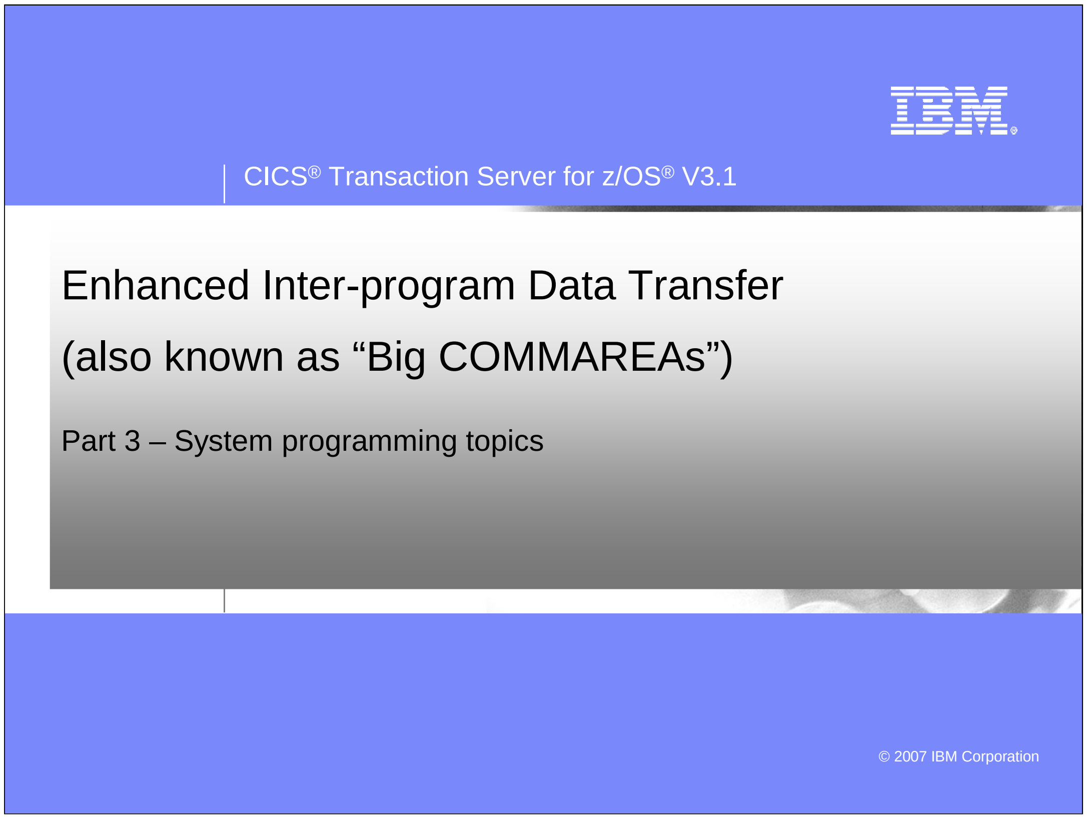


CICS® Transaction Server for z/OS® V3.1

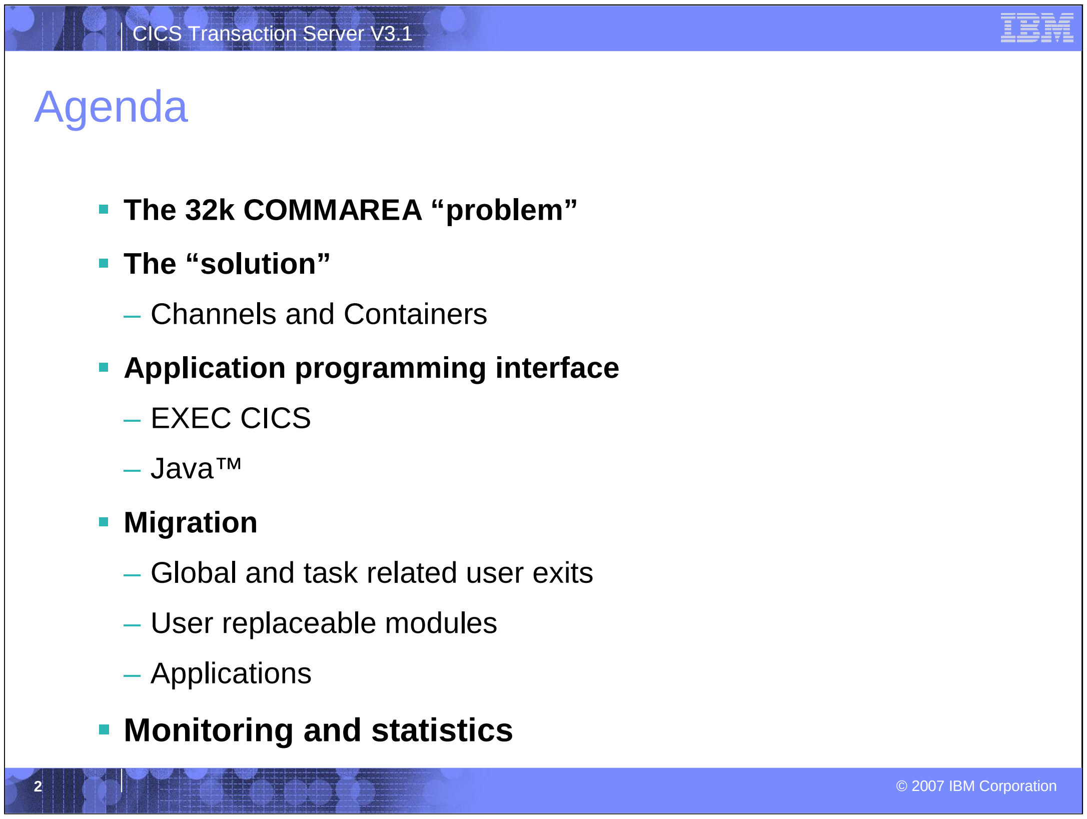
Enhanced Inter-program Data Transfer

(also known as “Big COMMAREAs”)

Part 3 – System programming topics

This presentation will describe the capabilities provided by the Enhanced Inter-program Data Transfer function introduced in CICS Transaction Server 3.1. This function will allow programs and transactions to exchange more than 32K of data when using a LINK, XCTL, START or RETURN TRANSID command.

While not technically correct, to facilitate the understanding of this capability you might think of this capability as being equivalent to “Big COMMAREAs”.



Agenda

- **The 32k COMMAREA "problem"**
- **The "solution"**
  - Channels and Containers
- **Application programming interface**
  - EXEC CICS
  - Java™
- **Migration**
  - Global and task related user exits
  - User replaceable modules
  - Applications
- **Monitoring and statistics**

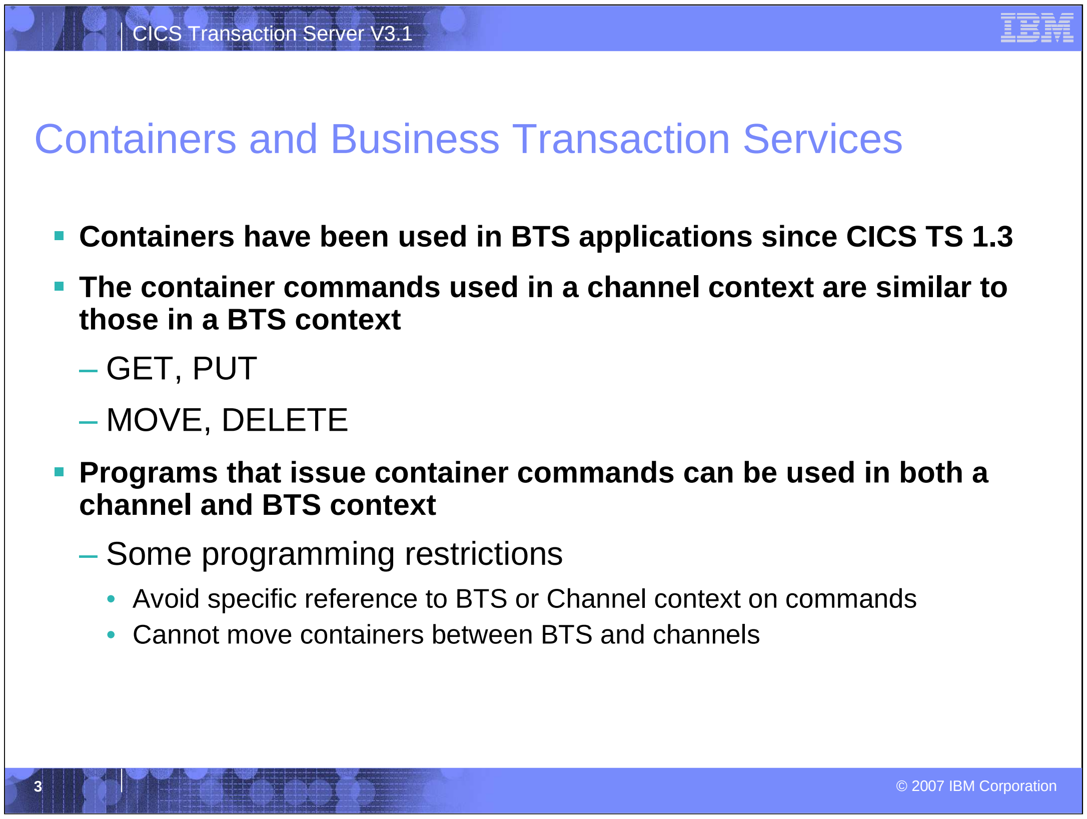
This presentation will discuss the problems that are encountered by programs encountering the 32K COMMAREA limitation, techniques that have been used to circumvent the 32K limitation and then will discuss the CICS solution to the problem.

