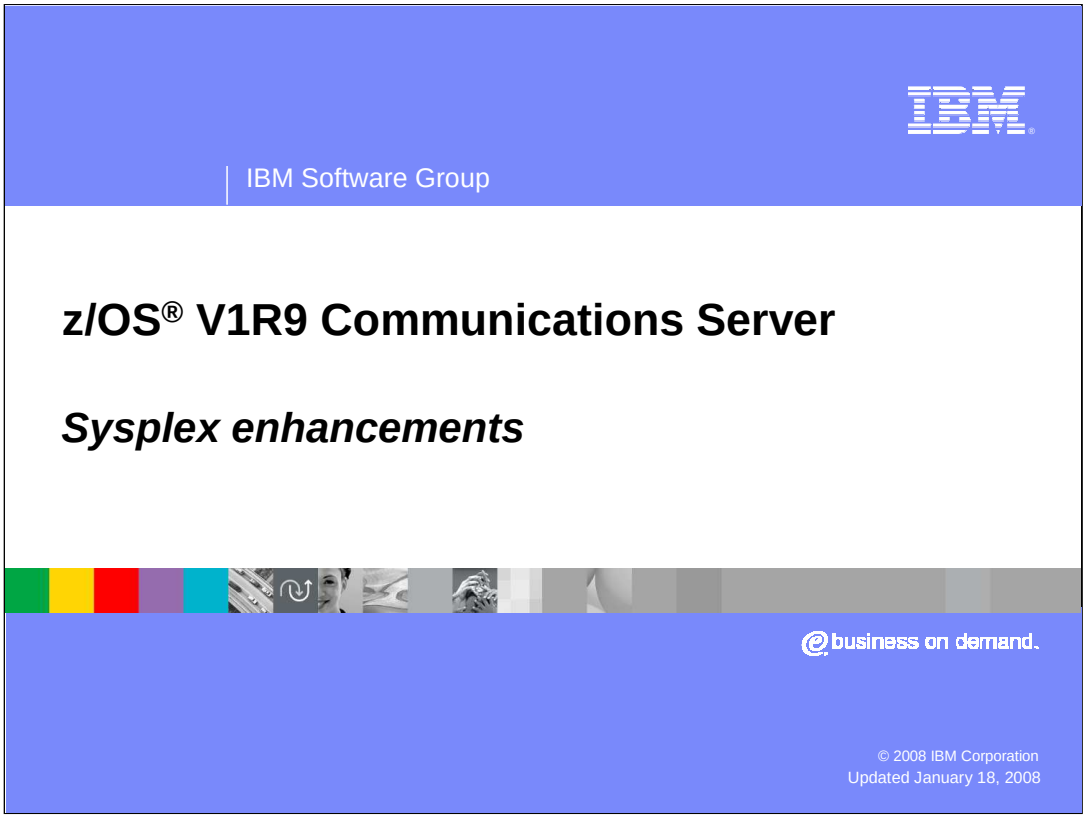
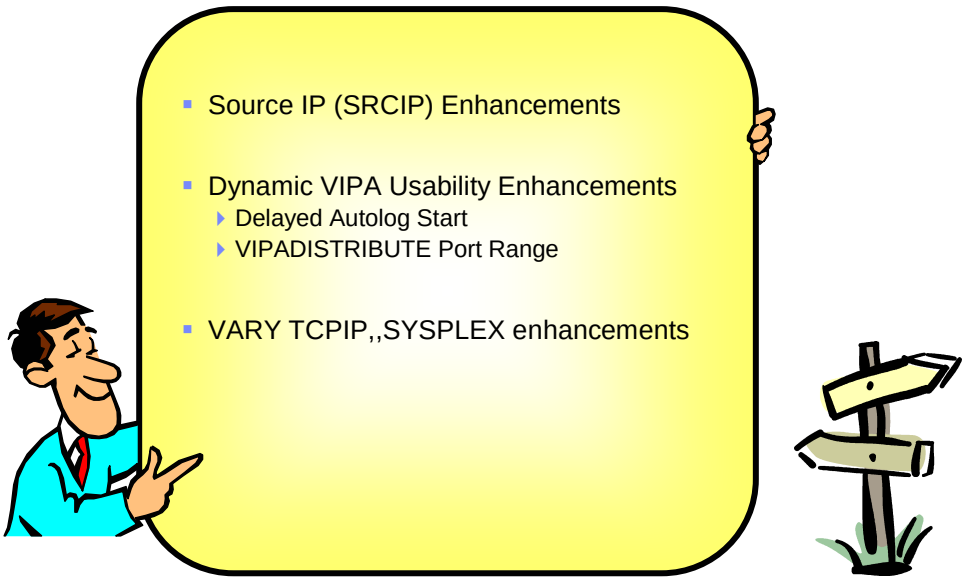


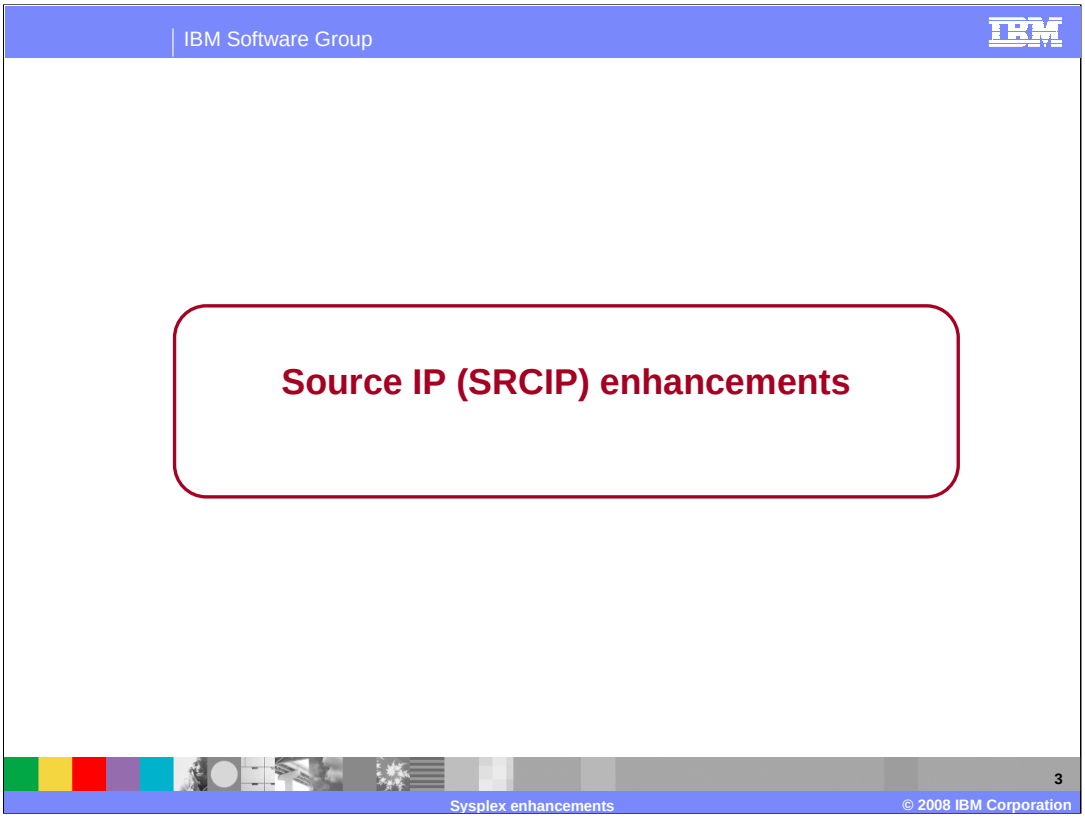
This presentation discusses the Sysplex enhancements in z/OS V1R9 Communications Server.

Agenda

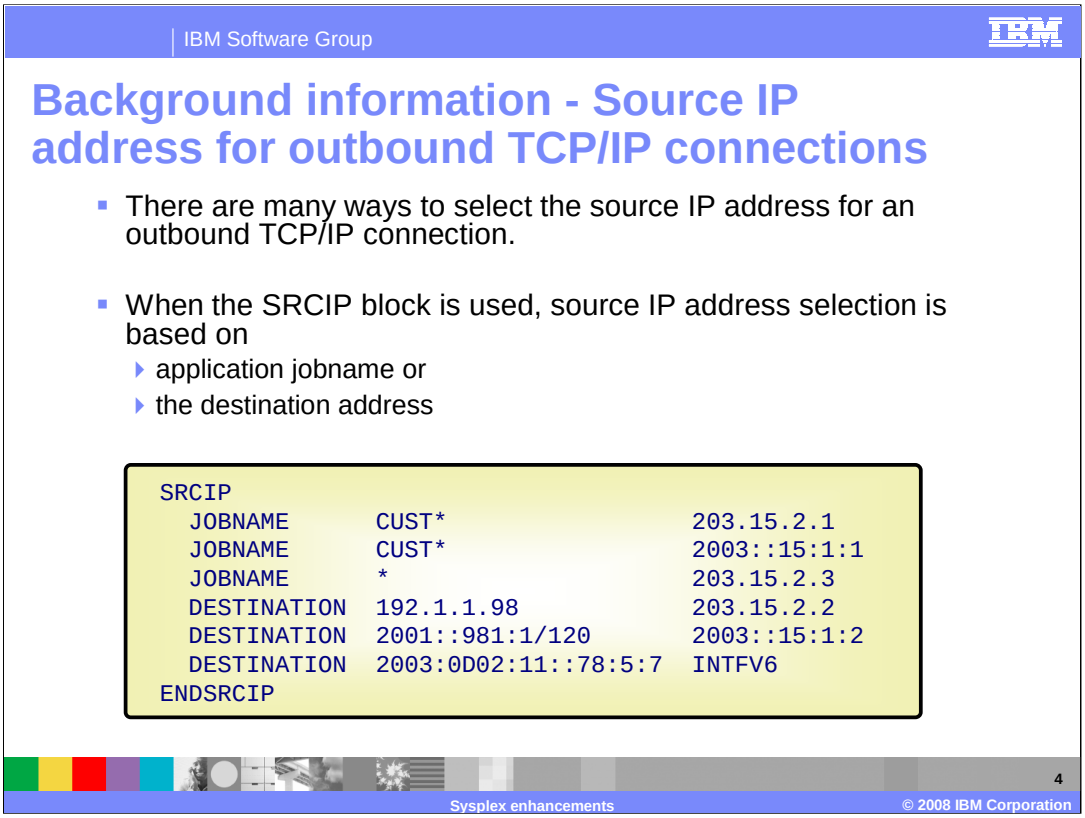


z/OS V1R9 Communications Server Sysplex enhancements include changes to the SRCIP block to give it more functionality.

