriate behavior is to choose one or the other.

Similarly, if the resource is a number that represents a bank account balance, then it should merge multiple simultaneous debits by adding the amounts of the debits together.

Thus, it is nice to re-use a Merge Type across multiple types of data; and each application has different needs for how its data should merge.

1.2. A Merge Type is specified as a function

A Merge Type is a function that produces a merged state when provided with any set of parent versions:

```
                    Merge Type

          (parent, parent, ..) -> merged state
```

A Merge Type must be a deterministic function. Given the same set of parents that descend from the same version history, it should always return the same resulting merged state. This guarantees eventual consistency.

To compute the result, a Merge Type has access to the entire version history that preceded each of the parents, but it cannot depend on information outside of that version history.

A Merge Type might only work on a particular Content-Type. For instance, a "counter" Merge Type only works on numbers. A Merge Type SHOULD state any limitations in the set of states that it is capable of operating upon, and any assumptions it makes about the data.

But given that it has data that meets those assumptions, a Merge Type MUST return a result when merging any set of versions. A Merge Type cannot perform any validation on new versions.

2. Definitions

For the purpose of this document we define "synchronization" as the resolution of conflicts. There are multiple approaches to resolving conflicts and we define "Merge Type" an identifier that corresponds to an approach of resolving conflicts.

3. Merge Type Naming

Naming of Merge Type follows naming requirements of media types as described in Section 4.2. of [RFC6838] with the following exception: Merge Type names consist only of a type-name without a sub-type.

Similarly, Merge Type parameters follow Section 4.3. of [RFC6838].

Example:

```
automerge-counter
sharedb-json; variant=1
biggest-wins; sortby=versionid
```

4. IANA Considerations

4.1. Merge Type Registry

The "Merge Type Registry" defines the namespace for the merge type names and refers to their corresponding specifications. The registry will be created and maintained at <http://www.iana.org/assignments/merge-types>.

4.1.1. Procedure

Registration of a Merge Type MUST include the following fields:

- Type name
- Required parameters
- Optional parameters
- Description
- Security considerations
- Interoperability considerations
- Published specification
- Person & email address to contact for further information

Values to be added to this namespace require IETF Review (see [RFC5226], Section 4.1).

4.1.2. Registrations

4.1.3. Comments on Merge Type Registrations

Comments on registered Merge Types may be submitted by members of the community to the IANA at iana@iana.org. These comments will be reviewed by the Merge Types reviewer and then passed on to the "owner" of the Merge Type if possible. Submitters of comments may request that their comment be attached to the Merge Type registration itself; if the IANA, in consultation with the Merge Types reviewer, approves, the comment will be made accessible in conjunction with the type registration.

4.1.4. Change Procedures

Once a Merge Type has been published by the IANA, the owner may request a change to its definition. The same procedure that would be appropriate for the original registration request is used to process a change request.

Merge Type registrations may not be deleted; Merge Types that are no longer believed appropriate for use can be declared OBSOLETE; such Merge Types will be clearly marked in the list published by the IANA.

Significant changes to a Merge Type's definition should be requested only when there are serious omissions or errors in the published specification. When review is required, a change request may be denied if it renders entities that were valid under the previous definition invalid under the new definition.

The owner of a Merge Type may pass responsibility to another person or agency by informing the IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a Merge Type. The most common case of this will be to enable changes to be made to types where the author of the registration has died, moved out of contact, or is otherwise unable to make changes that are important to the community.

5. Security Considerations

An analysis of security issues SHOULD be done for all registered Merge Types. However, regardless of what security analysis has or has not been done, all descriptions of security issues MUST be as accurate as possible. In particular, the security considerations MUST NOT state that there are "no security issues associated with this Merge Type". Security considerations for Merge Types MAY state that "the security issues associated with this Merge Type have not been assessed".

There is absolutely no requirement that Merge Types registered be secure or completely free from risks. Nevertheless, all known security risks MUST be identified in the registration of a Merge

Type.

The security considerations section of all registrations is subject to continuing evaluation and modification, and in particular MAY be extended by use of the "comments on Merge Types" mechanism described in Section 4.1.3 above.

Some of the issues that need to be examined and described in a security analysis of a Merge Type are:

- Merge Types might operate in a way that, while not directly harmful, may result in disclosure of information that either facilitates a subsequent attack or else violates user's privacy in some way.

- A Merge Type should be considered in concert with a Validator. For instance, it is possible for two transactions A and B to both be valid individually, but when merged, the result is not valid. For instance, if a bank account has $1, then two simultaneous debits of $1 are both valid individually, but the resulting merger, yielding -$2, drains the bank account balance below $0. This is the double-spend problem. A system using Merge Types SHOULD be aware of how its Merge Types interact with its Validators.

6. Conventions

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 [RFC2119].

7. Copyright Notice

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

8.2. Informative References

[CRDT] Shapiro, M., Preguica, N., Baquero, C., and M. Zawirski, "Conflict-free replicated data types", in SSS'11 Proceedings of the 13th international conference on Stabilization, safety, and security of distributed systems, October 2011.

[OT] Ellis, C. A., and S. J. Gibbs, "Concurrency control in groupware systems", in SIGMOD '89 Proceedings of the 1989 ACM SIGMOD international conference on Management of data, June 1989.

Authors' Addresses

For more information, the authors of this document are best contacted via Internet mail:

Michael Toomim
Invisible College, Berkeley
2053 Berkeley Way
Berkeley, CA 94704

EMail: toomim@gmail.com
Web: https://invisible.college/@toomim

Mitar Milutinovic
UC Berkeley, EECS Department
775 Soda Hall #1776
Berkeley, CA 94720-1776

EMail: mitar.ietf@tnode.com
Web: https://mitar.tnode.com/

Bryn Bellomy
Invisible College, Berkeley
2053 Berkeley Way
Berkeley, CA 94704

EMail: bryn@signals.io
Web: https://invisible.college/@bryn