

# *ES/1 NEO*

CSシリーズ

CS-Network

Packet Monitor

使用者の手引き



株式会社 アイ・アイ・エム

第20版 2016年8月

---

©版權所有者 株式会社 アイ・アイ・エム 2016年

**© COPYRIGHT IIM CORPORATION, 2016**

**ALL RIGHT RESERVED. NO PART OF THIS PUBLICATION MAY  
REPRODUCED OR TRANSMITTED IN ANY FORM BY ANY MEANS,  
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY RECORDING,  
OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM WITHOUT  
PERMISSION IN WRITING FROM THE PUBLISHER.**

**“RESTRICTED MATERIAL OF IIM “LICENSED MATERIALS – PROPERTY OF IIM**

# 目次

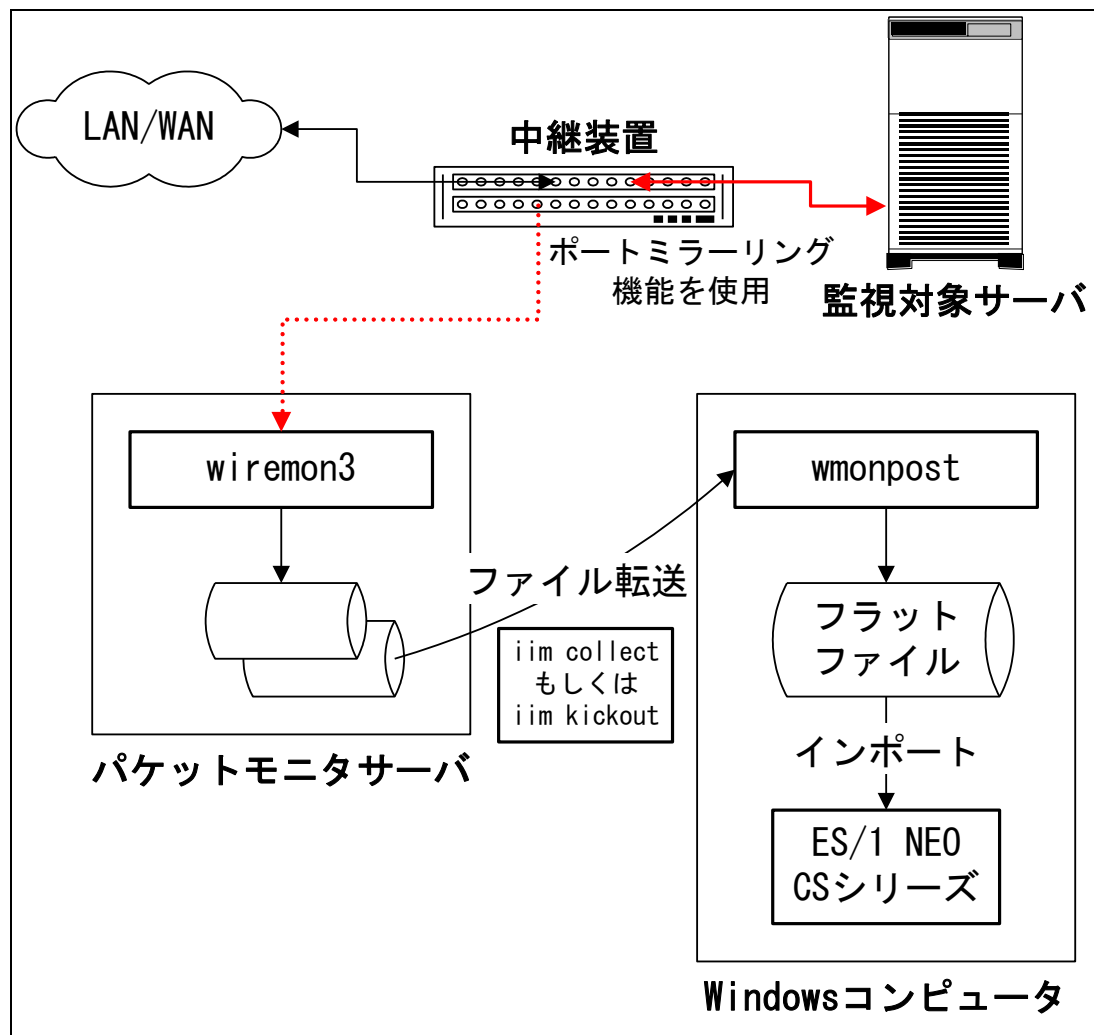
第 1 章	概要 .....	1
1.1.	全体構成 .....	1
第 2 章	パケットモニタのソフトウェア .....	2
2.1.	wiremon3.....	2
2.1.1.	wiremon3 の起動方法 .....	4
2.1.2.	wiremon3 の停止方法 .....	4
2.1.3.	wiremon3.conf の記述 .....	5
2.1.4.	パケット取得状況の確認 .....	9
2.2.	wmonpost .....	10
2.2.1.	動作設定 .....	11
2.2.2.	wmonpost の実行 .....	11
2.2.3.	複数の wiremon3 からの入力 .....	12
2.2.4.	ロギングの指定 .....	12
2.2.5.	フィルタについて .....	13
第 3 章	取得可能データ項目 .....	19
3.1.	全体でのパケット記録情報 .....	19
3.2.	各プロトコル層におけるパケット数/バイト数の情報 .....	19
3.3.	TCP セッション情報（TCP トレースログ解析情報） .....	21
3.4.	TCP レスpons時間分布 .....	23
3.5.	拡張 TCP セッション情報（Peer To Peer TCP トレースログ解析情報） .....	24
3.6.	オーバーフローパケット情報 .....	25

# 第1章 概要

本書は、稼働監視対象サーバの送受信パケットを取り込み、ネットワークのパフォーマンスデータを収集する、ES/1 NEO CS シリーズの Packet Monitor(パケットモニタ)について記述しています。