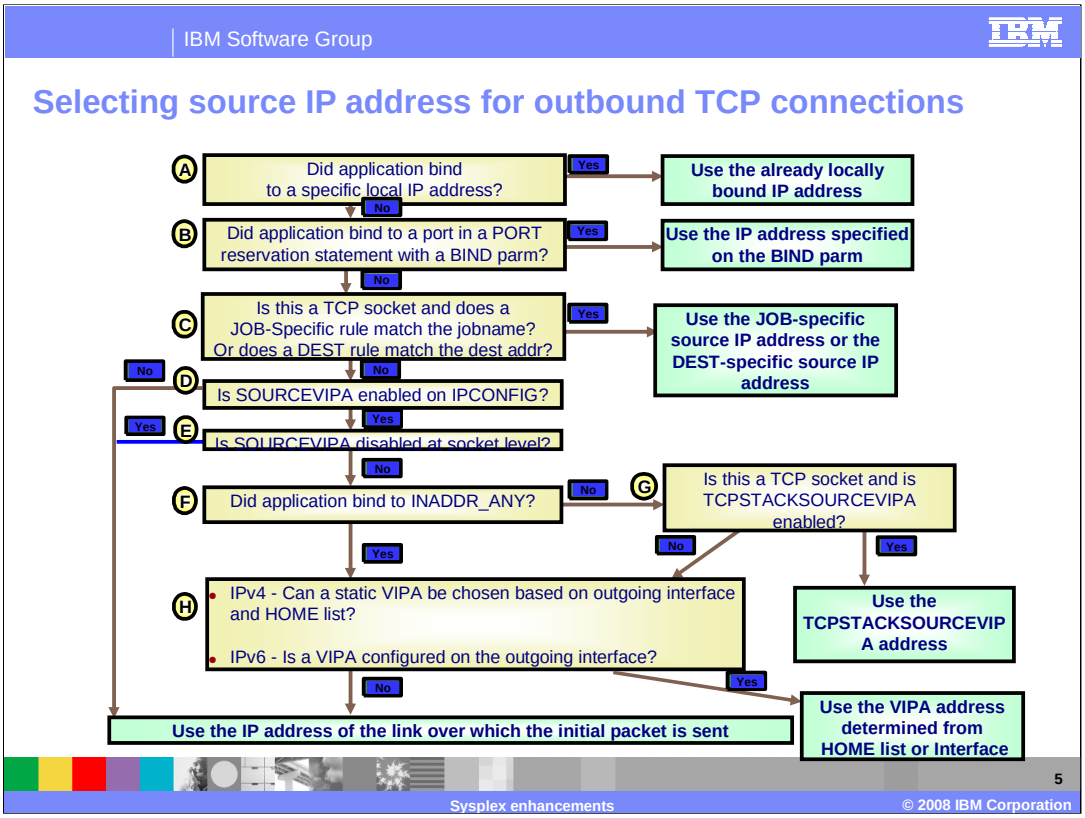
We will also be discussing the dynamic VIPA usability and the V TCPIP,,SYSPLEX enhancements.



This section describes the Source IP Enhancements for the z/OS V1R9 Communications Server.



TCP/IP provides a number of different mechanisms for selecting the source IP address of an outbound connection when an application has not explicitly specified one. The notes pages that follow describe the various selection methods. They will not all be discussed in this presentation. This presentation will focus on one of the mechanisms, the SRCIP block. The SRCIP block allows selection of a source IP address based on the applications jobname or based on the destination address. In the example shown, you can use the keyword JOBNAME to specify an application name (or part of a name, using the asterisk as a wildcard), and specify the source IP address to be used if there is a match on that name. Or you can use the DESTINATION keyword to supply a destination IP address. When a connection is made to that destination address, the associated source IP address will be used.



This chart illustrates the algorithm TCP/IP uses for selecting a source IP address.

As shown in step B a PORT profile statement can be specified with a BIND parameter supplying the source IP address. If an application BINDs to that port without specifying a specific source IP address, the IP address on the PORT statement will be used.

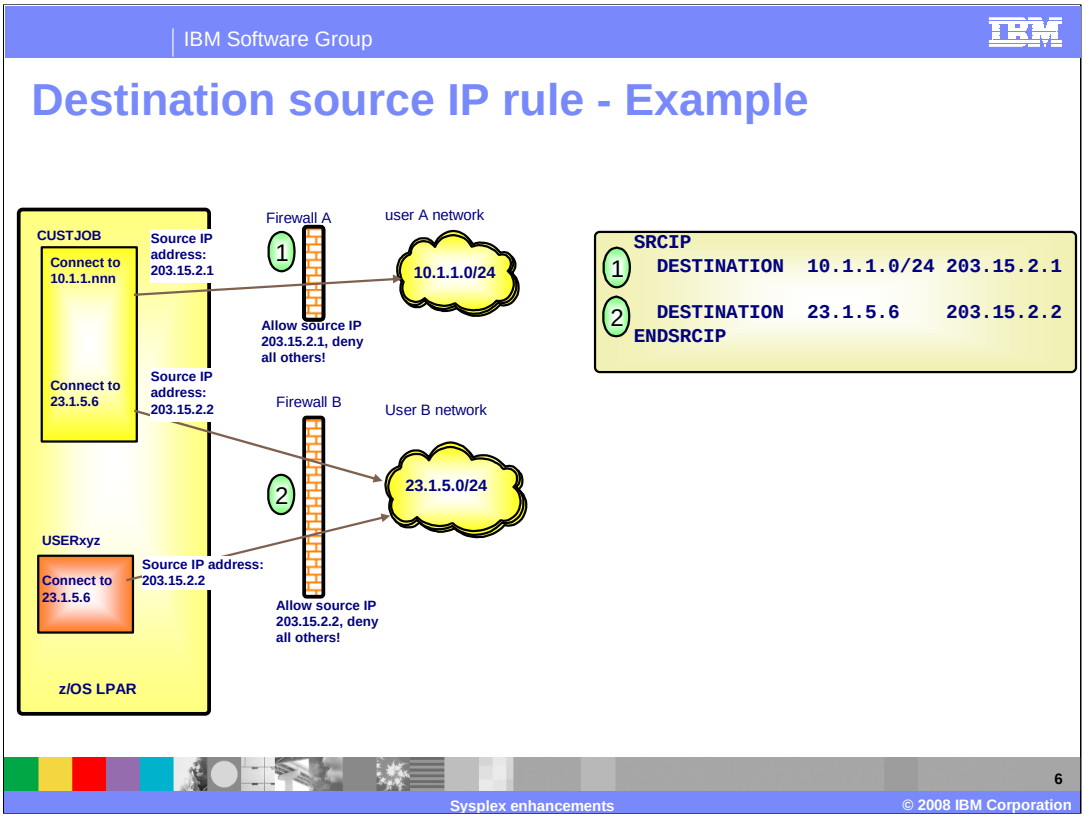
You can use the SRCIP block to select the source IP address based on the application jobname or based on the destination address. If an application has not issued a BIND on a socket (or has issued a BIND for INADDR_ANY) and then issues a CONNECT for that socket, if the application's jobname matches one of the JOBNAME entries in the SRCIP block, then the source IP address configured for that rule will be used. If the destination address for the connect matches one of the DESTINATION entries, the source IP address supplied on that entry will be used.

If TCPSTACKSOURCEVIPA is configured on IPCONFIG or IPCONFIG6 profile statement then SOURCEVIPA must also be configured. The TCPSTACKSOURCEVIPA address must be a static or Dynamic VIPA. Here is an example of it being configured on the IPCONFIG statement.

IPCONFIG SOURCEVIPA TCPSTACKSOURCEVIPA 203.1.15.99

From the example if an application issues a connect(), then 203.1.15.99 will be used as source IP address if SOURCEVIPA has not disabled for the socket and an explicit bind was not done for the socket (even to INADDR_ANY).
For SOURCEVIPA to be effective, you must also configure static VIPAs and order them within your HOME statement when using IPv4 and configure the SOURCEVIPAINT keyword on the INTERFACE statement for each interface on which you want SOURCEVIPA to take effect when using IPv6. If SOURCEVIPA is not disabled for the socket and the socket is not bound or is bound to INADDR_ANY then the source IP address used will be the first static VIPA in the HOME list that precedes the selected interface for the packet when using IPv4. The source IP address used will be the address specified by the SOURCEVIPAINT keyword on the INTERFACE statement for the selected interface when IPv6 is being used.

Refer to the IP Configuration Reference for more details on each of the configuration profile statements. Refer to the section "Source IP Address Selection" in the IP Configuration Guide for a more complete discussion of how TCP/IP determines the source IP address for an outbound UDP, TCP or RAW packet.



This diagram illustrates the use of the DESTINATION keyword in the SRCIP block for source IP selection. Suppose an application must connect to IP addresses in two different networks, and, let's say, those two networks are protected by firewalls which have different criteria as to which source IP addresses are allowed to pass through. The application must select an appropriate source IP address based on the destination address it is connecting to. In this example, application CUSTJOB connects to an address in the 10.1.1.0 network, and TCP/IP uses the first DESTINATION rule to provide a source IP address of 203.15.2.1, which can pass through firewall A. When CUSTJOB connects to address 23.1.5.6, TCP/IP will select the second DESTINATION rule to provide the source IP address of 203.15.2.2, which can pass through firewall B. Note in the example, that another application (USERxyz) can connect to a destination address that matches one of the SRCIP rules, and TCP/IP will select the corresponding source IP address for that application, also.

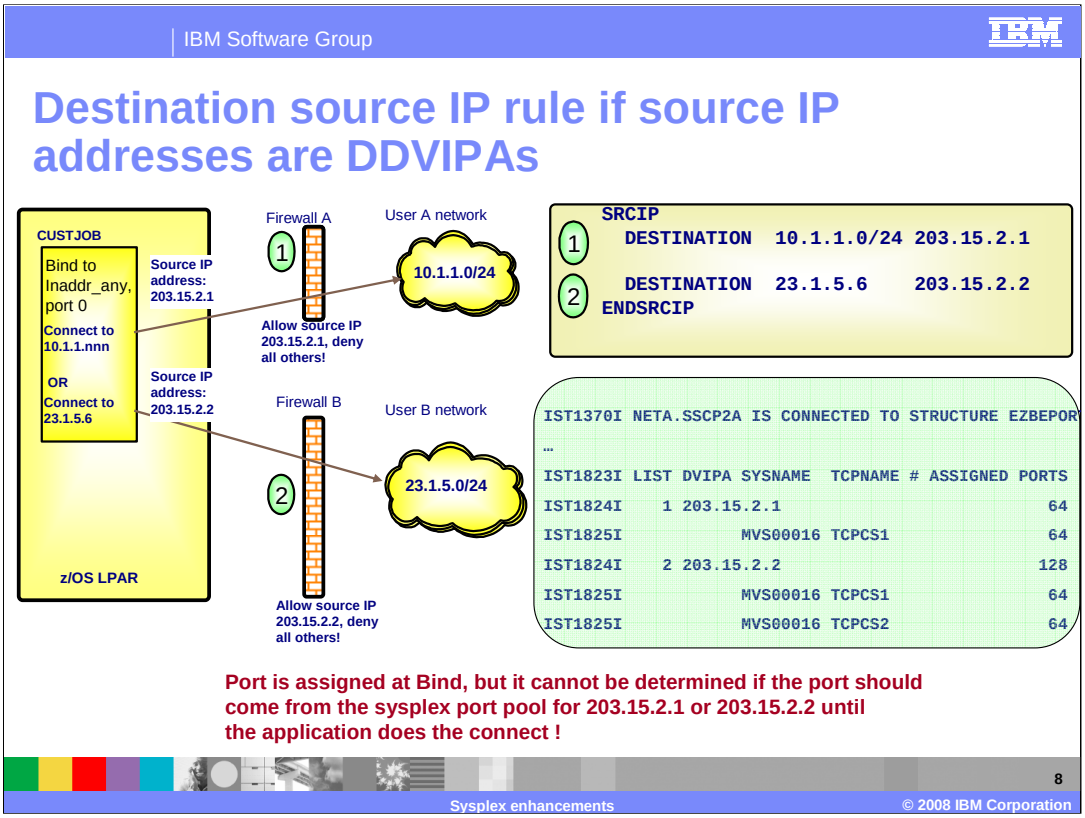
Problem Statement - SRCIP Limitations

- Source IP address on DESTINATION rule in SRCIP block cannot be a distributed DVIPA
  - Why?
    - ✓ Distributed DVIPAs (DDVIPAs) can be active on many stacks in the sysplex
    - ✓ To remove the possibility of duplicate outbound 4-tuples, source ports for DDVIPAs are coordinated across the sysplex
    - ✓ Ports are allocated from a DDVIPA-specific pool using the EZBEPORTvvtt structure when a BIND() is issued
    - ✓ When an application issues an explicit BIND() socket API to INADDR_ANY, port 0, a port must be assigned
      - Only at connect time is the source address known
      - A DDVIPA-specific pool cannot be used
      - How do you guarantee a unique port assignment at BIND even though you do not know the DDVIPA?
- SRCIP JOBNAME entries specify an IPv4 and IPv6 DDVIPA source IP address for rules that will match the same job name
  - If the socket is AF_INET6 and the connection destination will be an IPv4 address, the connection may fail
  - Why?
    - ✓ When an application running with that job name, using an AF_INET6 socket issues an explicit BIND() to IN6ADDR_ANY, port 0, a port must be assigned at this time
    - ✓ However, at BIND time, the stack doesn't know whether the destination will be an IPv6 or IPv4 destination
      - It can't determine whether to the assign the port from the IPv4 or the IPv6 DDVIPA's pool.
      - Currently the code uses the IPv6 pool since the socket is AF_INET6
- Users want to convert some Server applications that Bind() to the unspecified address to use a specific address
  - This cannot be done if the application BINDs to port 0

