INTERNET-DRAFT N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan
Intended Status: Proposed Standard
Expires: April 2015
October 20, 2014

IPPM Considerations for the IPv6 PDM Destination Option
draft-elkins-ippm-pdm-option-02

Table of Contents

Abstract

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header has been proposed in a companion document. This document specifies the field limits, calculations, and usage of the PDM in measurement.

Status of this Memo

Copyright and License Notice

1 Background

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header has been proposed in a companion document. This document specifies the field limits, calculations, and usage of the PDM in measurement.

1.1 Terminology

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].

1.2 End User Quality of Service (QoS)

The difference between timing values in the PDM traveling along with the packet will be used to estimate QoS as experienced by an end user device.

For many applications, the key user performance indicator is response time. When the end user is an individual, he is generally indifferent to what is happening along the network; what he really cares about is how long it takes to get a response back. But this is not just a matter of individuals' personal convenience. In many cases, rapid response is critical to the business being conducted.

When the end user is a device (e.g. with the Internet of Things), what matters is the speed with which requested data can be transferred -- specifically, whether the requested data can be transferred in time to accomplish the desired actions. This can be important when the relevant external conditions are subject to rapid change.

Response time and consistency are not just "nice to have". On many networks, the impact can be financial hardship or endanger human life. In some cities, the emergency police contact system operates over IP, law enforcement uses TCP/IP networks, transactions on our stock exchanges are settled using IP networks. The critical nature of such activities to our daily lives and financial well-being demand a solution.

1.3 Need for a Packet Sequence Number

While performing network diagnostics of an end-to-end connection, it often becomes necessary to find the device along the network path creating problems. Diagnostic data may be collected at multiple places along the path (if possible), or at the source and destination. Then, the diagnostic data corresponding to each packet at different observation points must be matched for proper measurements. A sequence number in each packet provides sufficient basis for the matching process. If need be, the timing fields may be used along with the sequence number to ensure uniqueness.

This method of data collection along the path is of special use to determine where packet loss or packet corruption is happening.

The packet sequence number needs to be unique in the context of the session (5-tuple).

1.4 Rationale for proposed solution

The current IPv6 specification does not provide timing nor a similar field in the IPv6 main header or in any extension header. So, we propose the IPv6 Performance and Diagnostic Metrics destination option (PDM) [ELK-PDM].

Advantages include:

1. Real measure of actual transactions.

2. Independence from transport layer protocols.
3. Ability to span organizational boundaries with consistent instrumentation
4. No time synchronization needed between session partners

The PDM provides the ability to quickly determine if the (latency) problem is in the network or in the server (application). More intermediate measurements may be needed if the host or network discrimination is not sufficient. At the client, TCP/IP stack time vs. applications time may still need to be broken out by client software.

2 Measurement Information Derived from PDM

Each packet contains information about the sender and receiver. In IP protocol, the identifying information is called a "5-tuple".

The 5-tuple consists of:

```
SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP, etc.)
```

The PDM contains the following metrics:

```
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTALR  : Delta Last Received
PSNLS    : Packet Sequence Number Last Sent
DELTALS  : Delta Last Sent
```

This information, combined with the 5-tuple, allows the measurement of the following metrics:

1. Round-trip delay
2. Server delay

2.1 Round-Trip Delay

Round-trip delay is the end-to-end delay for a packet from a source host to a destination host. This measurement has been defined, and the advantages and disadvantages discussed in "A Round-trip Delay Metric for IPPM" [RFC2681].

2.2 Server Delay

Server delay is the interval between when a packet is received by a device and the first corresponding packet is sent back in response. This may be "Server Processing Time". It may also be a delay caused by acknowledgements. Server processing time includes the time taken by the combination of the stack and application to return the response. The stack delay may be related to network performance. If this aggregate time is seen as a problem, and there is a need to make a clear distinction between application processing time and stack delay, including that caused by the network, then more client based measurements are needed.

3 Performance and Diagnostic Metrics Destination Option Layout

3.1 Destination Options Header

The IPv6 Destination Options Header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options Header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC2460 [RFC2460]. The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the Destination Options Header (Next Header value = 60). The PDM does not require time synchronization.

