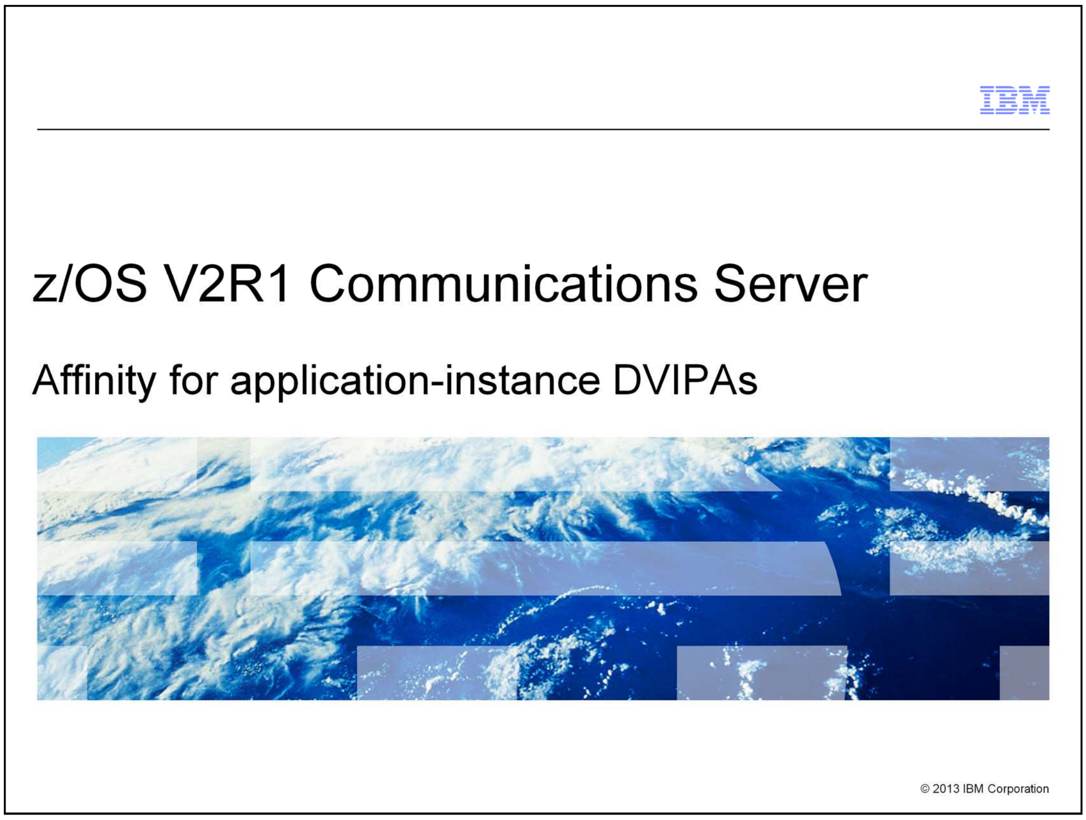
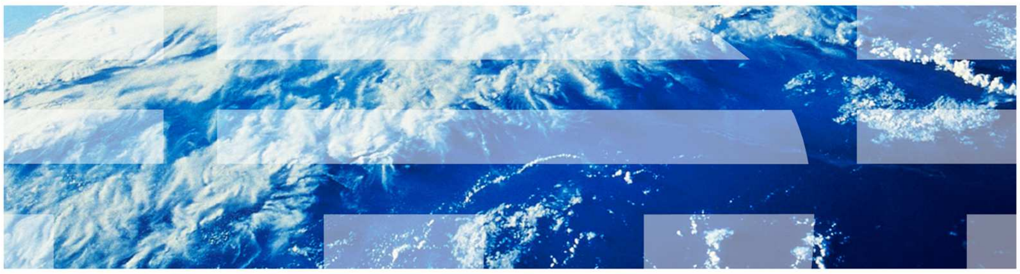


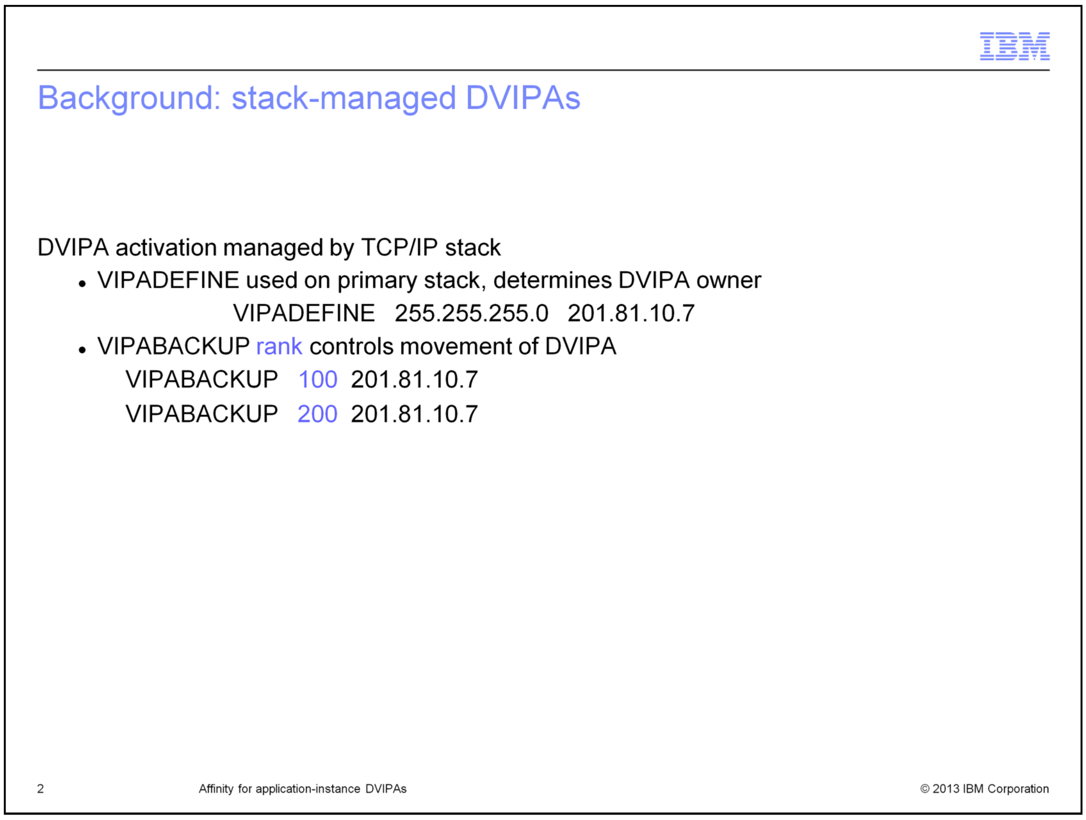
z/OS V2R1 Communications Server

Affinity for application-instance DVIPAs



© 2013 IBM Corporation

This presentation describes the support to provide affinity for application-instance DVIPAs.



Background: stack-managed DVIPAs

DVIPA activation managed by TCP/IP stack

- VIPADEFINE used on primary stack, determines DVIPA owner

```
VIPADEFINE   255.255.255.0   201.81.10.7
```

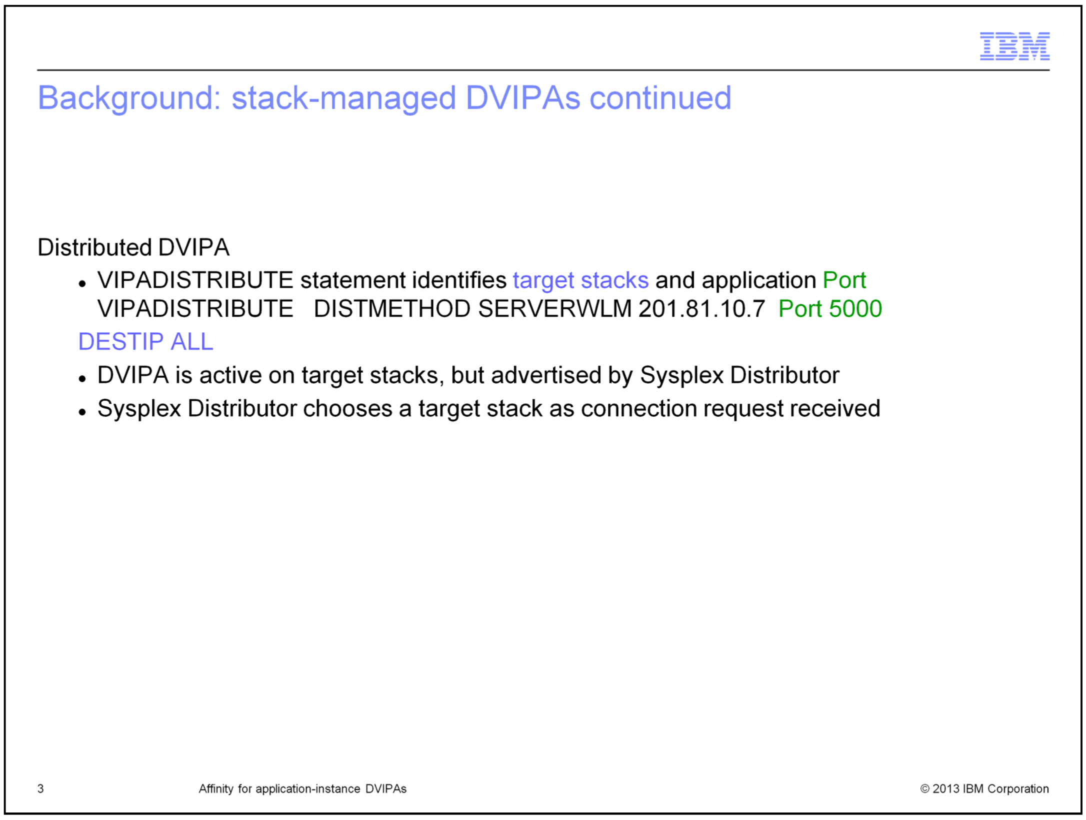
- VIPABACKUP rank controls movement of DVIPA

```
VIPABACKUP   100  201.81.10.7
VIPABACKUP   200  201.81.10.7
```

2 Affinity for application-instance DVIPAs © 2013 IBM Corporation

A stack-managed DVIPA is one in which the activation of the DVIPA and movement of the DVIPA is controlled by the TCP/IP stacks. A VIPADEFINE statement is configured on the stack that will *own* the DVIPA. As the stack is started, the DVIPA is activated and advertised by the *owning* stack. The slide contains an example of this configuration.

