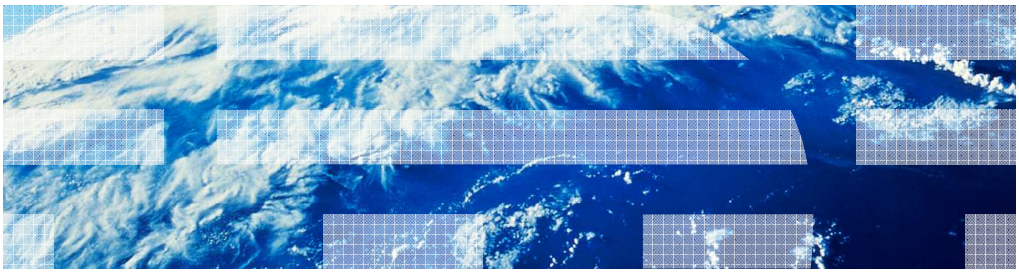
z/OS Communications Server
System management and monitoring



This presentation describes the new functions in z/OS V1R12 Communications Server that are contained in the system management and monitoring theme. System management covers any function that is needed to manage the networking environment on z/OS – commands, network management interfaces, SNMP support, SMF records, and so on.

System management and monitoring

- ✓Enhancements to TCP/IP callable NMI (EZBNMIFR)
- ✓SMF records for CSSMTP
- ✓SMF records for Sysplex events
- ✓Data trace records for socket data flow start and end
- ✓Packet trace filtering for encapsulated packets
- ✓Enhancements to SNMP manager API
- ✓Configuration Assistant

z/OS V1R12 Communications Server delivers a significant set of new network management interfaces (NMI) to give network management software from Tivoli and other vendors better insight into z/OS Communications Server.

Trace improvements were made to help with data flow diagnosis and packet trace filtering in a sysplex environment.

The SNMP manager API has been enhanced to be more compatible with other managers.

The configuration assistant has several usability enhancements.

Network management information sources

- **Network management and monitoring interfaces**
  - ✓TCP/IP callable network management interface (NMI), EZBNMIFR
  - ✓Real-time SMF data NMI (NETMONITOR SMFSERVICE), SYSTCPSM
  - ✓Real-time packet and data trace NMI, SYSTCPDA
  - ✓Packet and data trace formatting NMI, EZBCTAPI
  - – Local IPSEC and NSS NMIs
  - – SNA network monitoring NMI
- **IBM Health Checker**
- **SNMP**
- **Netstat reports**

There are several network management interfaces (NMIs) that relate to the Communications Server. In z/OS V1R12 Communications Server, enhancements include the TCP/IP callable NMI, the real-time SMF data NMI, and the packet and data trace formatting NMIs. A network management application can use the TCP/IP callable NMI, EZBNMIFR, as a high-speed, low-overhead callable programming interface to access data related to the TCP/IP stack. It can use the real-time SMF data NMI to collect many events without the need for SMF record capturing and further independent processing. And it can use the packet and data trace NMIs to programmatically obtain trace records and format them.

The IBM Health Checker is another way to monitor and report the health of IBM components, including the TCP/IP stack.

Some types of information are currently only available using SNMP or Netstat displays. SNMP can be difficult to configure and is slower than the NMIs described above. Various Netstat reports can provide information, but the report output can change every release as more information is added to each report. New information is obtained only after the application polls for new information updates. Many of the NMI interfaces and health checker functions are event driven.

New information available to applications and administrators

- **Applications want access to TCP/IP information and automatic notification**
- **Strategic interface and device attributes and statistics**
  – Four new NMI requests
- **Sysplex events – six new SMF records**
- **CSSMTP events – five new SMF records**
- **z/OS OMPROUTE number of indirect routes**
  – IBM Health Checker – two new checks

Network management applications want to use the NMI instead of using SNMP or polling reports on certain intervals to get needed information. Some statistics gathered from automatic notifications can indicate potential problems. By obtaining the statistics, the management applications can automatically alert customers to the potential problems.

Four areas have improved their network management interface in z/OS V1R12.