## 1.1. 全体構成

以下にパケットモニタの全体構成を示します。



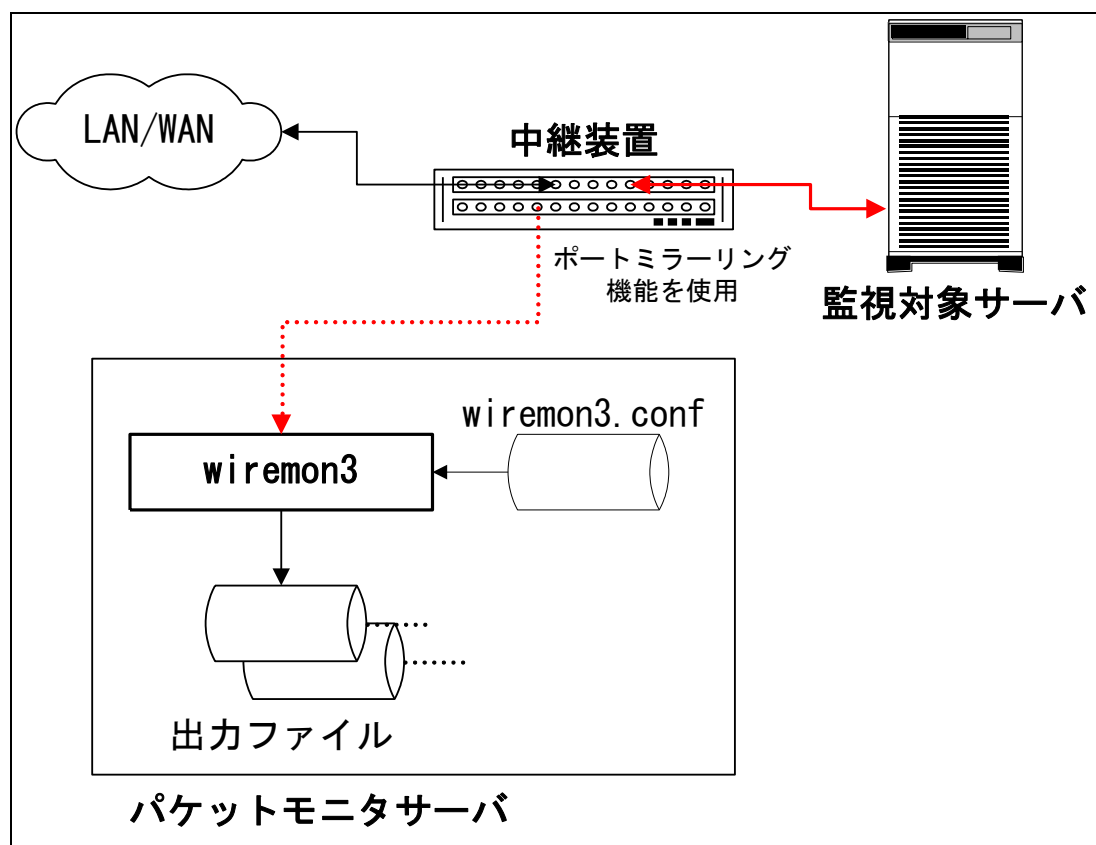
監視対象サーバの送受信パケットは専用のサーバ（以降パケットモニタサーバと記述します）上で動作するプログラム（wiremon3）により取り込まれ、独自のファイル形式にて出力後、Windows コンピュータ上で動作するプログラム（wmonpost）で ES/1 NEO CS シリーズの使用するフラットファイル形式へと変換されます。

## 第2章 パケットモニタのソフトウェア

この章ではパケットモニタを構成する 2 つのソフトウェア(wiremon3 と wmonpost)について説明します。

### 2.1. wiremon3

wiremon3 は、実行するコンピュータのネットワークインターフェースを無差別受信モード(promiscuous mode)に設定し、検出されたパケットを分類・集計・加工し、ファイルに記録するプログラムです。記録したファイルは後に後処理のプログラム(wmonpost)により、ES/1 NEO CS シリーズにて利用するフラットファイルの形式に変換することが可能です。



パケットモニタサーバのネットワークインターフェースは無差別受信モードに設定されますが、監視対象としたいサーバとの間の中継装置がスイッチの場合は、スイッチのポートミラーリングの機能を利用して監視対象サーバの入出力パケットがパケットモニタサーバのネットワークインターフェースにも流れるように設定する必要があります。

#### 注意！

- ・スイッチの種類によってはポートミラーリングの設定変更後に設定内容を保存しないと、スイッチを再起動した時に設定が変更前の状態に戻ってしまうことがあります。スイッチの仕様を確認の上、ミラーリングの設定を行ってください。
- ・パケットモニタサーバのネットワーク設定を変更する場合は、必ず wiremon3 を停止してから行ってください。また、設定変更後はパケットモニタサーバを必ず再起動してください。

Wiremon3 は設定ファイル wiremon3.conf の記述内容に従って、検出したパケットの分類・集計・記録を行います。  
wiremon3 が記録する情報は大きく分けて以下の 2 種類の情報になります。

- ・インターバル毎にプロトコル毎・ポート番号毎等により分類・集計したパケット数やバイト数  
(ファイル名 : WFnC\_yyyymmdd\_hhmmss.gz)
- ・TCP のトレースログ  
(ファイル名 : WFnS\_yyyymmdd\_hhmmss.gz)

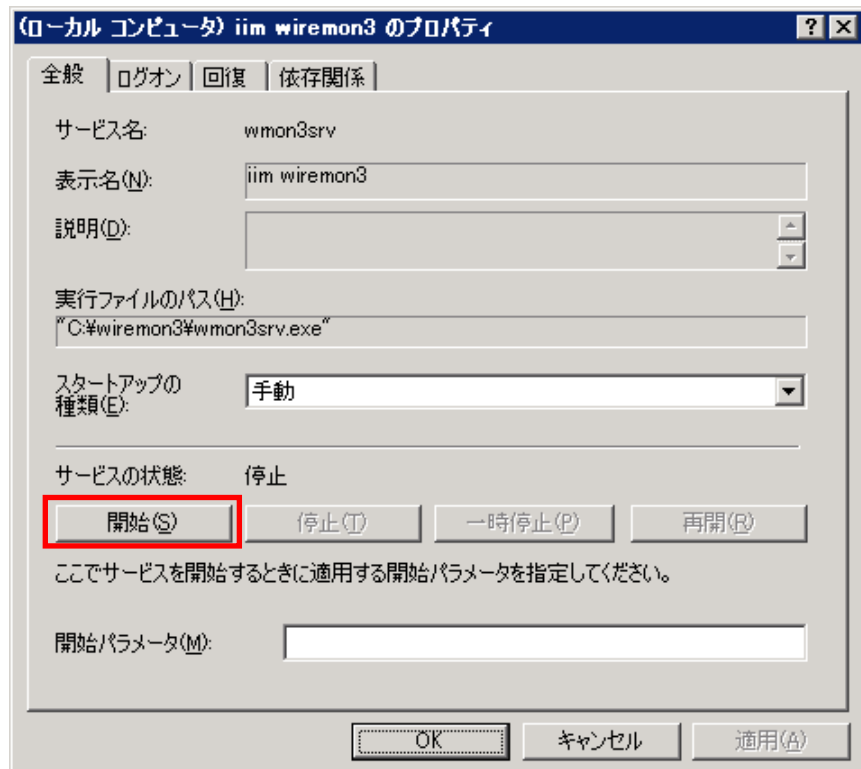
**メモ！**

ファイル名が WT で始まるファイルは現在書込みを行っているファイルですので、ファイル転送の対象外としてください。

wiremon3.conf には分類・記録対象のポート番号や IP アドレス等を指定します。

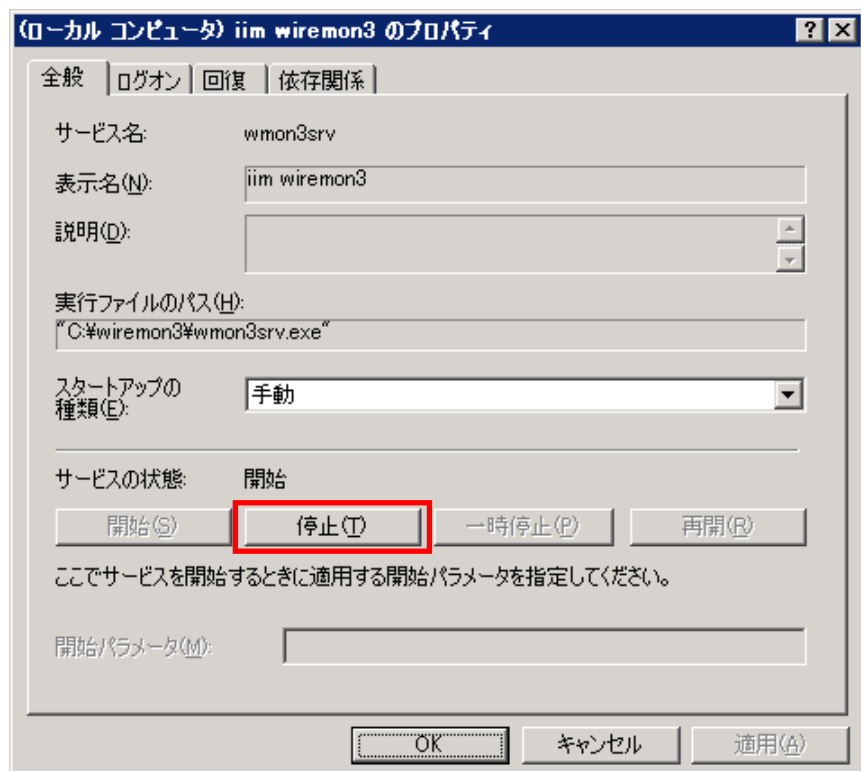
### 2.1.1. wiremon3 の起動方法

[スタート]メニュー→「設定」→「コントロールパネル」→「管理ツール」→「サービス」から「iim wiremon3」を選択し、サービスを開始します。iim wiremon3 のスタートアップの種類はデフォルトで手動に設定されています。