The CICS solution uses Channels and Containers to eliminate the problem. Channels are sets of Containers. Containers are name blocks of data that hold information to be passed between programs and transaction.

The CICS Application Programming Interface changes for both EXEC CICS commands and JCICS classes will be examined.

The effects of Channels and Containers on Global User Exits, Task Related User Exits and User Replaceable Modules will be described.

An example of how to migrate existing applications from their use of COMMAREAs to Channels and Containers will be presented.



Containers and Business Transaction Services

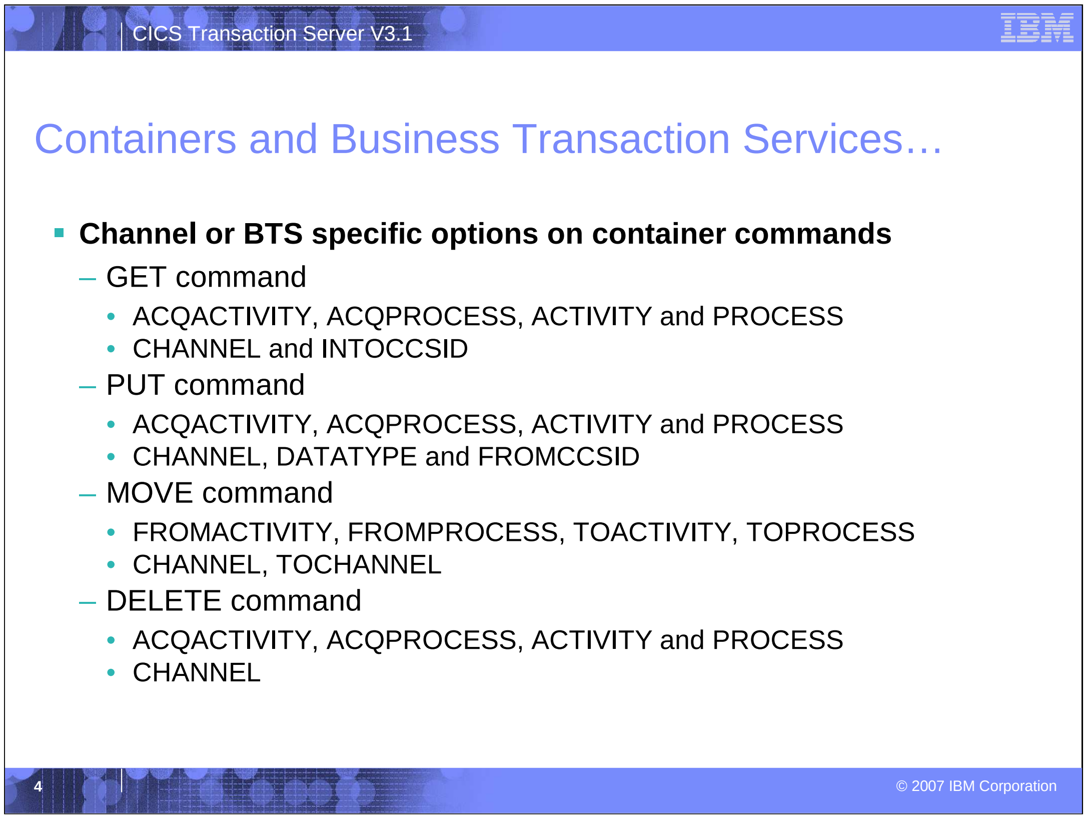
- **Containers have been used in BTS applications since CICS TS 1.3**
- **The container commands used in a channel context are similar to those in a BTS context**
  - GET, PUT
  - MOVE, DELETE
- **Programs that issue container commands can be used in both a channel and BTS context**
  - Some programming restrictions
    - Avoid specific reference to BTS or Channel context on commands
    - Cannot move containers between BTS and channels

For those of you that have used CICS Business Transaction Services (BTS), available since CICS TS 1.3, you will be familiar with containers. BTS implemented containers as a way of passing information between activities and processes. There is no limit to the size of a container in BTS. In fact, there have been white papers written to describe how a programmer might use BTS containers as a “Big COMMAREA”.

The containers used in the channel context are similar to those used in BTS and the commands used to access the container data are similar (for example GET, PUT, MOVE, DELETE).

It is possible to have the same server program invoked both a channel and BTS context. To accomplish this the server program must avoid the use of options that specifically identify the context.

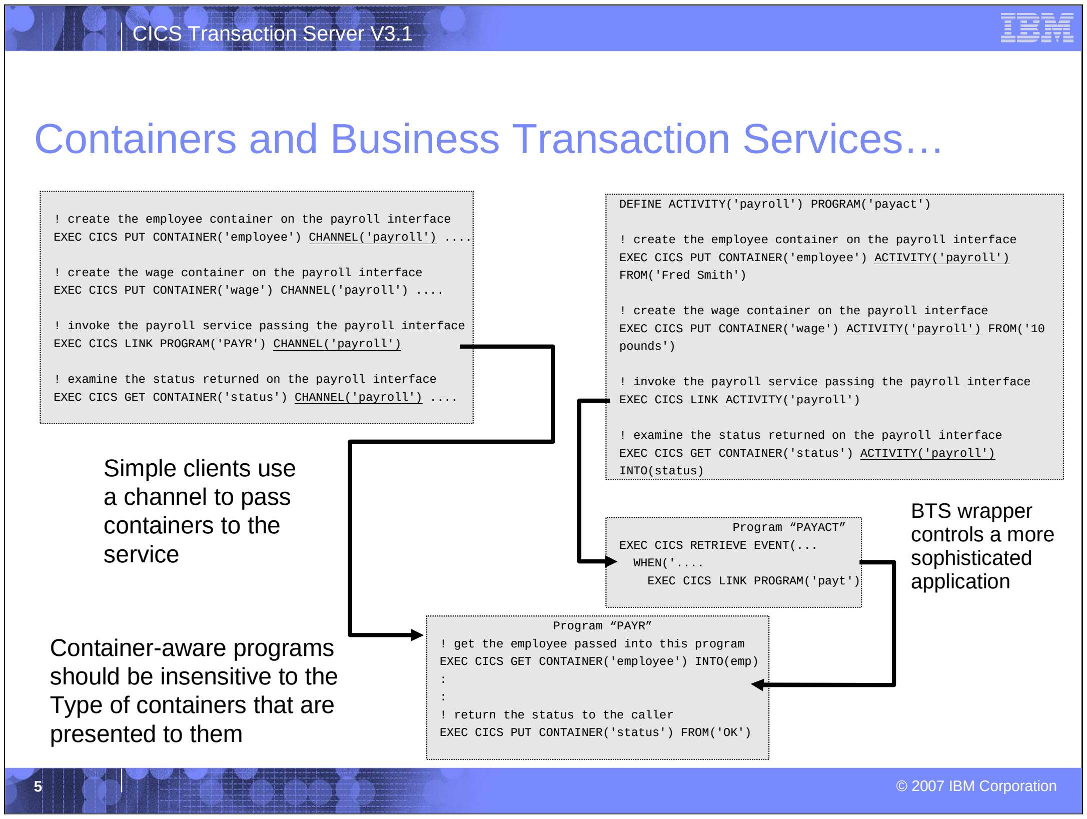
The server program must “call” CICS to determine the context of a command. When a container command is executed CICS will first check to see if there is a current channel. If there is, then the context of the command will be Channel. If there is no current channel, CICS will the check to see if this is part of a BTS activity. If this is part of a BTS activity, then the context will be BTS. If the program has no channel context and no BTS context than an INVREQ will be raised.



Containers and Business Transaction Services…

- **Channel or BTS specific options on container commands**
  - GET command
    - ACQACTIVITY, ACQPROCESS, ACTIVITY and PROCESS
    - CHANNEL and INTOCCSID
  - PUT command
    - ACQACTIVITY, ACQPROCESS, ACTIVITY and PROCESS
    - CHANNEL, DATATYPE and FROMCCSID
  - MOVE command
    - FROMACTIVITY, FROMPROCESS, TOACTIVITY, TOPROCESS
    - CHANNEL, TOCHANNEL
  - DELETE command
    - ACQACTIVITY, ACQPROCESS, ACTIVITY and PROCESS
    - CHANNEL

These are the options on the GET, PUT, MOVE and DELETE commands that specifically identify a Channel or BTS context.