VIPABACKUP statements are configured on each stack that will back up the stack that owns the DVIPA. If the owning stack gives up ownership of the DVIPA, a back up stack will take over the DVIPA. The owning stack might give up ownership because the stack leaves the sysplex or because the DVIPA is deactivated or deleted. The stack with highest rank VIPABACKUP statement takes over the DVIPA becoming the new *owner*. In the example, the stack with the VIPABACKUP statement with a rank of 200 will take over if the VIPADEFINE stack gives up ownership of the DVIPA.



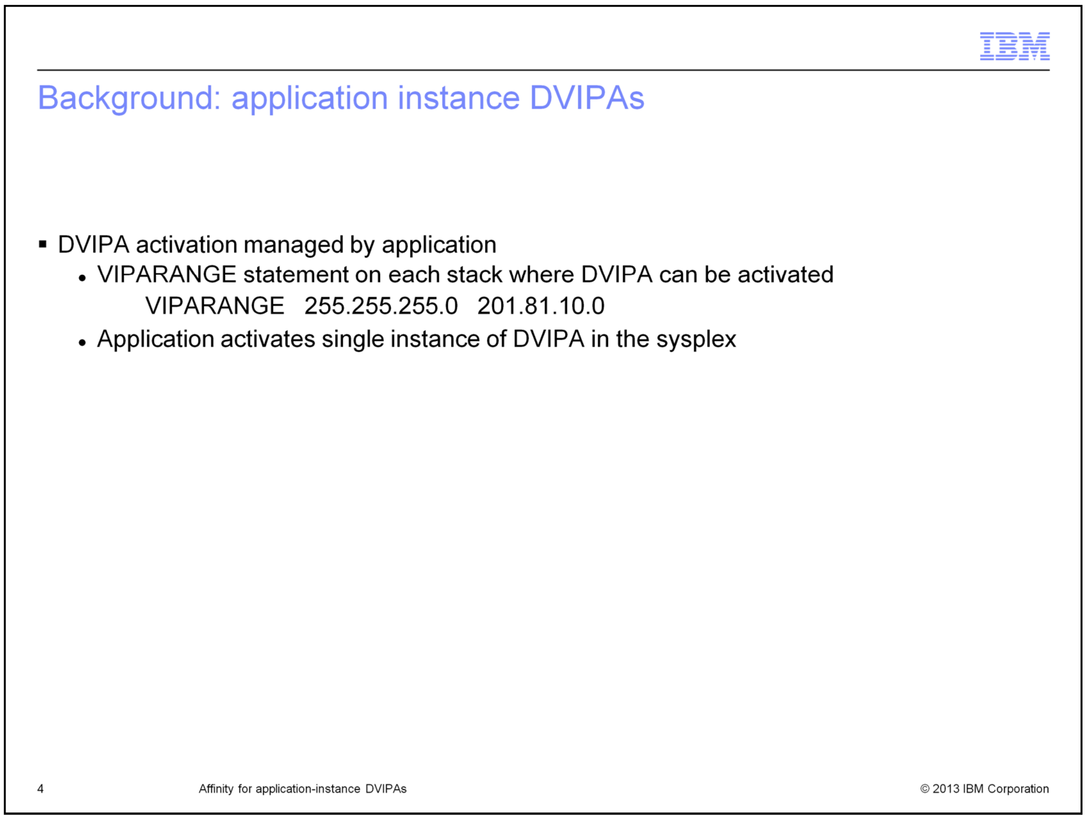
Background: stack-managed DVIPAs continued

Distributed DVIPA

- VIPADISTRIBUTE statement identifies target stacks and application Port
  VIPADISTRIBUTE   DISTMETHOD SERVERWLM 201.81.10.7 Port 5000
  DESTIP ALL
- DVIPA is active on target stacks, but advertised by Sysplex Distributor
- Sysplex Distributor chooses a target stack as connection request received

3   Affinity for application-instance DVIPAs   © 2013 IBM Corporation

A distributed DVIPA is used when multiple instances of an application are concurrently active on multiple stacks. In addition to the VIPADEFINE statement, a VIPADISTRIBUTE statement identifies target stacks where the application is active and the application's port.



Background: application instance DVIPAs

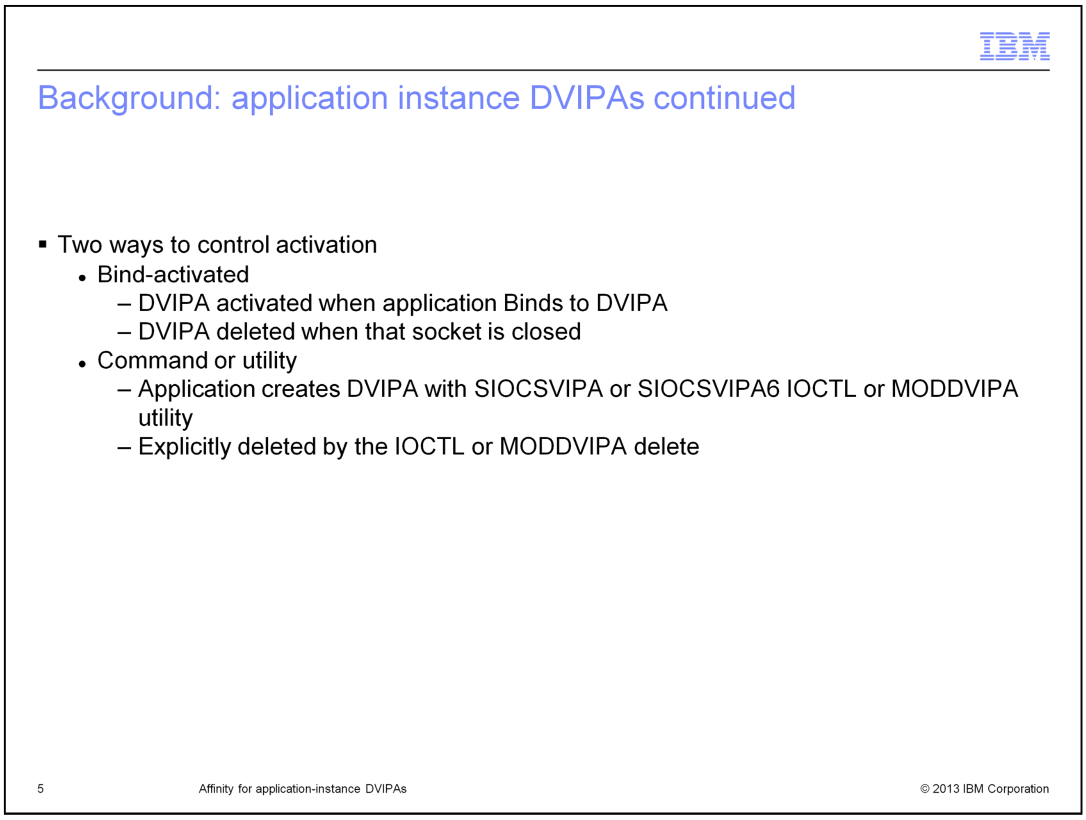
- DVIPA activation managed by application
  - VIPARANGE statement on each stack where DVIPA can be activated

    VIPARANGE 255.255.255.0 201.81.10.0
  - Application activates single instance of DVIPA in the sysplex

4 Affinity for application-instance DVIPAs © 2013 IBM Corporation

Another type of Dynamic VIPA is an application instance DVIPA. The activation and movement of this DVIPA is controlled by the application.

A VIPARANGE statement determines the range of application instance DVIPAs that can be activated on a stack. In the example, the VIPARANGE statement allows an application to create DVIPAs ranging from 201.81.10.1 - 201.81.10.254 on this stack.