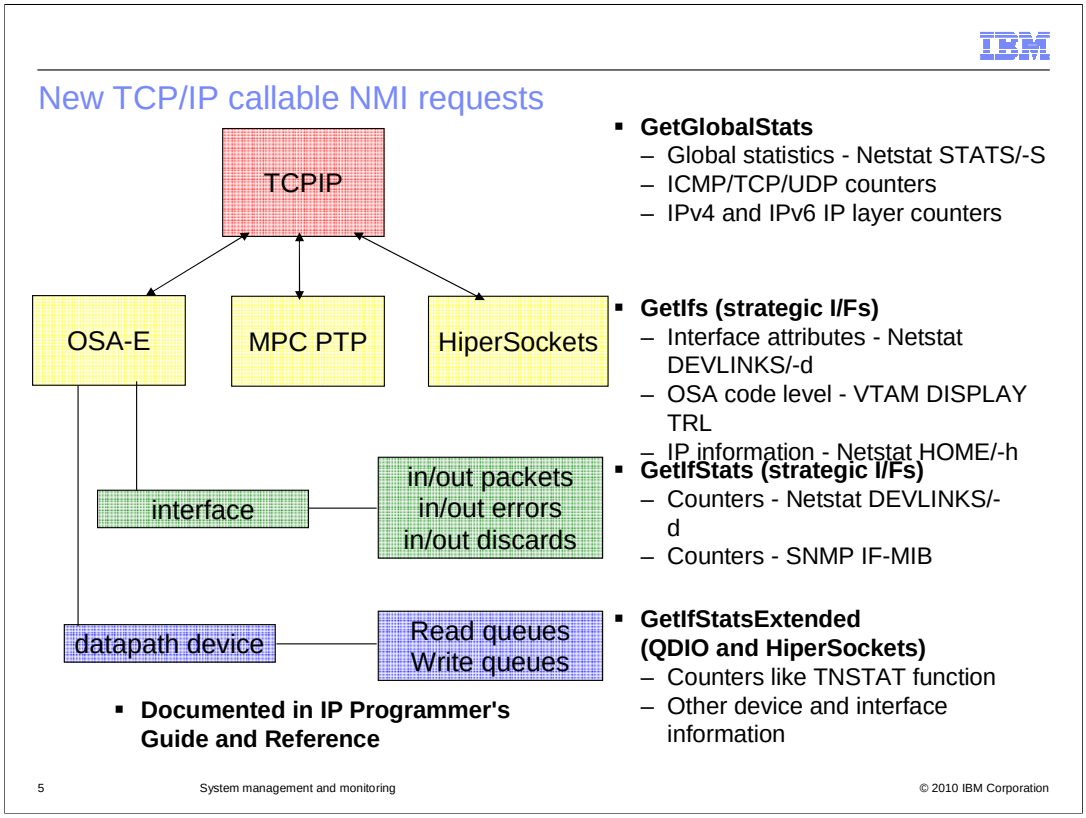
(1) Interface and device attributes and statistics for strategic interfaces have been improved by creating four new NMI requests. They combine the data from several different reports and displays and add additional information that was not available before.

(2) Sysplex monitoring has been improved by creating six new SMF subtype 119 event records that mirror the data provided by six existing SNMP records. Instead of using SNMP, the real-time SMF data NMI provides the data when the event occurs.

(3) CSSMTP monitoring was not part of the initial release of CSSMTP in z/OS V1R11. Five new SMF subtype 119 event records were created to provide real-time information about the CSSMTP client using the real-time SMF data NMI.

(4) The number of IPv4 and number of IPv6 indirect routes created by OMPROUTE are being monitored by the IBM Health Checker. The Health checker issues messages whenever the specified maximum number of routes is surpassed and provides data at specified intervals whether the maximum number has been reached or not.

See the "Network management interfaces" chapter of *z/OS V1R12 Communications Server: IP Programmer's Guide and Reference* for detailed information about these new NMI requests.

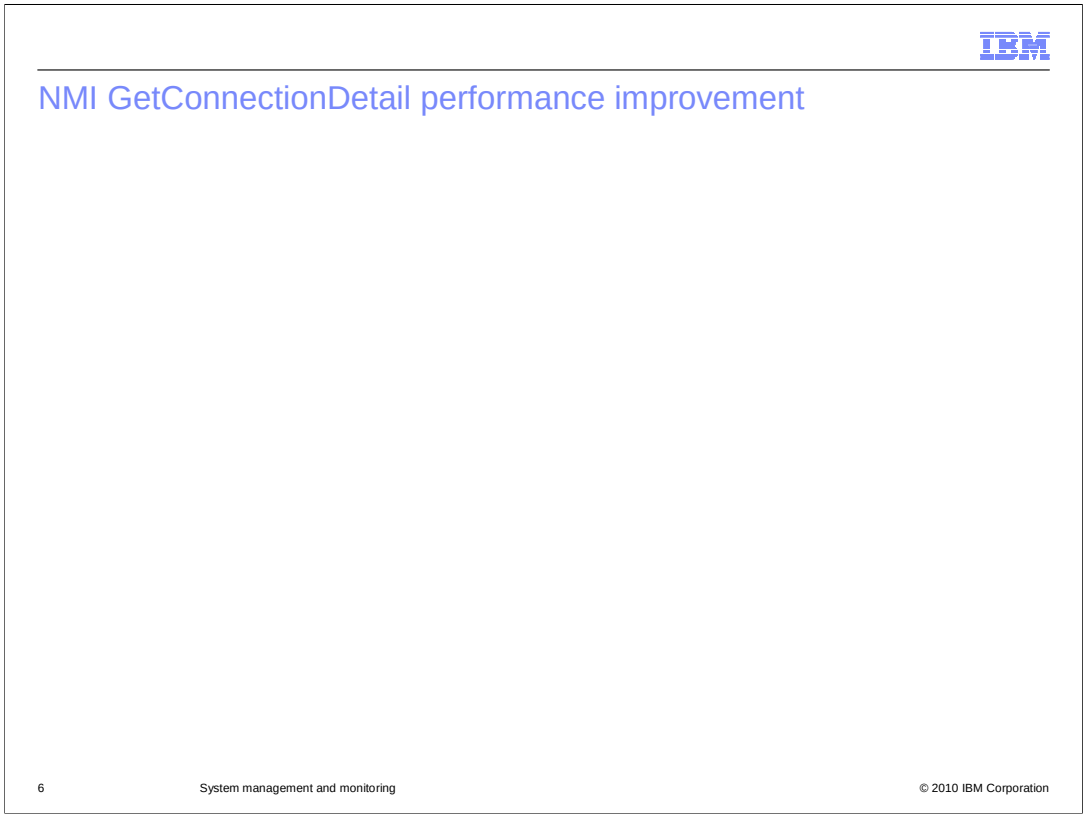


The GetGlobalStats request provides TCP/IP stack global statistics, similar to those on the Netstat STATS/-S report. IP, ICMP, TCP and UDP counters are provided. IP and ICMP counters for both IPv4 and IPv6 are provided.