### 2.1.2. wiremon3 の停止方法

[スタート]メニュー→「設定」→「コントロールパネル」→「管理ツール」→「サービス」から「iim wiremon3」を選択し、サービスを停止します。



### 2.1.3. wiremon3.conf の記述

wiremon3.conf ファイル内にはキーと値の組み合わせ('='でつなげます)を記述します。

キーと値の組み合わせは 1 行に収めなくてはなりません。

英文字(A-Z、a-z)以外で始まる行はコメントとして無視されます。

複数の値をキーに指定したい場合は同じキーの行を複数記述することが可能です。

(例)

```
udp_sport=53
```

```
udp_sport=137
```

```
...
```

また、キーによってはカンマ区切りによって複数の値を指定することが可能です。

(例)

```
udp_sport=53,137,...
```

以下に指定可能なキーについて説明します。

複数行による指定が可能なキーについてはキー名の後に '+' を、

複数行、かつ、カンマ区切りによる指定が可能なキーは ' \* ' をつけてあります。

#### sys\_output

wiremon3 が情報を記録するディレクトリを指定します。フルパスによる指定が必要です。

パケットモニタサーバ出荷時のデフォルト設定は以下の通りです。

```
sys_output=D:¥IIM_WORK¥WMONOUT
```

wiremon3 が情報を記録するファイルは以下の 2 種類のファイルです。

- WFnC\_YYYYMMDD\_HHMMSS.gz

- WFnS\_YYYYMMDD\_HHMMSS.gz

(上記 WFn の n の部分は 9 か 0 であり、YYYYMMDD\_HHMMSS はタイムスタンプになります。)

wmonpost ではこの 2 種類のファイルを入力としてフラットファイルを作成します。

(いずれのファイルもバイナリ形式の為、Windows マシンに ftp で転送する場合はバイナリモードが必須となります。)

#### sys\_fcyc

wiremon3 の出力するファイルを切り替える間隔を指定します。例えば 15 を指定した場合、毎時 0、15、30、45 の各分に出力ファイルを切り替えます。指定可能な数値は 2、3、5、10、15、20、30、60、120、180、240、360、480、720、1440 です。このオプションを指定しなかった場合のデフォルト値は 30(分)です。

#### sys\_devname

wiremon3 がモニタを行うネットワークインターフェース名を指定します。通常はパケットモニタサーバ出荷時に既に設定されています。ネットワークインターフェース名の確認はコマンドラインから "wiremon3 -D" を実行します。



(実行例)

C:¥wiremon3>wiremon3 -D

- 1 ¥Device¥NPF\_GenericDialupAdapter (Adapter for generic dialup and VPN capture)
- 2 ¥Device¥NPF\_{6A95DE43-1D19-464A-8A39-BE0C986DAEE2} (Broadcom NetXtreme Gigabit Ethernet Driver inet 172. 16. 1. 1 / 255.255. 0. 0)
- 3 ¥Device¥NPF\_{FA558CD2-8A20-462F-A0AA-0A4ADF532581} (Broadcom NetXtreme Gigabit Ethernet Driver)

“sys\_devname=”には“¥Device...{....}”の部分指定します。

(例)

sys\_devname=¥Device¥NPF\_{FA558CD2-8A20-462F-A0AA-0A4ADF532581}

sys\_mt

wiremon3 をマルチスレッド(キャプチャスレッド+記録スレッド)で起動します。

(例)

sys\_mt=1

ether\_addr+

IP パケット以外の Ethernet フレーム数/バイト数を記録する対象となるホストの Ethernet アドレスを指定します。

コロン(:)で区切られた 6 個の 16 進数で指定してください。

(例)

ether\_addr=00:0a:e4:40:25:cf

ether\_type\*

フレーム数/バイト数を個別に記録する Ethernet フレームのタイプ(16bit)を指定します。

10 進数、又は、'0x'で始まる 16 進数で指定してください。

ここで指定しなかったタイプのフレームは合算して記録されます。

(例)

ether\_type=0x0806

ipv4\_mask

キャプチャを行うサブネットの IPv4 サブネットマスクを指定してください。

255.255.0.0 の様な表現、又は、0xffff0000 の様な'0x'で始まる 16 進数表現が可能です。

(例)

ipv4\_mask=0xffff0000

(\*)UDP パケット等のブロードキャスト/ユニキャストの判定に必要です。

**ipv4\_addr+**

ICMP/UDP/TCP 以外の IPv4 パケット数/バイト数を記録する対象となるホストの IPv4 アドレスを指定します。  
ピリオド(.)で区切られた 4 つの 10 進整数で記述してください。

(例)

ipv4\_addr=172.16.1.1

**ipv4\_type\***

ICMP/UDP/TCP 以外の IPv4 パケット数/バイト数を個別に記録する IPv4 プロトコルタイプ(8bit)を指定します。  
10 進数、又は、'0x'で始まる 16 進数で指定してください。  
ここで指定しなかったタイプのパケットは合算して記録されます。

(例)

ipv4\_type=2

**udp\_4addr+**

UDP パケット数/バイト数を記録する対象となるホストの IPv4 アドレスを指定します。  
ピリオド(.)で区切られた 4 つの 10 進整数で記述してください。

(例)

udp\_addr=172.16.1.1

(\*)このキーは省略可能です。省略可能な場合は ipv4\_addr キーで指定したアドレスにより記録します。

**udp\_sport\***

個別にパケット数/バイト数を記録する UDP ポート番号を記録対象ホスト側のポート番号により指定します。  
10 進数、又は、'0x'で始まる 16 進数で指定してください。

(例)

udp\_sport=137

**udp\_dport\***

個別にパケット数/バイト数を記録する UDP ポート番号を記録対象ホストの通信相手側のポート番号により指定します。  
10 進数、又は、'0x'で始まる 16 進数で指定してください。

(例)

udp\_dport=53

(\*)udp\_sport と udp\_dport のどちらの条件にも一致しないパケットは合算して記録されます。

**tcp\_4addr+**

TCP パケット数/バイト数/トレースログを記録する対象となるホストの IPv4 アドレスを指定します。  
ピリオド(.)で区切られた 4 つの 10 進整数で記述してください。