Background: application instance DVIPAs continued

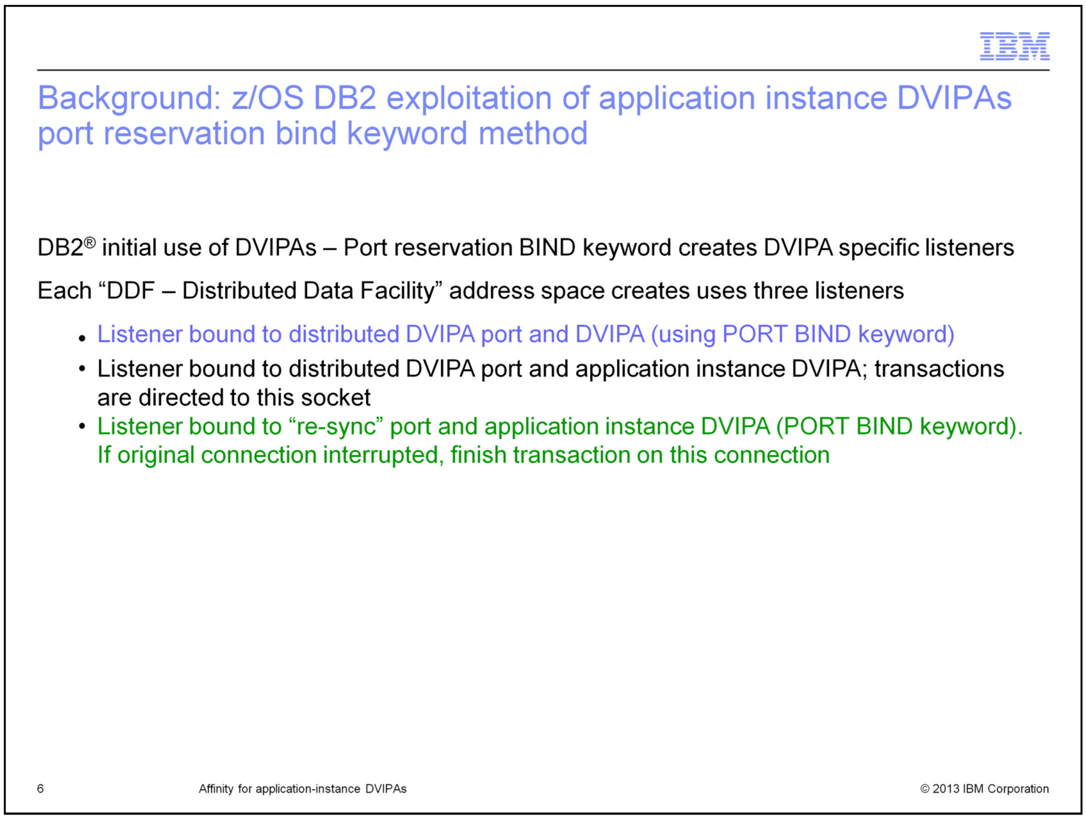
- Two ways to control activation
  - Bind-activated
    - DVIPA activated when application Binds to DVIPA
    - DVIPA deleted when that socket is closed
  - Command or utility
    - Application creates DVIPA with SIOCSVIPA or SIOCSVIPA6 IOCTL or MODDVIPA utility
    - Explicitly deleted by the IOCTL or MODDVIPA delete

5 Affinity for application-instance DVIPAs © 2013 IBM Corporation

There are two ways to control activation.

The DVIPA can be activated when the application issues a Bind explicitly specifying the DVIPA or it can be activated by using the BIND keyword on the Port statement. When the application binds to the port and inaddr_any (IPv4) or the unspecified address (IPv6), the IP address on the PORT BIND keyword is used. The DVIPA is deleted when the socket that issued the Bind is closed.

This type of DVIPA can also be activated by using the SIOCSVIPA (IPv4) or SIOCSVIPA6 (IPv6) IOCTL command (or the MODDVIPA utility). The DVIPA is explicitly deleted using the IOCTL or MODDVIPA utility.



Background: z/OS DB2 exploitation of application instance DVIPAs port reservation bind keyword method

DB2® initial use of DVIPAs – Port reservation BIND keyword creates DVIPA specific listeners

Each “DDF – Distributed Data Facility” address space creates uses three listeners

- Listener bound to distributed DVIPA port and DVIPA (using PORT BIND keyword)
- Listener bound to distributed DVIPA port and application instance DVIPA; transactions are directed to this socket
- Listener bound to “re-sync” port and application instance DVIPA (PORT BIND keyword). If original connection interrupted, finish transaction on this connection

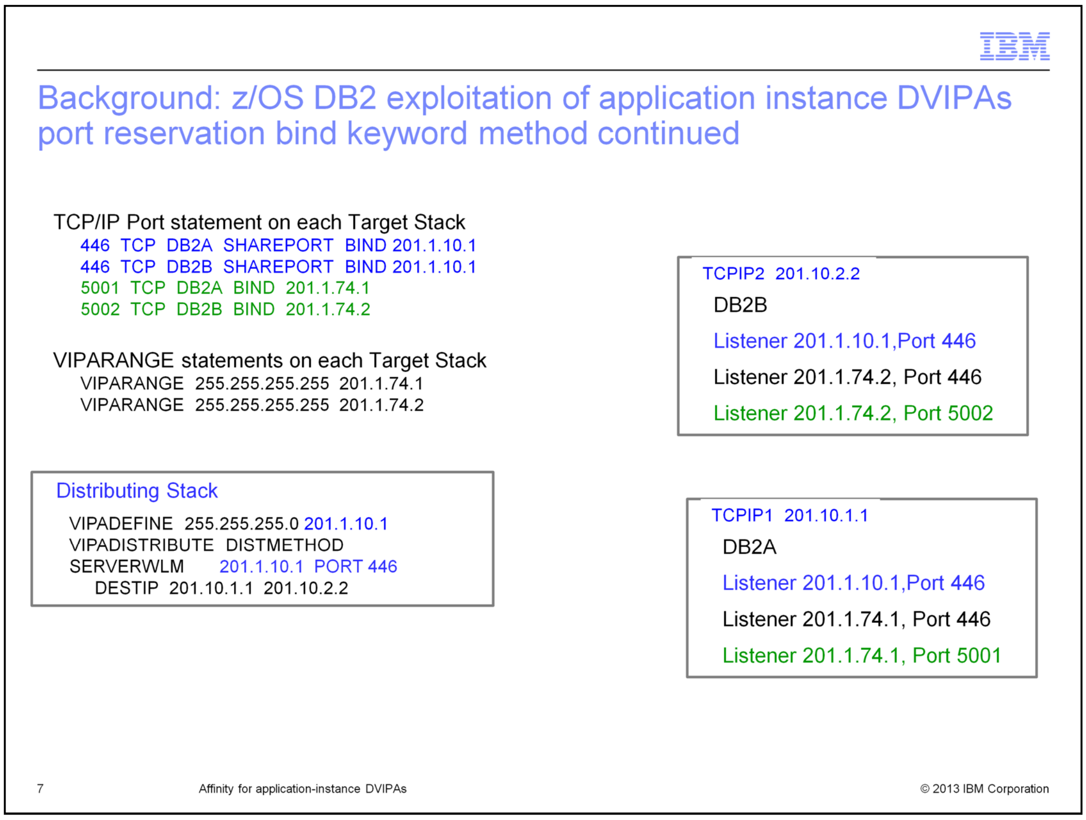
6 Affinity for application-instance DVIPAs © 2013 IBM Corporation

When DB2 first started using DVIPAs, the BIND keyword on the PORT reservation statement was used to bind listeners. Each application opens three listening sockets:

1) A listener bound to the distributed port and DVIPA using the BIND keyword on the PORT reservation statement.

2) A listener bound to its application instance DVIPA and the distributed DVIPA port.

3) A listener bound to a re-sync port and an application instance DVIPA using the BIND keyword on the PORT reservation statement. The re-sync port is used if the connection is interrupted before the transaction completes. The client reconnects to this port to complete the transaction.



Background: z/OS DB2 exploitation of application instance DVIPAs port reservation bind keyword method continued

TCP/IP Port statement on each Target Stack

```
446  TCP  DB2A  SHAREPORT  BIND 201.1.10.1
446  TCP  DB2B  SHAREPORT  BIND 201.1.10.1
5001  TCP  DB2A  BIND  201.1.74.1
5002  TCP  DB2B  BIND  201.1.74.2
```

VIPARANGE statements on each Target Stack

```
VIPARANGE  255.255.255.255  201.1.74.1
VIPARANGE  255.255.255.255  201.1.74.2
```

Distributing Stack

```
VIPADEFINE  255.255.255.0 201.1.10.1
VIPADISTRIBUTE  DISTMETHOD
SERVERWLM      201.1.10.1  PORT 446
   DESTIP  201.10.1.1  201.10.2.2
```