The TCP/IP stack's strategic interface types are: OSA-Express QDIO ethernet, MPCPTP, HiperSockets, Virtual IP address (VIPA) and Loopback. For strategic interface types (except for dynamic VIPA), the new GetIfs request provides attribute information similar to the Netstat DEVLINKS/-d report and interface IP address information similar to the Netstat HOME/-h report. It also provides the firmware code level for OSA-Express QDIO features. For the non-strategic interface types, only the interface name, type, status, and timestamp of last status change is provided.

For strategic interfaces (except for VIPA), the new GetIfStats request provides interface name, status, and interface counters similar to those on the Netstat DEVLINKS/-d report. It also provides SNMP IF-MIB MIB module data, from RFC 2233.

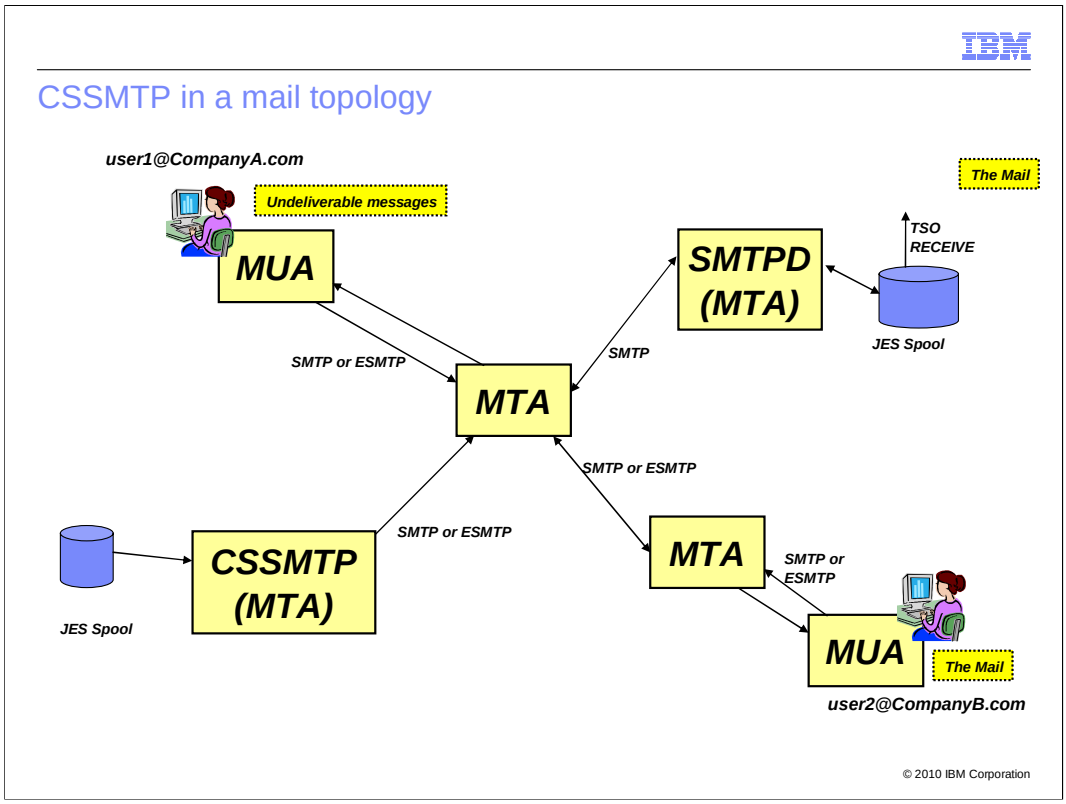
The new GetIfStatsExtended request provides DLC layer read and write queue counters for each datapath device being used by an active OSA-Express QDIO Ethernet or HiperSockets interface. The counters are similar to those provided by the TNSTAT display and SMF type 50 record. Due to performance considerations, the first GetIfStatsExtended request starts counter maintenance for all active interfaces. The read and write queue counters might be zero in the response for the first request. This request provides datapath device and interface information such as the read and write control device numbers and the TRLE name. These values are common to all interfaces defined by the same TRLE. The interface information provides the interface name and status for all interfaces sharing the datapath device.



The GetConnectionDetail NMI request provides detailed information about all active TCP connections. Each TCP/IP connection can be uniquely identified by four attributes of the connection: local IP address, local port, remote IP address, and remote port. These four attributes are called the 4-tuple of the connection.

When there are a large number of TCP connections, the NMI request can require a large amount of processing time to retrieve all the connection information. Filters can be specified for the request which contain the 4-tuple information for specific connections. But, even in this case, the NMI request still searches through all the TCP connections to satisfy the request.

The GetConnectionDetail NMI request has been improved to provide better performance when all the filters specified for the request contain the 4-tuple values of specific connections. The request directly retrieves the information for the connection identified by the 4-tuple values instead of searching through all the active TCP connections. The improvement not only reduces processor utilization but also improves the response time for the request. A maximum of four filters can be specified with each request invocation, to retrieve information for up to four specific connections. All of the filters must contain 4-tuple values in order to realize the performance improvement. For each of the four filters, additional filter values can be specified along with the 4-tuple value.