CICS Transaction Server V3.1

Containers and Business Transaction Services…

```
! create the employee container on the payroll interface
EXEC CICS PUT CONTAINER('employee') CHANNEL('payroll') ....

! create the wage container on the payroll interface
EXEC CICS PUT CONTAINER('wage') CHANNEL('payroll') ....

! invoke the payroll service passing the payroll interface
EXEC CICS LINK PROGRAM('PAYR') CHANNEL('payroll')

! examine the status returned on the payroll interface
EXEC CICS GET CONTAINER('status') CHANNEL('payroll') ....
```

```
DEFINE ACTIVITY('payroll') PROGRAM('payact')

! create the employee container on the payroll interface
EXEC CICS PUT CONTAINER('employee') ACTIVITY('payroll')
FROM('Fred Smith')

! create the wage container on the payroll interface
EXEC CICS PUT CONTAINER('wage') ACTIVITY('payroll') FROM('10
pounds')

! invoke the payroll service passing the payroll interface
EXEC CICS LINK ACTIVITY('payroll')

! examine the status returned on the payroll interface
EXEC CICS GET CONTAINER('status') ACTIVITY('payroll')
INTO(status)
```

Simple clients use a channel to pass containers to the service

```
                    Program “PAYACT”
EXEC CICS RETRIEVE EVENT(...
  WHEN('....
    EXEC CICS LINK PROGRAM('payt')
```

BTS wrapper controls a more sophisticated application

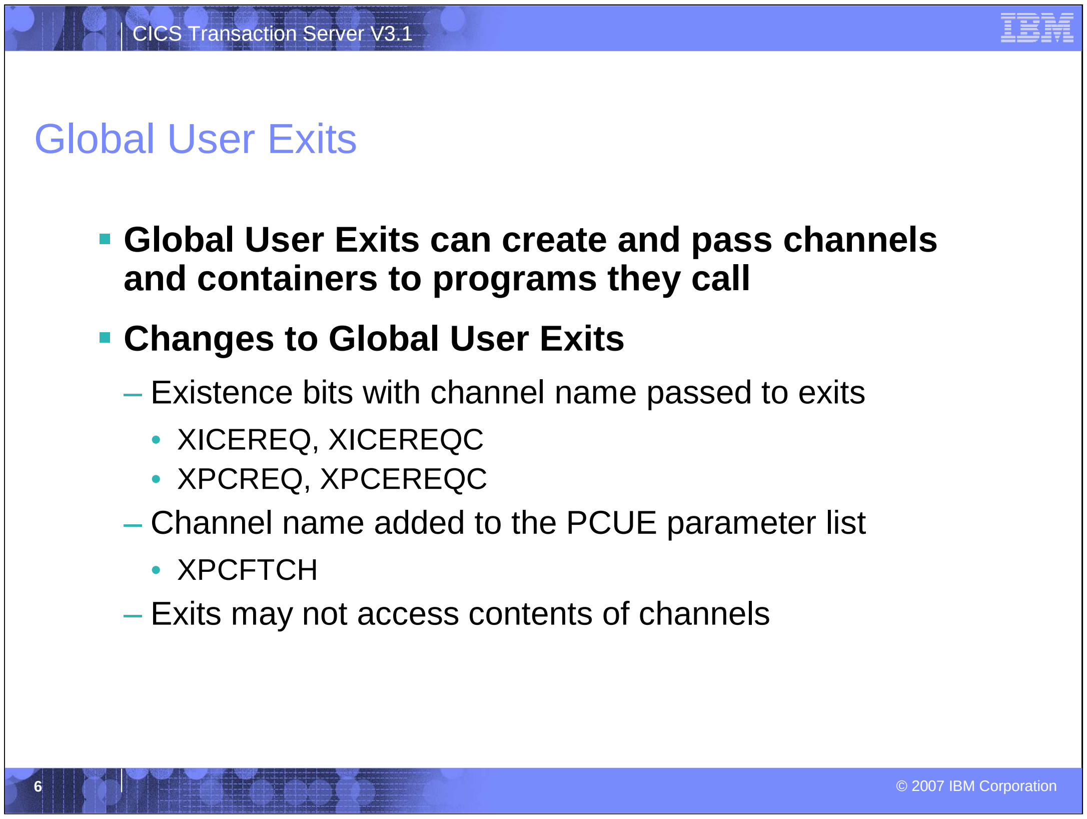
```
                Program “PAYR”
! get the employee passed into this program
EXEC CICS GET CONTAINER('employee') INTO(emp)
:
:
! return the status to the caller
EXEC CICS PUT CONTAINER('status') FROM('OK')
```

Container-aware programs should be insensitive to the Type of containers that are presented to them

5

© 2007 IBM Corporation

This is an example of how a program might be designed and written to use containers in both a Channel and BTS context.

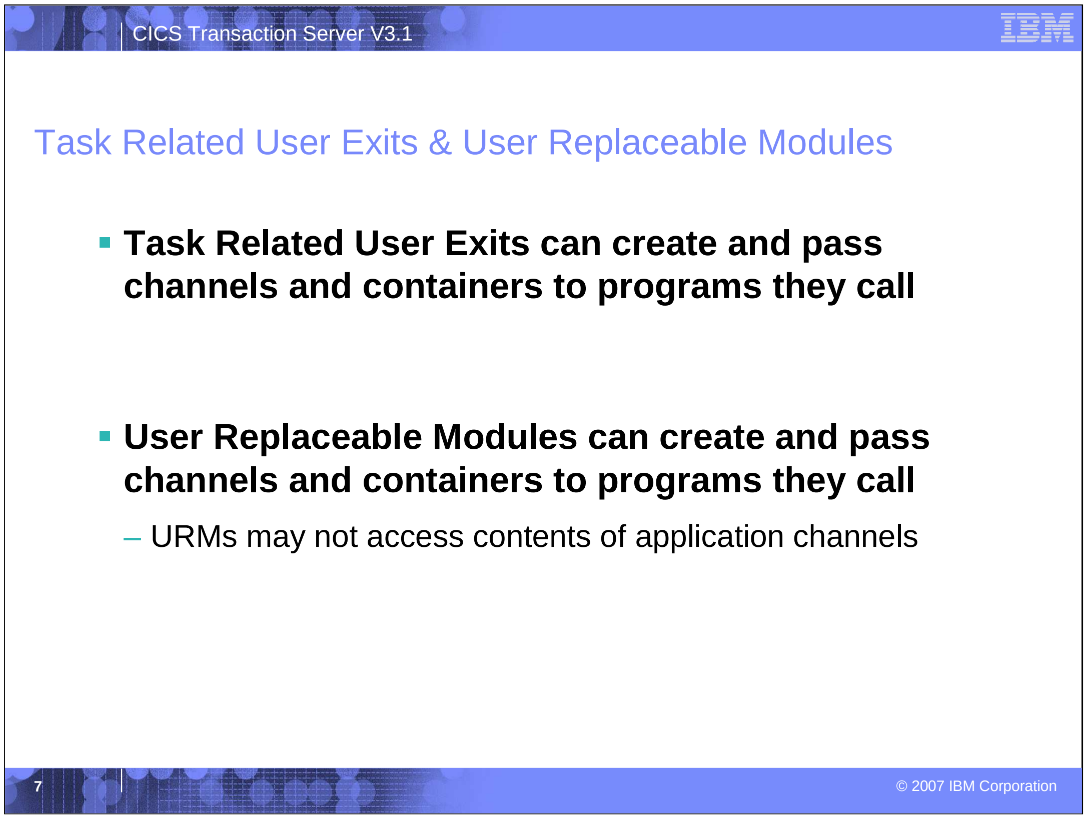


Global User Exits

- **Global User Exits can create and pass channels and containers to programs they call**
- **Changes to Global User Exits**
  - Existence bits with channel name passed to exits
    - XICEREQ, XICEREQC
    - XPCREQ, XPCEREQC
  - Channel name added to the PCUE parameter list
    - XPCFTCH
  - Exits may not access contents of channels

CICS Global User Exits (GLUEs) are eligible to create channels and containers for their own use.

The parameter list passed to a number of GLUEs changes slightly with the addition of existence bits to signify the presence of a application’s channel name. At this time the exit is not able to examine the contents of the channel. This restriction includes browsing the channel to determine container names as well as issuing a GET CONTAINER command to retrieve the application data.


CICS Transaction Server V3.1

Task Related User Exits & User Replaceable Modules

- **Task Related User Exits can create and pass channels and containers to programs they call**
- **User Replaceable Modules can create and pass channels and containers to programs they call**
  - URMs may not access contents of application channels

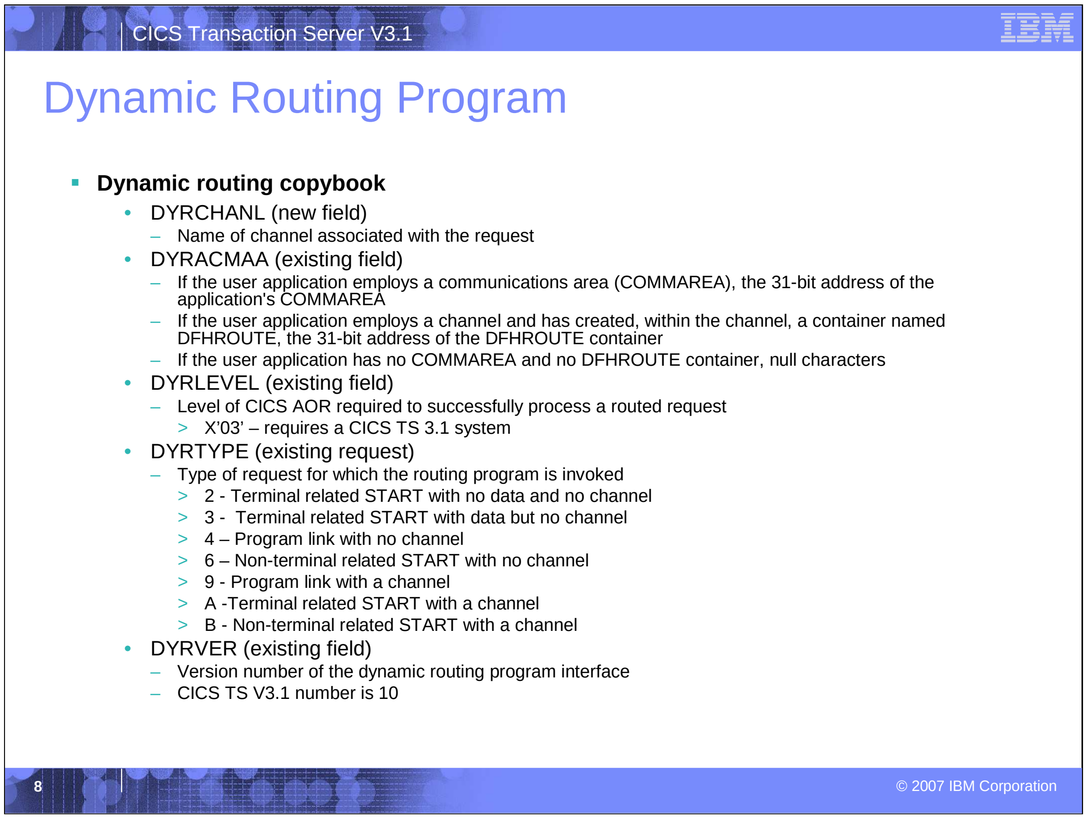
7

© 2007 IBM Corporation

CICS Task Related User Exits (TRUEs) are eligible to create channels and containers for their own use.