(例)

tcp\_4addr=172.16.1.1

(\*)このキーは省略可能です。省略可能な場合は ipv4\_addr キーで指定したアドレスにより記録します。

**tcp\_s\_sport\***

個別にパケット数/バイト数を記録する TCP ポート番号を記録対象ホスト側のポート番号により指定します。

10 進数、又は、'0x'で始まる 16 進数で指定してください。

(例)

tcp\_s\_sport=21,23

**tcp\_s\_dport\***

個別にパケット数/バイト数を記録する TCP ポート番号を記録対象ホストの通信相手側のポート番号により指定します。

10 進数、又は、'0x'で始まる 16 進数で指定してください。

(例)

tcp\_s\_dport=80,8080

(\*)tcp\_s\_sport と tcp\_s\_dport のどちらの条件にも一致しないパケットは合算して記録されます。

**tcp\_d\_dport\***

TCP トレースログ記録対象とする TCP ポート番号を記録対象ホストの通信相手側のポート番号により指定します。

10 進数、又は、'0x'で始まる 16 進数で指定してください。

(例)

tcp\_s\_dport=80,8080

**tcp\_d\_sport\***

TCP トレースログ記録対象とする TCP ポート番号を記録対象ホスト側のポート番号により指定します。

10 進数、又は、'0x'で始まる 16 進数で指定してください。

**tcp\_d\_all\_port**

記録対象ホストのすべての TCP セッションについてトレースログを記録するかどうかを指定します。

1 を指定するとポート番号に関係なくすべての TCP セッションのトレースログを記録します。

0 を指定すると tcp\_d\_dport と tcp\_d\_sport の指定に従いトレースログを記録します。

(例)

tcp\_d\_all\_port=1

**tcp\_d\_all\_addr**

アドレス指定に関わらずすべての TCP パケットのトレースを行うかどうかを指定します。

1 を指定するとミラーリングされているすべての TCP パケットを取得します。

0 を指定すると ipv4\_addr、tcp\_4addr の指定に従い対象サーバの TCP パケットを取得します。

(例)

tcp\_d\_all\_addr=1

(\*)tcp\_d\_all\_addr=1 と指定した場合、ipv4\_addr キーは省略可能となります。

#### 2.1.4. パケット取得状況の確認

Wiremon3 の初回実行時、あるいは wiremon3 の設定変更があった場合などに、wiremon3 インストールフォルダの captstat.exe をコマンドラインから実行することで、wiremon3 の TCP パケット取得状況を確認することができます。

下記に、captstat.exe の実行と出力の例を示します。

このコマンドは、最近の TCP パケット取得状況と起動してからの総 TCP パケット取得状況を、データ収集対象となっている IP アドレス毎に表示します。

TCP パケットを検出しなかった IP アドレスについては表示を行いません。

「TCP NOT ARRIVED.」はいずれの IP アドレスについても TCP パケットを検出しなかった場合に表示されます。

recv/drop はそれぞれ wiremon3 のデータ収集インターフェースが受信したパケット数と破棄したパケット数を示しています。

ここで、Send と Recv の項目に数値が出力されていることを確認してください。

(実行と出力の例)

C:¥wiremon3>captstat.exe

2005/05/12 11:10:00 - 11:11:00 recv:31, drop:0

-----Send----- Recv-----

TCP NOT ARRIVED.

2005/05/12 11:11:00 - 11:12:00 recv:6541, drop:0

-----Send----- Recv-----

172. 18. 11. 95:	600	700
172. 18. 11. 96:	600	700
172. 18. 11. 97:	600	700
172. 18. 11. 98:	600	700
172. 18. 11. 99:	600	700

2005/05/12 11:12:00 \*\*TOTAL\*\* recv:49631, drop:0

-----Send----- Recv-----

172. 18. 11. 95:	1800	2000
172. 18. 11. 96:	1800	2000
172. 18. 11. 97:	1800	2000
172. 18. 11. 98:	1800	2000
172. 18. 11. 99:	1800	2000

(\*)

tcp\_d\_all\_addr=1 の記述があり、ipv4\_addr キーの指定が省略されている場合については以下のように出力します。

(Send と Recv は同じになります)

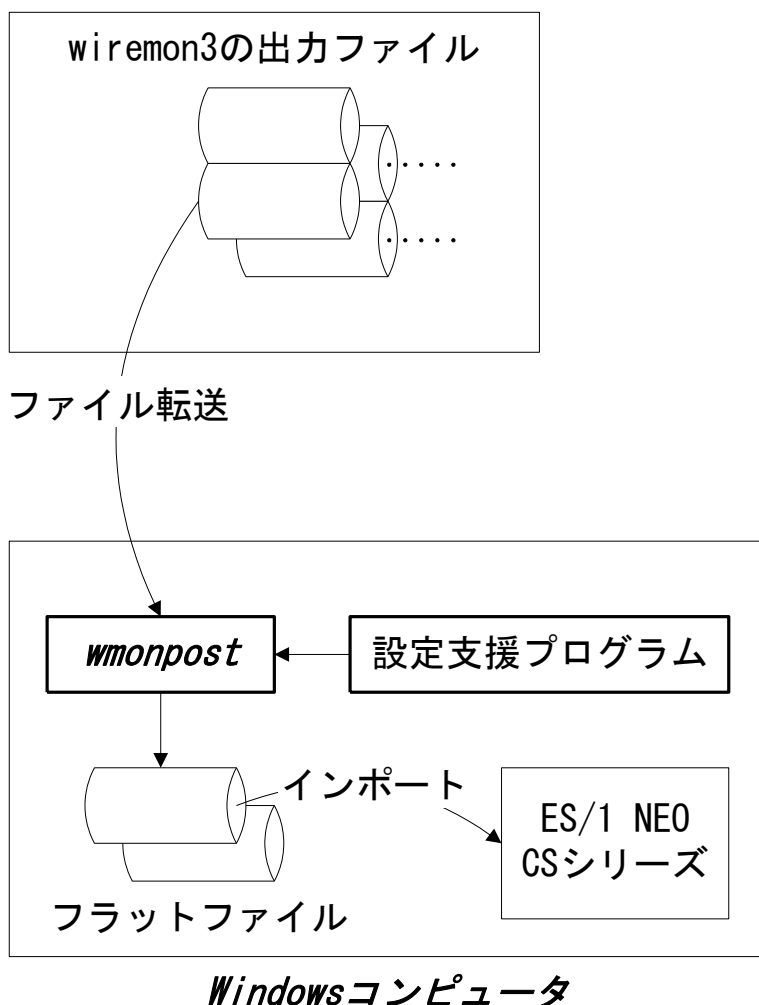
-----Send----- Recv-----

unknown hosts	1024	1024
---------------	------	------

## 2.2. wmonpost

wmonpost は wiremon3 で収集したデータを ES/1 NEO CS シリーズで処理可能なフラットファイル形式に変換するプログラムです。

### パケットモニタサーバ



wmonpost は設定支援プログラムでの記述内容に従って、wiremon3 の出力ファイル(WFxx\_yyyymmdd\_hhmmss.gz)をフラットファイルへ変換します。

wiremon3 が記録したインターバル毎のパケット数/バイト数といった情報(WFxC\_yyyymmdd\_hhmmss.gz)は IP アドレスや Ethernet アドレスにより、特定のサイト/システムの情報へと変換されます。

TCP トレースログ(WFxs\_yyyymmdd\_hhmmss.gz)からはシーケンス番号/確認応答番号やキャプチャ時刻から通信状況を解析し、再転送量やコネクト時間等の情報を計算・集計後、IP アドレスにより特定のサイト/システムの情報へと変換を行います。これらの情報はポート番号や通信相手先別に個別に分類・集計することが可能です。また、拡張設定を行うことで各パケットモニタサーバ(wiremon3)のデータ毎に IP アドレスの組み合わせ単位の TCP セッション情報(拡張 TCP セッション情報)を作成することも可能です。

### 2.2.1. 動作設定

動作設定については別紙マニュアル「CS-Utility iim configuration assistant 使用者の手引き」を参照してください。

### 2.2.2. wmonpost の実行

コマンドラインにて

```
install-path¥wmonpost.exe [-c rtt-timeout-sec] input-directory output-directory [interval-minute [ext  
-interval-minute]]
```