Communications Server Simple Mail Transport Protocol (CSSMTP) acts as a Mail Transfer Agent that reads JES spool files and separates the spool files into a series of mail messages. The individual mail messages are transferred or forwarded to mail target servers. The format and style of the mail messages are defined by RFC 2821 and 2822.

This diagram shows CSSMTP in a mail network, where it is just one source of messages added to a mail network. Mail User Agents (MUA) send and receive mail that is sent using the SMTP or ESMTP protocol to Mail Transfer Agents (MTA). SMTPD is a Mail Transfer Agent that can send mail to TSO users.

CSSMTP enhancements

- **New SMF records written to SMF and SYSTCPSM real time applications**
  - (48) CONFIG – A record written at initialization and each configuration refresh
  - (49) CONNECT – A record written at the end of each client connection
  - (50) MAIL – A record written at the completion of each mail message
    One record per mail message. Volume and size can be large.
  - (51) SPOOL – A record written at the completion of each spool file
  - (52) STATS – A record written at intervals and at termination with global statistics
- **APPLDATA update each CSSMTP client connection**
- **Maximum RETRYLIMIT changed from two hours to five days**

There were several requirements requesting accounting, statistics, and performance data regarding the mail message processing. Accounting data includes a description of the spool file with the job name, job identifier, the size of the spool file and other fields. For each mail message, the data includes the source of the mail message (which spool file), the mail-from name, the recipients of the mail message and subject. Statistics data includes time stamps about spool file and mail message processing, the size of the spool file, and each mail message. Performance data includes the traffic over each connection and each target server. CSSMTP provides SMF records about mail processing. There are five new subtype records for the type 119 records. The SMF119 configuration statement is added to the CSSMTP configuration file to define which SMF 119 subtype records are written to SMF. CONFIG contains the configuration data from initialization and after each MODIFY REFRESH command. CONNECT contains statistics about each connection to a target server that transfers mail messages. MAIL contains identification and statistics about each mail message. SPOOL contains identification and statistics about each spool file. STATS contains global and health check statistics at each SMF interval and at termination. They are written at the intervals defined by the SMF INTVAL and SYNCVAL parameters - the same interval used to write other interval SMF119 records.

By providing APPLDATA for the server connections, the state of the CSSMTP connections can be monitored. CSSMTP adds APPLDATA to its connections with the mail servers. The application connection data can be displayed using NETSTAT or acquired from the SMF connection termination record.

The current RETRYLIMIT value is two hours. Some customers requested a longer interval. The default minimum and maximum values of the COUNT and INTERVAL parameters stay the same. The range of values for COUNT is 0 to 120. The range of values for INTERVAL is 0 to 120. The maximum value of the product of COUNT and INTERVAL is raised from 120 (two hours) to 7200 (five days).

Configuration statements for CSSMTP SMF event records

- **NETMONITOR SMFService profile statement for real-time SMF NMI**
  - New options
    - CSSMTP (default) or NOCSSMTP
    - NOCSMAIL (default) or CSMAIL
- **CSSMTP writes to only one TCP/IP stack**
  (The stack must have the CSSMTP or CSMAIL option active)
  - The stack identified with -p start option, or
  - The controlling stack for subtype 49 connections, or
  - The first stack with the CSSMTP or CSMAIL option active

The NETMONITOR SMFSERVICE statement in the TCP/IP profile defines which SMF records are written to the real-time SMF NMI. The statement was updated to add the CSSMTP|NOCSSMTP and CSMAIL|NOCSMAIL options. The CSSMTP|NOCSSMTP option controls writing the subtype 48, 49, 51 and 52 records to the SYSTCPSM NMI. The CSMAIL|NOCSMAIL option controls writing the subtype 50 (mail) records to the SYSTCPSM NMI.

Note the defaults for these options are CSSMTP and NOCSMAIL if just NETMONITOR SMFSERVICE is coded. NOCSMAIL is the default because each mail message creates one SMF record. Installations that create many mail messages must allow for the increased number of records to be saved and analyzed The mail message SMF record becomes larger with an increasing number of recipients. If there are too many recipients, the record is truncated. There is a flag in the SMF record that indicates truncation has occurred. The SYSTCPSM real time application and SMF reporting programs must be updated before creating the CSSMTP SMF records.

CSSMTP writes the SMF record to only one TCP/IP stack:

If the -p start option was used to name a TCP/IP stack then that stack is used if the corresponding CSMAIL or CSSMTP option was set.

Otherwise these criteria are used for each subtype.
For subtype 49 records (connection records), the stack that is controlling the connection is used if the CSSMTP option was set for that stack.

For subtype 48, 51 and 52 records (configuration, spool and statistic records), the first active stack with CSSMTP option set is used.

For subtype 50 records (mail records), the first active stack with CSMAIL option set is used.

