N/A
Internet-Draft
Intended status: Informational
Expires: March 14, 2013

T. Keiser
A. Deason, Ed.
Sine Nomine
September 10, 2012

Extensible XDR Discriminated Union Primitive Type
draft-keiser-afs3-xdr-union-06

Abstract

AFS-3 relies upon XDR to carry Rx RPC call payloads. XDR discriminated unions are ill-suited to cases where the protocol needs to evolve without inventing new RPCs, i.e., unknown discriminant values cause the entire XDR payload to fail the decoding step. While this can be circumvented through the use of opaque payloads (and recursive XDR invocations), such solutions are inelegant and difficult to implement. This memo defines a new XDR primitive type, "ext-union", which is derived from the XDR discriminated union primitive type, but with two key variations: 1) each leg contains a length field, and 2) no default leg is supported.

Internet Draft Comments

Comments regarding this draft are solicited. Please include the AFS-3 protocol standardization mailing list (afs3-standardization@openafs.org) as a recipient of any comments.

AFS-3 Document State

This document is in state "draft", as per the document state definitions set forth in [I-D.wilkinson-afs3-standardisation].

Status of this Memo

This Internet-Draft will expire on March 14, 2013.

Copyright Notice

Table of Contents

1. Introduction

AFS-3 [CMU-ITC-88-062] [CMU-ITC-87-068] is a distributed file system that has its origins in the VICE project [CMU-ITC-84-020] [CMU-ITC-85-039] at the Carnegie Mellon University Information Technology Center [CMU-ITC-83-025], a joint venture between CMU and IBM. VICE later became AFS when CMU moved development to a new commercial venture called Transarc Corporation, which later became IBM Pittsburgh Labs. AFS-3 is a suite of un-standardized network protocols based on a remote procedure call (RPC) suite known as Rx [AFS3-RX]. While de jure standards for AFS-3 fail to exist, the various AFS-3 implementations have agreed upon certain de facto standards, largely helped by the existence of an open source fork called OpenAFS that has served the role of reference implementation. In addition to using OpenAFS as a reference, IBM wrote and donated developer documentation that contains somewhat outdated specifications for the Rx protocol and all AFS-3 remote procedure calls, as well as a detailed description of the AFS-3 system architecture.

The Rx RPC protocol utilizes XDR [RFC4506] as its means of encoding RPC call and response payloads. XDR provides a discriminated union type. However, the semantics of the discriminated union base type do not lend themselves to evolution of the discriminant namespace: introduction of new discriminants--when there is no default leg--cause the remainder of the XDR octet stream to be un-parseable (due to the lack of a length field in the encoding) by older peers. This memo introduces a new XDR primitive type that is identical to the XDR discriminated union, except that:

1. each leg contains a length field, and

2. the default leg is disallowed.

1.1. Use Case

Given that this design doubles the overhead from 4 to 8 octets (relative to the XDR discriminated union primitive type), it is ill-suited for use with small implied legs. Within the AFS-3 protocol suite, the primary use case for the extensible union type is to wrap large data structures, rather than small primitive types.

1.2. Abbreviations

AFS - Historically, AFS stood for the Andrew File System; AFS no longer stands for anything

RPC - Remote Procedure Call

RPC-L - Rx RPC Interface Definition Language (fork of ONC RPC [RFC5531] .x file format)

Rx - The Remote Procedure Call mechanism utilized by AFS-3 [AFS3-RX]

XDR - eXternal Data Representation [RFC4506]

2. 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 RFC 2119 [RFC2119].

3. Extensible Discriminated Union

The extensible discriminated union will contain a length field in every leg so that decoding peers can compute the offset of the next object in the XDR octet stream, regardless of whether the discriminant is recognized. For small legs, this will result in significant encoding inefficiency, but it is necessary to permit the union to evolve over time (without peers failing to decode the entire XDR octet stream).

3.1. Extensible Union Type

The definition of the extensible discriminated union is derived from the XDR union defined in section 4.15 of the XDR specification [RFC4506]. Unlike XDR discriminated unions, the XDR types mapped to each arm of the union need not be defined a priori. Instead, the length of the arm is always included in the wire encoding along with the discriminant value, thus permitting the decoder to continue decoding past an unknown discriminant in an XDR octet stream.