として実行してください。

-c rtt-timeout-sec は RTT 計測対象パケットのタイムアウト時間を 1 以上の整数の秒単位で指定します。省略値は、60(推奨値)です。通常は指定する必要はありません。

install-path は管理用コンピュータの ES/1 NEO CS シリーズインストールフォルダ以下の wmonpost ディレクトリです。

input-directory は wiremon3 の出力ファイル (WFxx\_yyyymmdd\_hhmmss.gz) が存在するディレクトリを、

output-directory は ES/1 NEO CS シリーズのインポート元ディレクトリを、interval-minute はフラットファイルに変換する際の集約単位 (インターバル長) を分単位で指定してください。指定可能な値は、2、3、5、10、15、20、30、60 のいずれかです。また、wiremon3 の監視対象サーバにて athene-acquire によるデータ収集も行っている場合は、その収集インターバルに合わせることをお勧めします。省略値は、15(推奨値)です。

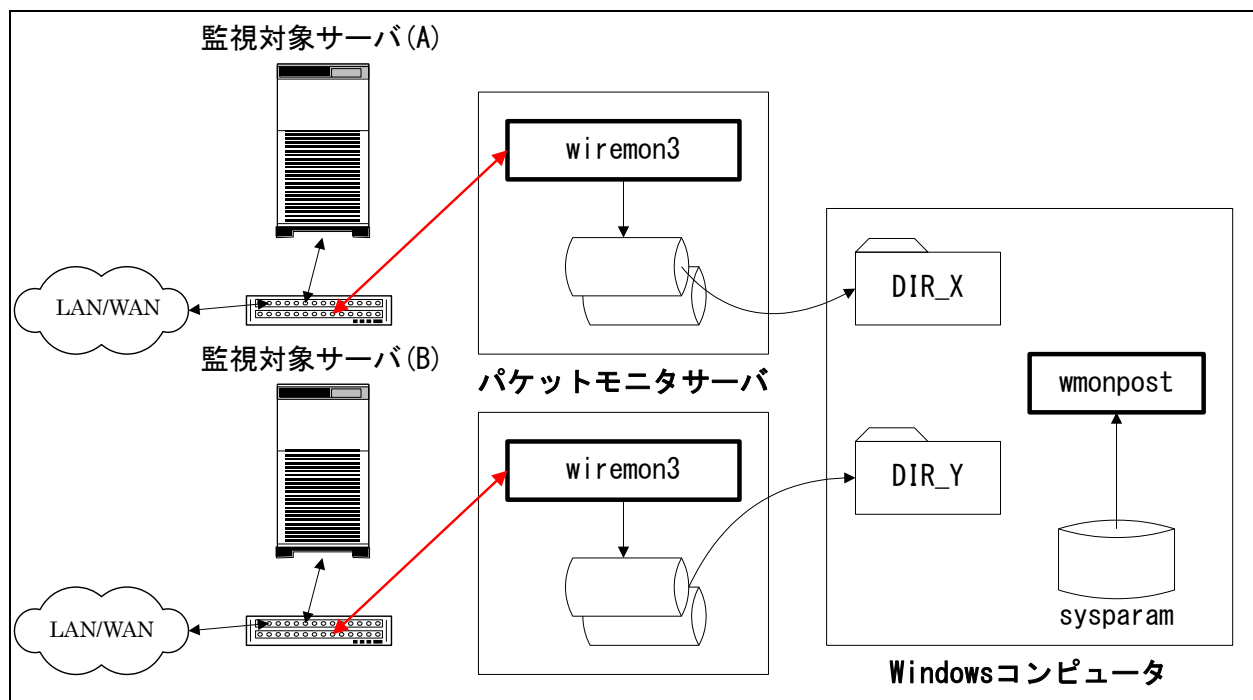
ext-interval-minute は「3.5. 拡張 TCP セッション情報」を作成する場合の集約単位を分単位で指定してください。指定可能な値は、2、3、5、10、15、20、30、60、120、180、240、360、480、720、1440 のいずれかです。この値を指定しなかった場合は拡張 TCP セッション情報は作成しません。この値を指定する場合は interval-minute の指定が必須です。

#### **メモ!**

**wmonpost の実行時間が長くなり運用に支障をきたす場合、wiremon にて片側のみのパケットを収集している可能性があります。rtt-timeout-sec に小さい値を指定することで、片側のみのパケットを破棄し、実行時間を短縮することが可能です。**

### 2.2.3. 複数の wiremon3 からの入力

複数の箇所で wiremon3 によるキャプチャを行った場合の構成のイメージは以下のようになります。



複数の wiremon3 からの入力を扱う際には、ファイル名の重複を避ける為、wmonpost 側で個々に wiremon3 の出力ファイル用のディレクトリを割り当てます。

sysparam には上図監視対象サーバ(A)と監視対象サーバ(B)両方の設定を記述します。

wmonpost の実行には入力元ディレクトリの指定が必要ですので、以下の様に 2 回の wmonpost の実行が必要です。

```
x:¥...¥wmonpost x:¥...¥DIR_X x:¥...¥output nn
```

```
x:¥...¥wmonpost x:¥...¥DIR_Y x:¥...¥output nn
```

### 2.2.4. ロギングの指定

wmonpost の実行ログはテキストファイル(wmonpost ディレクトリ内の“wmonpost.log”)、及び、イベントログに記録することが可能です。

ロギングの指定は wmonpost ディレクトリ内の wmonpost.ini ファイルにて行います。

wmonpost.ini ファイルは[LOG]セクションにより構成されます。

[LOG]セクションについては別紙マニュアル「Log-Utility 使用者の手引き 8. ログ情報出力レベルの設定」を参照してください。

### 2.2.5. フィルタについて

ここでは、wiremon3 が収集したデータを wmonpost がフラットファイルに変換する際に、処理対象とするパケットの条件を記述するフィルタについて説明します。

#### 要求/処理/応答時間の算出について

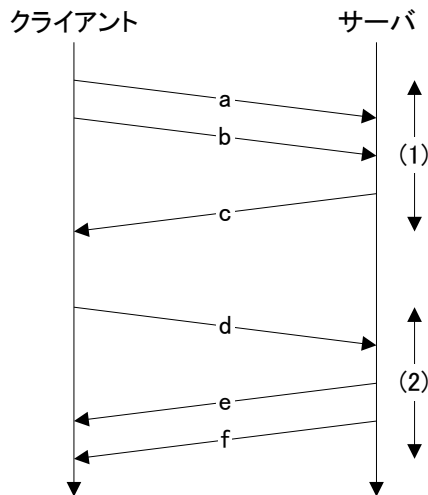
wmonpost は TCP データについて“要求送信時間”、“処理時間”、“応答送信時間”という値を出力します。

これらの値の算出には(アプリケーションの)データを含んだパケット(以降データパケットと表記)を使用しています。

通常の算出方法では、クライアントからの連続した(間にサーバからのデータパケットが存在しない)データパケットの流れを 1 つの要求とし、それに続くサーバからの連続した(間にクライアントからのデータパケットが存在しない)データパケットの流れをそれに対する応答としています。