TCPIP2 201.10.2.2

DB2B

Listener 201.1.10.1,Port 446

Listener 201.1.74.2, Port 446

Listener 201.1.74.2, Port 5002

TCPIP1 201.10.1.1

DB2A

Listener 201.1.10.1,Port 446

Listener 201.1.74.1, Port 446

Listener 201.1.74.1, Port 5001

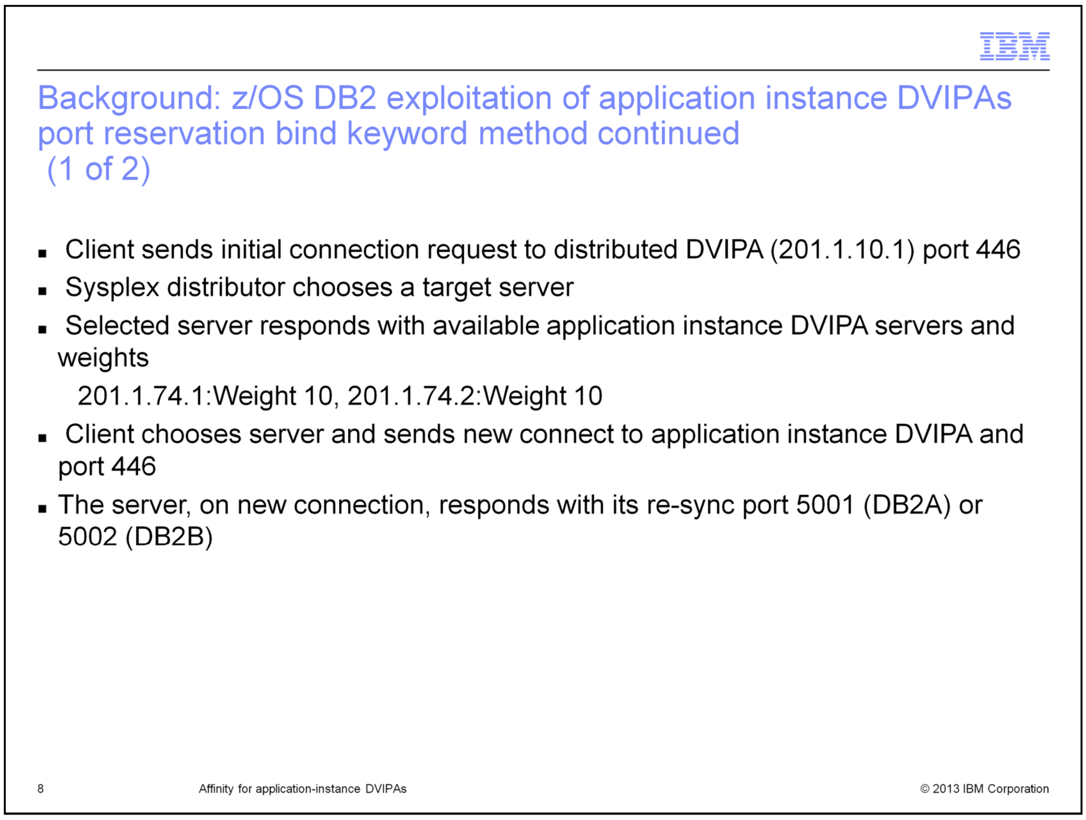
7 Affinity for application-instance DVIPAs © 2013 IBM Corporation

This slide provides an example of the configuration for z/OS® DB2 exploitation of the Application Instance DVIPA function. It shows the configuration necessary when using the BIND keyword on the PORT reservation statement to bind listeners. The applications DB2A and DB2B opens three listening sockets.

1) A listener bound to the distributed port and DVIPA using the BIND keyword on the PORT reservation statement.

2) A listener bound to its application instance DVIPA and the distributed DVIPA port.

3) A listener bound to a re-sync port and an application instance DVIPA using the BIND keyword on the PORT reservation statement. complete the transaction.



Background: z/OS DB2 exploitation of application instance DVIPAs port reservation bind keyword method continued (1 of 2)

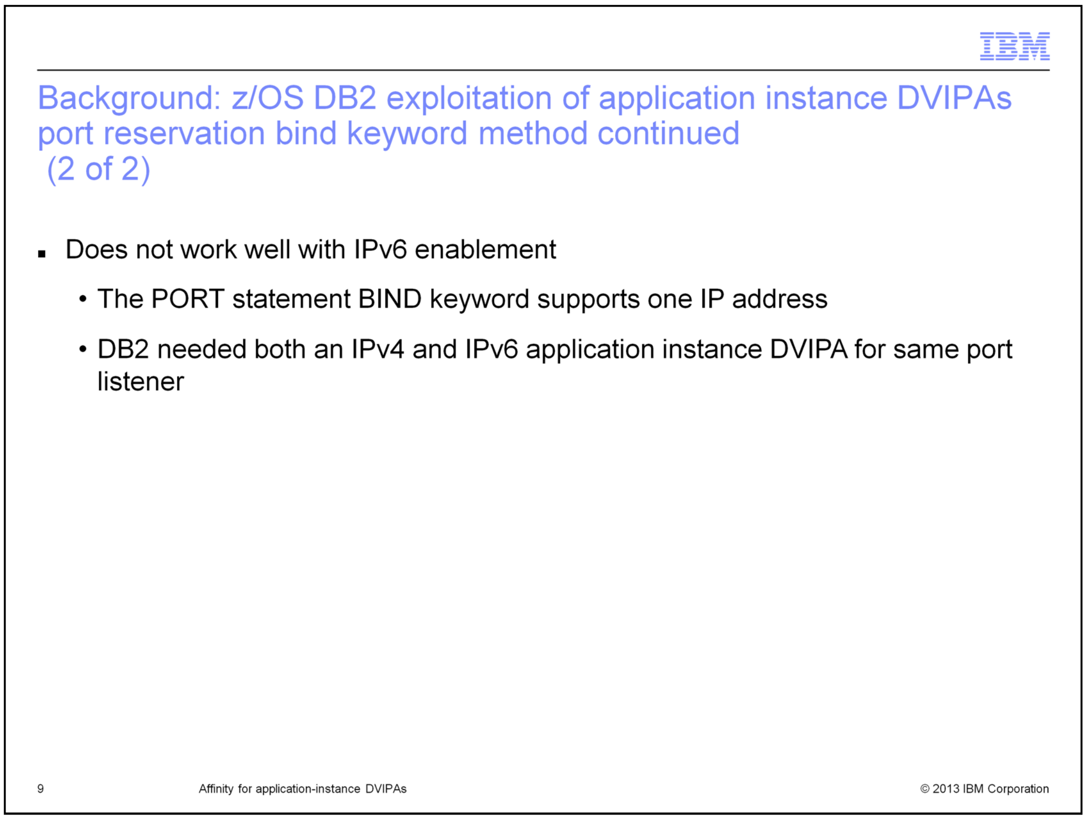
- Client sends initial connection request to distributed DVIPA (201.1.10.1) port 446
- Sysplex distributor chooses a target server
- Selected server responds with available application instance DVIPA servers and weights

  201.1.74.1:Weight 10, 201.1.74.2:Weight 10
- Client chooses server and sends new connect to application instance DVIPA and port 446
- The server, on new connection, responds with its re-sync port 5001 (DB2A) or 5002 (DB2B)

8 Affinity for application-instance DVIPAs © 2013 IBM Corporation

The client sends its initial connection request to the distributed DVIPA and port. The sysplex distributor chooses a server. The server responds to the client with a list of all available application instance servers and their weights. Based on the weight, the client chooses a server, sending a new connection request to the application instance DVIPA for its transaction.

On the new connection, the server responds with its re-sync port which is used if the connection is interrupted before completing the transaction.

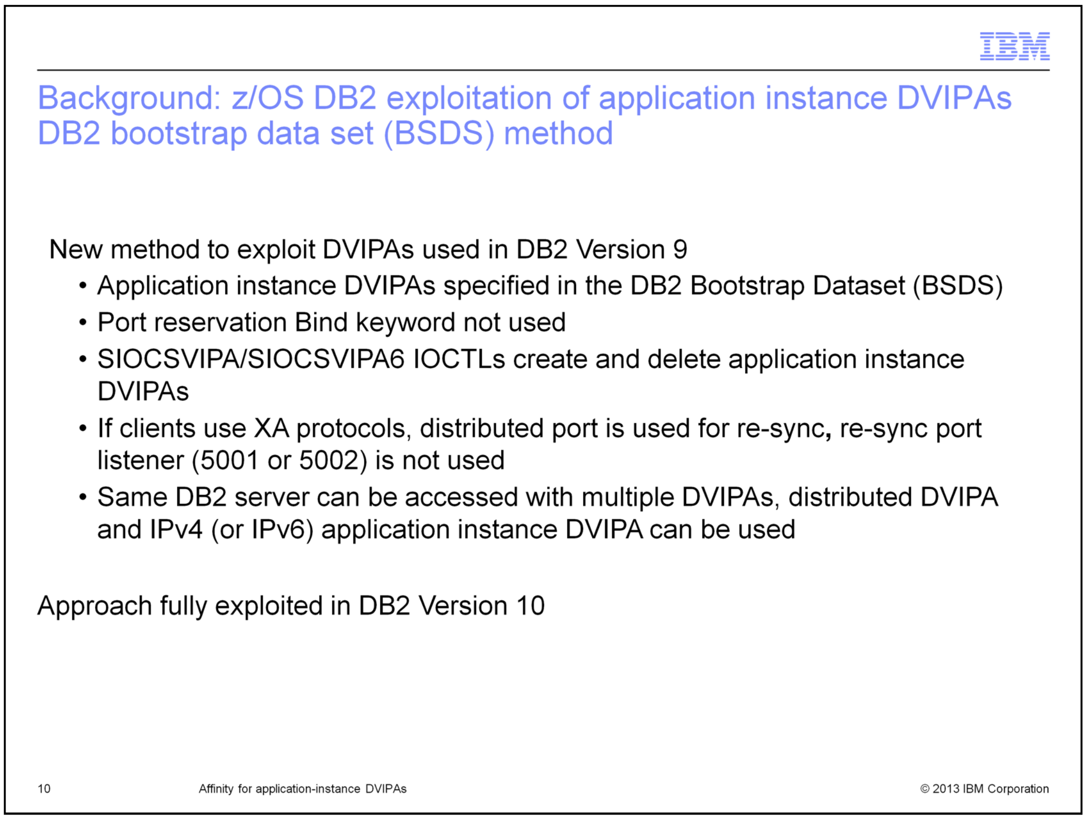


Background: z/OS DB2 exploitation of application instance DVIPAs port reservation bind keyword method continued (2 of 2)

- Does not work well with IPv6 enablement
  - The PORT statement BIND keyword supports one IP address
  - DB2 needed both an IPv4 and IPv6 application instance DVIPA for same port listener

9 Affinity for application-instance DVIPAs © 2013 IBM Corporation

This did not work well for IPv6 enablement. The BIND keyword on the PORT statement only supports a single IP address. DB2 needed to create both an IPv4 and IPv6 application specific DVIPA for the same port listener.