3.2 Performance and Diagnostic Metrics Destination Option

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

```
TIMEBASE : Base timer unit
SCALEDL  : Scale for Delta Last Received
SCALEDS  : Scale for Delta Last Sent
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTALR  : Delta Last Received
DELTALS  : Delta Last Sent
```

The PDM destination option is encoded in type-length-value (TLV) format as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  | Option Length |TB |ScaleDL        |   ScaleDS   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   PSN This Packet             |  PSN Last Received              |
|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Delta Last Received         |  Delta Last Sent                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780]

Option Length

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 16.

Time Base

2-bit unsigned integer. It will indicate the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 picosecond.

This field is being included so that a device may choose the granularity which most suits its timer ticks.   That is, so that it

does not have to do more work than needed to convert values required for the PDM.

The possible values of Time Base are as follows:

00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds

Scale Delta Last Received (SCALEDLR)

7-bit signed integer. This is the scaling value for the Delta Last Received (DELTALR) field. The possible values are from -128 to +127. See Section 4 for further discussion on Timing Considerations and formatting of the scaling values.

Scale Delta Last Sent (SCALEDLS)

7-bit signed integer. This is the scaling value for the Delta Last Sent (DELTALS) field. The possible values are from -128 to +127.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use. That is, while to be used while analyzing packet traces.

Initialized at a random number and monotonically incremented for each packet on the 5-tuple. The 5-tuple consists of the source and destination IP addresses, the source and destination ports, and the upper layer protocol (ex. TCP, ICMP, etc). The random number initialization is to make it harder to spoof and insert such packets.

Operating systems MUST implement a separate packet sequence number counter per 5-tuple. Operating systems MUST NOT implement a single counter for all connections.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned integer. This is the PSN of the packet last received on the 5-tuple.

Delta Last Received (DELTALR)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDLR.

DELTALR = Send time packet 2 - Receive time packet 1

Delta Last Sent (DELTALS)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDS.

Delta Last Sent = Receive time packet 2 - Send time packet 1

Option Type

The two highest-order bits of the Option Type field are encoded to indicate specific processing of the option; for the PDM destination option, these two bits MUST be set to 00. This indicates the following processing requirements:

00 - skip over this option and continue processing the header.

RFC2460 [RFC2460] defines other values for the Option Type field. These MUST NOT be used in the PDM. The other values are as follows:

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

In keeping with RFC2460 [RFC2460], the third-highest-order bit of the Option Type specifies whether or not the Option Data of that option

can change en-route to the packet's final destination.

In the PDM, the value of the third-highest-order bit MUST be 0. The possible values are as follows:

0 - Option Data does not change en-route

1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

4 Considerations of Timing Representation

4.1 Encoding the Delta-Time Values

This section makes reference to and expands on the document "Encoding of Time Intervals for the TCP Timestamp Option" [TRAM-TCPM].

4.2 Timer registers are different on different hardware

One of the problems with timestamp recording is the variety of hardware that generates the time value to be used. Different CPUs track the time in registers of different sizes, and the most-frequently-iterated bit could be the first on the left or the first on the right. In order to generate some examples here it is necessary to indicate the type of timer register being used.

As described in the "IBM z/Architecture Principles of Operation" [IBM-POPS], the Time-Of-Day clock in a zSeries CPU is a 104-bit register, where bit 51 is incremented approximately every microsecond:

```
                                                                   1
0        1         2         3         4         5         6         0
+--------+---------+---------+---------+---------+---------+--+...+
|        |         |         |         |         |*        |      |
+--------+---------+---------+---------+---------+---------+--+...+
^                                                  ^              ^
0                                                51 = 1 usec    103
```

To represent these values concisely a hexadecimal representation will be used, where each digit represents 4 binary bits. Thus:

```
0000 0000 0000 0001 = 1 timer unit (2**-12 usec, or about 244 psec)
0000 0000 0000 1000 = 1 microsecond
```



```
0000 0000 003E 8000 = 1 millisecond
0000 0000 F424 0000 = 1 second
0000 0039 3870 0000 = 1 minute
0000 0D69 3A40 0000 = 1 hour
0001 41DD 7600 0000 = 1 day
```

Note that only the first 64 bits of the register are commonly represented, as that represents a count of timer units on this hardware. Commonly the first 52 bits are all that are displayed, as that represents a count of microseconds.