How undefined discriminants are handled by the decoder is deliberately left unspecified by this document. Rather, this memo merely specifies which error conditions must be flagged to the caller (see Section 3.4.1). The error handling semantics--for both length mismatches and unknown discriminants--are left up to the definition of any type built upon the ext-union primitive type. While this lends significant flexibility to the design, it also permits XDR

decoding to continue when the expected implied arm length doesn't match the length on the wire. It is RECOMMENDED that implementations fail to decode the entire XDR stream when such a length mismatch is encountered, as such a length mismatch is indicative of either:

1. a significant divergence in RPC-L definitions across the peers, or

2. an undetected bit error in the XDR octet stream.

Extensible discriminated unions are defined in RPC-L as follows:

```
    ext-union [ max-unknown-leg-length=X ]
       switch (discriminant-definition) {
    case discriminant-value-A:
       arm-declaration-A;
    case discriminant-value-B:
       arm-declaration-B;
    ...
    } identifier;
```

Figure 1

Because the discriminant namespace of an extensible union must be capable of evolving over time, it is not possible to support a default leg.

The max-unknown-leg-length optional parameter specifies the maximum permissible leg length for any union leg whose discriminant value is not known to the decoder. This is necessary to limit the scope of denial of service attacks: by permitting the decoder to detect inordinately large payloads--after only receiving the first few octets of the extensible union.

The extensible discriminated union is encoded on the wire as: a 4-octet discriminant, followed by a 4-octet arm length, and finally the variable-length implied arm. The arm length field SHALL count the length of the implied arm in octets, and only the implied arm. In other words, the 8 octets occupied by the discriminant and arm length fields SHALL NOT be counted as part of the arm length value.

```
             0   1   2   3
           +---+---+---+---+---+---+---+---+---+---+---+---+
           |  discriminant |   arm length  |  implied arm  |
           +---+---+---+---+---+---+---+---+---+---+---+---+
           |<---4 octets-->|<---4 octets-->|
```

Figure 2

It should be noted that this design makes it convenient to implement extensible discriminated unions on top of existing XDR primitive types. Thus, in terms of the existing XDR primitives [RFC4506], we can describe an extensible discriminated union as follows:

```
struct ext_union {
    unsigned int discriminant;
    opaque implied_leg<>;
};
```

Figure 3

3.2. RPC-L Changes

In order to implement the above, the XDR grammar, as specified in Section 6.3 of [RFC4506], will need to be modified in the following ways:

- "type-specifier" will require a new production rule mapping to "ext-union-type-spec", and
- an "ext-union-type-spec" production rule will need to be defined.

The "type-specifier" grammar will now include a new production rule for "ext-union-type-spec":

```
type-specifier:
     [ "unsigned" ] "int"
   | [ "unsigned" ] "hyper"
   | "float"
   | "double"
   | "quadruple"
   | "bool"
   | enum-type-spec
   | struct-type-spec
   | union-type-spec
   | identifier
   | ext-union-type-spec
```

Figure 4

The new "ext-union-type-spec" production rule, and the production rule for its nonterminal symbol dependency "ext-union-body", are defined as follows:

```
   ext-union-type-spec:
      "ext-union" ext-union-body
    | "ext-union" ext-union-options ext-union-body

   ext-union-options:
      "[" ext-union-options-body "]"

   ext-union-options-body:
      ext-union-option
    | ext-union-option "," ext-union-options-body

   ext-union-option:
      "max-unknown-leg-length" "=" value

   ext-union-body:
      "switch" "(" declaration ")" "{"
         case-spec
         case-spec *
      "}"
```

Figure 5

3.3. Encoding

It is RECOMMENDED that encoding of an AFS-3 extensible union proceed using the following algorithm:

1. encode an XDR unsigned 32-bit integer (see Section 4.2 of [RFC4506]) containing the discrimant,

2. XDR encode into temporary storage the implied leg (according to the type definition of the type specified for this discriminant value in the ext-union definition), and

3. encode the implied leg using the XDR variable-length opaque specification (see Section 4.10 of [RFC4506]).

3.4. Decoding

It is RECOMMENDED that decoding of an AFS-3 extensible union proceed using the following algorithm:

1. XDR decode the 32-bit unsigned integer containing the discriminant;

2. XDR decode the variable-length opaque blob into temporary storage;

3. If this is a known discriminant:

    1. XDR decode the implied leg payload using the appropriate decoder for the discriminant value;

    2. Compare the length of the decoded XDR payload against the previously-decoded extensible union implied leg length. If the lengths do not match, then mark the union as failed to decode due to a length mismatch;

4. However, if this is an unknown discriminant, then mark the union as failed to decode due to an unknown discriminant;

5. XDR decoding continues at the current offset plus the length of the previously-decoded XDR variable-length opaque.

3.4.1. Error Handling

While the specific decoding algorithm used is left up to the implementor, error handling MUST be implemented as described in this section.

