Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: March 20, 2014

J. Hadi Salim
Mojatatu Networks

September 16, 2013

ForCES Protocol Extensions
draft-ietf-forces-protoextension-00

Abstract

Experience in implementing and deploying ForCES architecture has demonstrated need for a few small extensions both to ease programmability and to improve wire efficiency of some transactions. This document describes a few extensions to the ForCES Protocol Specification [RFC5810] semantics to achieve that end goal.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on March 20, 2014.

Table of Contents

1. Terminology and Conventions 2
1.1. Requirements Language 2
1.2. Definitions 2
2. Introduction 4
3. Problem Overview 4
3.1. Table Ranges 4
3.2. Error codes 5
4. Protocol Update Proposal 5
4.1. Extending Result-TLV 5
4.2. Table Ranges 6
4.3. Error Codes 7
4.3.1. New Codes 7
4.3.2. Extending Result TLV 8
5. IANA Considerations 8
6. Security Considerations 8
7. References 8
7.1. Normative References 9
7.2. Informative References 9
Author's Address 9

1. Terminology and Conventions

1.1. Requirements Language

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].

1.2. Definitions

This document reiterates the terminology defined by the ForCES architecture in various documents for the sake of clarity.

FE Model - The FE model is designed to model the logical processing functions of an FE. The FE model proposed in this document includes three components; the LFB modeling of individual Logical Functional Block (LFB model), the logical interconnection between LFBs (LFB topology), and the FE-level attributes, including FE capabilities. The FE model provides the basis to define the information elements exchanged between the CE and the FE in the ForCES protocol [RFC5810].

LFB (Logical Functional Block) Class (or type) - A template that represents a fine-grained, logically separable aspect of FE processing. Most LFBs relate to packet processing in the data path. LFB classes are the basic building blocks of the FE model.

LFB Instance - As a packet flows through an FE along a data path, it flows through one or multiple LFB instances, where each LFB is an instance of a specific LFB class. Multiple instances of the same LFB class can be present in an FE's data path. Note that we often refer to LFBs without distinguishing between an LFB class and LFB instance when we believe the implied reference is obvious for the given context.

LFB Model - The LFB model describes the content and structures in an LFB, plus the associated data definition. XML is used to provide a formal definition of the necessary structures for the modeling. Four types of information are defined in the LFB model. The core part of the LFB model is the LFB class definitions; the other three types of information define constructs associated with and used by the class definition. These are reusable data types, supported frame (packet) formats, and metadata.

LFB Metadata - Metadata is used to communicate per-packet state from one LFB to another, but is not sent across the network. The FE model defines how such metadata is identified, produced, and consumed by the LFBs, but not how the per-packet state is implemented within actual hardware. Metadata is sent between the FE and the CE on redirect packets.

ForCES Component - A ForCES Component is a well-defined, uniquely identifiable and addressable ForCES model building block. A component has a 32-bit ID, name, type, and an optional synopsis description. These are often referred to simply as components.

LFB Component - An LFB component is a ForCES component that defines the Operational parameters of the LFBs that must be visible to the CEs.

ForCES Protocol - Protocol that runs in the Fp reference points in the ForCES Framework [RFC3746].

ForCES Protocol Layer (ForCES PL) - A layer in the ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, and the ForCES protocol architecture itself as defined in the ForCES Protocol Specification [RFC5810].

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability,

ordering, etc. the ForCES SCTP TML [RFC5811] describes a TML that is mandated for ForCES.

2. Introduction

Experience in implementing and deploying ForCES architecture has demonstrated need for a few small extensions both to ease programmability and to improve wire efficiency of some transactions. This document describes a few extensions to the ForCES Protocol Specification [RFC5810] semantics to achieve that end goal.

This document describes and justifies the need for 2 small extensions which are backward compatible.

1. A table range operation to allow a controller or control application to request an arbitrary range of table rows.

2. Improved Error codes returned to the controller (or control application) to improve granularity of existing defined error codes.

3. Problem Overview

In this section we present sample use cases to illustrate the challenge being addressed.

3.1. Table Ranges

Consider, for the sake of illustration, an FE table with 1 million reasonably sized table rows which are sparsely populated. Assume, again for the sake of illustration, that there are 2000 table rows sparsely populated between the row indices 23-10023.