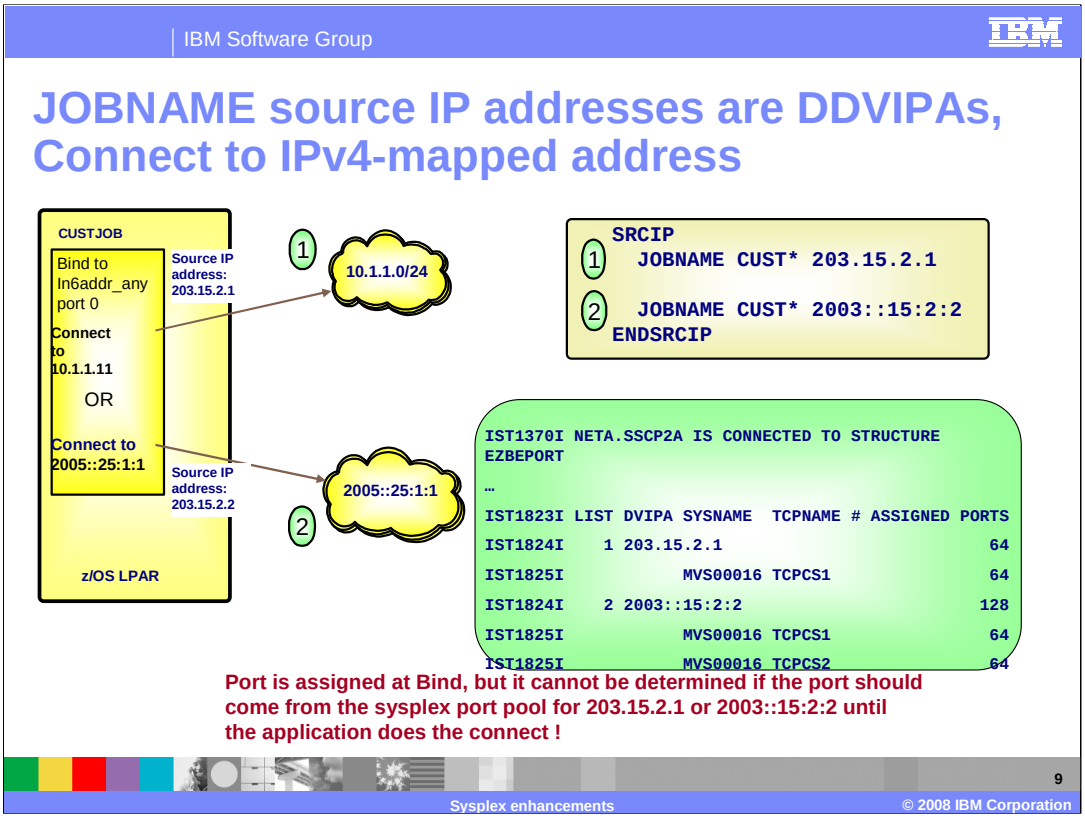
In previous z/OS Communications Server TCP/IP releases, there is a restriction on the type of IP address that can be specified on a SRCIP block DESTINATION rule. Specifically, it could not be a distributed DVIPA. Because distributed DVIPAs may be active on many stacks in the sysplex, source ports that are allocated for connections from these DVIPAs must be coordinated across the sysplex. If they were not, an application on node A in the sysplex, connecting to a destination IP address and port using a specific distributed DVIPA might choose the same port as an application on node B, using the same distributed DVIPA to connect to the same destination IP address and port. This would result in two connection requests with the same 4-tuple (of source IP address, source port, destination IP address, destination port) being sent to the same destination. To prevent this, allocation of source ports (known as sysplexports) for distributed DVIPAs is coordinated using the EZBEPORTvvtt structure, which establishes an allocated source port pool for each specific distributed DVIPA. A problem occurs when an application uses an explicit BIND to INADDR_ANY and port 0 before issuing a CONNECT. The BIND protocols require that a port be assigned at this time. However, since the CONNECT has not yet been issued, TCP/IP does not know the destination address, so it would not know to allocate the port from the sysplex port pool associated with the matching DESTINATION rule's source IP address, if that source IP address were a distributed DVIPA.

Another problem that can occur is if there are JOBNAME rules in a SRCIP block that can match the same jobname and have an IPv4 and an IPv6 distributed DVIPA source address. Here, if a CONNECT is issued on an AF_INET6 socket and the destination address is an IPv4 address, then the JOBNAME rule with the IPv4 source IP address will be selected, and the source IP port will be allocated from the IPv4 address's sysplexport pool. If the destination is an IPv6 address, then the JOBNAME rule specifying the IPv6 source IP address will be selected, and the source IP port will be allocated from the IPv6 address's sysplexport pool. However, if an explicit BIND for IN6ADDR_ANY and port 0 is done before the connect, TCP/IP does not know which sysplexport pool to allocate the port from. TCP/IP chooses the IPv6 pool, since the socket is AF_INET6, which may cause the connection to fail, if the connection is to the IPv4 destination address.

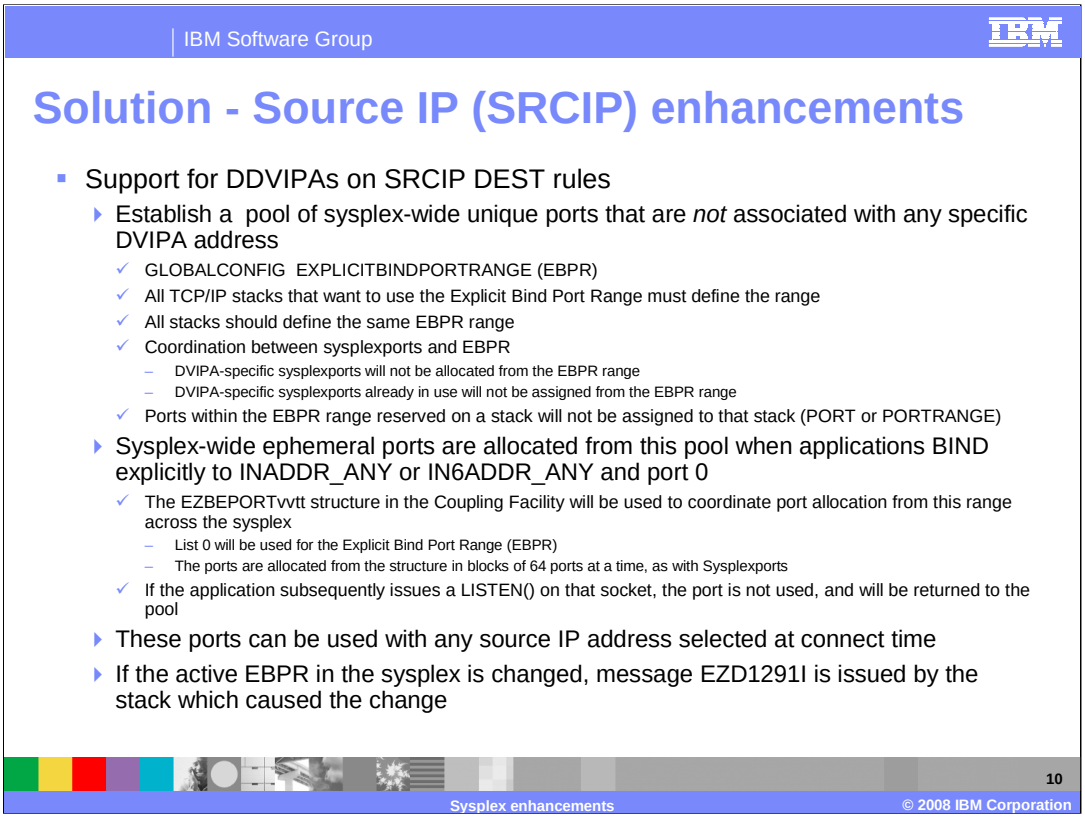
The BIND parameter on the port statement can be used to specify the source IP address when the application binds to the unspecified address and a port in the range of 1 to 65535. However this function can not be used when the application binds to port 0. Users would like the ability to convert the unspecified address to a specific IP address for this case. Here is an example of when this may be required. A user has multiple z/OS WebSphere Application Server instances running in the same z/OS image. One way to provide isolation of these instances is to have each instance associated with a unique VIPA for all of its listening sockets. A single WebSphere Application Server instance may have multiple address spaces acting as TCP listeners and some of these listeners may use ephemeral ports. Existing WebSphere Application Server configuration options and scripts could be used to force each Listener to Bind to specific IP addresses but this process can be complex. A more straight forward way of accomplishing this is desired



In this example, you can see that the application CUSTJOB can connect to either an address in the 10.1.1.0 network, or the address 23.1.5.6 in another network. If distributed DVIPAs are specified as the source IP address on the corresponding DESTINATION rules, the source ports should be allocated from the sysplexports pool associated with the specific DVIPA, as depicted in the green box. However, if the CUSTJOB application issues an explicit BIND to inaddr_any and port 0 before doing the CONNECT, TCP/IP cannot tell which specific distributed DVIPA pool to allocate the source port from.



This diagram illustrates the problem with mapped IPv4 addresses and JOBNAME rules that specify distributed DVIPAs. Here, application CUSTJOB can match either of the two SRCIP JOBNAME rules, one of which specifies an IPv4 distributed DVIPA address, and the other an IPv6 distributed DVIPA address. If CUSTJOB issues a CONNECT to a destination IP address in the IPv4 network 10.1.1.0, TCP/IP can allocate the source IP port from the sysplexport pool associated with the IPv4 source IP address. If CUSTJOB connects to a destination address in the IPv6 network, TCP/IP can allocate the source IP port from the sysplexport pool associated with the IPv6 source IP address. However, if CUSTJOB issues a BIND to in6addr_any and port 0 on an AF_INET6 socket before the CONNECT, TCP/IP cannot know which sysplexport pool to allocate the source port from.



Support for DDVIPAs on the SRCIP DEST rule is allowed by a new sysplex-wide port range called the Explicit Bind Port Range. It is configured with a new parameter on the TCP/IP GLOBALCONFIG statement. There are two new GLOBABLCONFIG parameters for enabling and disabling an Explicit Bind Port Range processing. NOEXPLICITBINDPORTRANGE indicates that the stack will not participate in EBPR processing, when handling an explicit bind() of a TCP socket to an IP address of INADDR_ANY or IN6ADDR_ANY and port 0. EXPLICITBINDPORTRANGE indicates that the stack will participate in EBPR processing, when processing an explicit bind() of a TCP socket to an IP address of INADDR_ANY or IN6ADDR_ANY and port 0. It also specifies the range of ports, starting at *1st_port* for *num_ports* ports, that will define that pool. This parameter will define the range used by all stacks participating in EBPR processing throughout the sysplex. The EZBEPORTvvtt structure will coordinate port allocation across the sysplex in the new range, using list 0 of the structure.