サーバからのデータパケットに続くクライアントからのデータパケットがあった場合、それは別の要求として認識します。

例えば、下図のようなデータパケットの交換があった場合、(1)、(2)という 2 つの要求/応答の交換があったとしてそれぞれの値を計算します。



クライアントからのデータパケットが複数であった場合は、先頭のパケットの検出時刻と最後のパケットの検出時刻の差を“要求送信時間”とします（上図の a-b）。

サーバからのデータパケットからは同様に“応答送信時間”を算出します(上図の e-f)。

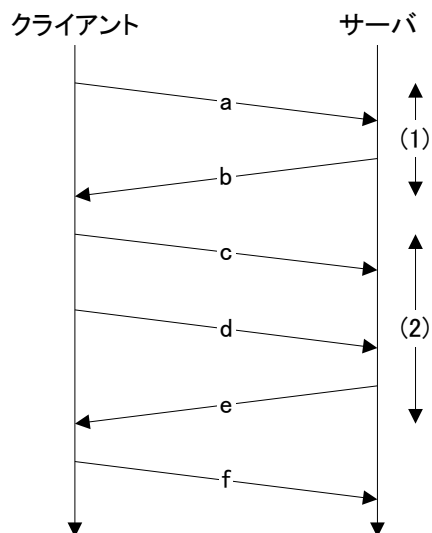
いずれの場合もデータパケットが 1 つであった場合は時間を 0 とします(上図の c、d)。



また、クライアントからの最後のデータパケットとサーバからの最初のデータパケットの検出時刻の差を“処理時間”とします(上図の b-c、d-e)。

ただし、アプリケーションによっては上記の算出方法が不適切な場合があります。

例えば、サーバからの応答を受信後にクライアントが確認の為にデータパケットを送信するようなアプリケーションにおいて、以下の図のようなパケットの交換が行われたとします。



上図における c と f はそれぞれ b と e に対する確認のデータパケットとします。

このような場合は(1)と(2)という2つの要求/応答の交換があったものとして計算を行いますが、通常の計算方法では c と d の間にサーバからのデータパケットが存在しない為、c-d を(2)における“要求送信時間”とします。c は確認の為にデータパケットなので本来は d だけを要求のパケットとして認識し“要求送信時間”を 0 とする必要があります。

フィルタは上記の例に挙げたような、単純な要求/応答の交換でない型のアプリケーションにおいて通常の計算方法に変更を加える為の定義です。

## フィルタの記述

フィルタは wmonpost インストールディレクトリ内の wmonflt.txt ファイルに記述(定義)します。  
フィルタファイル中では英大文字小文字の区別はありません。

### (1)ステートメント

フィルタの定義は 1 つ以上のステートメントの連続からなります。

ステートメント …繰り返し
------------------

ステートメントには PASS、DROP、HEAD の 3 つの種類が存在し、それぞれ以下のような形式で記述します。

PASS{条件式}

DROP{条件式}

HEAD{条件式}

フィルタが定義されている場合、それぞれのデータパケットについてステートメントの条件式の判定を行い、条件式が真の時に対応する動作を行います。

ステートメントを複数定義した場合にはステートメントの条件判定は記述順(上から下)に行われ、いずれかのステートメント(条件式)がマッチした時点で判定を終了し、以降のステートメントの判定は行いません。

また、いずれのステートメントにもマッチしなかったデータパケットについては通常の計算方法が適用されます。それぞれのステートメントの意味(動作)は以下のようになります。

PASS	パケットを各時間の算出に使用します
DROP	パケットを各時間の算出に使用しません
HEAD	パケットを要求(応答)の開始とします

PASS ステートメントは特定の条件に当てはまるパケットについて通常の計算方法を適用したい場合に使用します。例えばポート 8080 番に関するパケットについては通常の計算方法を適用したい場合は、

PASS {curr.support == 8080 or curr.dport == 8080}

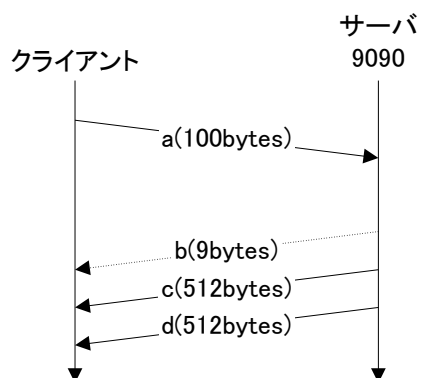
と記述します(条件式の意味については後の記述を参照してください)。

DROP ステートメントは特定の条件に当てはまるパケットについて通常の計算方法の適用から除外したい場合に使用します。例えばポート番号 9090 を使用しているサーバにおいてペイロードのサイズが 10 バイト未満のパケットについては応答パケットとみなさない場合は、

DROP{curr.support == 9090 or curr.plsz < 10}

と記述します。

例えば以下のようなセッションにおいては b のパケットが除外されます。

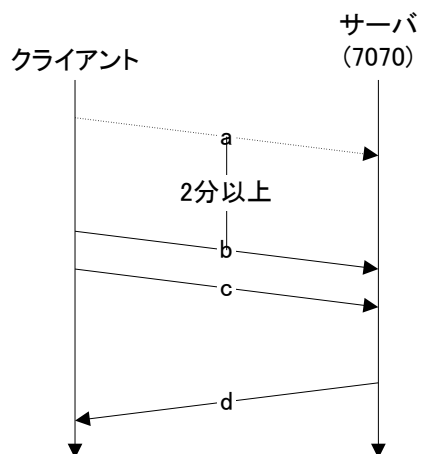


DROP ステートメントが行うことは、「そのパケットはなかったことにする」という動作です。

HEAD ステートメントは特定の条件に当てはまるパケットを要求/応答の先頭としたい場合に使用します。例えばポート番号 7070 を使用しているサーバに対するクライアントからのパケットが 2 分以上の間隔を空けて送信された時に後のパケットを要求の開始とするには、

HEAD {curr.dport == 7070 and stsreqsent() and tdlastreq >= MIN(2)}

と記述します。例えば以下のようなセッションにおいては b のパケットを要求の開始パケットとし、b-c が要求送信時間となります。



HEAD ステートメントが行うことは、「それまでのパケットをなかったことにする」という動作です。

## (2)条件式

条件式には比較条件式、条件判定関数、および論理条件式があります。

比較条件式は変数や定数の関係を比較演算子で記述したものです。

比較演算子の種類と意味は以下の通りです。

==	等しい
!=	等しくない
>	左辺値が右辺値より大きい
<	左辺値が右辺値より小さい
>=	左辺値が右辺値以上である
<=	左辺値が右辺値以下である

比較演算子の左辺値と右辺値に指定可能な値を以下に列記します。

・パケット識別子+"."+属性の形で記述した、パケットの属性を示す変数

パケット識別子の種類と意味は以下の通りです。

curr	現在フィルタで判定しているパケット
prev	現在フィルタで判定しているパケットの直前のパケット
lastreq	直近の要求パケットとして認識されたパケット
lastrsp	直近の応答パケットとして認識されたパケット

属性の種類と意味は以下の通りです。

saddr	パケットの発信元アドレス
daddr	パケットの宛て先アドレス
sport	パケットの発信元ポート番号
dport	パケットの宛て先ポート番号
plsz	パケットのペイロード(データ)サイズ

例えば、curr.plsz は現在判定を行っているパケットのペイロード(データ)のサイズを示します。

- ・0 以上の整数の定数 (ex.0, 10, 64)
- ・IP アドレスの定数 (ex.172.1.10.1, 10.3.2.10)
- ・以下の 3 つの変数

tdprev	現在フィルタで判定しているパケットと直前のパケットとの時間間隔
tdlastreq	現在フィルタで判定しているパケットと直近の要求パケットとして認識されたパケットの時間間隔
tdlastrsp	現在フィルタで判定しているパケットと直近の応答パケットとして認識されたパケットの時間間隔