ForCES GET requests sent from a controller (or control app) are prepended with a path to a component and sent to the FE. In the case of indexed tables, the component path can either be to a table or a table row index. A control application attempting to retrieve the first 2000 table rows appearing between row indices 23 and 10023 can achieve its goal in one of:

- Dump the whole table and filter for the needed 2000 table rows.

- Send upto 10000 ForCES PL requests with monotonically incrementing indices and stop when the needed 2000 entries are retrieved.

- Use ForCES batching to send fewer large messages (several path requests at a time with incrementing indices until you hit the require number of entries).

All of these approaches are programmatically (from an application point of view) unfriendly, tedious, and are seen as abuse of both compute and bandwidth resources.

3.2. Error codes

[RFC5810] has defined a generic set of error codes that are to be returned to the CE from an FE. Deployment experience has shown that it would be useful to have more fine grained error codes. As an example, the error code E_NOT_SUPPORTED could be mapped to many FE error source possibilities that need to be then interpreted by the caller based on some understanding of the nature of the sent request. This makes debugging more time consuming.

4. Protocol Update Proposal

This section describes proposals to update the protocol for issues discussed in Section 3

4.1. Extending Result-TLV

We extend the RESULT-TLV (0x114) to additionally carry an optional description of the result. This is illustrated in Figure 1.

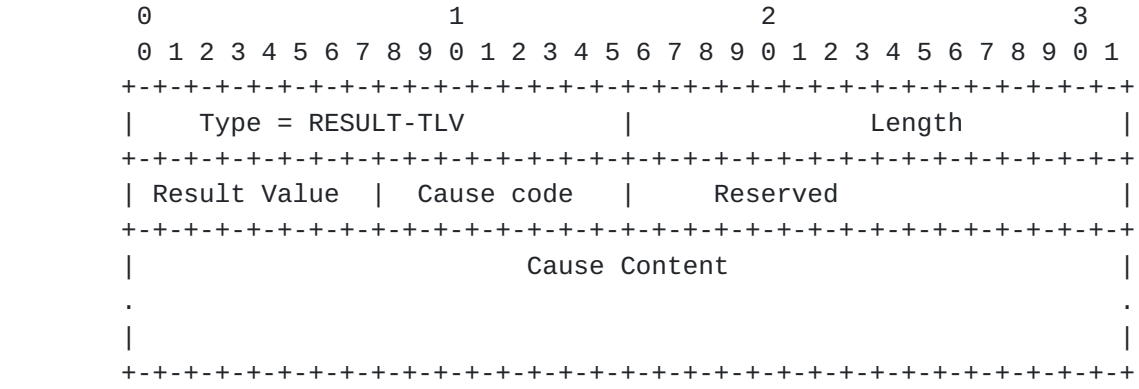

Figure 1: Extended Result TLV

- As before, the Result TLV is expected to be 32 bit aligned.
- The Result Value is derived from the same current namespace as specified in RFC 5810, section 7.1.7 with some new values added in Section 4.3.
- The cause code is an enumeration which describes additional content. This field was originally part of a reserved field. By definition, the user was not supposed to interpret the reserved field and the sender was expected to set it to 0. By default,

therefore, we assume 0 to imply the status quo i.e ignore cause content if present. For this reason, we expect this new extension to be both backward compatible and forward compatible because old implementations ignore the reserved fields and always set them to zero and new implementations will set and intepret the cause code.

4.2. Table Ranges

We propose to add a Table-range TLV (type ID 0x117) that will be associated with the PATH-DATA TLV in the same manner the KEYINFO-TLV is.

```
OPER = GET
       PATH-DATA:
       flags = F_SELTABRANGE,  IDCount = 2, IDs = {1,6}
       TABLERANGE-TLV = {11,23}
```

Figure 2: ForCES table range request

Figure 2 illustrates a GET request for a a table range for rows 11 to 23 of a table with component path of 1/6.

Path flag of F_SELTABRANGE (0x2 i.e bit 1, where bit 0 is F_SELKEY as defined in RFC 5810) is set to indicate the presence of the Table-range TLV. The pathflag bit F_SELTABRANGE can only be used in a GET and is mutually exclusive with F_SELKEY. The FE MUST enforce those constraints and reject a request with an error code of E_INVALID_FLAGS with an english description of what the problem is (refer to Section 4.3).