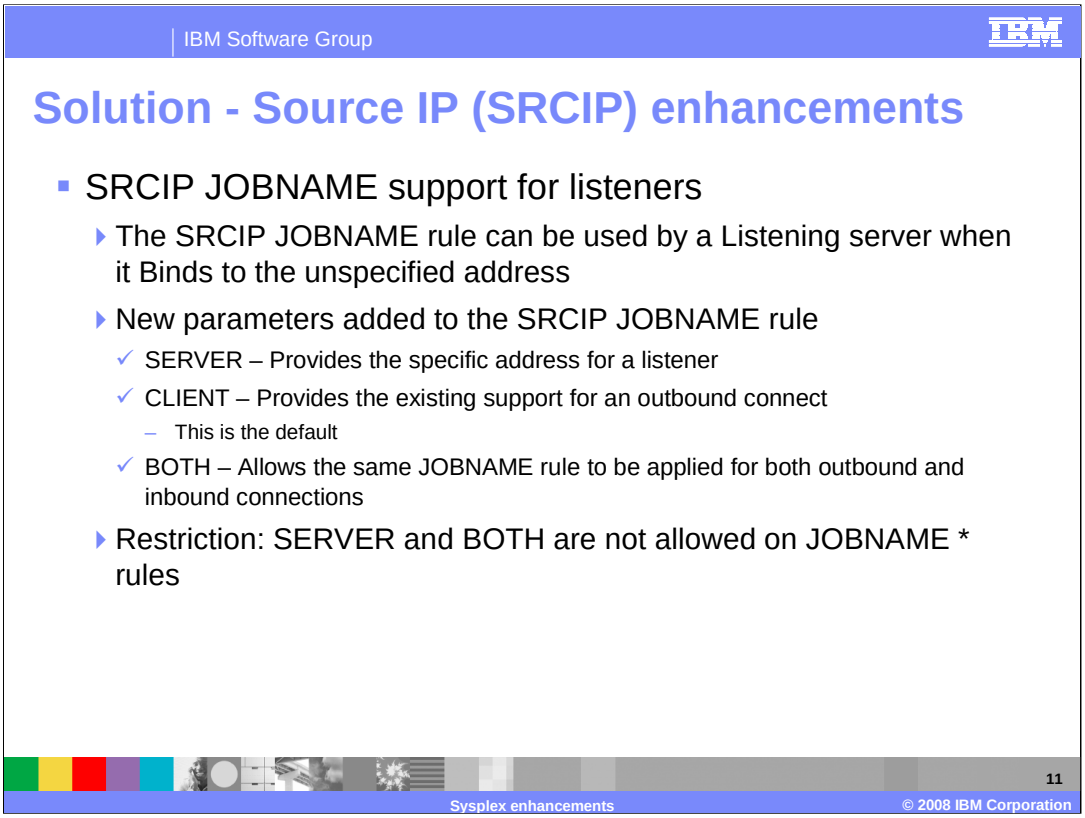
All TCP/IP stacks in the sysplex that will participate in EXPLICITBINDPORTRANGE processing should specify the same port range. This can be done by specifying the GLOBALCONFIG EXPLICITPORTRANGE statement in a file that is specified in an INCLUDE statement in the TCP profiles data sets of all the participating stacks. If stacks define different ranges, the last configuration processed defines the EBPR range for the entire sysplex (or subplex). The port range defined on the EXPLICITBINDPORTRANGE parameter should not overlap any existing port reservations of any TCP/IP stacks in the sysplex. Note that any reserved ports that are within the EXPLICITBINDPORTRANGE range will be excluded from the EXPLICITBINDPORTRANGE port pool, effectively making the pool smaller. The EXPLICITBINDPORTRANGE port range must be large enough to accommodate all applications in the sysplex that may issue explicit bind() calls for INADDR_ANY (or IN6ADDR_ANY) with port 0. If additional TCP/IP stacks or systems are introduced into the sysplex, the extent of the port range defined by EXPLICITBINDPORTRANGE should be re-evaluated. If the size of the port range defined by EXPLICITBINDPORTRANGE is too large, there will be fewer ports available for local ephemeral port allocation.

Changing the range specified on the EXPLICITBINDPORTRANGE parameter of the GLOBALCONFIG statement affects every stack in the sysplex that has configured a GLOBALCONFIG EXPLICITBINDPORTRANGE. Future port allocations for all such stacks will use the new port range. Ports in the EXPLICITBINDPORTRANGE range are typically assigned to a stack in blocks of 64 ports. When expanding the range, you should use multiples of 64 times the number of stacks using GLOBALCONFIG EXPLICITBINDPORTRANGE.

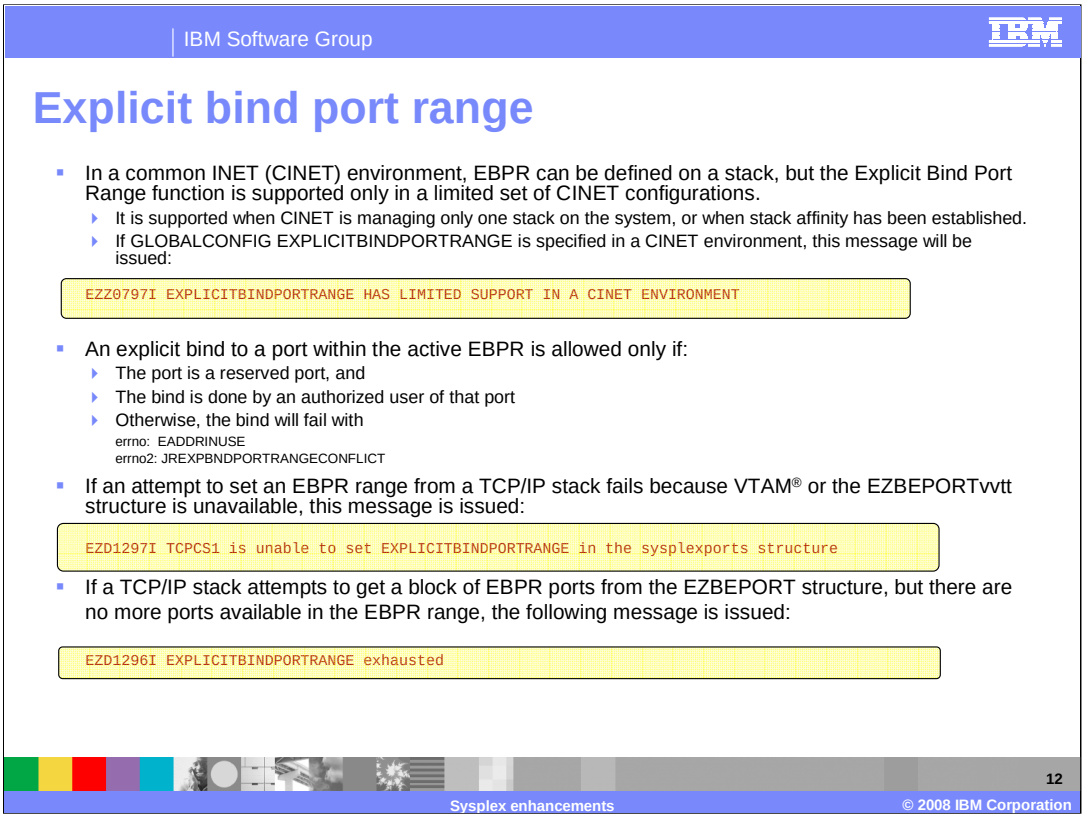
A stack with an Explicit Bind Port Range configured will attempt to set the range in the EZBEPORTvvtt structure. If the range in the structure is successfully changed to a different range, the stack setting the range will issue message EZD1291I to display the new range. Here is an example of the message that may be issued:

EZD1291I Active EXPLICITBINDPORTRANGE changed to 01024 – 02047 by TCPCS1 on MVS001

If the attempt to set the range did not change the range, no message will be issued.

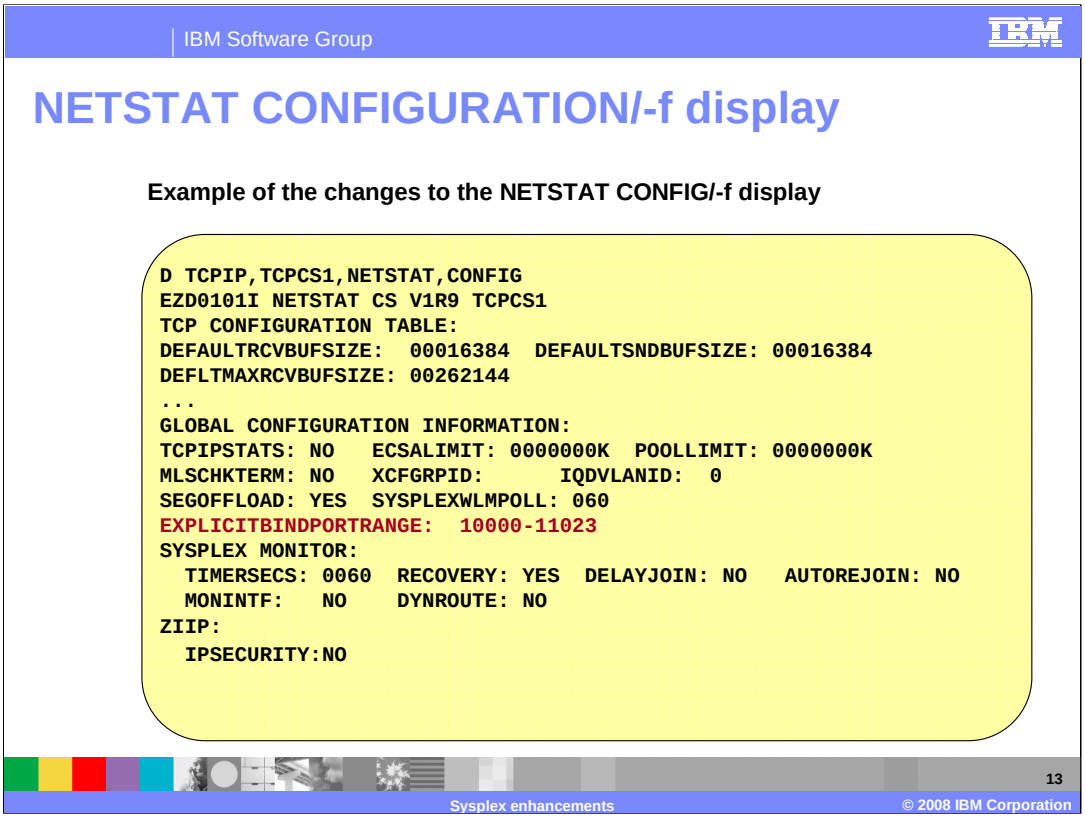


The SRCIP JOBNAME rule is extended to allow users to specify a specific address for a Server application that Binds to INADDR(6)_ANY. Three new parameters can be specified on the JOBNAME rule. The SERVER parameter will cause a Server application with a matching jobname to have the source IP address on the matching rule used in place of INADDR(6)_ANY on the Listen(). The CLIENT parameter (which is the default), will indicate that existing behavior of substituting the source IP address for outbound connections done by an application matching the specified JOBNAME (Clients). The BOTH parameter will allow the same JOBNAME rule to be used for both client and server applications. The SERVER and BOTH parameters are not allowed on JOBNAME * rules.