(\*)時間間隔の単位はマイクロ秒です

・時間(間隔)をあらわす以下の 3 つの定数式

MSEC(n)	n ミリ秒に相当するマイクロ秒を示します
SEC(n)	n 秒に相当するマイクロ秒を示します
MIN(n)	n 分に相当するマイクロ秒を示します

条件判定関数にはセッションの状態が現在どのような状況になっているかを判定する以下の関数があります。

stsreqsent()	クライアントからの要求パケットが送信されている状態
stsrpsent()	サーバからの応答パケットが送信されている状態
synprev()	syn パケットの交換がされた直後の状態

論理条件式には以下の 3 つがあります。

expr AND expr	左右両方の条件式が真だった時に真
expr OR expr	左右どちらかの条件式が真だった時に真
NOT expr	条件式が偽だった時に真

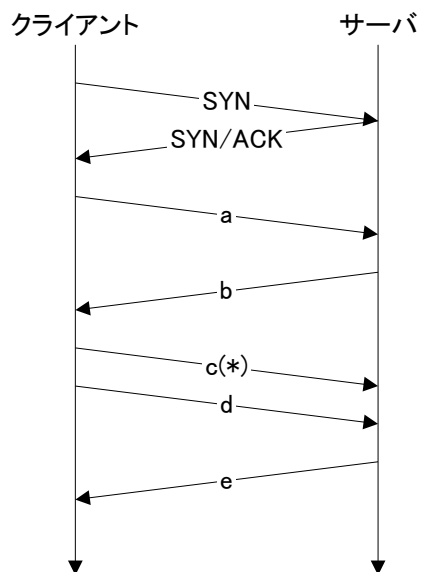
AND と OR を組み合わせた論理式においては AND の方が優先的に結合されます。

例えば、"a OR b AND c" という論理式は、「a あるいは、b かつ c」として解釈されます。ただし括弧を用いてこの優先順位を変更することが可能です。"(a OR b) AND c" という論理式は、「a または b、かつ c」として解釈されます。

### (3) 変数の参照について

以下のようなパケット交換においてフィルタを適用した場合の動作を説明します。

(ACK のみのパケットは省いてあるものとし、また、c のパケットのみフィルタで DROP と判定されたものとします。)



フィルタは各データパケット a から e について適用されます。以下に各パケットのフィルタ適用時における prev/lastreq/lastrsp を使用した変数が参照するパケット、および、各条件判定関数の真偽値を示します。

適用パケット	prev	lastreq	lastrsp	synprev()	stsreqsent()	stsrspsent()
a				真	偽	偽
b	a	a		偽	真	偽
c	b	a	b	偽	偽	真
d	c	a	b	偽	偽	真
e	d	d	b	偽	真	偽

prev は常に直前のパケットを参照しますが、フィルタで DROP と判定されたパケットを lastreq/lastrsp が参照することはありません。また、DROP と判定されたパケットは条件判定関数の結果に影響を与えません。

## 第3章 取得可能データ項目

ES/1 NEO CS シリーズのクエリ定義機能で取り扱うことが可能なデータ項目を以下に示します。

### 3.1. 全体でのパケット記録情報

クエリ定義機能での表（テーブル）名：NETSYS

列（フィールド）名	説明
INTVL	集約インターバル長
RINTVL	実インターバル長
CAPT	キャプチャパケット数
DROP	ドロップパケット数

### 3.2. 各プロトコル層におけるパケット数/バイト数の情報

クエリ定義機能での表（テーブル）名：NETIO

列（フィールド）名	説明
INTVL	集約インターバル長
RINTVL	実インターバル長
TYPE	プロトコルタイプ(*1)
ADDR	IPv4 アドレス、又は、Ethernet アドレス(プロトコルタイプが Ethernet の場合)
ETCNO	プロトコルタイプに応じた番号(*2)
SRCDEST	発信元 or 宛先 識別(*3)
FLG	プロトコルタイプに応じたフラグ(*4)
SNDCNT	送信パケット数
SNDBYTE	送信バイト数(*5)
RCVCNT	受信パケット数
RCVBYTE	受信バイト数(*5)

(\*1)TYPE フィールドは以下の値のいずれかをとりま。

値	意味
ETH	IPv4 以外の Ethernet フレームの情報
IP4	ICMPv4/UDP/TCP 以外の IPv4 パケットの情報
CM4	ICMPv4 の情報
UDP	UDP の情報
TCP	TCP の情報

(\*2)ETCNO フィールドは、TYPE フィールドの値に応じて以下の値をとります。

TYPE フィールドの値	ETCNO フィールドの値
ETH	Ethernet フレームタイプ(0x + 16 進表現)
IP4	IPv4 プロトコルタイプ(10 進表現)
CM4	ICMPv4 メッセージタイプ + コード(16 進表現)
UDP	UDP ポート番号
TCP	TCP ポート番号

(\*3)SRCDEST フィールドは TYPE フィールドが UDP/TCP の時のみ以下の値をとります。

値	意味
S	ホスト側のポート番号が ETCNO フィールドの値であったパケット
D	相手側のポート番号が ETCNO フィールドの値であったパケット

(\*4)FLG フィールドは TYPE フィールドの値に応じて以下の値をとります。  
(TYPE フィールドの値が CM4 の場合はこのフィールドに値は入りません。)

TYPE フィールドの値	FLG フィールドの値	意味
ETH	U	ユニキャスト
	M	マルチキャスト
	B	ブロードキャスト
IP4 又は UDP	U	ユニキャスト
	M	マルチキャスト
	L	リミテッドブロードキャスト(255.255.255.255 宛)
	N	リミテッドブロードキャスト以外のブロードキャスト
TCP	S	SYN セグメント(ACK を含まない)
	C	SYN/ACK セグメント
	A	データを含まない ACK のみのセグメント
	F	FIN フラグの付いたセグメント
	R	RST フラグの付いたセグメント
	D	上記以外のデータを含んだセグメント
	T	すべてのセグメント

(\*5)TYPE フィールドが TCP で FLG フィールドが T 以外の場合はこのフィールドに値はセットされません。

### 3.3. TCP セッション情報 (TCP トレースログ解析情報)

クエリ定義機能での表 (テーブル) 名 : TCPSESS

列 (フィールド) 名	説明
INTVL	集約インターバル長
RINTVL	実インターバル長
ADDR	IPv4 アドレス(設定画面にて入力したサーバのアドレス)
PORTNO	TCP ポート番号(設定画面にて入力したポート条件の名称、または、ポート番号)
SRCDST	発信元 or 宛先 識別(3.2. *3と同様)
PEER	相手先(設定画面にて入力したゾーンの名称)
SNDSEG	送信セグメント数
RCVSEG	受信 "
SNDBYTE	送信バイト数
RCVBYTE	受信 "
ACCCNT	SYN セグメントに対する SYN/ACK セグメントを検出した回数
ACCAVGTM	SYN と SYN/ACK セグメント間の平均時間(msec)
ACCMAXTM	" 最大時間(msec)
ESTCNT	セッションを確立(3ウェイハンドシェイクを完了)したコネクション数
ESTAVGTM	セッション確立に要した平均時間
ESTMAXTM	" 最大時間
REFUSED	SYN セグメントに対する RST セグメントの検出回数
SYNCNT	SYN セグメント数
SYNACKCNT	SYN/ACK セグメント数
SNDMSGCNT	送信データセグメント数
RCVMSGCNT	受信 "
SNDMSGBYTE	送信データバイト数
RCVMSGBYTE	受信 "
RESYNCNT	再送 SYN セグメント数
RESYNACKCNT	再送 SYN/ACK セグメント数
RESNDCNT	再送信データセグメント数
RERCVCNT	再受信 "
RESNDBYTE	再送信データバイト数
RERCVBYTE	再受信 "
DELSND	遅延(順番が入れ替わって)送信されたセグメント数
DELRCV	" 受信されたセグメント数
TDACKRCV	重複 ACK(3重)受信回数
TDACKSND	" 送信回数
RESPCNT	要求応答交換の回数
REQAVGTM	平均要求送信時間
REQMAXTM	最大 "
PROCAVGTM	平均処理時間
PROCMAXTM	最大 "
RSPAVGTM	平均応答送信時間
RSPMAXTM	最大 "
REQAVGRTT	クライアントのデータとサーバの ACK 間の平均時間
REQRTTCNT	上記値のサンプリング回数
REQMAXRTT	クライアントのデータとサーバの ACK 間の最大時間
RSPAVGRTT	サーバのデータとクライアントの ACK 間の平均時間
RSPRTTCNT	上記値のサンプリング回数
RSPMAXRTT	サーバのデータとクライアントの ACK 間の最大時間
WSZEROSND	Window サイズゼロ送信回数
WSZERORCV	Window サイズゼロ受信回数