Display commands

- **Modify CSSMTP,Display,Config to see SMF119 and RetryLimit parameters**

```
SMF119:
 CONFIG              : YES         CONNECT         : YES
 MAIL                : NO          SPOOL           : YES
 STATS               : YES
RETRYLIMIT:
 COUNT               : 12          INTERVAL        : 15
```

- **Netstat CONFIG/-f to see the NETMONITOR options**

```
SMFSRV:    YES
  IPSECURITY: NO   PROFILE: NO   CSSMTP: YES  CSMAIL: YES  DVIPA: NO
```

- **Netstat CONN/-c with APPLD=EZASMTP***

```
EZD0101I NETSTAT CS V1R12 TCPCS 362
 USER ID  CONN     STATE
 CSSMTP   0000002E ESTBLSH
   LOCAL SOCKET:   127.0.0.1..1026
   FOREIGN SOCKET: 127.0.0.1..25
   APPLICATION DATA:  EZASMTPC XYZ      SB
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```


The MODIFY CSSMTP,DISPLAY,CONFIG command displays the values of the SMF119 parameters and shows that the RETRYLIMIT parameters are set for a maximum interval of 180 minutes (12x15) or three hours.

The NETSTAT CONFIG command shows the sub-parameter values of the NETMONITOR SMFSERVICE TCP/IP profile statement parameter. The values of the CSMAIL and CSSMTP parameters are shown.

The NETSTAT CONN command shows the connections to SMTP target servers. The display includes APPLDATA which consists of an application identifier, the external writer name, an SMTP or ESMTP indicator, and secure connection fields. For more information regarding the CSSMTP APPLDATA layout, see appendix F, Application data, in the IP Programmer's Guide and Reference.

Provide programming interface for sysplex events

- **Add SMF 119 records for sysplex events**
  - Written to MVS SMF datasets
  - Added to real-time SMF data NMI
- **6 new SMF 119 record subtypes are defined, comparable to SNMP traps**:
  - ibmMvsDVIPAStatusChange – (32) DVIPA Status Change
  - ibmMvsDVIPARemoved – (33) DVIPA Removed
  - ibmMvsDVIPATargetAdded – (34) DVIPA Target Added
  - ibmMvsDVIPATargetRemoved – (35) DVIPA Target Removed
  - ibmMvsDVIPATargetServerStarted – (36) DVIPA Target Server Started
  - ibmMvsDVIPATargetServerEnded – (37) DVIPA Target Server Ended

z/OS V1R12 Communications Server has added additional SMF 119 records for sysplex distributor events. These records can be written to the MVS SMF data sets, added to the real-time TCP/IP NMI, or both.

Six new SMF 119 records are defined, each corresponding to one of the existing SNMP sysplex distributor traps.

The DVIPA status change and DVIPA removed records contain the DVIPA address, the destination XCF address, and the status, origin, rank (for a backup DVIPA), and move options of the DVIPA. The DVIPA status change record also includes the activation timestamp. The DVIPA status change record is generated when a DVIPA is created or when the DVIPA's status changes. The DVIPA removed record is generated when a DVIPA is deleted.

The DVIPA target added and DVIPA target removed records contain the DVIPA address and type, the destination XCF address and type, and the distributed port number. The DVIPA target added record is generated when a new target stack is identified through a VIPADISTRIBUTE DEFINE statement. The DVIPA target removed record is generated when a stack is removed as a distribution target through a VIPADISTRIBUTE DELETE statement.

The DVIPA target server started and DVIPA target server ended records contain the DVIPA address and type, the destination XCF address and type, the distributed port number and the ready count. The DVIPA target server started record is generated when a server application becomes ready to accept distributed connections. The DVIPA target server ended record is generated when a server application becomes unavailable for distributed connections.

Configuration statements for Sysplex SMF event records

- **SMFCONFIG profile statement**
  - New TYPE119 options
    - NODVIPA (default) or DVIPA
- **NETMONITOR profile statement**
  - New SMFService options
    - DVIPA (default) or NODVIPA

The SMFCONFIG profile statement controls the writing of the six sysplex event records to the MVS SMF data sets. If the keyword DVIPA is specified, the sysplex events are written to the MVS SMF data sets. If NODVIPA is specified, the events are not written to the MVS SMF data sets. NODVIPA is the default.

The NETMONITOR profile statement controls whether the six sysplex events are made available to the real-time SMF NMI. If DVIPA is specified on the NETMONITOR statement, the records are written to the NMI interface. If NODVIPA is specified, the records are not written to the NMI interface. DVIPA is the default.

Netstat CONFIG/-f command example

- **The Netstat CONFIG/-f report reflects the new SMFCONFIG and NETMONITOR parameter settings**

```
  :
  :
SMF Parameters:
Type 118:
  TcpInit:      00   TcpTerm:    02   FTPClient:    03
  TN3270Client: 04   TcpIpStats: 05
Type 119:
  TcpInit:      Yes  TcpTerm:    Yes  FTPClient:    Yes
  TcpIpStats:   Yes  IfStats:    Yes  PortStats:    Yes
  Stack:        Yes  UdpTerm:    Yes  TN3270Client: Yes
  IPSecurity:   No   Profile:    Yes  DVIPA:        Yes
  :
  :
Network Monitor Configuration Information:
PktTrcSrv: Yes  TcpCnnSrv: Yes  MinLifTim: 3  NtaSrv: Yes
SmfSrv:    Yes
  IPSecurity: Yes  Profile: Yes   DVIPA: Yes
  :
  :
```


The Netstat CONFIG/-f report can be used to display the SMFCONFIG and NETMONITOR SMFSERVICE DVIPA settings.

Data trace – Data flow Start and End records

- **Data trace traces socket data through the physical file system (PFS)**
  - No indication when data flow starts and ends
  - Difficult to understand the flow of data on a socket
  - Packet trace indicates data flow start and end but encrypted when using IPSec or AT-TLS
- **Two new records created to trace start and end of data flow**
  - Only supported for TCP and UDP sockets – not RAW
  - Start record is written on the first socket read or write operation
  - End record written when the socket is closed
  - For AT-TLS connections, data trace does not show application data unless the CtraceClearText parameter in the AT-TLS policy is set to On