Background: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method

New method to exploit DVIPAs used in DB2 Version 9

- Application instance DVIPAs specified in the DB2 Bootstrap Dataset (BSDS)
- Port reservation Bind keyword not used
- SIOCSVIPA/SIOCSVIPA6 IOCTLs create and delete application instance DVIPAs
- If clients use XA protocols, distributed port is used for re-sync, re-sync port listener (5001 or 5002) is not used
- Same DB2 server can be accessed with multiple DVIPAs, distributed DVIPA and IPv4 (or IPv6) application instance DVIPA can be used

Approach fully exploited in DB2 Version 10

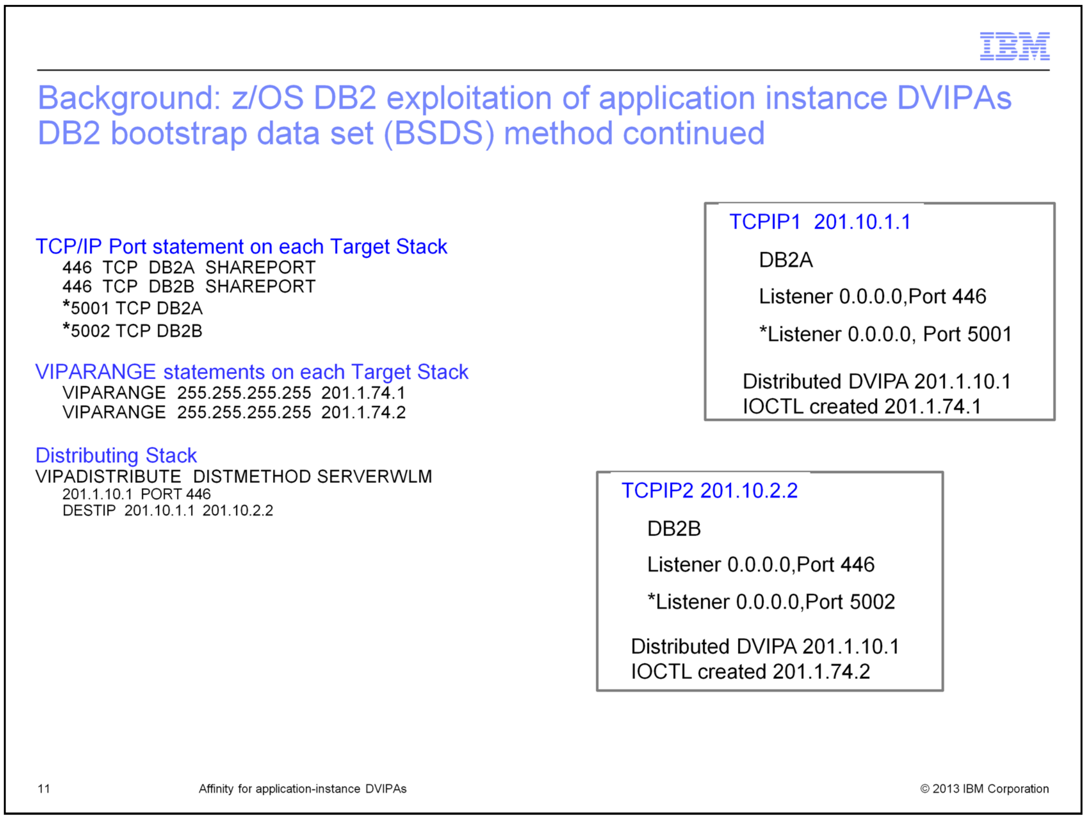
10 Affinity for application-instance DVIPAs © 2013 IBM Corporation

DB2 provided a new method for activating DVIPAs in DB2 V9. All application instance IP addresses are specified in the DB2 Bootstrap Dataset (BSDS). Instead of using the Port reservation Bind keyword, DB2 uses a single listener bound to the distributed DVIPA port and inaddr_any for IPv4 or the unspecified address for IPv6. SIOCSVIPA/SIOCSVIPA6 IOCTLS are used to create and delete application instance DVIPAs as needed.

If clients are using XA protocols for transaction coordination, the distributed port is also used for re-sync. A re-sync port is not used.

Using this method the DB2 server can be accessed with multiple DVIPAs; the distributed DVIPA and an IPv4 or IPv6 application instance DVIPA can all be used with the same listener.

This approach is fully exploited in DB2 V10 and is now the preferred mechanism for configuring DVIPAs for DB2.



Background: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method continued

TCP/IP Port statement on each Target Stack

```
    446  TCP  DB2A  SHAREPORT
    446  TCP  DB2B  SHAREPORT
    *5001 TCP DB2A
    *5002 TCP DB2B
```

VIPARANGE statements on each Target Stack

```
    VIPARANGE  255.255.255.255  201.1.74.1
    VIPARANGE  255.255.255.255  201.1.74.2
```

Distributing Stack

```
VIPADISTRIBUTE  DISTMETHOD SERVERWLM
    201.1.10.1  PORT 446
    DESTIP  201.10.1.1  201.10.2.2
```

TCPIP1 201.10.1.1

DB2A

Listener 0.0.0.0,Port 446

*Listener 0.0.0.0, Port 5001

Distributed DVIPA 201.1.10.1
IOCTL created 201.1.74.1

TCPIP2 201.10.2.2

DB2B

Listener 0.0.0.0,Port 446

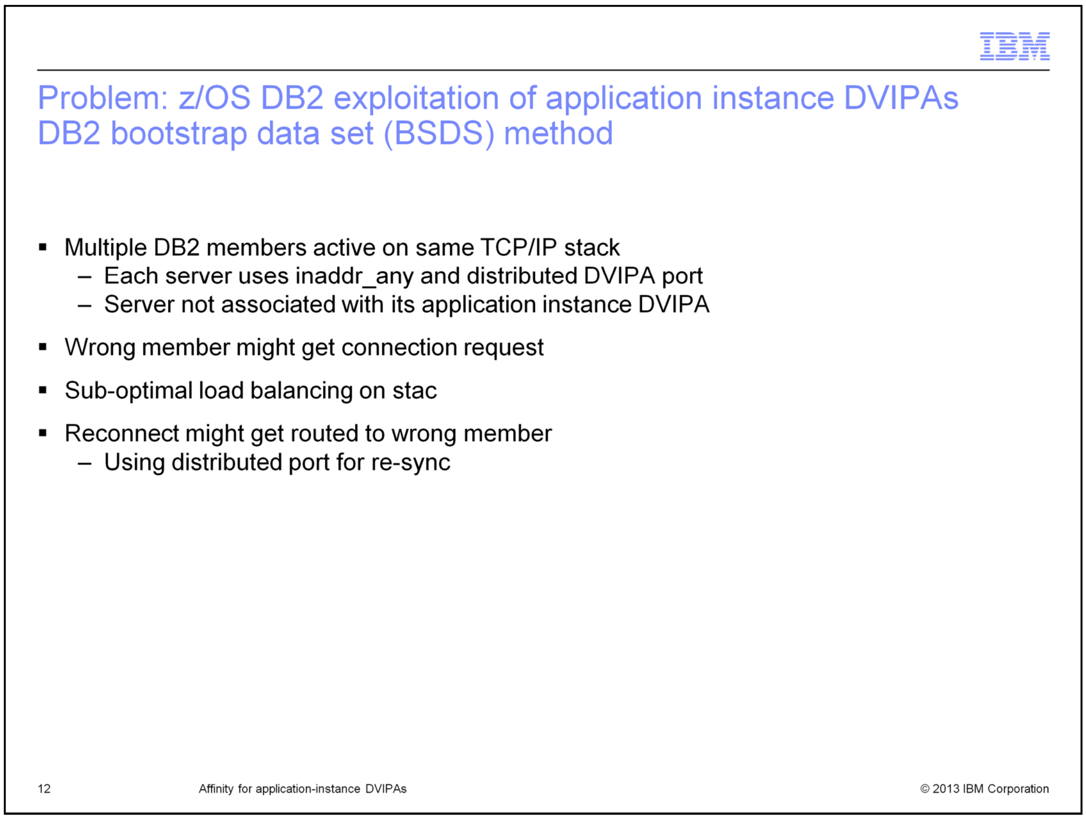
*Listener 0.0.0.0,Port 5002

Distributed DVIPA 201.1.10.1
IOCTL created 201.1.74.2

11 Affinity for application-instance DVIPAs © 2013 IBM Corporation

Here is a possible TCP/IP configuration when the DB2 Bootstrap Dataset method is used. Note that the Port reservation Bind keyword is not used. Instead, DB2 uses a single listener bound to the distributed DVIPA port and inaddr_any for IPv4 or the unspecified address for IPv6.