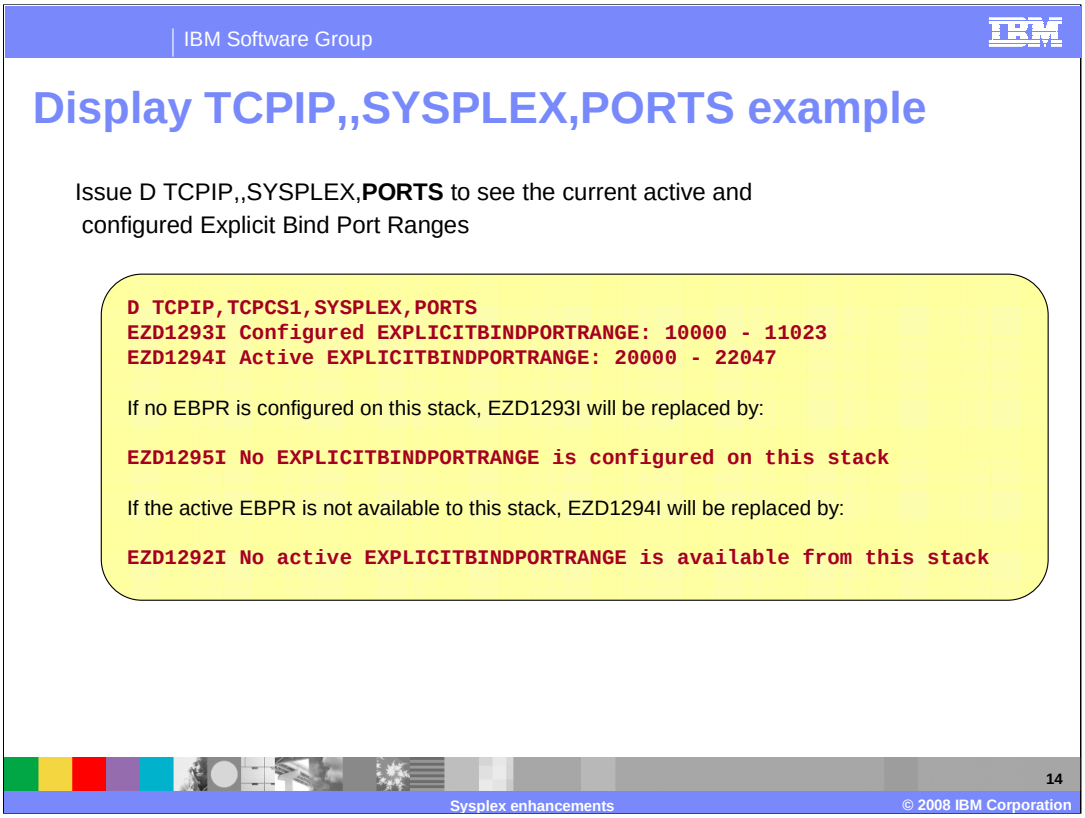


When operating in a CINET environment where CINET is managing more than one stack and stack affinity has not been established, CINET will substitute a port from the INADDRANYPORT port range defined in the BPXPRMxx parmlib when an application Binds to INADDR_ANY or IN6ADDR_ANY and port 0, before passing the BIND() request to the TCP/IP stack. The TCP/IP stack will not see a Bind to port 0, but will instead see a BIND specifying a specific port. Therefore, it will not assign a port from the ExplicitBindPortRange pool. Subsequently, if a SRCIP DESTINATION match selects a distributed DVIPA to be the source IP address, the connect will fail with the JRSRCIPDistDVIPA reason code, indicating the port is not from the EBPR pool.

Binding explicitly to a port number that is within the active explicit bind port range can only be done by authorized users. If the port is not reserved by the PORT or PORTRANGE profile statement or if the application that issues the bind is not authorized for the port, the bind will fail with EADDRINUSE and a new errnojr that indicates that this port is available only as an ephemeral port for explicit binds to INADDR_ANY or IN6ADDR_ANY and port 0.



The NETSTAT CONFIG/-f command will display the Explicit Bind Port Range under the Global Configuration Information section. If no Explicit Bind Port Range is configured on this stack, 00000-00000 will be displayed as the range.



The current active and configured EBPR ranges can be displayed using the new D TCPIP,,SYSPLEX,PORTS command. The configured range is the range that was defined on the specified stack. The active range is the range that is actually being used in the EZBEPORTvvtt structure.

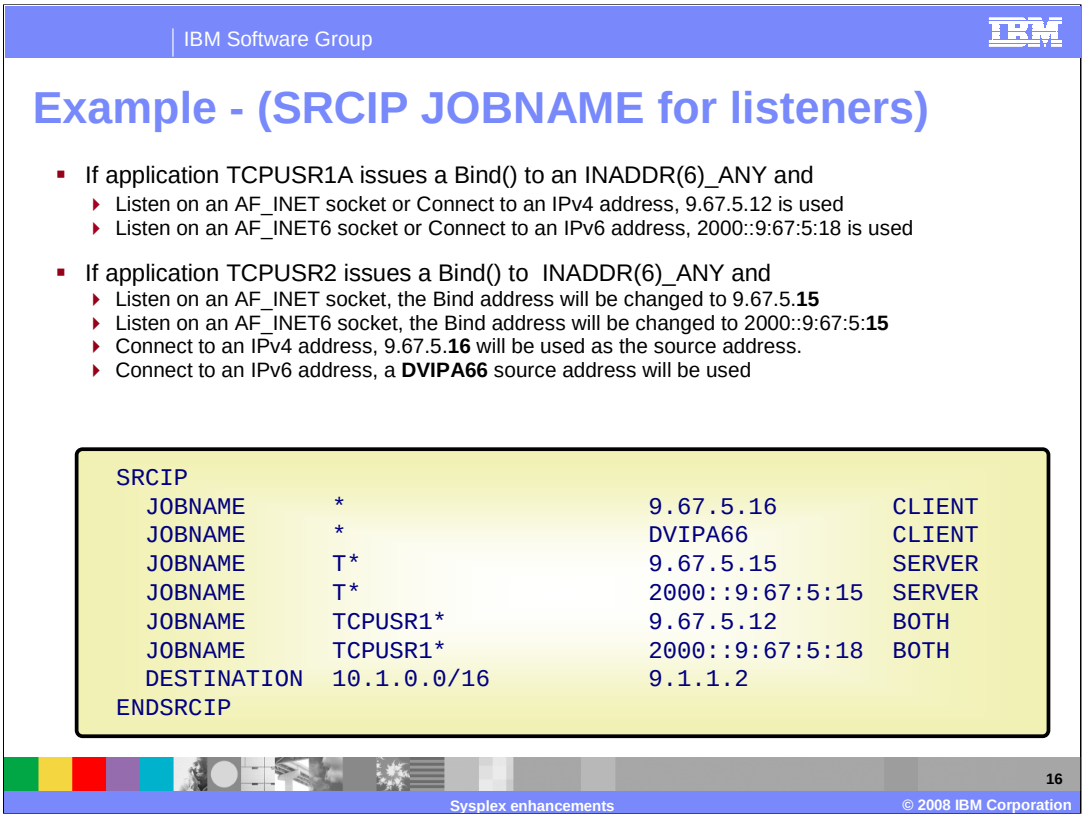
Message EZD1292I may be issued if the stack has not yet fully completed establishing an Explicit Bind Port Range with the Coupling Facility or if access to the Coupling Facility structure has failed.

Display NET,STATS,TYPE=CFS command example for EZBEPORTvvtt

Issue D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvvtt,LIST=ALL to show the EXPLICITBINDPORTRANGE allocations in the CFS structure

```
20.32.31  IST350I DISPLAY TYPE = STATS,TYPE=CFS
IST1370I NETA.SSCP2A IS CONNECTED TO STRUCTURE EZBEPORT
IST1797I STRUCTURE TYPE = LIST
IST1517I LIST HEADERS = 1024 - LOCK HEADERS = 1024
IST1373I STORAGE ELEMENT SIZE = 256
IST924I -------------------------------------------------------------
IST1374I                              CURRENT     MAXIMUM  PERCENT
IST1375I STRUCTURE SIZE                 8192K      15104K    *NA*
IST1376I STORAGE ELEMENTS                  64       22400       0
IST1377I LIST ENTRIES                       3         700       0
IST924I -------------------------------------------------------------
IST2221I EXPLICITBINDPORTRANGE - START: 20000  END: 22047
IST1823I LIST DVIPA SYSNAME  TCPNAME                # ASSIGNED PORTS
IST1824I    0 EXPLICITBINDPORTRANGE                          128
IST1825I            MVS00001 TCPCS1                           64
IST1825I            MVS00002 TCPCS11                          64
IST1824I    1 203.16.2.1                                      64
IST1825I            MVS00001 TCPCS1                           64
...
```


The active Explicit Bind Port Range can also be display from VTAM, by displaying the structure information for the EZBEPORTvvtt structure. If there is no active EBPR range, message IST2221I will not be displayed, and no information on list 0 will be displayed.



This slide illustrates the use of the new SRCIP JOBNAME rule parameters.

Note that if the application issues a Bind to INADDR_ANY on an AF_INET6 socket, an IPv4 source address will be chosen over an IPv6 address since the application is intending to use the socket to receive mapped IPv4 packets. So in the example above, if application TCPUSR2 issues a Bind to INADDR_ANY & a Listen on an AF_INET6 socket, the Bind address would be changed to 9.67.5.15.

NETSTAT SRCIP display

Example of the changes to the NETSTAT SRCIP/-J display

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPCS
20:30:49
Source IP Address Based on Job Name:
Job Name  Type  Flg  Source
--------  ----  ---  ------
*         IPV4  C    9.67.5.16
*         IPV6  C    DVIPA66
T*        IPV4  S    9.67.5.15
T*        IPV6  S    2000::9:67:5:15
TCPUSR1*  IPV4  B    9.67.5.12
TCPUSR2*  IPV6  B    DVIPA62