```
  63 MVS182   DATA     00000005 13:18:52.705189 Data Trace
To Jobname      : USER12                                Full=0
 Tod Clock        : 2009/06/22 13:18:52.705187            Cid: 00000058
 Domain           : AF_Inet          Type: Stream         Protocol: TCP
 State            : API Data Flow Starts         /  API Data Flow Ends
 Segment #        : 0                Flags:  Out /  None
 Source           : 10.81.2.5
 Destination      : 10.81.2.1
 Source Port      : 2000             Dest Port: 2000  Asid: 0041 TCB: 006FF1D8
```


The data trace can be used to trace socket data into and out of the physical file system (PFS). Data trace records can be obtained by formatting the records under IPCS or by an application using the real-time packet and data trace NMI and the packet and data trace formatting NMI.

In release z/OS V1R11 and earlier, data trace records do not provide an indication of the start or end of socket data flow. This makes it difficult for management applications and users to understand the flow of data for a socket. Packet trace records do provide an indication of the start or end of socket data flow. But, when using AT-TLS or IPSec, the information in the packet trace record is encrypted. The encryption makes it impossible to interpret the start and end information.

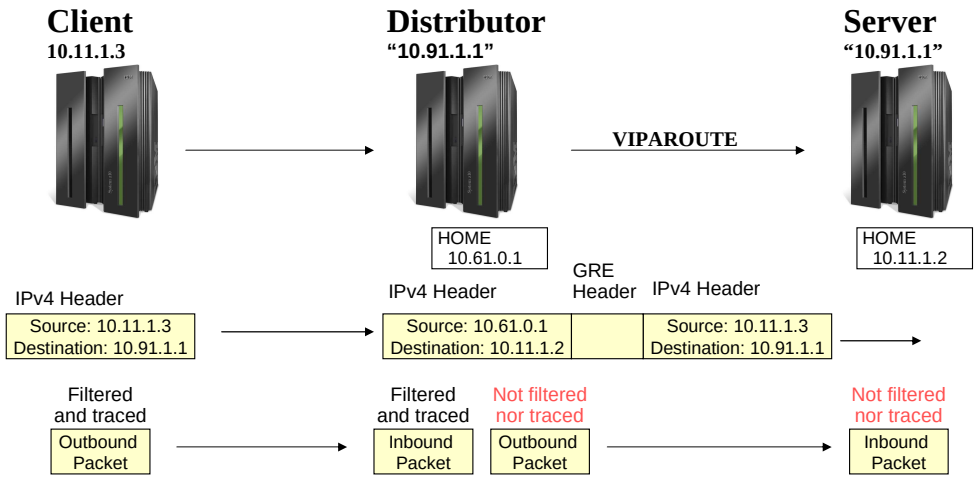
The data trace has been changed to generate two additional records to indicate the start and end of data flow for a socket. This is currently supported only for TCP and UDP packets. The start record is written when the first socket read or write operation is performed and has a state field containing the literal, “API Data Flow Starts”. This record contains all the session information such as source and destination IP address, source and destination port, CID, and so on. But it does not contain the data that is traced by data trace. Then a series of standard data records follow this start record. These records do show data traced between two end points. The CID is the same as that in the start record. Finally, the end record is written when the socket is closed and again shows the state field, with the literal “API Data Flow Ends”.

AT-TLS data trace packets are, by default, not shown. In order to view AT-TLS packet data in the data trace, the AT-TLS policy must be changed. If the CtraceClearText parameter in the AT-TLS policy is set to Off (the default), the data trace does not show application data for AT-TLS connections.

An example of the start and end data trace records are shown on the slide with the only difference being the literals in the state field.

Sysplex distributor with VIPAROUTE packet encapsulation

- **Packets encapsulated when sent to the target using VIPAROUTE**
  - IPv4 packets encapsulated with a *base* GRE header
  - IPv6 packets encapsulated with another IPv6 header



When VIPAROUTE statements are defined to a sysplex distributor to select routes, the sysplex distributor encapsulates the IPv4 packet with a GRE header before sending it to the target stack. IPv6 packets are encapsulated with an additional IPv6 header. Multiple versions of GRE headers have been defined, but the sysplex distributor uses version 0 defined in RFC 1701.

The example on the slide illustrates the processing of VIPAROUTE traffic within a sysplex for an IPv4 network.

The client at 10.11.1.3 sends a packet to the sysplex distributor at 10.91.1.1. The sysplex distributor encapsulates this packet with a GRE header. This payload is encapsulated by a new IPv4 header with the target address defined on the VIPAROUTE statement as the destination address. Packets cannot be filtered by the original client or final target's IP addresses or ports for VIPAROUTE traffic. The packet is encapsulated by an additional header which uses the sysplex distributor and target home address as the source and destination addresses.

Packet trace example - encapsulation of VIPAROUTE traffic

- **Filter based on the encapsulated packet**
- **Add next hop address**

Filter on client source IP address 10.11.1.3 :

```
   3 MVS182   PACKET   00000004 07:48:04.757037 Packet Trace
 To Interface      : MPC4121L         Device: Mpc Ptp          Full=86
  Tod Clock        : 2009/04/21 07:48:04.757037                Intfx: 19
  Segment #        : 0                Flags:  Tunnel Out
  Source           : 10.61.0.1
  Destination      : 10.11.1.2
  Source Port      : 0                Dest Port: 0     Asid: 0033 TCB: 00000000
  Next Hop         : 10.11.1.2
 IpHeader: Version : 4                Header Length: 20
 Generic Routing Encapsulation Header
Ip Header          : 20               IP: 10.61.0.1, 10.11.1.2 Offset:  0
000000 45000056 00800000 3F2F65AF 0A3D0001  0A0B0102
```