The DB2 server can be accessed with multiple DVIPAs; the distributed DVIPA and an IPv4 or IPv6 application instance DVIPA can all be used with the same listener.



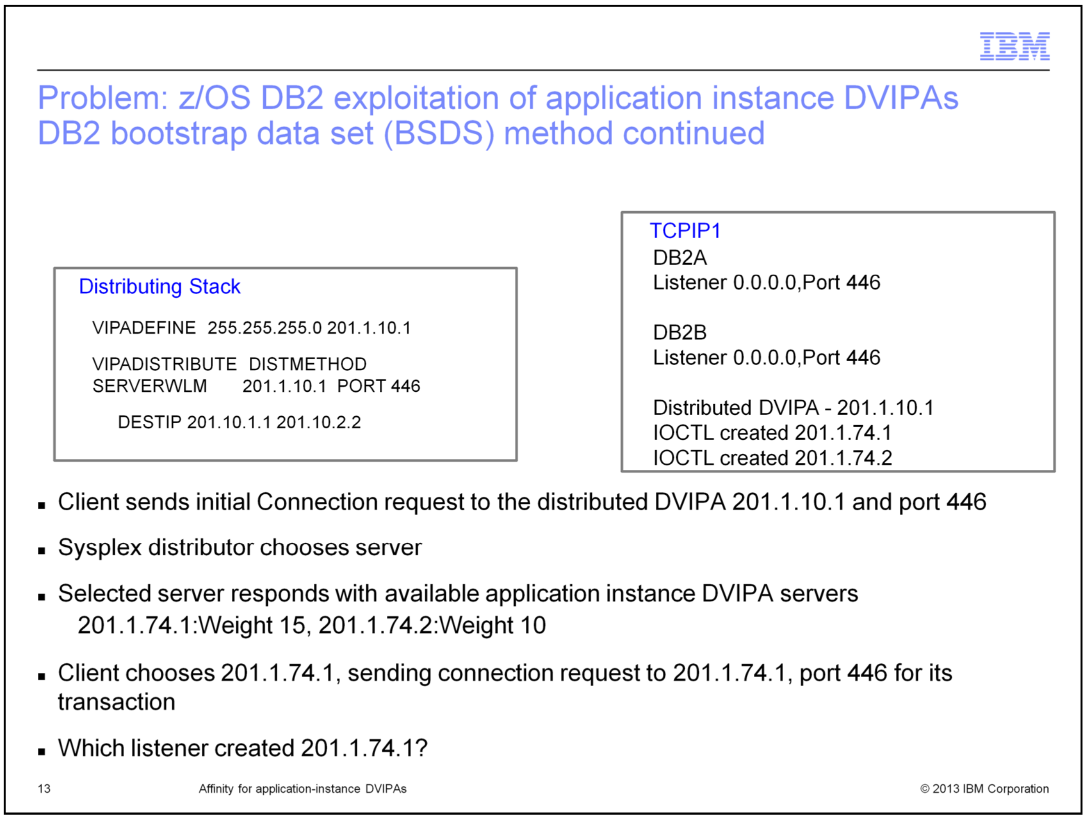
Problem: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method

- Multiple DB2 members active on same TCP/IP stack
  - Each server uses inaddr_any and distributed DVIPA port
  - Server not associated with its application instance DVIPA
- Wrong member might get connection request
- Sub-optimal load balancing on stac
- Reconnect might get routed to wrong member
  - Using distributed port for re-sync

12 Affinity for application-instance DVIPAs © 2013 IBM Corporation

There is a problem when multiple DB2 members are active on the same TCP/IP stack. Since there is only one server for each application using inaddr_any and the same distributed DVIPA port. There is no way to associate a specific application instance DVIPA with a listener.

This can result in the wrong member getting a connect request which results in sub-optimal load balancing on the stack. A reconnect might get routed to the wrong application if the distributed port is used for re-sync causing the transaction to fail.



Problem: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method continued

Distributing Stack

```
VIPADEFINE  255.255.255.0 201.1.10.1

VIPADISTRIBUTE  DISTMETHOD
SERVERWLM      201.1.10.1  PORT 446

    DESTIP 201.10.1.1 201.10.2.2
```

TCPIP1

DB2A
Listener 0.0.0.0,Port 446

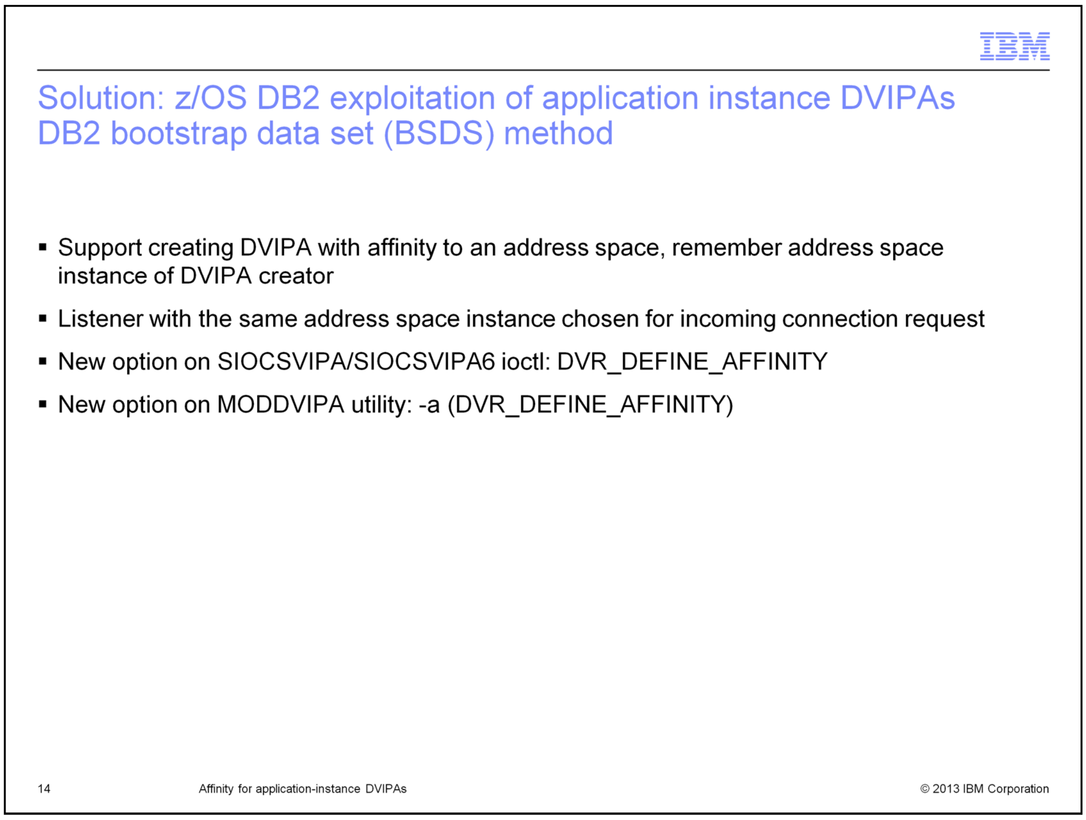
DB2B
Listener 0.0.0.0,Port 446

Distributed DVIPA - 201.1.10.1
IOCTL created 201.1.74.1
IOCTL created 201.1.74.2

- Client sends initial Connection request to the distributed DVIPA 201.1.10.1 and port 446
- Sysplex distributor chooses server
- Selected server responds with available application instance DVIPA servers
  201.1.74.1:Weight 15, 201.1.74.2:Weight 10
- Client chooses 201.1.74.1, sending connection request to 201.1.74.1, port 446 for its transaction
- Which listener created 201.1.74.1?

In the example, the client chooses to use the application instance DVIPA 201.1.74.1. It sends a connect request to this IP address and port 446 for its transaction. But there are two listeners on TCP/IP stack TCPIP1. There is no way to determine which application instance created this DVIPA.

This can result in the wrong member getting a connect request. Even worse, a transaction failure can occur if the distributed port is used for re-sync and the wrong application is chosen.