列（フィールド）名	説明
REQDROPRTTCNT	要求確認破棄パケット数（*1）
RSPDROPRTTCNT	応答確認破棄パケット数（*1）

（\*1）

回線遅延時間（RTT）は、データがネットワークで一往復するのに要する時間です。Packet Monitor では、送信したデータに対する確認応答(ACK)との時間差を繰り返し計測し、平均を算出して回線遅延時間を求めています。

一往復するのに要する時間がタイムアウト値（デフォルト 60 秒）以上の場合、回線遅延時間の算出からそのパケット情報を破棄して回線遅延時間を算出します。このタイムアウト値は回線遅延時間の算出にのみ影響し、その他の項目には影響ありません。

### 3.4. TCP レスpons時間分布

クエリ定義機能での表（テーブル）名：TCPRESPDIST

列（フィールド）名	説明
RINTVL	実インターバル長
RECID	レコード ID
TOTAL	総件数
LT100MS	件数（0.1 秒未満）
GE100LT200MS	件数（ $0.1 \leq r < 0.2$ ）
GE200LT300MS	件数（ $0.2 \leq r < 0.3$ ）
GE300LT400MS	件数（ $0.3 \leq r < 0.4$ ）
GE400LT500MS	件数（ $0.4 \leq r < 0.5$ ）
GE500LT600MS	件数（ $0.5 \leq r < 0.6$ ）
GE600LT700MS	件数（ $0.6 \leq r < 0.7$ ）
GE700LT800MS	件数（ $0.7 \leq r < 0.8$ ）
GE800LT900MS	件数（ $0.8 \leq r < 0.9$ ）
GE900LT1000MS	件数（ $0.9 \leq r < 1.0$ ）
GE1000MS	件数（1 秒以上）(*1)
GE1LT2SEC	件数（ $1 \leq r < 2$ ）
GE2LT3SEC	件数（ $2 \leq r < 3$ ）
GE3LT4SEC	件数（ $3 \leq r < 4$ ）
GE4LT5SEC	件数（ $4 \leq r < 5$ ）
GE5LT6SEC	件数（ $5 \leq r < 6$ ）
GE6LT7SEC	件数（ $6 \leq r < 7$ ）
GE7LT8SEC	件数（ $7 \leq r < 8$ ）
GE8LT9SEC	件数（ $8 \leq r < 9$ ）
GE9LT10SEC	件数（ $9 \leq r < 10$ ）
GE10SEC	件数（10 秒以上）(*2)
GE10LT15SEC	件数（ $10 \leq r < 15$ ）
GE15LT20SEC	件数（ $15 \leq r < 20$ ）
GE20LT25SEC	件数（ $20 \leq r < 25$ ）
GE25LT30SEC	件数（ $25 \leq r < 30$ ）
GE30SEC	件数（30 秒以上）

(\*1)

1 秒以上の分布件数を示すフィールド(GE1LT2SEC、GE2LT3SEC、GE3LT4SEC、GE4LT5SEC、GE5LT6SEC、GE6LT7SEC、GE7LT8SEC、GE8LT9SEC、GE9LT10SEC、GE10LT15SEC、GE15LT20SEC、GE20LT25SEC、GE25LT30SEC、GE30SEC)の合計に等しい

(\*2)

10 秒以上の分布件数を示すフィールド(GE10LT15SEC、GE15LT20SEC、GE20LT25SEC、GE25LT30SEC、GE30SEC)の合計に等しい

### 3.5. 拡張 TCP セッション情報 (Peer To Peer TCP トレースログ解析情報)

クエリ定義機能での表 (テーブル) 名 : TCPPTOP

列 (フィールド) 名	説明
RINTVL	実インターバル長
ADDRLL	IPv4 アドレス(Low)(*1)
ADDRHH	IPv4 アドレス(High)(*1)
SNDSGL	送信セグメント数(Low)
SNDSGH	送信セグメント数(High)
SNDBYTEL	送信バイト数(Low)
SNDBYTEH	送信バイト数(High)
RESNDSGL	再送信セグメント数(Low)
RESNDSGH	再送信セグメント数(High)
RESNDBYTEL	再送信データバイト数(Low)
RERCVBYTEH	再送信データバイト数(High)
TDACKSNDL	3 重重複 ACK 送信回数(Low)
TDACKSNDH	3 重重複 ACK 送信回数(High)
WSZEROSNDL	Window サイズゼロ送信回数(Low)
WSZEROSNDH	Window サイズゼロ送信回数(High)
RTTCNTL	RTT サンプリング回数(Low)
RTTAVGL	平均 RTT(ms)(Low)(*2)
RTTMAXL	最大 RTT(ms)(Low)(*2)
RTTCNTH	RTT サンプリング回数(High)
RTTAVGH	平均 RTT(ms)(High)(*3)
RTTMAXH	最大 RTT(ms)(High)(*3)
DROPRTTCNTL	RTT 破棄パケット数(Low) (*4)
DROPRTTCNTH	RTT 破棄パケット数(High) (*4)
TTLAVGL	平均 TTL(Low)
TTLMAXL	最大 TTL(Low)
TTLMINL	最小 TTL(Low)
TTLAVGH	平均 TTL(High)
TTLMAXH	最大 TTL(High)
TTLMINH	最小 TTL(High)

(\*1)

Low/High は通信の両端のホストの IP アドレスを比較し、小さい方を Low、大きい方を High とします。

(例)172.16.1.1 と 172.16.1.2 の場合、172.16.1.1 が Low、172.16.1.2 が High になります。

(\*2)

Low の送信したデータとそれに対する High からの ACK の時間差を RTT として計算しています。

(\*3)

High の送信したデータとそれに対する Low からの ACK の時間差を RTT として計算しています。

(\*4)

回線遅延時間 (RTT) は、データがネットワークで一往復するのに要する時間です。Packet Monitor では、送信したデータに対する確認応答(ACK)との時間差を繰り返し計測し、平均を算出して回線遅延時間を求めています。

一往復するのに要する時間がタイムアウト値 (デフォルト 60 秒) 以上の場合、回線遅延時間の算出からそのパケット情報を破棄して回線遅延時間を算出します。このタイムアウト値は回線遅延時間の算出にのみ影響し、その他の項目には影響ありません。

### 3.6. オーバーフローパケット情報

クエリ定義機能での表（テーブル）名：NETSYS\_TCP

列（フィールド）名	説明
INTVL	集約インターバル長
RINTVL	実インターバル長
OVERFLOW	オーバーフローパケット数(*1)

(\*1)

wiremon3 の記憶領域不足により破棄した TCP パケット数を示します。