```
Protocol Header    : 4                Protocol: 47(GRE)        Offset: 14
000000 00000800
Data               : 62     Data Length: 62                   Offset: 18
  Tos              : 00               QOS: Routine Normal Service
  Packet Length    : 62               ID Number: 0080
  Fragment         :                  Offset: 0
  TTL              : 63               Protocol: TCP            CheckSum: 64D1 FF
  Source           : 10.11.1.3
  Destination      : 10.91.1.1

Ip Header          : 20               IP: 10.11.1.3, 10.91.1.1 Offset: 18
000000 4500003E 00800000 3F0664D1 0A0B0103  0A5B0101
```


The solution to the problem is to automatically look beyond the encapsulation header for VIPAROUTE traffic. This allows filtering to be performed on the inner packet.

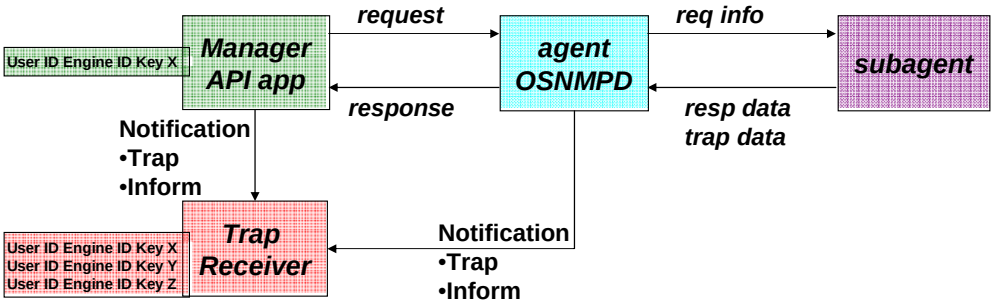
Additionally, the next hop IP address is provided for all outbound packets. This information is only viewable if the packet trace is formatted with the “FULL” option.

TCP/IP formatting code was modified to recognize and format the new, next hop field in packet trace data. This information is available externally by way of the real-time packet trace NMI. On the slide is an example of a sysplex distributor sending a packet to the target stack IP address defined on a VIPAROUTE statement. The next hop address for this packet is now always displayed. The outer IPv4 address contains the target address of 10.11.1.2. The GRE header is formatted next, followed by the original packet. The original packet addresses are filtered allowing the packets to be captured when outbound from the sysplex distributor and inbound to the target server.

As a reminder, the packet trace formatter has always been able to format and select packets encapsulated in other packets using the FIRST/LAST option and the FORMAT option. To select packets for the inner IP address, use OPTIONS((LAST IPADDR(10.11.1.3))).

SNMP Manager API for SNMP managers

- **For SNMPv3, trap receivers must know user ID and engine ID**
  - Both are used for table lookup to use correct security key
  - SNMP Manager API generates a random engine ID
- **V1R12 allows configuration of the engine ID**
  - Trap receivers receiving SNMPv3 notifications can find the correct security key to decrypt data
  - snmpInitialize() or snmpBuildSession()



- **New SNMP_SYNTAX_UNSIGNED32 value defined**
  - Supports snmpValueCreateUnsigned32() and snmpGetValue() functions

SNMP is an IETF standards-based protocol for network management. There are three main SNMP management entities: a manager, an agent, and a subagent. The manager typically sends SNMP requests to agents to retrieve management data. It can also send or receive notifications from agents or other managers. Notifications are alerts in the form of either traps or informs, which are confirmed traps. Network management applications typically provide the SNMP manager function.

The z/OS Communications Server SNMP Manager API can be used to create a network management application. Besides providing the ability to send SNMP requests and process responses, it also provides the ability to send SNMP notifications, which is not normally a manager function. Notifications contain a user ID and engine ID to identify the sender. The user ID value is sufficient for sender identification for most notifications, therefore the API generates a random engine ID. However, SNMPv3 notifications are encrypted and use a unique security key for encryption and decryption. The SNMPv3 specification requires that the key be mapped to a user ID and engine ID pair - not just a user ID. Therefore, the receiver must have the user ID and engine ID configured when searching for a key to use for decryption. A random engine ID will not work.

A new configurable engine ID value is now supported for use with SNMPv3 notifications. To request this support, you must invoke the snmpInitialize() function with a new value for the functionsRequested parameter. The engine ID can be configured on a configuration statement, or it can be specified as an input to the snmpBuildSession() function.

A new syntax constant for Unsigned32 has been defined for use in your SNMP Manager API application. Use the new snmpValueCreateUnsigned32() function to create the Unsigned32 data. Use the enhanced snmpGetValue() function to retrieve SNMP data of type Unsigned32 from an SNMP response.

Configuration Assistant (CA) for z/OS Communications Server