At this time the task related user exit is not able to examine the contents of the channel. This restriction includes browsing the channel to determine container names as well as issuing a GET CONTAINER command to retrieve the application data.

CICS User Replaceable Modules (URMs) are eligible to create channels and containers for their own use.



Dynamic Routing Program

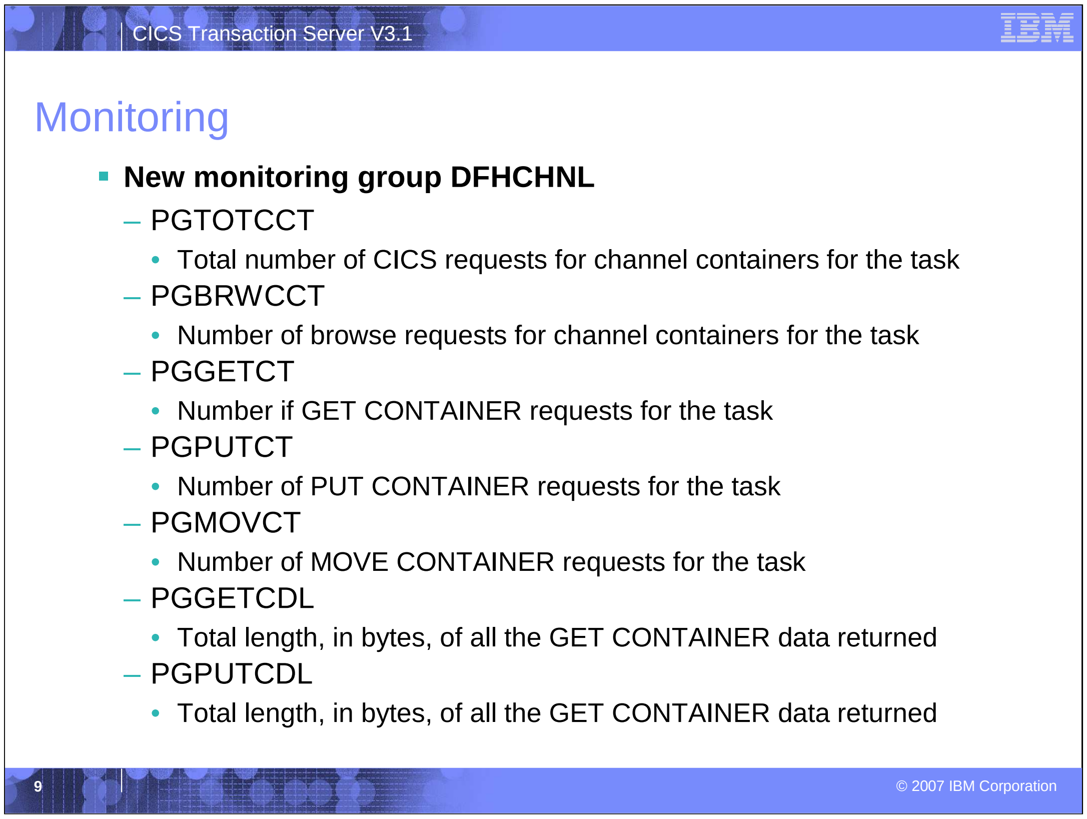
- **Dynamic routing copybook**
  - DYRCHANL (new field)
    - Name of channel associated with the request
  - DYRACMAA (existing field)
    - If the user application employs a communications area (COMMAREA), the 31-bit address of the application's COMMAREA
    - If the user application employs a channel and has created, within the channel, a container named DFHROUTE, the 31-bit address of the DFHROUTE container
    - If the user application has no COMMAREA and no DFHROUTE container, null characters
  - DYRLEVEL (existing field)
    - Level of CICS AOR required to successfully process a routed request
      - X’03’ – requires a CICS TS 3.1 system
  - DYRTYPE (existing request)
    - Type of request for which the routing program is invoked
      - 2 - Terminal related START with no data and no channel
      - 3 - Terminal related START with data but no channel
      - 4 – Program link with no channel
      - 6 – Non-terminal related START with no channel
      - 9 - Program link with a channel
      - A -Terminal related START with a channel
      - B - Non-terminal related START with a channel
  - DYRVER (existing field)
    - Version number of the dynamic routing program interface
    - CICS TS V3.1 number is 10

CICS User Replaceable Modules (URMs) are eligible to create channels and containers for their own use.

The COMMAREA (parameter list) passed to the Dynamic Routing Programs changes slightly with the addition of the channel name in user by the application and changes to the target AOR level and the type of request.

At this time the URM is not able to examine the contents of the application containers. This restriction includes browsing the channel to determine container names as well as issuing a GET CONTAINER command to retrieve the application data.

The application may create a container with the name DFHROUTE. The address of this container will be passed in DYRACMAA to the dynamic routing program.



Monitoring

- **New monitoring group DFHCHNL**
  - PGTOTCCT
    - Total number of CICS requests for channel containers for the task
  - PGBRWCCT
    - Number of browse requests for channel containers for the task
  - PGGETCT
    - Number if GET CONTAINER requests for the task
  - PGPUTCT
    - Number of PUT CONTAINER requests for the task
  - PGMOVCT
    - Number of MOVE CONTAINER requests for the task
  - PGGETCDL
    - Total length, in bytes, of all the GET CONTAINER data returned
  - PGPUTCDL
    - Total length, in bytes, of all the GET CONTAINER data returned

CICS adds new task performance monitoring information for channel and container usage.

Group DFHCHNL contains the following performance data:

321 (TYPE-A, 'PGTOTCCT', 4 BYTES)

The number of CICS requests for channel containers issued by the user task.

322 (TYPE-A, 'PGBRWCCT', 4 BYTES)

The number of CICS browse requests for channel containers issued by the user task.

323 (TYPE-A, 'PGGETCCT', 4 BYTES)

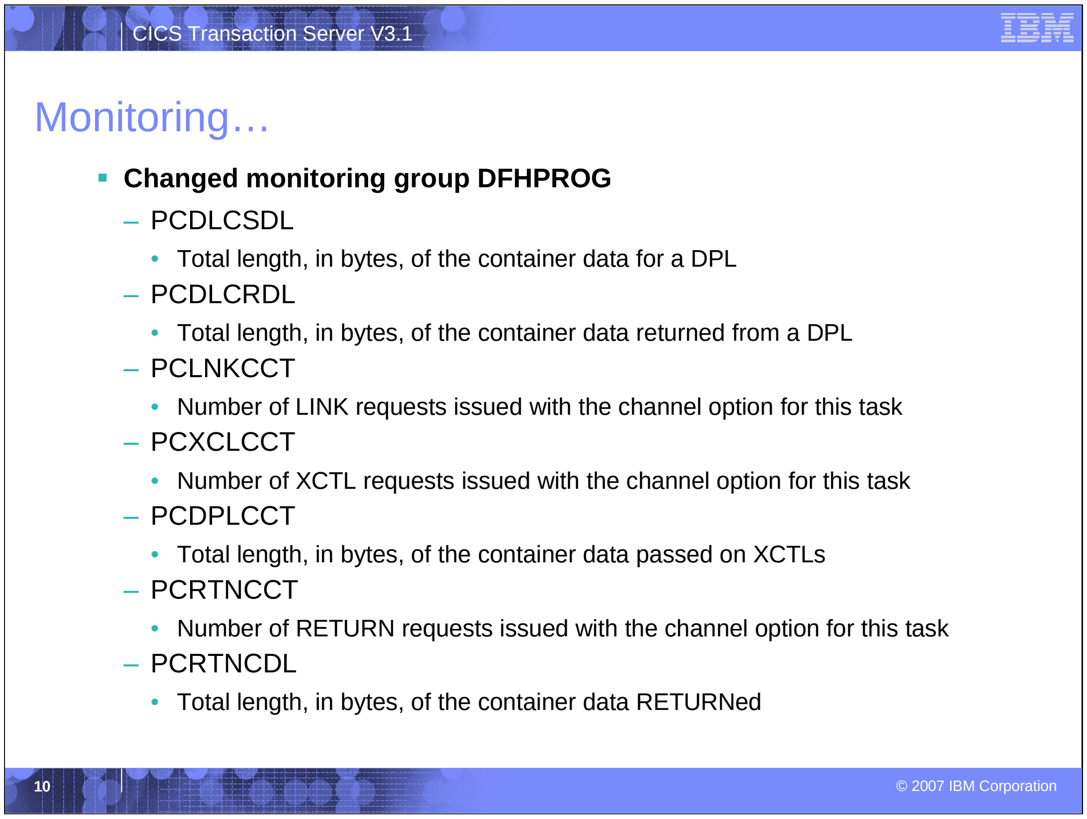
The number of GET CONTAINER requests for channel containers issued by the user task.