Source IP Address Based on Destination:
Destination: 10.1.0.0/16
  Source:    9.1.1.2
```


The NETSTAT SRCIP/-J display shows the new JOBNAME parameters by adding a flag column, FLG. In that column, C is for CLIENT, S is for SERVER, and B is for BOTH.

Diagnosis

- Diagnosing explicit bind port range
  - Use D TCPIP,,SYSPLEX,PORTS to determine
    - the configured EXPLICITBINDPORTRANGE for this stack
    - the active EXPLICITBINDPORTRANGE in the sysplex (or subplex).
  - Verify from the console log that this message was issued:
    - IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORTvvtt
  - After a Bind to IN(6)ADDR_ANY and port 0.
    - Use D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvvtt,LIST=0 to verify that ports from the EBPR pool were allocated for the stack
    - Use NETSTAT ALLCONN/-a to determine if the port is from the Explicit Bind Port Range.

If the configured EBPR is different from the active EBPR, this indicates another TCP/IP stack has subsequently configured a different range

Ensure that all stacks that configure EBPR specify the same range of ports

If the display indicates that no EBPR is configured, then verify that the GLOBALCONFIG EXPLICITBINDPORTRANGE statement is syntactically correct. If the display indicates that no active EBPR is available from this stack then determine if access to the EZBEPORTvvtt structure has been established.

Look for the message IST1370I. This will determine if VTAM has successfully connected to the EZBEPORT structure on behalf on the TCP/IP stack. If sysplex subplexing is being used, the structure name will be in the form EZBEPORTvvtt, where vv is the VTAM subplex group ID, and tt is the TCP/IP stack subplex group ID. Alternatively, issue D XCF,STR,STRNAME=EZBEPORTvvtt to determine if this VTAM node is connected to the structure.

After an application has BIND to IN(6)ADDR_ANY and port 0 then issue D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvvtt,LIST=0 to display the EBPR range and how many ports have been allocated for that range. There should be at least 64 ports allocated in the EBPR pool, and at least 64 ports should be listed as allocated to the TCP/IP stack that the application was associated with. If message EZD1296I EXPLICITBINDPORTRANGE exhausted is issued, then there are no more ports available in the EBPR range. Determine whether the active EBPR range may overlap with other reserved port ranges on this stack. Otherwise, consider increasing the size of the EBPR range.

NETSTAT ALLCONN/–a will display the connections for this stack. If an EBPR port was assigned to this connection, it will be displayed as part of the local socket.

How to determine the optimal size of the EBPR

1. Define an EXPLICITBINDPORTRANGE with 128 ports for each participating stack. Do not initially make the change to use distributed DVIPAs on the DESTINATION rules.
2. Reach the typical steady state connection load for your sysplex environment. All connections from sockets that were explicitly bound to IN(6)ADDR_ANY and port 0 will use ports from the new range.
3. Periodically check to see how many ports from the new range are in use:

   D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvvtt,LIST=0 command
4. Check for message EZD1296I which will be issued if a local ephemeral port was used because the EXPLICITBINDPORTRANGE was exhausted
5. If the above message is issued
   1. Change the GLOBALCONFIG EXPLICITBINDPORTRANGE parameter to use a larger explicit bind port range; as a guideline increase the size by 64 ports for each participating stack
   2. Issue a VARY TCPIP,,OBEYFILE command to change the range and return to step 3 above.
6. Change your SRCIP block to use distributed DVIPAs on DESTINATION rules on each participating stack.

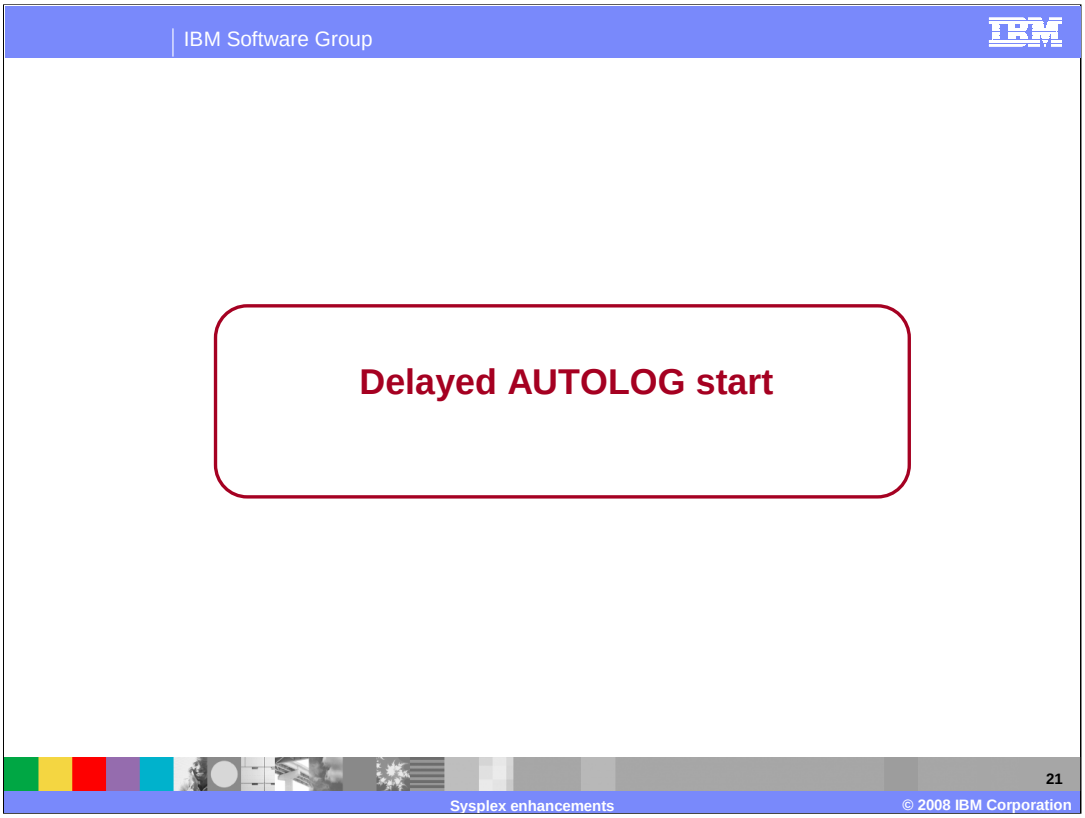
➔ The same EXPLICITBINDPORTRANGE should be configured on each participating stack.

This page describes how to determine an optimal size for the EBPR range. EZD1296I is issued if an EXPLICITBINDPORTRANGE port is not available during Bind processing, and it has been at least 5 minutes since this message was previously issued.

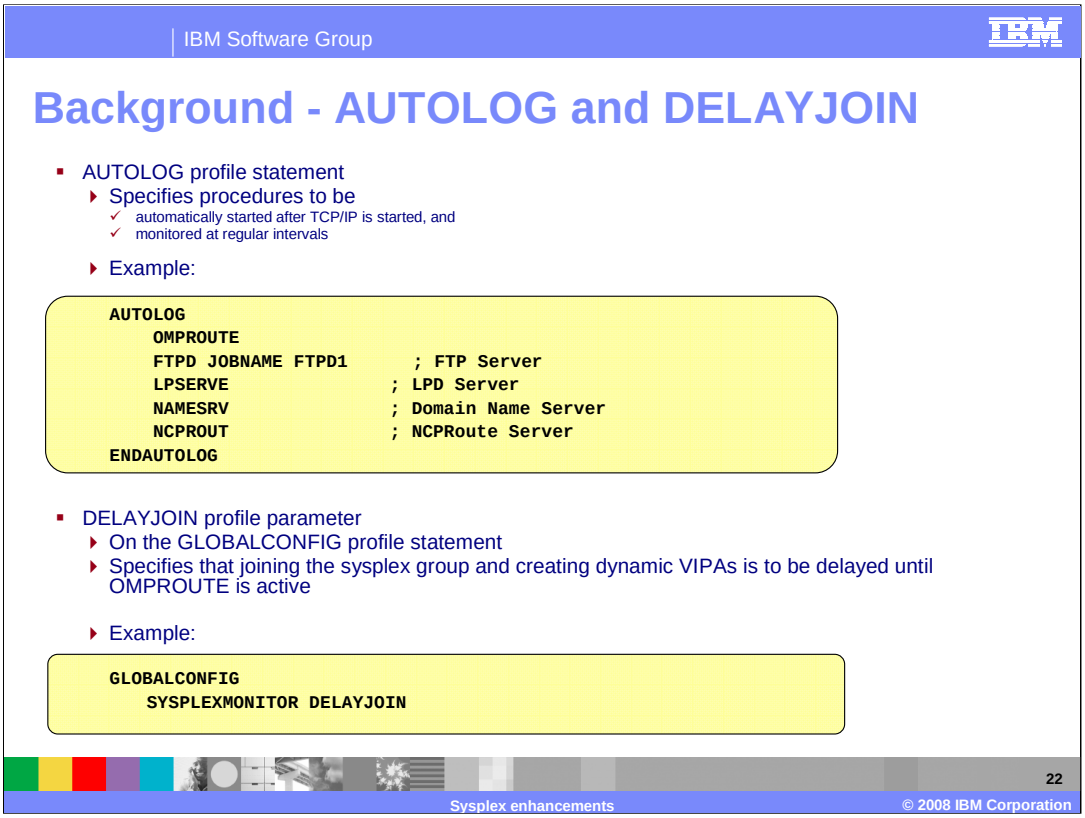
Things to think about

- Explicit Bind Port Range considerations
  - If EBPR is used in a sysplex containing pre-V1R9 systems
    - ✓ Stacks in the sysplex not using EBPR must either be V1R9 or specify a portrange statement reserving the EBPR range
      - – This prevents EBPR port assignments from DVIPA-specific pools
  - The addition of the new EBPR pool managed by the coupling facility may require that the EZBEPORTvvtt structure be resized
  - In a CINET environment with multiple stacks active, an application must establish stack affinity
- SRCIP JOBNAME support for listeners considerations
  - getsockname() after bind() will not retrieve the IP address specified on the matching JOBNAME rule
    - ✓ The IP address substitution is not made until the connect() or listen()
    - ✓ This is different from the operation of the PORT statement with the BIND parameter. There, the IP address is available after the bind().
  - When using a SRCIP JOBNAME rule for an IPv6 server application, an IPv6 address and not an IPv6 Interface should be specified.
    - ✓ If an interface is specified, TCP might not select the best IPv6 address for the application to be bound to.

This slide describes how to handle mixed configurations, that include z/OS V1R9 Communications Server systems and pre-V1R9 systems when using the explicit bind port range function. Also, in a CINET environment if there is no stack affinity and multiple stacks, CINET will assign a port before the stack even receives the Bind. Therefore the explicit bind port range processing will not occur for the request.
Applications that bind to INADDR(6)_ANY, and match on a SRCIP JOBNAME or DESTINATION statement, will not have the designated IP address as its source address upon completion of the bind() call. The source address will not be set to the designated address until completion of the subsequent connect() (client applications) or listen() (server applications) call.
When using the BIND parameter on the PORT statement, the designated IP address will be set upon completion of the bind() call.
If an IPv6 server application matches a JOBNAME rule specifying an IPv6 Interface, rather than an IPv6 address, TCP will choose the first IPv6 address in the interface as the Bind() address. This may not be the most appropriate IP address for the inbound connections.



The enhancement for AUTOLOG is concerned with the timing of when AUTOLOG starts a procedure.

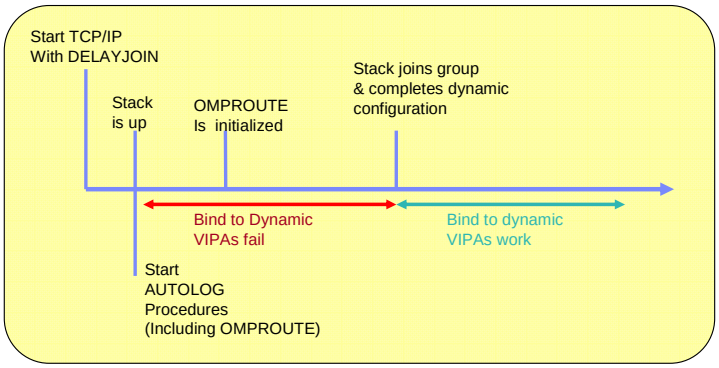


The procedures to be automatically started as soon as TCP/IP has been initialized are specified on the AUTOLOG profile statement. The example shown here will cause AUTOLOG to start the five specified procedures as soon as TCP/IP initialization is completed.

DELAYJOIN is another configuration parameter. It is specified on the GLOBALCONFIG profile statement, as shown in the example on this slide. When DELAYJOIN is specified, TCP/IP will not join the sysplex group until OMPROUTE is active . Since the stack's dynamic VIPA configuration is not processed until after the stack has joined the sysplex group, this delay in joining the sysplex group ensures that OMPROUTE will be active and ready to advertise dynamic VIPAs when they are created on this stack.

Problem - Using AUTOLOG with DELAYJOIN

- When DELAYJOIN is configured
  - AUTOLOGed procedures may be started before OMPROUTE is active
  - Binds to dynamic VIPAs will fail until
    - ✓ OMPROUTE is initialized
    - ✓ TCP/IP has joined the sysplex group and created the dynamic VIPAs



When DELAYJOIN is configured, the stack will not join the sysplex group and create dynamic VIPAs until OMPROUTE signals that it is ready to advertise them. OMPROUTE and the other AUTOLOGed procedures will be started at the same time. AUTOLOGed procedures that bind to dynamic VIPAs may fail due to the delay in creating these DVIPAs.

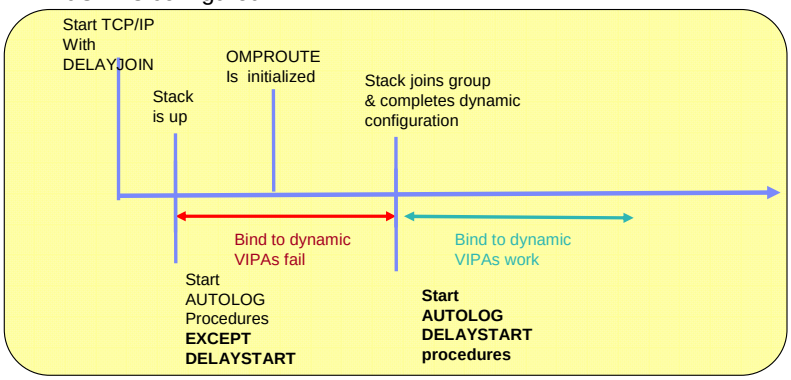
This slide contains a time line showing what happens to bind requests to dynamic VIPAs when TCP/IP is started with DELAYJOIN specified.

When the TCP/IP stack is started with DELAYJOIN configured,

1. the stack completes its basic initialization (the stack is up) but it is not in the sysplex group yet.
2. At this point, AUTOLOG will start the specified procedures. In this example, this includes OMPROUTE.
3. When OMPROUTE has completed its initialization and is active, it notifies the stack.
4. Now the stack will join the sysplex group and then process its dynamic configuration, which includes creating its dynamic VIPAs. Until the stack has completed its dynamic configuration processing, any bind request to a dynamic VIPA will fail.

Solution - Delayed AUTOLOG start

- New optional keyword, DELAYSTART, for procedures specified on the AUTOLOG profile statement
  - Indicates that the procedure is not to be started until after TCP/IP has joined the sysplex group and completed its dynamic sysplex configuration
  - Prevents a procedure from being started (and issuing a bind) before the dynamic VIPAs and VIPARANGEs have been created
- Do not specify AUTOLOG DELAYSTART for OMPROUTE when GLOBALCONFIG DELAYJOIN is configured



z/OS V1R9 Communications Server provides a new optional keyword, DELAYSTART, for procedures specified on the AUTOLOG profile statement. DELAYSTART is used to identify procedures that should not be automatically started until after the stack has joined the sysplex group and its dynamic sysplex configuration is completed. At that point, bind requests to dynamic VIPAs can succeed.

One word of caution: when DELAYJOIN is configured, do not specify DELAYSTART for your OMPROUTE procedure. If you do, the stack will complete its initialization but OMPROUTE will never be started (because AUTOLOG is waiting for the stack to join the sysplex group) and the stack will not join the group and create its dynamic VIPAs because the stack is waiting for OMPROUTE to be active.

The slide contains a time line showing AUTOLOG processing when DELAYSTART is specified for some procedures and not specified for other procedures.

As you saw earlier, when the TCP/IP stack is started with DELAYJOIN configured,

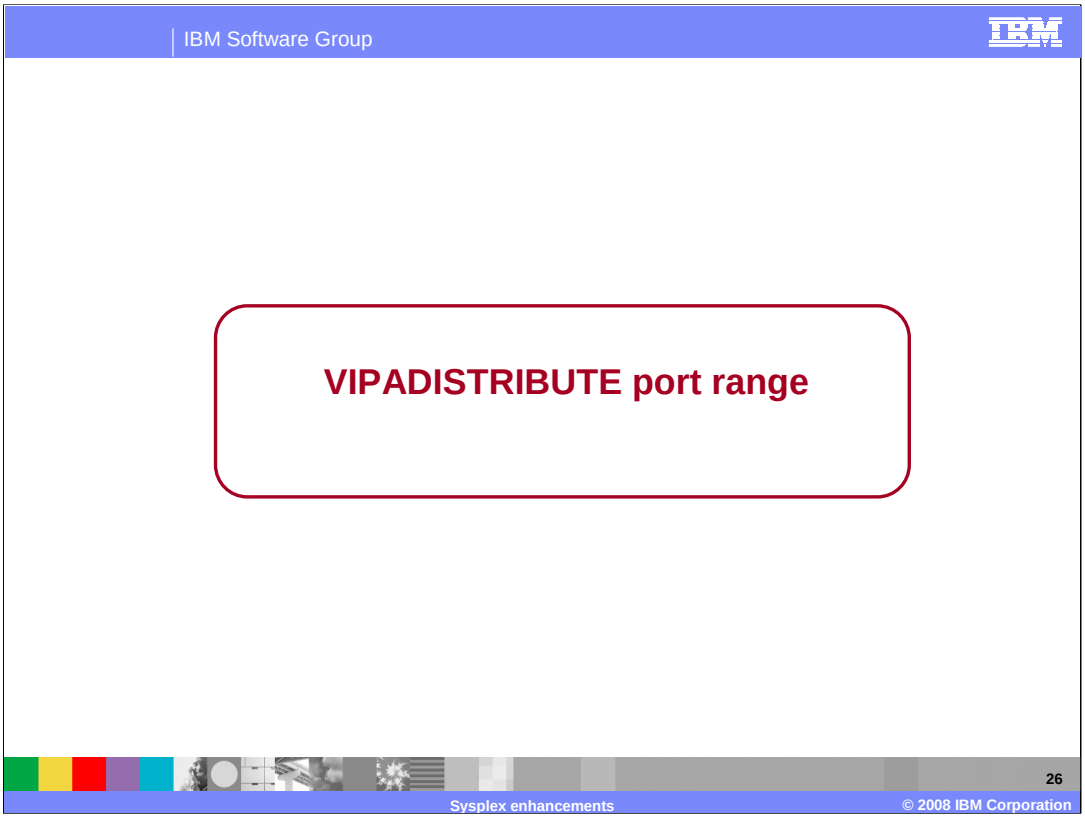
1. the stack completes its basic initialization (the stack is up) but it is not in the sysplex group yet.
2. At this point, AUTOLOG will start the specified procedures. However, it will ignore procedures that have DELAYSTART specified.
3. When OMPROUTE has completed its initialization and is active, it notifies the stack.
4. Now the stack will join the sysplex group and then process its dynamic configuration, which includes creating its dynamic VIPAs. Until the stack has completed its dynamic configuration processing, any bind request to a dynamic VIPA will fail.
5. If DELAYSTART has been configured for any AUTOLOGed procedure, the stack notifies AUTOLOG when the dynamic configuration is complete and AUTOLOG then starts all procedures with DELAYSTART specified.

Display command example

Use Netstat CONFIG/-f to verify whether DELAYJOIN and DELAYSTART are configured

