Introduction ...

20..21...22.................4096

RADIUS Packet

“\nThe minimum length is 20 and maximum length is 4096.”
-- RFC 2865 sec 3.

0..1...2..........4096.................8192................12288......65535

RADIUS Packet RADIUS Packet RADIUS Packet ...

Assembled RADIUS Packet (UDP)

.. or ..

RADIUS Packet (TCP)

Any Request

Any Response
Goals

Minimize Surprise

• Fragment-Data used only where necessary
• Clients transmit large requests only while supported by server
• Servers transmit large responses only while supported by client

Plug and Play

• Client and server automatically discover large packet support
• Clients automatically obtain administrative limits from servers
• Servers discover large response support of client and proxy path
• Signal server support for TCP and TCP large packet to clients
Extended-Request

**Section 3** “An Extended-Request packet is sent to the RADIUS server requesting an action whose purpose is determined by an attribute present immediately after RADIUS header within the RADIUS packet”

**Head Attributes**

- **Fragment-Data** Facilitates fragmentation of RADIUS packets beyond 4096 bytes
- **Fragment-Inquire** Requests fragment related capabilities and parameters from RADIUS server
## Extended-Response

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Authenticator**

**Attributes ...**

Indicates success in response to Extended-Request. Response attributes may be included per Extended-Request head attribute specification.

## Extended-Reject

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Authenticator**

**Attributes ...**

Communicates failure of Extended-Request. Error-Cause attribute may be included to provide feedback to client.
**Fragment-Data**

**Command Code Summary**

**Extended-Request**
- On RADIUS Request transmits (Inner) Request to Server.
- During RADIUS Reply used to request next Fragment from Server.

**Extended-Response**
- On RADIUS Request transmits Fragment Ack to Client.
- During RADIUS Reply transmits (Inner) Reply to Client.

**Extended-Reject**
- On RADIUS Request transmits Fragment failure to Client.
- During RADIUS Reply Extended-Reject is unused.
Fragment-Data

**RADIUS REQUEST within Fragment-Data**

- **Code** = Extended-Request ID, Len, **Authenticator** = Same as **Accounting-Request**
- **Type** = Fragment-Data
- **Code** = Think 802.1Q
- **Auth/Acct/CoA/Disc...etc.**
- **Flags** = **More** Data | **Cont**=0
- **Sequence** = 1.2.3...65535

**Authenticator** = **Code** specific Authenticator, doubles as “State” for tracking Fragment-Data requests

**Attributes** = **Code** Specific Request Attributes

Consistent 24 bytes overhead per fragment ...
RADIUS REQUEST Extended-Response, Extended-Reject

Extended-Response

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Extended-Reject

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attributes...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Code = Extended-Response
ID, Len, Authenticator = Same as Accounting-Response

Attributes = Error-Cause
• Missing Attribute
• Administratively Prohibited
• Unsupported Extension
• ...
Fragment-Data

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Authenticator

<table>
<thead>
<tr>
<th>Type</th>
<th>Len</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MCLR</td>
</tr>
</tbody>
</table>

Sequence

<table>
<thead>
<tr>
<th>Authenticator</th>
</tr>
</thead>
</table>

Attributes...

RADIUS REPLy within Fragment-Data

Code = Extended-Response
ID, Len, Authenticator = Same as Accounting-Response

Type = Fragment-Data
Code = Think 802.1Q
Auth/Acct/CoA/Disc...etc.
Flags = More Data | Cont=0
Sequence = 1.2.3...65535

Authenticator = Code specific
Reply Authenticator, doubles as “state” when issuing
Extended-Request to retrieve next response

Attributes = Code Specific
Reply Attributes

Consistent 24 bytes overhead per fragment ...
**Fragment-Data**

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Len</th>
<th>Code</th>
<th>MCLR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Authenticator</th>
</tr>
</thead>
</table>

| Attributes... |

**RADIUS REPLY Extended-Request**

- **Code** = Extended-Request
- **ID, Len, Authenticator** = Same as **Accounting-Request**

**Type**=Fragment-Data
- **Code**, **Flags** (*Cont*=1), **Sequence**, **Authenticator** = Echoed from last received Extended-Response
Sample Flow (Access Request)

Extended-Req C=0, M=1, Seq 1
Extended-Response

Extended-Req C=0, M=1, Seq 2
Extended-Response

Extended-Req C=0, M=0, Seq 3

Fragments assembled internally as Access-Request then processed normally.

Final Extended-Req acknowledged by Extended-Response

Extended-Res C=0, M=1, Seq 1

Extended-Req C=1, M=1, Seq 1

Extended-Res C=0, M=0, Seq 2

Assembled fragment processed normally as Access-Accept or Access-Reject.
Retransmission Overview

1. **RADIUS CLIENT** drives retransmission.
   
   Outer Extended Request/Response identical behavior to Accounting Request/Response.
   
   At any time if response is not received last unacknowledged Request or **Extended-Request** is retransmitted.