324 (TYPE-A, 'PGPUTCCT', 4 BYTES)

The number of PUT CONTAINER requests for channel containers issued by the user task.

325 (TYPE-A, 'PGMOVCCT', 4 BYTES)

The number of MOVE CONTAINER requests for channel containers issued by the


CICS Transaction Server V3.1

Monitoring…

- **Changed monitoring group DFHPROG**
  - PCDLCSDL
    - Total length, in bytes, of the container data for a DPL
  - PCDLCRDL
    - Total length, in bytes, of the container data returned from a DPL
  - PCLNKCCT
    - Number of LINK requests issued with the channel option for this task
  - PCXCLCCT
    - Number of XCTL requests issued with the channel option for this task
  - PCDPLCCT
    - Total length, in bytes, of the container data passed on XCTLs
  - PCRTNCCT
    - Number of RETURN requests issued with the channel option for this task
  - PCRTNCDL
    - Total length, in bytes, of the container data RETURNed

10 © 2007 IBM Corporation

CICS adds new task performance monitoring information for channel and container usage.

The following new fields are added to group DFHPROG:

286 (TYPE-A, 'PCDLCSDL', 4 BYTES)

The total length, in bytes, of the data in the containers of all the distributed program link (DPL) requests issued with the CHANNEL option by the user task.

287 (TYPE-A, 'PCDLCRDL', 4 BYTES)

The total length, in bytes, of the data in the containers of all DPL RETURN CHANNEL commands issued by the user task.

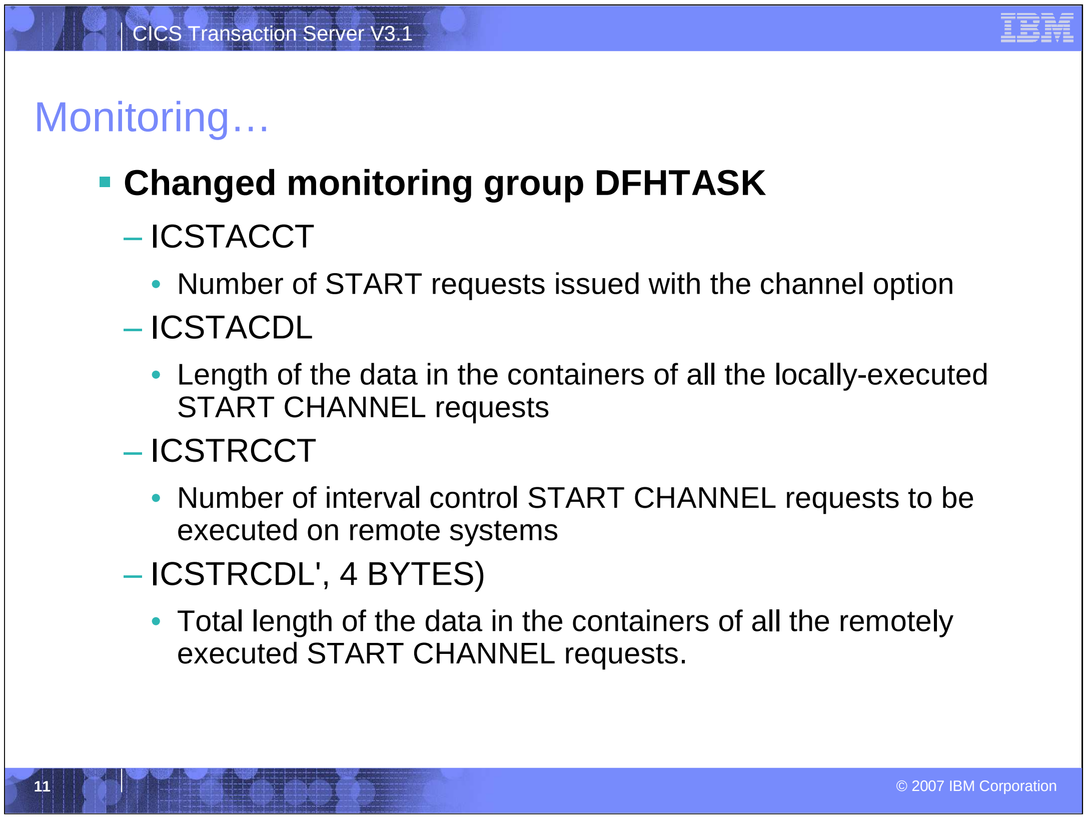
306 (TYPE-A, 'PCLNKCCT', 4 BYTES)

Number of program LINK requests issued with the CHANNEL option by the user task.

307 (TYPE-A, 'PCXCLCCT', 4 BYTES)

Number of program XCTL requests issued with the CHANNEL option by the user task.

308 (TYPE-A, 'PCDPLCCT', 4 BYTES)


CICS Transaction Server V3.1

Monitoring…

- **Changed monitoring group DFHTASK**
  - ICSTACCT
    - Number of START requests issued with the channel option
  - ICSTACDL
    - Length of the data in the containers of all the locally-executed START CHANNEL requests
  - ICSTRCCT
    - Number of interval control START CHANNEL requests to be executed on remote systems
  - ICSTRCDL', 4 BYTES)
    - Total length of the data in the containers of all the remotely executed START CHANNEL requests.

11 © 2007 IBM Corporation

CICS adds new task performance monitoring information for channel and container usage.

The following new fields are added to group DFHTASK:

065 (TYPE-A, 'ICSTACCT', 4 BYTES)

Total number of local interval control START requests, with the CHANNEL option, issued by the user task.

345 (TYPE-A, 'ICSTACDL', 4 BYTES)

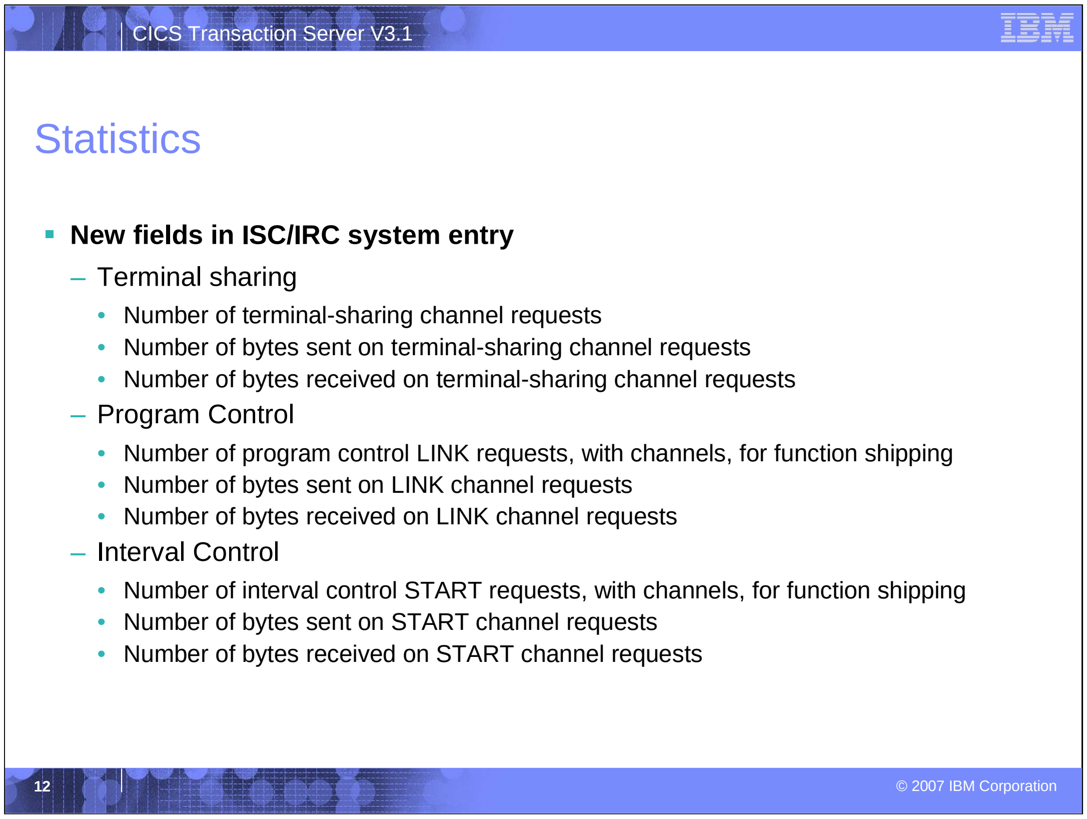
Total length, in bytes, of the data in the containers of all the locally-executed START CHANNEL requests issued by the user task. This total includes the length of any headers to the data.

346 (TYPE-A, 'ICSTRCCT', 4 BYTES)

Total number of interval control START CHANNEL requests, to be executed on remote systems, issued by the user task.

347 (TYPE-A, 'ICSTRCDL', 4 BYTES)

Total length, in bytes, of the data in the containers of all the remotely-executed START CHANNEL requests issued by the user task. This total includes the length of any headers to the data.



Statistics

- **New fields in ISC/IRC system entry**
  - Terminal sharing
    - Number of terminal-sharing channel requests
    - Number of bytes sent on terminal-sharing channel requests
    - Number of bytes received on terminal-sharing channel requests
  - Program Control
    - Number of program control LINK requests, with channels, for function shipping
    - Number of bytes sent on LINK channel requests
    - Number of bytes received on LINK channel requests
  - Interval Control
    - Number of interval control START requests, with channels, for function shipping
    - Number of bytes sent on START channel requests
    - Number of bytes received on START channel requests

There are additions to "ISC/IRC system entry: Resource statistics" and to the "Connections and Modenames Report", both of which are mapped by the DFHA14DS DSECT. The new fields relate to channel data flowing across the connection.