- **Windows-based stand-alone:**
  - Versions for z/OS V1R7 – V1R12 are available for download
  - http://tinyurl.com/cgoqsa
  - Support is “as-is”
- **Configuration Assistant for z/OS V1R12 Communications Server is also shipped with the z/OS MF product**
  - Runs on z/OS
  - Accessed from a web browser
  - Support is through normal IBM support channels
  - Improved look/feel

The IBM Configuration Assistant for z/OS V1R12 Communications Server will in due time be made available at the location shown above. You can use the Configuration Assistant to configure a variety of policy-based functions within z/OS Communications Server, such as IP Security or Application Transparent Transport Layer Security (AT-TLS).

The Configuration Assistant in z/OS V1R12 is delivered both as a Windows-based stand-alone application and as part of the z/OS Management Facility offering on z/OS.

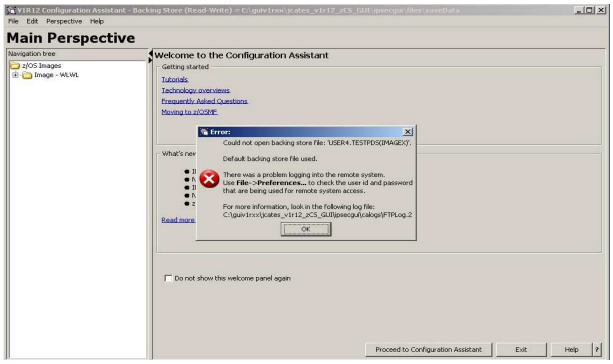
The Configuration Assistant began as a web download stand-alone Windows-based application. Versions of the code are available that support Communications Server from V1R7 to V1R12.

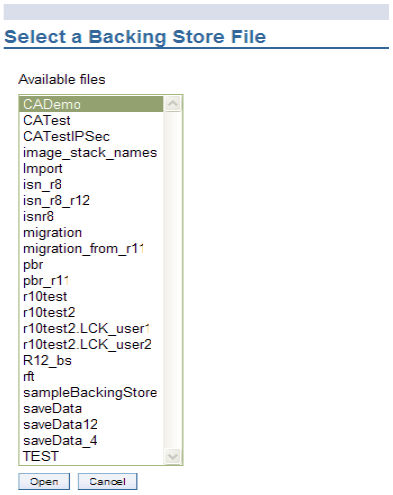
In the z/OS Management Facility (z/OSMF) environment, the Configuration Assistant is accessed through a web browser. Configuration data is generated for the Communications Server for releases V1R11 and V1R12. This code is fully supported as part of the z/OSMF product through the normal IBM support channels.

Several enhancements are made to the interface of the z/OSMF client. In V1R11, small messages issued by the Configuration Assistant took over the entire panel. In V1R12, such messages appear in a popup window instead. In V1R11, panels with tabs showed the tabs along the left side of the panel. Usability practices prefer tabs to be shown as a row along the top of the panel. Tabs on top are now shown for tabbed panels in V1R12. Also, several changes are made to make the Configuration Assistant more accessible to people with disabilities.

Enhancements to backing store file management

- **Backing Store files on z/OS**
  - Locally back up the Backing Store before FTP transfer
  - Better informational messages if back up fails
  - Sort list of Backing Store files
- **Assist with the password expiration problem**





Enhancements have been made to backing store file management.

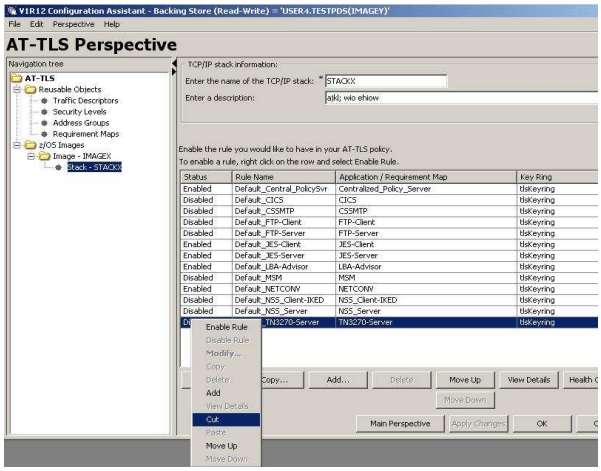
When you save backing store files on z/OS, the data transfer uses FTP. To protect the backing store from any FTP transmission difficulties, the backing store is automatically copied locally. If the FTP transmission is disrupted, error messages inform you where the backup copy was made.

When you save backing store files on z/OS, your FTP credentials (id and password) are saved in the Configuration Assistant. The Configuration Assistant uses this information when it is started to fetch and open the most recent backing store. With this new function, failed FTP credentials cause a temporary backing store to not open, and specific instructions are provided about how to remedy the problem with the credentials.

In z/OSMF, the collection of backing store files being managed were previously listed in no particular order. In z/OS V1R12, the list of backing store files is sorted alphabetically.

Enhancements to AT-TLS Perspective

- New rules for JES, IMS, DB2, NSS
- Rules to be reordered (cut and paste)
- AT-TLS default rules integrate with NSS



In V1R11, the Configuration Assistant provided default AT-TLS rules for common applications that support the AT-TLS function. This has proven to be a good methodology for most users who do not know the details of each application's needs. In V1R12, the list of default rules has been expanded to include JES, IMS, DB2 and NSS.

The list of AT-TLS rules is ordered. The interface to change this order is improved to allow groups of rules to be cut and pasted.

When NSS is configured, it must use AT-TLS to protect its communications. AT-TLS provides default NSS client and server rules. The health checker makes sure that these AT-TLS rules are compatible with NSS client and server specifications. If NSS is active and the default NSS client or server rules have not been enabled, generated AT-TLS rules are provided.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_SysMgmt.ppt

This module is also available in PDF format at: ../SysMgmt.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information