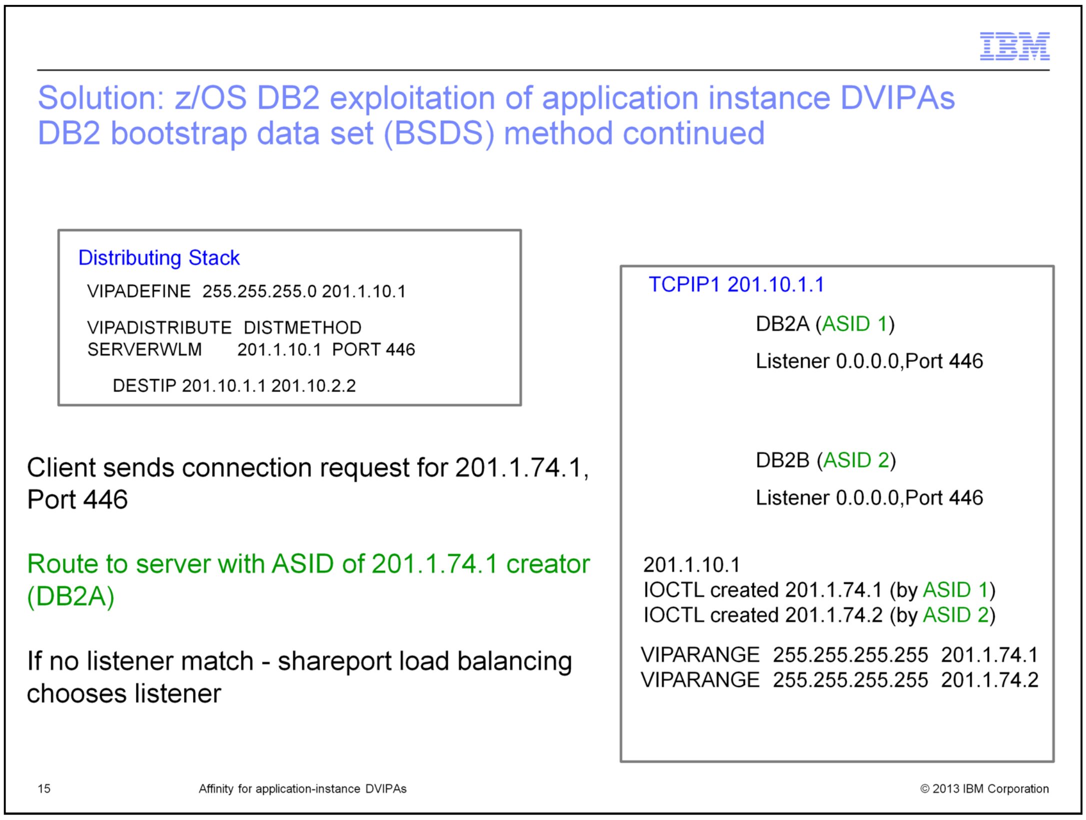
Solution: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method

- Support creating DVIPA with affinity to an address space, remember address space instance of DVIPA creator
- Listener with the same address space instance chosen for incoming connection request
- New option on SIOCSVIPA/SIOCSVIPA6 ioctl: DVR_DEFINE_AFFINITY
- New option on MODDVIPA utility: -a (DVR_DEFINE_AFFINITY)

14 Affinity for application-instance DVIPAs © 2013 IBM Corporation

z/OS V2R1 Communications Server provides support to create an application instance DVIPA with affinity to an address space. When a VIPARANGE DVIPA is created, TCP/IP will remember the address space that created it. When a connect request is received for a VIPARANGE DVIPA created with affinity, TCP/IP will choose the listener of the same address space that created the DVIPA.

A new option, -a, is added to the MODDVIPA utility to create an application instance DVIPA with affinity. The DVR-DEFINE-AFFINITY option can be used on the SIOCSVIPA/SIOCSVIPA6 IOCTLs to create an application instance DVIPA with Affinity.



Solution: z/OS DB2 exploitation of application instance DVIPAs DB2 bootstrap data set (BSDS) method continued

Distributing Stack

```
VIPADEFINE  255.255.255.0 201.1.10.1

VIPADISTRIBUTE  DISTMETHOD
SERVERWLM       201.1.10.1  PORT 446

   DESTIP 201.10.1.1 201.10.2.2
```

Client sends connection request for 201.1.74.1, Port 446

Route to server with ASID of 201.1.74.1 creator (DB2A)

If no listener match - shareport load balancing chooses listener

TCPIP1 201.10.1.1

DB2A (ASID 1)

Listener 0.0.0.0,Port 446

DB2B (ASID 2)

Listener 0.0.0.0,Port 446

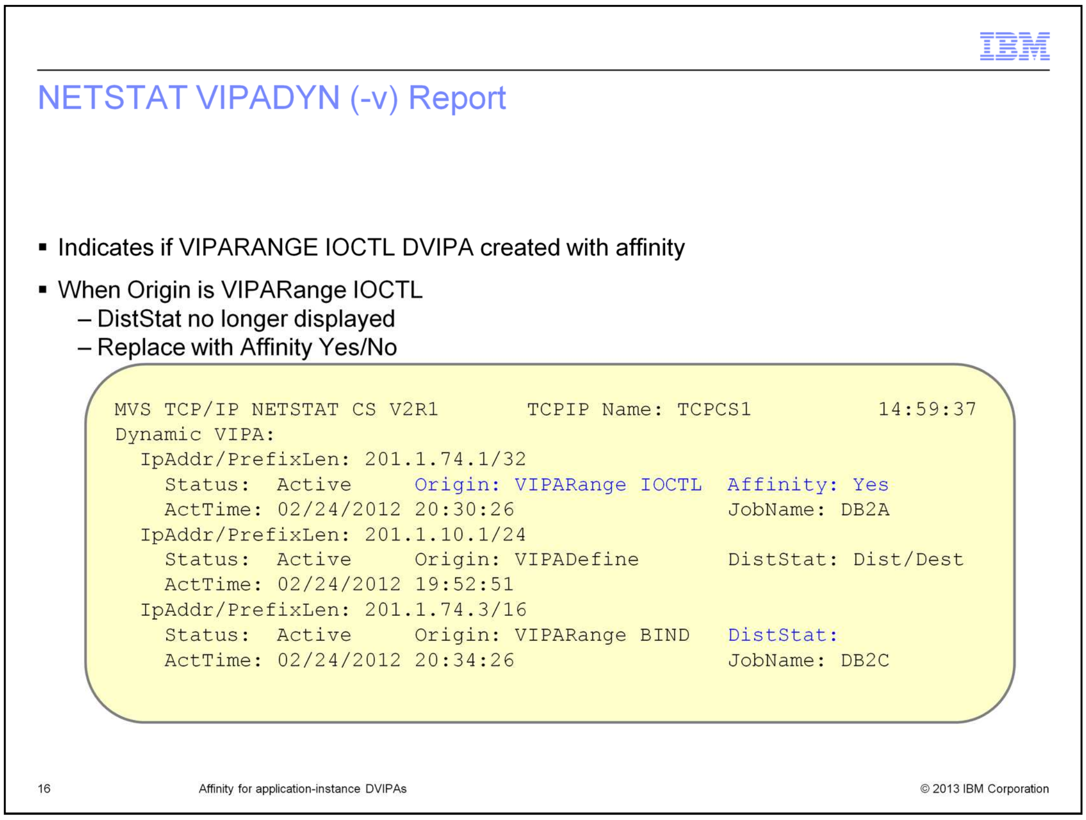
```
201.1.10.1
IOCTL created 201.1.74.1 (by ASID 1)
IOCTL created 201.1.74.2 (by ASID 2)

VIPARANGE  255.255.255.255  201.1.74.1
VIPARANGE  255.255.255.255  201.1.74.2
```

15 Affinity for application-instance DVIPAs © 2013 IBM Corporation

In the example, a connection request for 201.1.74.1 is routed to DB2A since the saved DVIPA creation address space instance is the same as the DB2A listener address space instance.

If there is no listener with the same address space instance as the DVIPA, existing shareport load balancing is used to choose a listener.