A14ESTTC_CHANNEL is the number of terminal-sharing channel requests.

Number of bytes sent on terminal-sharing channel requests

A14ESTTC_CHANNEL_SENT is the number of bytes sent on terminal-sharing channel requests. This is the total amount of data sent on the connection, including any control information.

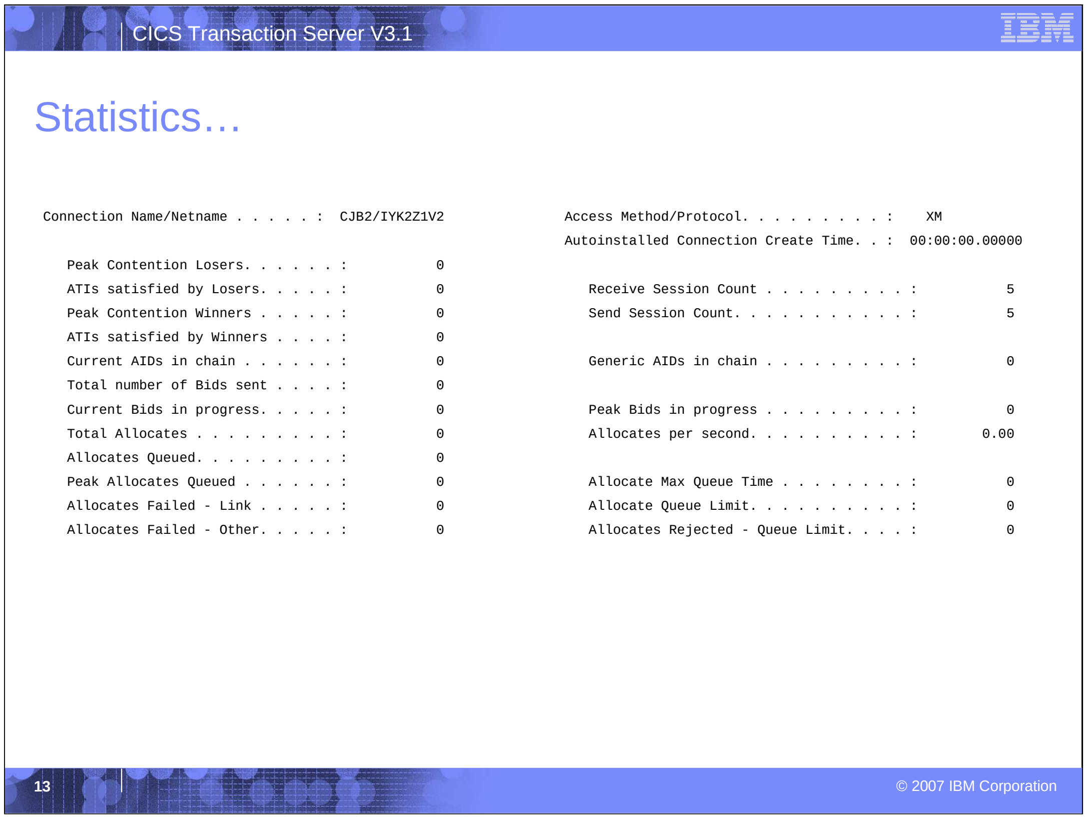
A14ESTTC_CHANNEL_RCVD is the number of bytes received on terminal-sharing channel requests. This is the total amount of data sent on the connection, including any control information.

A14ESTPC_CHANNEL is the number of program control LINK requests, with channels, for function shipping. This is a subset of the number in A14ESTPC.

A14ESTPC_CHANNEL _SENT is the number of bytes sent on LINK channel requests. This is the total amount of data sent on the connection, including any control information.

A14ESTPC_CHANNEL _RCVD is the number of bytes received on LINK channel requests. This is the total amount of data received on the connection, including any control information.

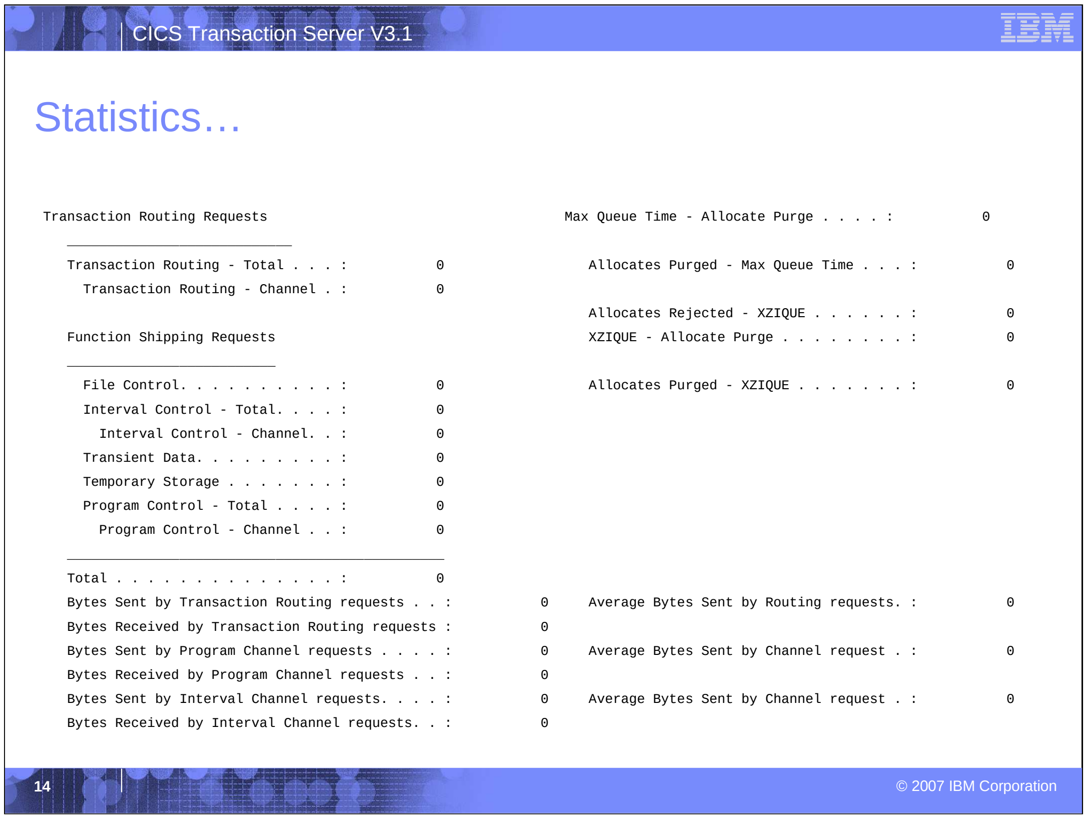
A14ESTIC_CHANNEL is the number of interval control START requests, with channels,



Statistics…

```
Connection Name/Netname . . . . . :  CJB2/IYK2Z1V2           Access Method/Protocol. . . . . . . . . :    XM
                                                             Autoinstalled Connection Create Time. . :  00:00:00.00000
   Peak Contention Losers. . . . . . :           0
   ATIs satisfied by Losers. . . . . :           0              Receive Session Count . . . . . . . . . :           5
   Peak Contention Winners . . . . . :           0              Send Session Count. . . . . . . . . . . :           5
   ATIs satisfied by Winners . . . . :           0
   Current AIDs in chain . . . . . . :           0              Generic AIDs in chain . . . . . . . . . :           0
   Total number of Bids sent . . . . :           0
   Current Bids in progress. . . . . :           0              Peak Bids in progress . . . . . . . . . :           0
   Total Allocates . . . . . . . . . :           0              Allocates per second. . . . . . . . . . :        0.00
   Allocates Queued. . . . . . . . . :           0
   Peak Allocates Queued . . . . . . :           0              Allocate Max Queue Time . . . . . . . . :           0
   Allocates Failed - Link . . . . . :           0              Allocate Queue Limit. . . . . . . . . . :           0
   Allocates Failed - Other. . . . . :           0              Allocates Rejected - Queue Limit. . . . :           0
```