Unknown Discriminant:

When a decoder encounters an unknown discriminant, it MUST mark the discriminant as unknown, and SHOULD proceed to decoding the next element in the XDR stream by seeking past the length and implied leg fields.

Unexpected Length for Discriminant:

When a decoder encounters a length field that doesn't agree with the length expected for this discriminant, it MUST mark the discriminant as failed to decode due to a length mismatch, and SHOULD fail to decode the rest of the XDR octet stream.

Excessively Long Payload:

When the following conditions are satisfied:

1. the max-unknown-leg-length value is specified in the extensible union type definition, and

2. the discriminant value is unknown, and

3. the implied leg length exceeds the max-unknown-leg-length specified in the type definition,

then the decoder MUST mark the discriminant as failed to decode due to excessive length, and SHOULD fail to decode the rest of the XDR octet stream. In addition, an implementation MAY choose to mark a discriminant as failed to decode--and MAY fail to decode the rest of the XDR octet stream--when a known implied leg's length exceeds the max-unknown-leg-length in the extensible union type specification.

4. Acknowledgements

The author would like to thank Jeffrey Hutzelman for proposing standardization of a new XDR primitive type; and Matt Benjamin, Derrick Brashear, Andrew Deason, Steven Jenkins, Michael Meffie, Mark Vitale, and Simon Wilkinson for helping to refine the design of this extensible union type. Finally, the author would like to thank the attendees of the 2011 Pittsburgh AFS Hackathon for their review and comments regarding the -01 draft.

5. IANA Considerations

This memo includes no request to IANA.

6. AFS Assigned Numbers Registrar Considerations

This memo includes no request to the AFS Assigned Numbers Registrar.

7. Security Considerations

Users of this extensible type should understand that any Rx XDR payload is only as secure as the security class bound to the Rx connection in question. This document merely standardizes a primitive type; it is up to the authors of standards defining new types (upon the "ext-union" primitive type) to ensure that the contents of their types are only marshalled over sufficiently-secure security classes.

Decoders should take special care when encountering unexpected implied arm lengths. This could be indicative of serious errors, such as octet stream bit errors that were undetected by lower-layer checksums. At the very least, this error condition implies that the peers do not agree upon their ext-union type-to-discriminant mappings. It is RECOMMENDED that decoders treat this as a hard error and fail to decode the remainder of the XDR octet stream.

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.

[RFC4506] Eisler, M., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

8.2. Informative References

[AFS3-RX] Zayas, E., "AFS-3 Programmer's Reference: Specification for the Rx Remote Procedure Call Facility", Transarc Corp. Tech. Rep. FS-00-D164, August 1991.

[CMU-ITC-83-025] Morris, J., Van Houweling, D., and K. Slack, "The Information Technology Center", CMU ITC Tech. Rep. CMU-ITC-83-025, 1983.

[CMU-ITC-84-020] West, M., "VICE File System Services", CMU ITC Tech. Rep. CMU-ITC-84-020, August 1984.

[CMU-ITC-85-039] Satyanarayanan, M., Howard, J., Nichols, D., Sidebotham, R., Spector, A., and M. West, "The ITC Distributed File System: Principles and Design", Proc. 10th ACM Symp. Operating Sys. Princ. Vol. 19, No. 5, December 1985.

[CMU-ITC-87-068] Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan, M., Sidebotham, R., and M. West, "Scale and Performance in a Distributed File System", ACM Trans. Comp. Sys. Vol. 6, No. 1, pp. 51-81, February 1988.

[CMU-ITC-88-062] Howard, J., "An Overview of the Andrew File System"", Proc. 1988 USENIX Winter Tech. Conf. pp. 23-26, February 1988.

[I-D.wilkinson-afs3-standardisation] Wilkinson, S., "Options for AFS Standardisation", draft-wilkinson-afs3-standardisation-00 (work in progress), June 2010.

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol

Specification Version 2", RFC 5531, May 2009.

Authors' Addresses

Thomas Keiser
Sine Nomine Associates
43596 Blacksmith Square
Ashburn, VA 20147
USA

Email: tkeiser@gmail.com

Andrew Deason (editor)
Sine Nomine Associates
43596 Blacksmith Square
Ashburn, Virginia 20147-4606
USA

Phone: +1 703 723 6673
Email: adeason@sinenomine.net