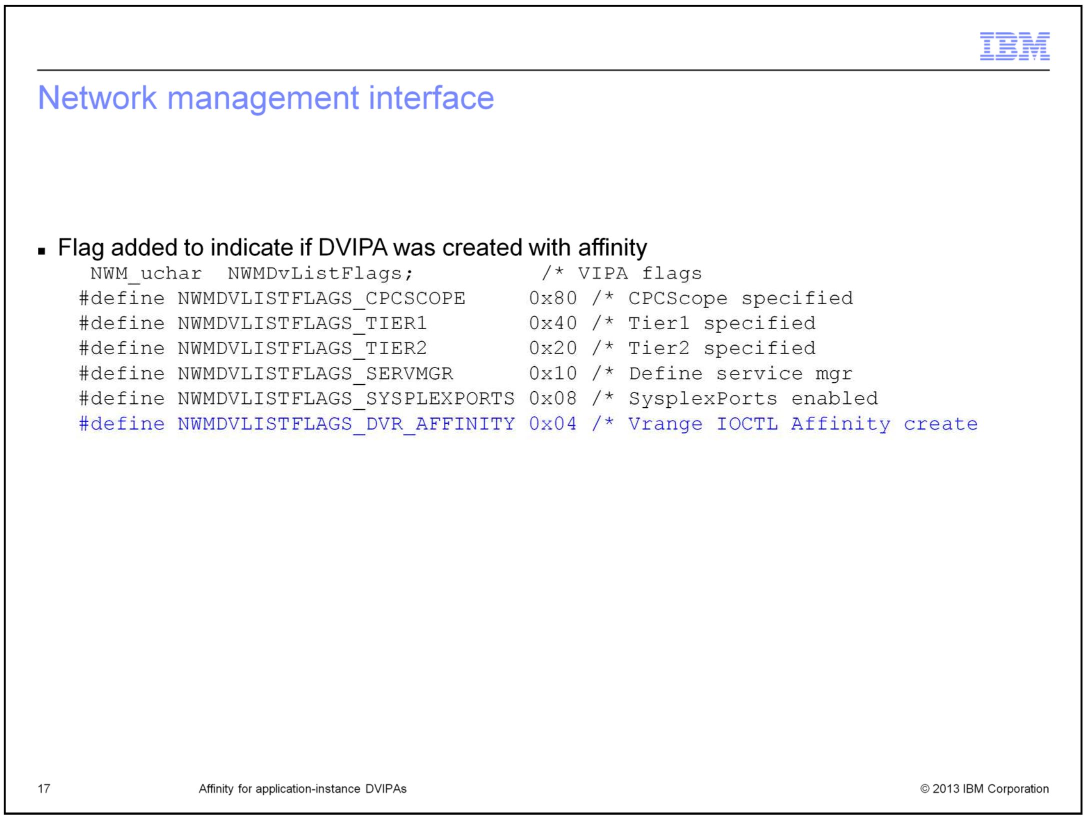
NETSTAT VIPADYN (-v) Report

- Indicates if VIPARANGE IOCTL DVIPA created with affinity
- When Origin is VIPARange IOCTL
  - DistStat no longer displayed
  - Replace with Affinity Yes/No

```
MVS TCP/IP NETSTAT CS V2R1       TCPIP Name: TCPCS1          14:59:37
Dynamic VIPA:
  IpAddr/PrefixLen: 201.1.74.1/32
    Status:  Active     Origin: VIPARange IOCTL  Affinity: Yes
    ActTime: 02/24/2012 20:30:26                 JobName: DB2A
  IpAddr/PrefixLen: 201.1.10.1/24
    Status:  Active     Origin: VIPADefine       DistStat: Dist/Dest
    ActTime: 02/24/2012 19:52:51
  IpAddr/PrefixLen: 201.1.74.3/16
    Status:  Active     Origin: VIPARange BIND   DistStat:
    ActTime: 02/24/2012 20:34:26                 JobName: DB2C
```

16 Affinity for application-instance DVIPAs © 2013 IBM Corporation

The Netstat VIPADYN(-v) report is modified. For a VIPARANGE DVIPA, DistStat or distribution status will no longer be displayed. In prior releases, this field was left blank for VIPARANGE DVIPAs since these DVIPAs are never distributed. Starting in z/OS V2R1, if a VIPARANGE DVIPA is created with an IOCTL, the display is changed to show if it was created with Affinity to an address space.



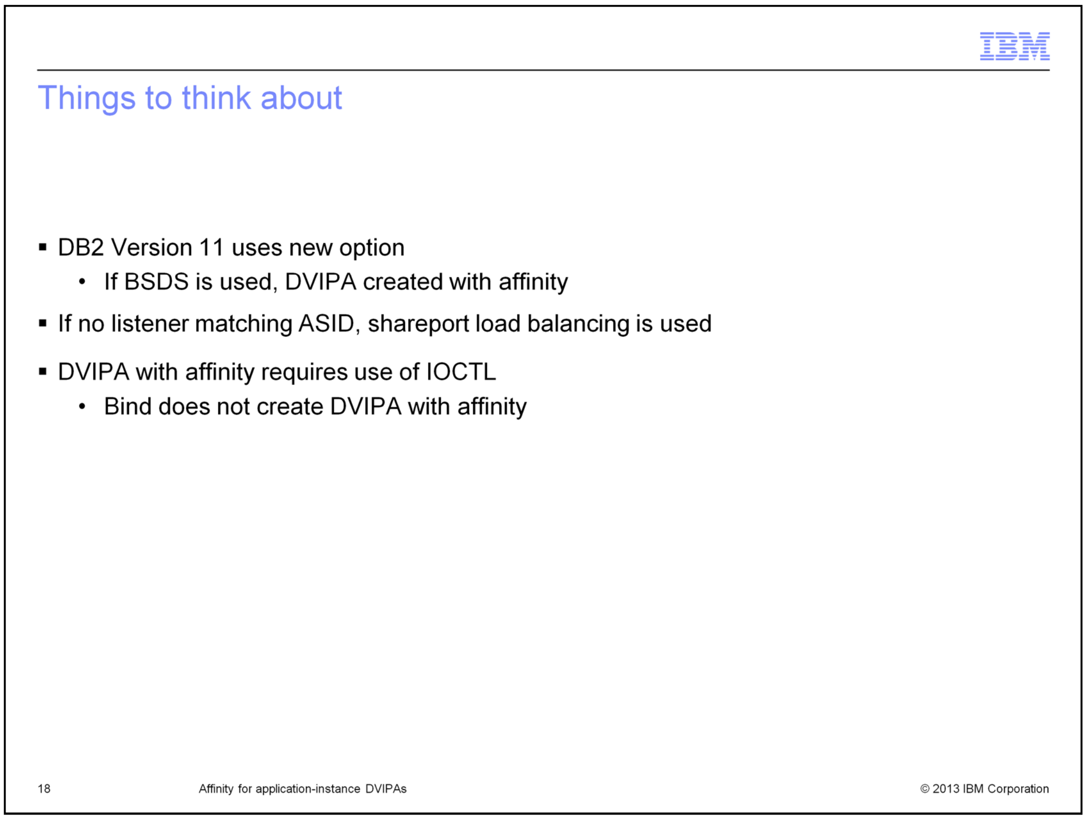
Network management interface

- Flag added to indicate if DVIPA was created with affinity

```
    NWM_uchar  NWMDvListFlags;           /* VIPA flags
   #define NWMDVLISTFLAGS_CPCSCOPE     0x80 /* CPCScope specified
   #define NWMDVLISTFLAGS_TIER1        0x40 /* Tier1 specified
   #define NWMDVLISTFLAGS_TIER2        0x20 /* Tier2 specified
   #define NWMDVLISTFLAGS_SERVMGR      0x10 /* Define service mgr
   #define NWMDVLISTFLAGS_SYSPLEXPORTS 0x08 /* SysplexPorts enabled
   #define NWMDVLISTFLAGS_DVR_AFFINITY 0x04 /* Vrange IOCTL Affinity create
```

17 Affinity for application-instance DVIPAs © 2013 IBM Corporation

The NMI interface that is used to retrieve information about DVIPAs is changed to use a new flag to indicate if the DVIPA was created with Affinity to an address space.

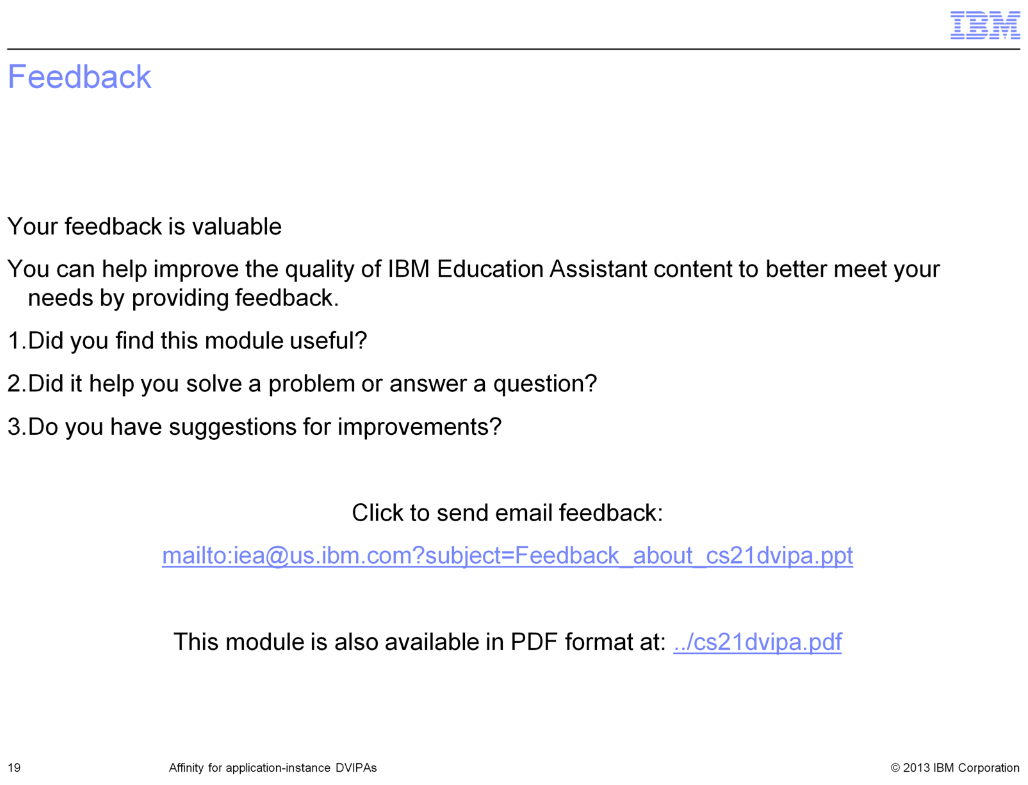


Things to think about

- DB2 Version 11 uses new option
  - If BSDS is used, DVIPA created with affinity
- If no listener matching ASID, shareport load balancing is used
- DVIPA with affinity requires use of IOCTL
  - Bind does not create DVIPA with affinity

18 Affinity for application-instance DVIPAs © 2013 IBM Corporation

If DB2 Version 11 with the bootstrap dataset (BSDS) is used, then an application instance DVIPA will always be created with affinity. If there is no matching listener, then the listener is selected using existing shareport load balancing. Creating a DVIPA with the affinity option requires use of the IOCTL. Bind will not create a DVIPA with affinity.



IBM

Feedback

Your feedback is valuable

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

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

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_cs21dvipa.ppt

This module is also available in PDF format at: ../cs21dvipa.pdf

© 2013 IBM Corporation

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

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, Approach, DB2, and z/OS are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2013. All rights reserved.