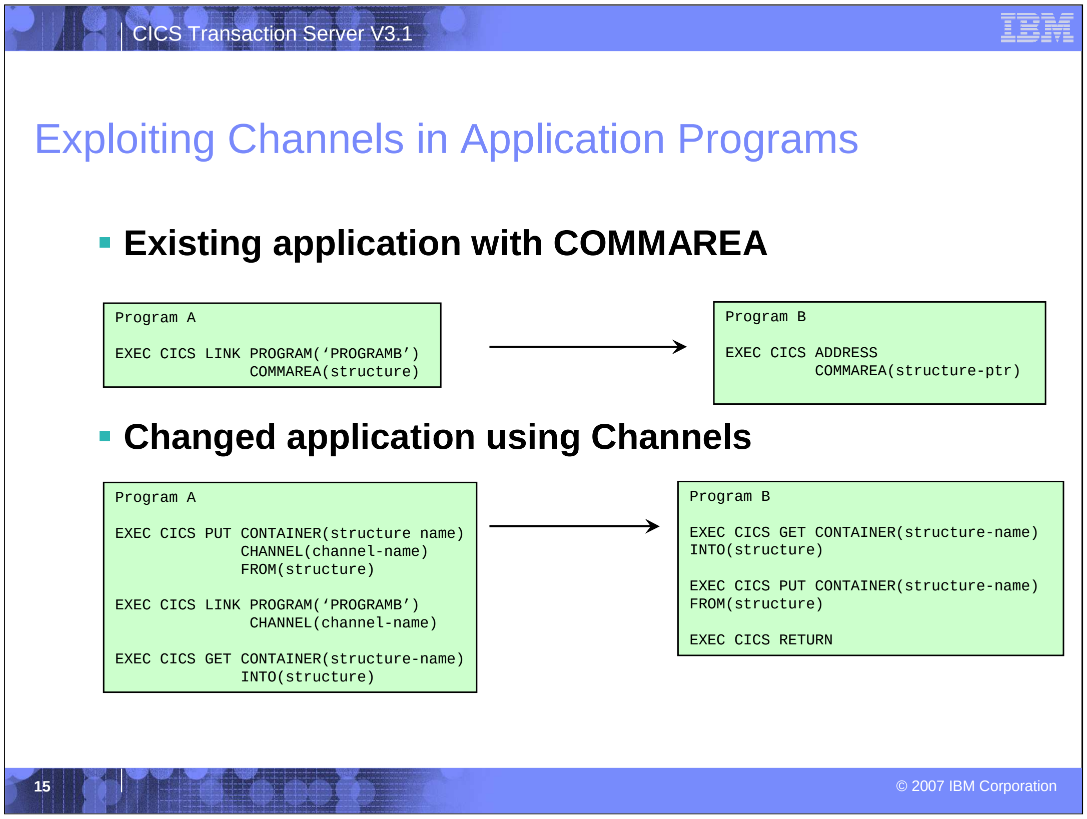
This is an example of the statistics reports produced for channel and container usage.



Statistics…

```
Transaction Routing Requests                                      Max Queue Time - Allocate Purge . . . . :           0
   ____________________________
   Transaction Routing - Total . . . :           0                   Allocates Purged - Max Queue Time . . . :           0
     Transaction Routing - Channel . :           0
                                                                     Allocates Rejected - XZIQUE . . . . . . :           0
   Function Shipping Requests                                        XZIQUE - Allocate Purge . . . . . . . . :           0
   __________________________
     File Control. . . . . . . . . . :           0                   Allocates Purged - XZIQUE . . . . . . . :           0
     Interval Control - Total. . . . :           0
       Interval Control - Channel. . :           0
     Transient Data. . . . . . . . . :           0
     Temporary Storage . . . . . . . :           0
     Program Control - Total . . . . :           0
       Program Control - Channel . . :           0
   _______________________________________________
   Total . . . . . . . . . . . . . . :           0
   Bytes Sent by Transaction Routing requests . . :           0     Average Bytes Sent by Routing requests. :           0
   Bytes Received by Transaction Routing requests :           0
   Bytes Sent by Program Channel requests . . . . :           0     Average Bytes Sent by Channel request . :           0
   Bytes Received by Program Channel requests . . :           0
   Bytes Sent by Interval Channel requests. . . . :           0     Average Bytes Sent by Channel request . :           0
   Bytes Received by Interval Channel requests. . :           0
```

This is an example of the statistics reports produced for channel and container usage.



Exploiting Channels in Application Programs

- **Existing application with COMMAREA**

```
Program A

EXEC CICS LINK PROGRAM('PROGRAMB')
               COMMAREA(structure)
```

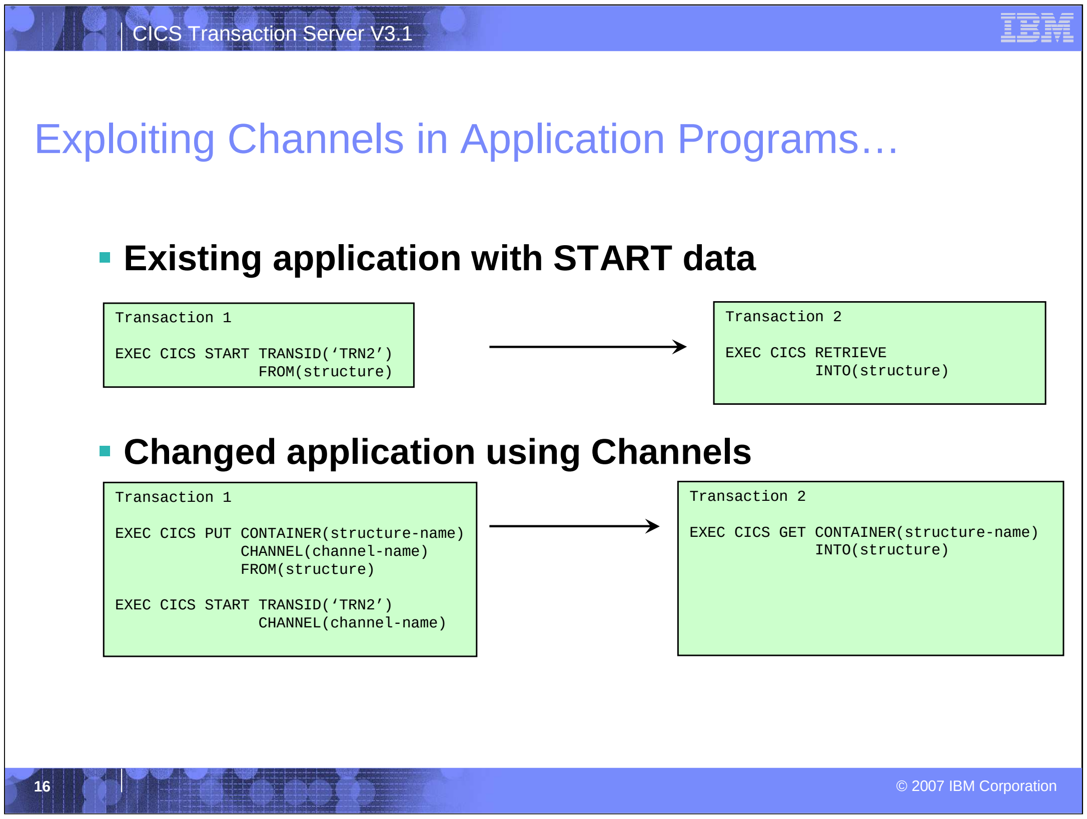
```
Program B

EXEC CICS ADDRESS
          COMMAREA(structure-ptr)
```

- **Changed application using Channels**

```
Program A

EXEC CICS PUT CONTAINER(structure name)
              CHANNEL(channel-name)
              FROM(structure)

EXEC CICS LINK PROGRAM('PROGRAMB')
               CHANNEL(channel-name)

EXEC CICS GET CONTAINER(structure-name)
              INTO(structure)
```

```
Program B

EXEC CICS GET CONTAINER(structure-name)
INTO(structure)

EXEC CICS PUT CONTAINER(structure-name)
FROM(structure)

EXEC CICS RETURN
```

This is an example of the changes necessary to convert an application program that is using a COMMAREA to one using a channel and container. The example here only shows the commands which need to be added or changed. There is no attempt in this example to describe how the copybook structure can be simplified. Refer to the "Best Practices" page for discussion on how the COMMAREA copybook should be evaluated.



Exploiting Channels in Application Programs…

- **Existing application with START data**

```
Transaction 1

EXEC CICS START TRANSID(‘TRN2’)
                FROM(structure)
```

$\longrightarrow$

```
Transaction 2

EXEC CICS RETRIEVE
          INTO(structure)
```

- **Changed application using Channels**

```
Transaction 1

EXEC CICS PUT CONTAINER(structure-name)
              CHANNEL(channel-name)
              FROM(structure)

EXEC CICS START TRANSID(‘TRN2’)
                CHANNEL(channel-name)
```

$\longrightarrow$

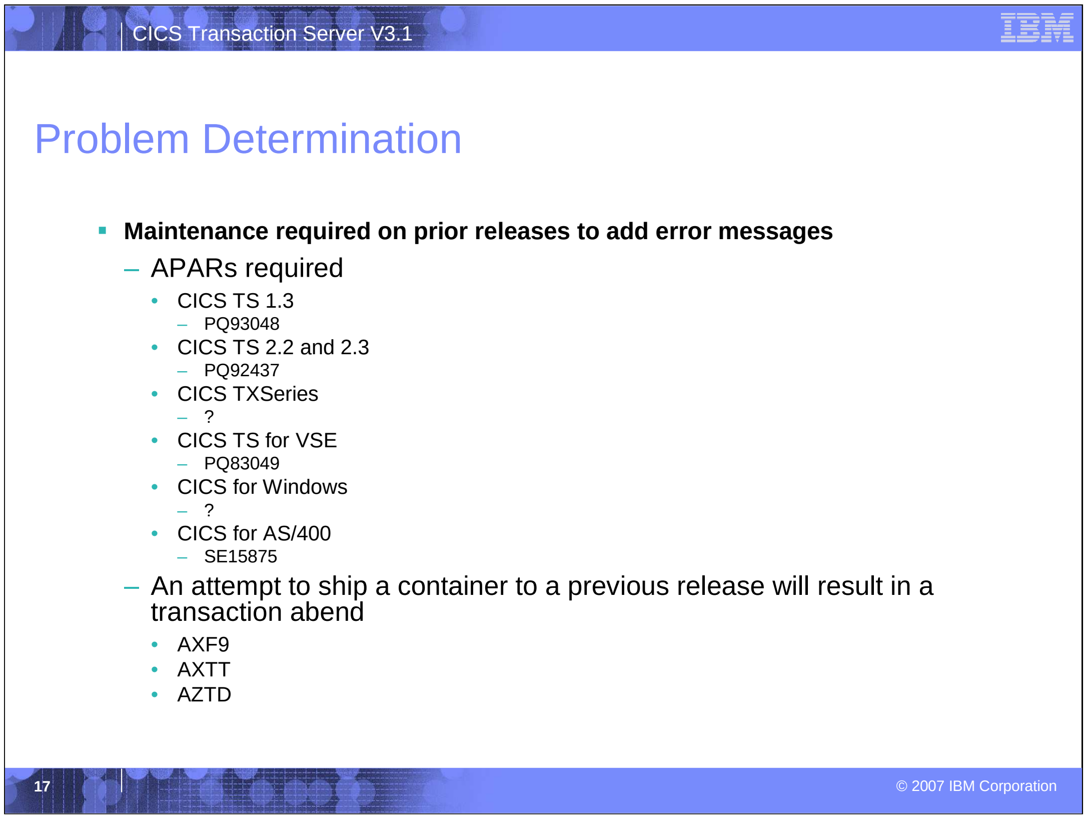
```
Transaction 2

EXEC CICS GET CONTAINER(structure-name)
              INTO(structure)
```