4.3 Timer Units on Other Systems

This encoding method works the same with other hardware clock formats. The method uses a microsecond as the basic value and allows for large time differentials.

4.4 Time Base

We propose a base unit for the time. This is a 2-bit integer indicating the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 picosecond.

The possible values of Time Base are as follows:

```
00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds
```

Time base is not necessarily equivalent to length of one timer tick. That is, on many, if not all, systems, the timer tick value will not be in complete units of nanoseconds, milliseconds, etc. For example, on an IBM zSeries machine, one timer tick (or clock unit) is 2 to the -12th microseconds.

Therefore, some amount of conversion may be needed to approximate Time Base units.

4.5 Timer-value scaling

As discussed in [TRAM-TCPM] we propose storing not an entire time-interval value, but just the most significant bits of that value, along with a scaling factor to indicate the magnitude of the time-interval value. In our case, we will use the high-order 16 bits. The

scaling value will be the number of bits in the timer register to the right of the 16th significant bit. That is, if the timer register contains this binary value:

```
1110100011010100101001010001000000000000
<-16 bits     -><-24 bits              ->
```

then, the values stored would be 1110 1000 1101 0100 in binary (E8D4 hexadecimal) for the time value and 24 for the scaling value. Note that the displayed value is the binary equivalent of 1 second expressed in picoseconds.

The below table represents a device which has a TimeBase of picosecond (or 00). The smallest and simplest value to represent is 1 picosecond; the time value stored is 1, and the scaling value is 0. Using values from the table below, we have:

Delta time	Time value in picoseconds	Encoded value	Scaling decimal
1 picosecond	1	1	0
1 nanosecond	3E8	3E8	0
1 microsecond	F4240	F424	4
1 millisecond	3B9ACA00	3B9A	16
1 second	E8D4A51000	E8D4	24
1 minute	3691D6AFC000	3691	32
1 hour	cca2e51310000	CCA2	36
1 day	132f4579c980000	132F	44
365 days	1b5a660ea44b80000	1B5A	52

Sample binary values (high order 16 bits taken)

```
   1 psec           1                                                     0001
   1 nsec         3E8                                           0011 1110 1000
   1 usec       F4240                                 1111 0100 0010 0100 0000
   1 msec    3B9ACA00           0011 1011 1001 1010 1100 1010 0000 0000
   1 sec    E8D4A51000 1110 1000 1101 0100 1010 0101 0001 0000 0000 0000
```


4.6 Limitations with this encoding method

If we follow the specification in [TRAM-TCPM], the size of one of these time-interval fields is limited to this 11-bit value and five-bit scale, so that they fit into a 16-bit space. With that limitation, the maximum value that could be stored in 16 bits is:

```
    11-bit value   Scale
    =============  ======
    1111 1111 111  1 1111
```

or an encoded value of 3FF and a scale value of 31. This value corresponds to any time differential between:

```
                 |<Count of zeroes is the Scale value>|
   11 1111 1111 1000 0000 0000 0000 0000 0000 0000 0000  (binary)
   3  F    F    8    0    0    0    0    0    0    0     (hexadecimal)
```

and

```
   11 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111  (binary)
   3  F    F    F    F    F    F    F    F    F    F     (hexadecimal)
```

This time value, 3FFFFFFFFFF, converts to 50 days, 21 hours, 40 minutes and 46.511103 seconds. A time differential 1 microsecond longer won't fit into 16 bits using this encoding method.

4.7 Lack of precision induced by timer value truncation

When the bit values following the first 11 significant bits are truncated, obviously loss of precision in the value. The range of values that will be truncated to the same encoded value is 2**(Scale)-1 microseconds.