```
Sysplex Monitor:
  TimerSecs: 3600  Recovery: Yes  DelayJoin: Yes  AutoRejoin: Yes
  MonIntf:   No    DynRoute: No
...
Autolog Configuration Information: Wait Time: 0120
ProcName: FTPD      JobName: FTPD1    DelayStart: Yes
  ParmString:
ProcName: OMPROUTE  JobName: OMPROUTE DelayStart: No
  ParmString:
```


The Netstat CONFIG display command can be used to verify the new AUTOLOG DELAYSTART values which are shown in bold type in this example. The same Netstat command also displays the Sysplex Monitor DELAYJOIN value.



With the z/OS V1R9 Communications Server, the VIPADISTRIBUTE profile statement is enhanced to allow you to specify a range of ports.

Background - VIPADISTRIBUTE profile statement

- VIPADISTRIBUTE profile statement is used to configure the distribution targets for connection requests to a dynamic VIPA.
  - A dynamic VIPA and optionally one or more ports may be specified.
- Example:

VIPADISTRIBUTE 9.2.3.4 PORT 3000 3001 3002 DESTIP ALL

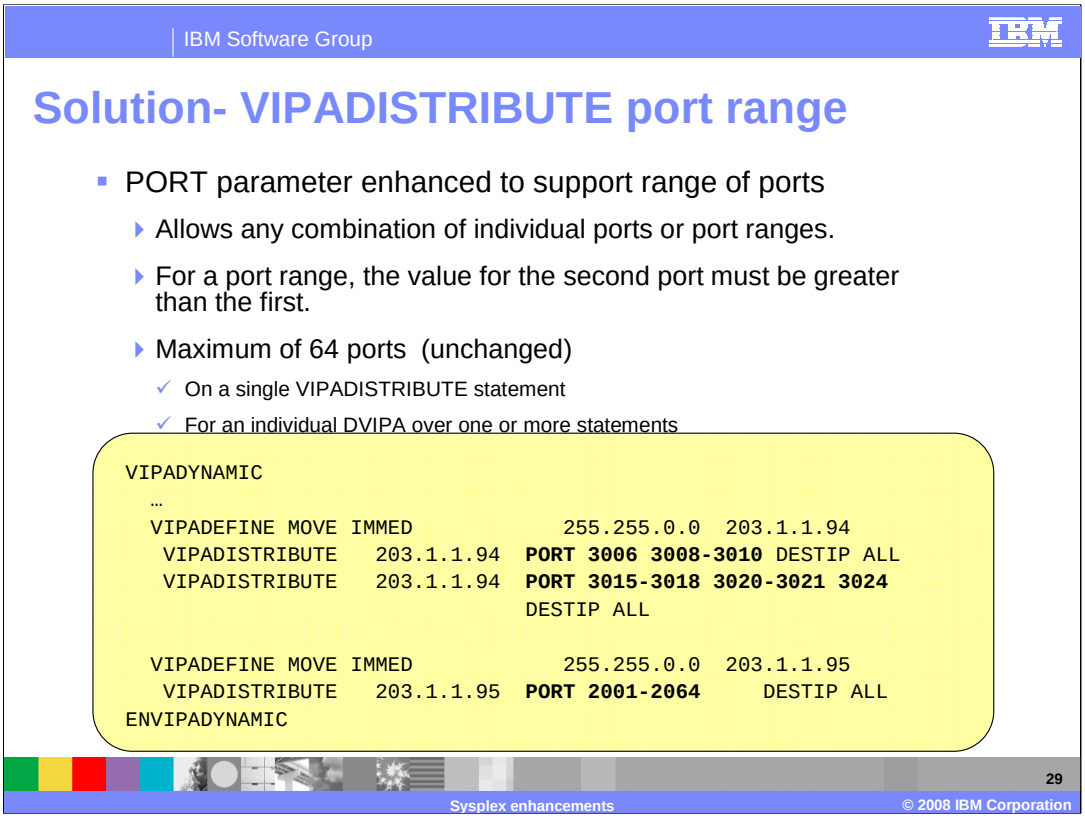
The VIPADISTRIBUTE profile statement specifies how inbound connection requests to a dynamic VIPA are to be distributed among the stacks in the sysplex group. The example on this slide specifies that new connection requests can be forwarded to any stack in the sysplex group which has a socket listening on the dynamic VIPA 9.2.3.4 or inaddr_any, and one of the ports (3000, 3001, 3002) .

Problem - Ports must be specified individually

- Up to 64 specified ports may be specified for a distributed DVIPA
- Currently each port must be individually specified

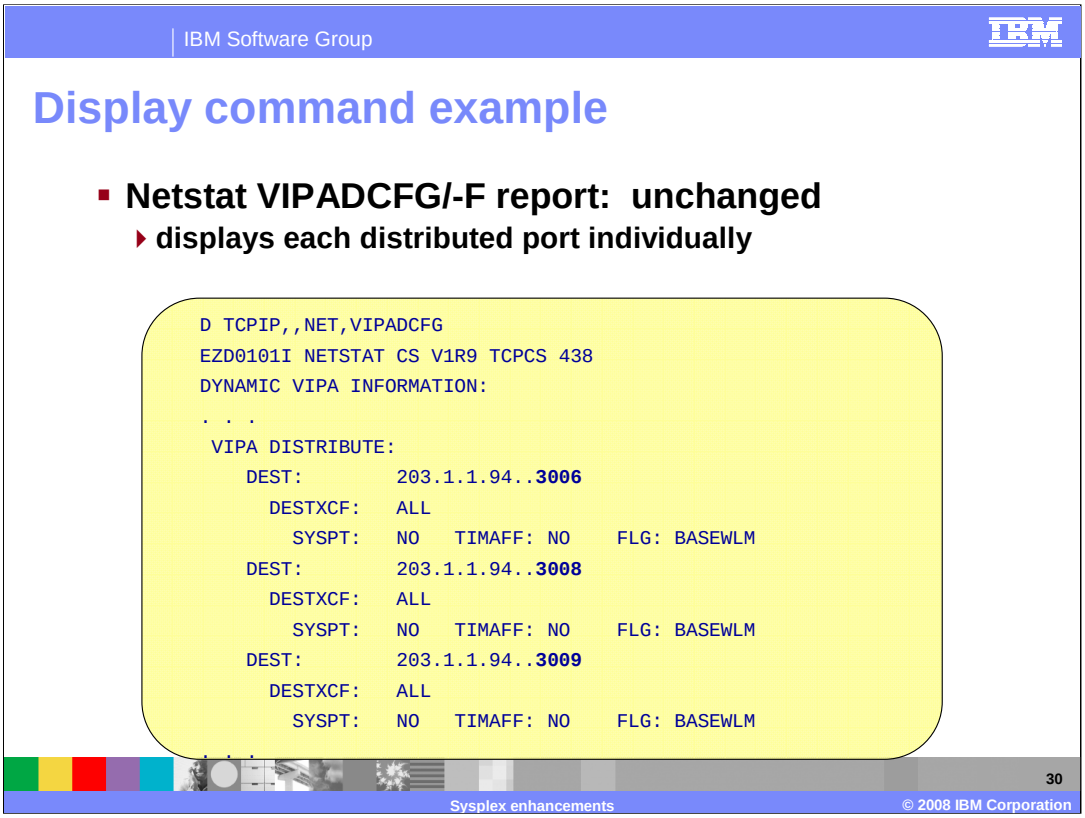
```
VIPADISTRIBUTE  9.2.3.4
  PORT
   3001 3002 3003  3004 3005  3006  3007  3008  3009 3010
   3011 3012 3013  3014 3015  3016  3017  3018 3019  3020
   3021 3022 3023  3024 3025  3026  3027  3028 3029  3030
   …
   3051 3052 3053  3054 3055  3056  3057  3058 3059  3060
   3061 3062 3063  3064