2. Acknowledgement of final (More=0) fragment is **Extended-Response** containing RADIUS response (e.g. Access-Accept)
   
   **Clients** may elect to reduce retry timers when transmitting non-final (More=1) **Extended-Requests**.
Retransmission Example (Request)

- Extended-Request, Seq 1
- Extended-Response
- Extended-Request, Seq 2
- Extended-Response
- Extended-Request, Seq 2
- Extended-Response
- Extended-Request, Seq 3
- ...
Retransmission Example (Reply)

Extended-Request (Echo 1)

Extended-Response, Seq 1

Extended-Request (Echo 2)

Extended-Response, Seq 2

Extended-Request (Echo 2)

Extended-Response, Seq 3

Extended-Response, Seq 3

...

Extended-Response, Seq 3
Fragment-Data

**New**

- Network I/O
- AVP Decode and Validation
- Fragment Processing
- AVP Decode and Validation
- Request Processing
- AVP Encode
- AVP Encode
- Fragment Processing
- Network I/O

**Implementation changes?**

"Inner" Authenticator
**Access-Request**
“the Authenticator value is a 16 octet random number” --RFC 2865

**Access-Accept, Access-Reject, Access-Challenge**
“MD5(Code + ID + Length + RequestAuth + Attributes + Secret)” --RFC 2865

**Message-Authenticator**
“HMAC-MD5 [RFC2104] hash of the entire Access-Request packet, including Type, ID, Length and Authenticator, using the shared secret as the key”
--RFC 3579

**Accounting-Request, Disconnect-Request, CoA-Request**
MD5(Code + ID + Length + 16 zero octets + request attributes + Secret)
--RFC 2866

**Accounting-Response, Disconnect-ACK/NAK, CoA-ACK/NAK**
MD5(Code + ID + Length + RequestAuth + response attributes + Secret)
--RFC 2866

Inner packet generated normally with ID field set 0.
Constructing the “Inner” Packet

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Len</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCLR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
</tr>
</tbody>
</table>

Authenticator

Sequence

Attributes

<table>
<thead>
<tr>
<th>Code</th>
<th>ID=0</th>
<th>Len=24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sum frag attr</td>
</tr>
</tbody>
</table>

Authenticator

Attributes

Attributes

Attributes

Fragment Seq 2

Fragment Seq 3

Fragment Seq 4
**Fragment-Data**

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Authenticator

<table>
<thead>
<tr>
<th>Type</th>
<th>Len</th>
<th>Code</th>
<th>MCLR</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Sequence
- Authenticator

- Attributes…

**“Outer” vs. “Inner” Processing**

- ++++++++Outer Packet+++++++:

Consumes head attribute (Fragment-Data) only.

**ALL** additional attributes are appended to “Inner” request byte for byte with no changes.

AVP length fields validated against total “Outer” packet length. No checking is done with respect to type or content of attributes at this stage.

- ++++++++Inner Packet+++++++:

Once all fragments are assembled inner packet is constructed and processed normally as if received on “wire”.

- ++++++++:
Responding to Fragmented Request
• The “Inner” Fragmented request (Authenticator) is used to produce Fragmented response.

Responding to Non-Fragmented Request
• Non-Fragmented request (Authenticator) is used to produce either a Fragmented or Non-Fragmented response.
**Fragment-Inquire**

- **Code** = Extended-Request
- **ID**, **Len**, **Authenticator** = Same as **Accounting-Request**

**Requesting fragment related parameters from server**

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Authenticator**

<table>
<thead>
<tr>
<th>Type</th>
<th>Len</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Optional Request Attributes**

- **Fragment-Stream-Limit**:
  Client is capable of receiving response packets up to length indicated

- **Fragment-Reply-Supported**:
  Client is capable of receiving fragmented response packets

**Type** = Fragment-Inquire
**Value** = 1

**Attributes** = Optional client fragment related parameters communicated to server. To be used only with session based transport.
Fragment-Inquire

<table>
<thead>
<tr>
<th>Code</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Authenticator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attributes…</td>
</tr>
</tbody>
</table>

Providing fragment related parameters to client

**Code** = Extended-Response
**ID**, **Len**, **Authenticator** = Same as **Accounting-Response**

Optional Response Attributes

- **Fragment-Reply-Allowed**: Server implements Fragment-Reply-Supported attribute (Section 6.1)
- **Fragment-Stream-Limit**: Maximum RADIUS packet length supported by server over TCP
- **Fragment-Limit**: Maximum fragmented inner packet length supported by server
- **Fragment-Inquire-Interval**: Interval at which server recommends clients poll for parameter changes
- **Framed-MTU**: Server MTU hint
- **Event-Timestamp**: Server time of Extended-Response
**Fragment-Inquire**

**Fragment-Reply-Supported** is forwarded toward each downstream destination only if downstream has advertised fragment support via Fragment-Inquire response containing **Fragment-Reply-Allowed**.

RADIUS Server is prevented from generating a fragmented response in the event RADIUS Client or any intermediary (e.g. **Proxy B** in example below) does not support Fragment-Data.

When RADIUS Client and all intermediaries support fragments then **Fragment-Reply-Supported** reaches RADIUS Server. Server may then safely issue a fragmented response.
How do you proxy a Extended-Request?

Each “hop” assembles all fragments into an “inner” packet. This packet may then be forwarded by disassembling packet into fragments to next hop.
Must all systems in proxy chain support Fragments?

Unfortunately if any system in the chain does not support fragments then RADIUS packets are limited to 4096 bytes.
Raining on my parade...

Comments, Suggestions and IDEAS welcome!!!

Peter Deacon
peterd@iea-software.com