The smallest time differential value that will be truncated is

    1000 0000 0000 = 2.048 msec

The value

    1000 0000 0001 = 2.049 msec

will be truncated to the same encoded value, which is 400 in hex, with a scale value of 1. With the scale value of 1, the value range is calculated as 2**1 - 1, or 1 usec, which you can see is the difference between these minimum and maximum values.

With that in mind, let's look at that table of delta time values again, where the Precision is the range from the smallest value corresponding to this encoded value to the largest:

Delta time	Time value in microseconds	Encoded value	Scale	Precision
1 microsecond	1	1	0	0:00.000000
1 millisecond	38E	38E	0	0:00.000000
1 second	F4240	7A1	9	0:00.000511
1 minute	3938700	727	15	0:00.032767
1 hour	D693A400	6B4	21	0:02.097151
1 day	141DD76000	507	26	1:07.108863
Maximum value	3FFFFFFFFFF	7FF	31	35:47.483647

So, when measuring the delay between transmission of two packets, or between the reception of two packets, any delay shorter than 50 days 21 hours and change can be stored in this encoded fashion within 16

bits. When you encode, for example, a DTN response time delay of 50 days, 21 hours and 40 minutes, you can be assured of accuracy within 35 minutes.

5 Sample Implementation Flow PDM

Following is a sample simple flow for the PDM with one packet sent from Host A and one packet received by Host B. The PDM does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics are shown below each packet sent or received.

Each packet, in addition to the PDM contains information on the sender and receiver. As discussed before, a 5-tuple consists of:

```
SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP)
```

It should be understood that the packet identification information is in each packet. We will not repeat that in each of the following steps.

5.1 Step 1

Packet 1 is sent from Host A to Host B. The time for Host A is set initially to 10:00AM.

The time and packet sequence number are saved by the sender internally. The packet sequence number and delta times are sent in the packet.

Packet 1

```
                +----------+           +----------+
                |          |           |          |
                |   Host   | ----------> |   Host   |
                |    A     |           |    B     |
                |          |           |          |
                +----------+           +----------+
```

PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:     25
PSNLR    : Packet Sequence Number Last Received:   -
DELTALR  : Delta Last Received:                    -
SCALEDL  : Scale of Delta LR:                      0
DELTALS  : Delta Last Sent:                        -
SCALEDS  : Scale of Delta LS:                      0
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

Internally, within the sender, Host A, it must keep:

```
Packet Sequence Number of the last packet sent:     25
Time the last packet was sent:                  10:00:00
```

Note, the initial PSNTP from Host A starts at a random number. In this case, 25. The timestamp is in seconds for the sake of simplicity.

5.2 Step 2

Packet 1 is received at Host B. Its time is set to one hour later than Host A. In this case, 11:00AM

Internally, within the receiver, Host B, it must note:

```
Packet Sequence Number of the last packet received:    25
Time the last packet was received                :    11:00:03
```


Note, this timestamp is in Host B time. It has nothing whatsoever to do with Host A time. The Packet Sequence Number of the last packet received will become PSNLR which will be sent out in the packet sent by Host B in the next step. The time last received will be used to

calculate the DELTALR value to be sent out in the packet sent by Host B in the next step.

5.3 Step 3

Packet 2 is sent by Host B to Host A. Note, the initial packet sequence number (PSNTP) from Host B starts at a random number. In this case, 12.   Before sending the packet, Host B does a calculation of deltas. Since Host B knows when it is sending the packet, and it knows when it received the previous packet, it can do the following calculation:

Sending time (packet 2) - receive time (packet 1)

We will call the result of this calculation: Delta Last Received

That is:

DELTALR = Sending time (packet 2) - receive time (packet 1)

Note, both sending time and receive time are saved internally in Host B. They do not travel in the packet. Only the Delta is in the packet.