The Table-range TLV contents constitute:

- A 32 bit start index. An index of 0 implies the beggining of the table row.

- A 32 bit end index. A value of 0xFFFFFFFFFFFFFFFF implies the last entry. XXX: Do we need to define the "end wildcard"?

The response for a table range query will either be:

- The requested table data returned (when at least one referenced row is available); in such a case, a response with a path pointing to the table and whose data content contain the row(s) will be sent to the CE. The data content MUST be encapsulated in sparsedata TLV. The sparse data TLV content will have the "I" (in ILV) for each table row indicating the table indices.

- A result TLV when:

* data is absent where the result code of E_NOT_SUPPORTED (typically returned in current implementations when accessing an empty table entry) with an english message describing the nature of the error (refer to Section 4.3).

* When both a path key and path table range are reflected on the the pathflags, an error code of E_INVALID_FLAGS with an english message describing the nature of the erro (refer to Section 4.3).

* other standard ForCES errors (such as ACL constraints trying to retrieve contents of an unreadable table), accessing unknown components etc.

4.3. Error Codes

We propose two things:

o A new set of error codes.

o A cause string to be carried in the new proposed RESULT-TLV.

4.3.1. New Codes

The following error codes are added.

Result Value	Value	Definition
E_TIMED_OUT	0x18	A time out occured while processing the message
E_CONGEST_NT	0x19	The message was successfully processed but there is congestion detected.
E_EMPTY	0x1A	A requested for table in a GET operation is empty
E_INVALID_PATH_FLGS	0x1B	The submitted path flags in a request are invalid
E_UNKNOWN	0x1C	A generic error catch all error code. To be useful, presented only in association with extended Result TLV from below and carries a string to further extrapolate what the error implies.

Table 1

XXX: More error codes to be added in later doc revisions.

4.3.2. Extending Result TLV

We introduce a cause content of a string to further describe the error code. The result TLV is shown in Figure 3. The content code will be 1 indicating the cause content is an UTF-8 string[N] cause description.

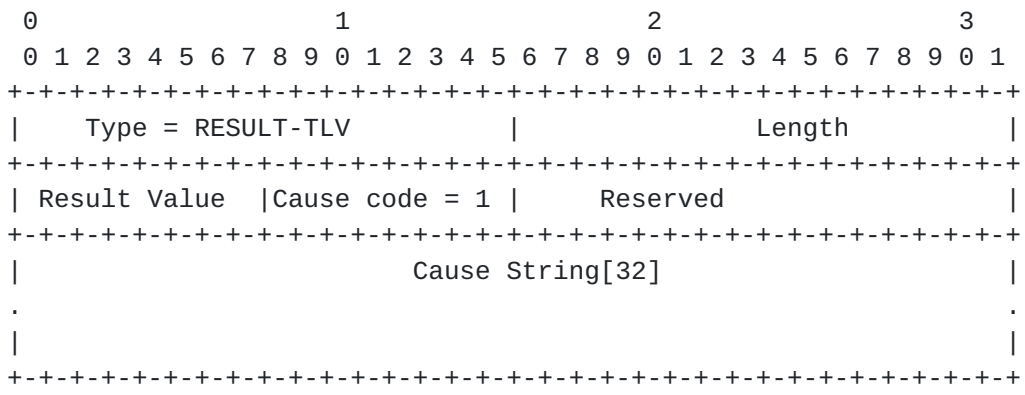

Figure 3: Extending The Result TLV

It is recommended that the maximum size of the cause string should not exceed 32 bytes. We do not propose the cause string be standardized.

5. IANA Considerations

This document registers two new top Level TLVs and two new path flags.

The following new TLVs are defined:

- Table-range TLV (type ID 0x117)

- EXTENDED RESULT-TLV Cause codes.

The following new path flags are defined:

- F_SELTABRANGE (value 0x2 i.e bit 1)

6. Security Considerations

TBD

7. References

7.1. Normative References

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

[RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

7.2. Informative References

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

Author's Address

Jamal Hadi Salim
Mojatatu Networks
Suite 400, 303 Moodie Dr.
Ottawa, Ontario K2H 9R4
Canada

Email: hadi@mojatatu.com