  DESTIP  ALL
```


On a VIPADISTRIBUTE statement, you can specify up to 64 ports, each of which must be individually specified. The example on this slide shows a VIPADISTRIBUTE statement that specifies ports 3001 through 3064.

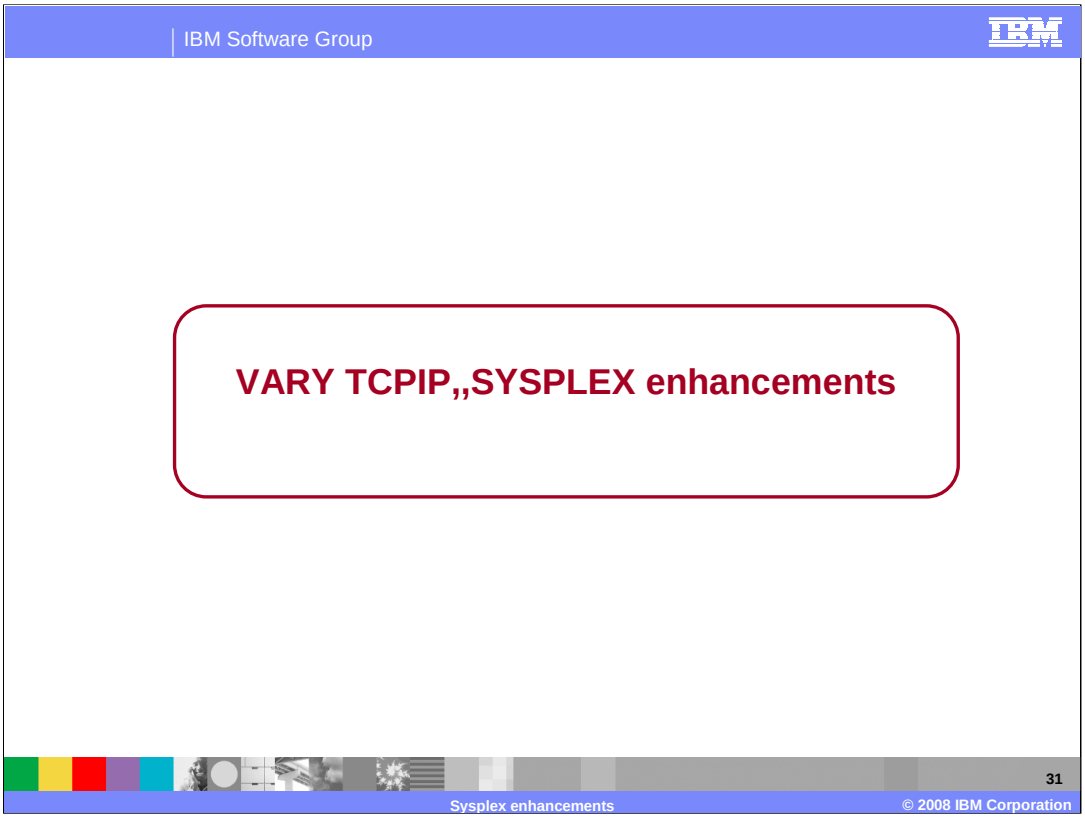


With the z/OS V1R9 Communications Server, the VIPADISTRIBUTE statement syntax is more flexible. It will now accept individual port numbers, a range of port numbers, or a combination of individual ports and ranges for the PORT keyword. The maximum number of ports that can be specified on a VIPADISTRIBUTE statement or for a DVIPA over multiple statements remains 64.

The example on this slide contains a VIPADYNAMIC block containing several VIPADISTRIBUTE statements that demonstrate the new syntax for the PORT values. Connections to DVIPA 203.1.1.94 are to be distributed for ports 3006, 3008, 3009, and 3010 (because of the first VIPADISTRIBUTE statement) and ports 3015, 3016, 3017, 3018, 3020, 3021, and 3024 (because of the second VIPADISTRIBUTE statement). Connections to DVIPA 203.1.1.95 are to be distributed for the full range of 64 ports from 2001 through 2064



The Netstat VIPADCFG command displays the distribution specifications that are configured for dynamic VIPAs on this stack. Although you can now specify ranges of port numbers on the VIPADISTRIBUTE statement, this Netstat report has not changed. As in previous releases, it displays distribution specifications for each individual DVIPA-port combination.

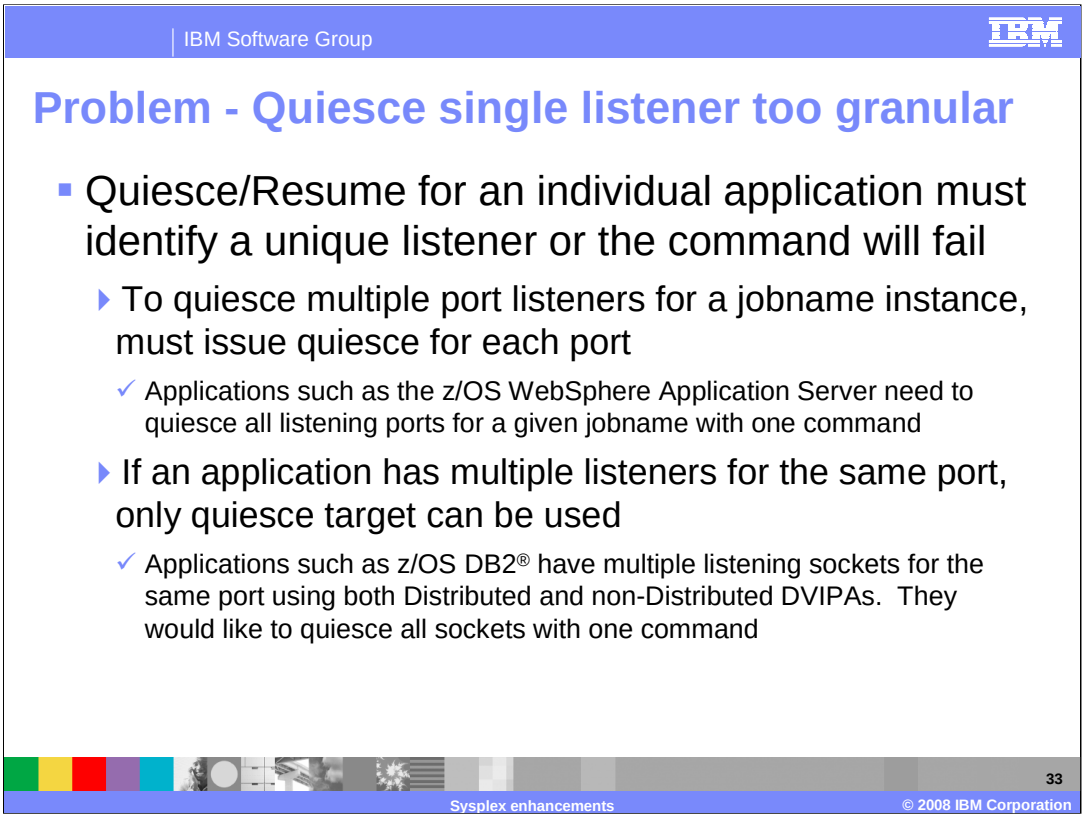


The next slides describe an enhancement to the Sysplex distributor quiesce/resume command

Background information

- The VARY TCPIP,,SYSPLEX command allows operators to quiesce/resume applications from sysplex distribution
- Quiesce/Resume sysplex distribution for individual applications identified by port and, optionally, jobname and asid
- Quiesce/Resume sysplex distribution for all applications on a target stack
- No impact to existing connections
- Must be issued on the stack where the application runs

This is the background information that explains how the existing quiesce and resume functions work.



The existing VARY TCPIP,,SYSPLEX command allows operators to quiesce or resume applications from sysplex distribution. These operations do not impact the existing connections. Furthermore the command must be issued on the stack where the application is running.
In prior releases to quiesce multiple port listeners for a jobname instance, an operator must issue the quiesce flavor of the command for each port. An application instance may have listeners on different ports. Each of them must be quiesced with a separate command. For example:
Quiesce Port=50001,Jobname=jobxyz

Quiesce Port=50002,Jobname=jobxyz

…

Quiesce Port=50025,Jobname=jobxyz

In prior releases, if an application instance has multiple listeners for the same port, only quiesce target can be used. An application may have multiple listeners on port 50001 with the same asid (71). This command will fail since a unique listener is not identified:

Quiesce Port=50001,jobname=jobxyz,asid=71

The only option is Quiesce Target, but this will quiesce **all** target applications on this stack from sysplex distribution.

Solution - Allow single quiesce for multiple listeners

When the command is issued all matching listeners must have the same jobname and asid, but not port

- New - Quiesce using jobname (and optionally asid)
  - Quiesce jobname - all matching listeners, but they must have the same asid
  - Quiesce jobname, asid – quiesce all matching listeners
  - All matching listeners quiesced regardless of port
- Existing - Quiesce using port (and optionally jobname and asid)
  - Quiesce port – quiesce all matching listeners if they have the same jobname and asid
  - Quiesce port, jobname - quiesce all matching listeners if they have the same asid
  - Quiesce using port, jobname, asid - quiesce all matching listeners

A new flavor of the Quiesce and Resume Sysplex command is now allowed. A user can now quiesce and resume applications using the jobname and optionally the associated asid. When jobname is used and all listeners identified by jobname do not have the same asid, the command will fail. The listeners will not be quiesced or resumed.

The existing Quiesce/Resume, port or Quiesce/Resume, port,jobname command is changed in z/OS V1R9. When either of these flavors of the command is used, if all listeners do not have the same jobname and asid, the command will fail. The listeners will not be quiesced or resumed. In previous releases, if a unique listener was not identified, the command would fail.

Command syntax

Vary TCPIP SYSplex quiesce/resume enhancements

```
>>-Vary --TCPIP--,--+----------+--,SYSplex--------------------->
                    '-procname-'

>-+-QUIesce,POrt=portnum-+-------------------------------+
                         +-,JOBNAME=jobname-+------------+
                                            +-,ASID=asid-+
  +-QUIesce,JOBNAME=jobname-+------------+
                            +-,ASID=asid-+
  +-QUIesce,TARGET

  +-RESUME,POrt=portnum--+--------------------------------+
                         +-,JOBNAME=jobname-+------------+-+
                                            +-,ASID=asid-+
  +-RESUME,JOBNAME=jobname-+------------+
                           +-,ASID=asid-+
  +-RESUME,TARGET
```


This slide shows the various ways the quiesce/resume command can now be issued. Vary TCPIP Sysplex also supports Leavegroup, Joingroup, Deactivate, and Reactivate; these are not shown here.

Things to think about

- The command must be issued on the target stack where the listening application is
- Quiesce using Port is changed to quiesce multiple listeners (with matching jobname and asid).
  - Previously if more than one listener was identified, the command would fail.

The first bullet of this slide emphasizes a restriction. The second bullet points out how the Quiesce Port flavor of the command works differently from previous releases.

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 e-mail feedback:

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

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

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

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2 VTAM z/OS

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.