Assume that within Host B is the following:

Packet Sequence Number of the last packet received:	25
Time the last packet was received:	11:00:03
Packet Sequence Number of this packet:	12
Time this packet is being sent:	11:00:07

We can now calculate a delta value to be sent out in the packet. DELTALR becomes:

4 seconds = 11:00:07 - 11:00:03

This is the derived metric: Server Delay.   The time and scaling factor must be calculated. Then, this value, along with the packet sequence numbers will be sent to Host A as follows:

Packet 2

```
                    +----------+           +----------+
                    |          |           |          |
                    |   Host   | <---------- |   Host   |
                    |    A     |           |    B     |
                    |          |           |          |
                    +----------+           +----------+
```

PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    12
PSNLR    : Packet Sequence Number Last Received:  25
DELTALR  : Delta Last Received:                 3A35 (4 seconds)
SCALEDL  : Scale of Delta LR:                     25
DELTALS  : Delta Last Sent:                        -
SCALEDS  : Scale of Delta LS:                      0
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

The metric left to be calculated is the Round-Trip Delay. This will be calculated by Host A when it receives Packet 2.

5.4 Step 4

Packet 2 is received at Host A. Remember, its time is set to one hour earlier than Host B. Internally, it must note:

```
Packet Sequence Number of the last packet received:    12
Time the last packet was received                 :    10:00:12
```

Note, this timestamp is in Host A time. It has nothing whatsoever to do with Host B time.

So, now, Host A can calculate total end-to-end time. That is:

End-to-End Time = Time Last Received - Time Last Sent

For example, packet 25 was sent by Host A at 10:00:00. Packet 12 was received by Host A at 10:00:12 so:

End-to-End time = 10:00:12 - 10:00:00 or 12 (Server and Network RT delay combined)

This derived metric we will call DELTALS or Delta Last Sent.

We can now also calculate round trip delay. The formula is:

Round trip delay = DELTALS - DELTALR

Or:

Round trip delay = 12 - 4 or 8

Now, the only problem is that at this point all metrics are in Host A only and not exposed in a packet. To do that, we need a third packet.

5.5 Step 5

Packet 3 is sent from Host A to Host B.

```
+----------+            +----------+
|          |            |          |
|   Host   | ---------> |   Host   |
|    A     |            |    B     |
|          |            |          |
+----------+            +----------+
```

PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    26
PSNLR    : Packet Sequence Number Last Received:  12
DELTALR  : Delta Last Received:                    0
SCALEDL  : Scale of Delta LR                       0
DELTALS  : Delta Last Sent:                     105e (12 seconds)
SCALEDL  : Scale of Delta LR                      26
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

To calculate Two-Way Delay, any packet capture device may look at these packets and do what is necessary.

6 Security Considerations

TBD. It is conceivable that in allowing this Destination Option through a firewall, that other malicious traffic may be allowed

through.

7 IANA Considerations

Option Type TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780].

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.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[IBM-POPS] IBM Corporation, "IBM z/Architecture Principles of Operation", SA22-7832, 1990-2012

8.2 Informative References

[ELK-PDM] Elkins, N., "draft-elkins-6man-ipv6-pdm-dest-option-09", Internet Draft, October 2014. [Work in Progress]

[TRAM-TCPM] Trammel, B., "Encoding of Time Intervals for the TCP Timestamp Option-01", Internet Draft, July 2013. [Work in Progress]

9 Acknowledgments

The authors would like to thank Keven Haining, Al Morton, Brian Trammel, David Boyes, and Rick Troth for their comments and assistance.

Authors' Addresses

Nalini Elkins
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA 93924
United States
Phone: +1 831 659 8360
Email: nalini.elkins@insidethestack.com
http://www.insidethestack.com

Robert Hamilton
Chemical Abstracts Service
A Division of the American Chemical Society

2540 Olentangy River Road
Columbus, Ohio 43202
United States
Phone: +1 614 447 3600 x2517
Email: rhamilton@cas.org
http://www.cas.org

Michael S. Ackermann
Blue Cross Blue Shield of Michigan
P.O. Box 2888
Detroit, Michigan 48231
United States
Phone: +1 310 460 4080
Email: mackermann@bcbsmi.com
http://www.bcbsmi.com