16

© 2007 IBM Corporation

This is an example of the changes necessary to convert an application program that is using an EXEC CICS START with data to a START passing a channel. The example here only shows the commands which need to be added or changed. There is no attempt in this example to describe how the copybook structure can be simplified. Refer to the “Best Practices” page for discussion on how the data area copybook should be evaluated.

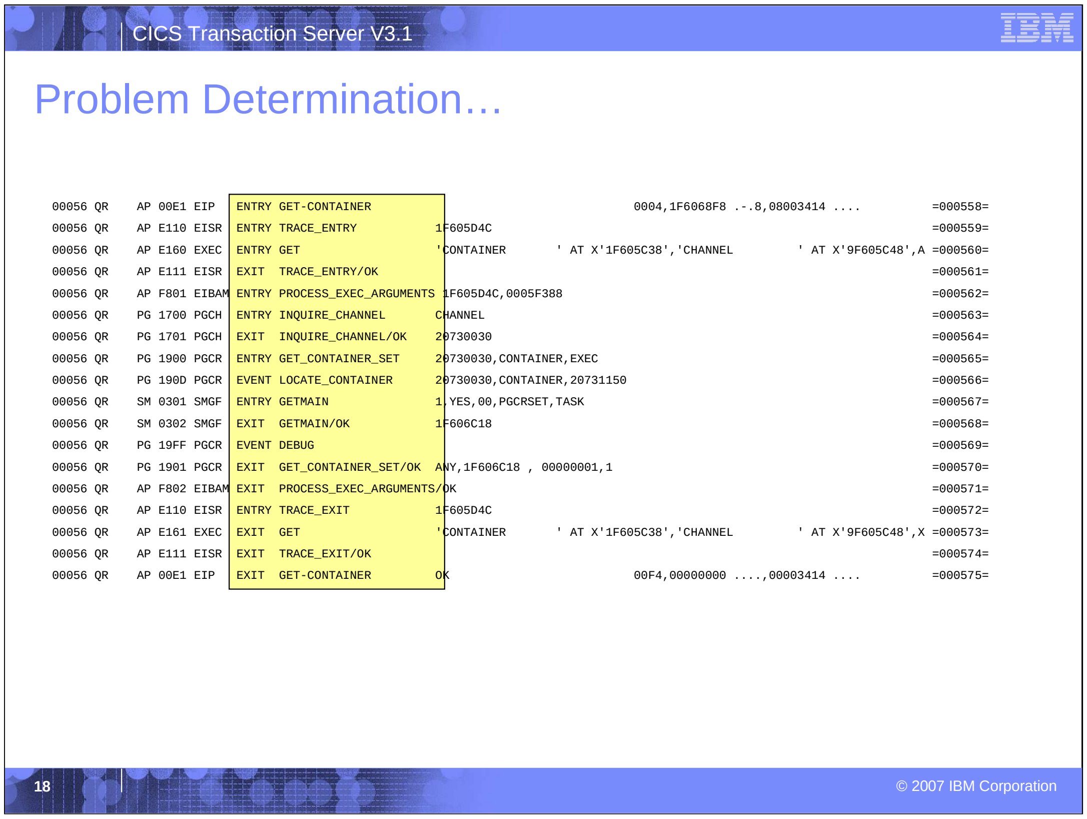
Today a program may issue multiple STARTs with data for a single transaction id. CICS will start one instance of the transaction. The program can issue multiple RETRIEVEs to get the data. When using the channel option on the start, CICS will start one transaction for each start request. The started transaction will be able to access the contents of a single channel. The started transaction will get a copy of the channel.



Problem Determination

- **Maintenance required on prior releases to add error messages**
  - APARs required
    - CICS TS 1.3
      - PQ93048
    - CICS TS 2.2 and 2.3
      - PQ92437
    - CICS TXSeries
      - ?
    - CICS TS for VSE
      - PQ83049
    - CICS for Windows
      - ?
    - CICS for AS/400
      - SE15875
  - An attempt to ship a container to a previous release will result in a transaction abend
    - AXF9
    - AXTT
    - AZTD

Maintenance will be required on prior releases of CICS to enable a clean error message to be produced if you attempt to use the channel option on a command shipped to a pre CICS TS 3.1 release.



CICS Transaction Server V3.1

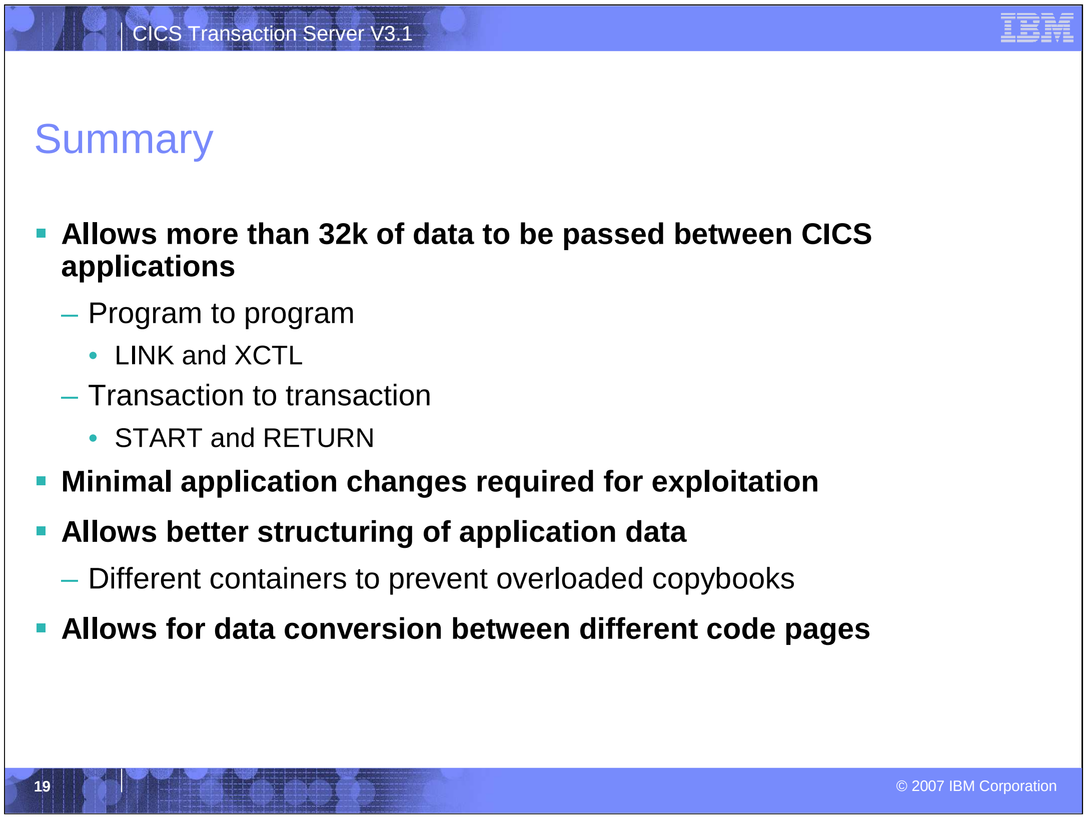
Problem Determination…

```
00056 QR    AP 00E1 EIP   ENTRY GET-CONTAINER                                     0004,1F6068F8 .-.8,08003414 ....          =000558=
00056 QR    AP E110 EISR  ENTRY TRACE_ENTRY           1F605D4C                                                              =000559=
00056 QR    AP E160 EXEC  ENTRY GET                   'CONTAINER       ' AT X'1F605C38','CHANNEL         ' AT X'9F605C48',A =000560=
00056 QR    AP E111 EISR  EXIT  TRACE_ENTRY/OK                                                                              =000561=
00056 QR    AP F801 EIBAM ENTRY PROCESS_EXEC_ARGUMENTS 1F605D4C,0005F388                                                    =000562=
00056 QR    PG 1700 PGCH  ENTRY INQUIRE_CHANNEL       CHANNEL                                                               =000563=
00056 QR    PG 1701 PGCH  EXIT  INQUIRE_CHANNEL/OK    20730030                                                              =000564=
00056 QR    PG 1900 PGCR  ENTRY GET_CONTAINER_SET     20730030,CONTAINER,EXEC                                               =000565=
00056 QR    PG 190D PGCR  EVENT LOCATE_CONTAINER      20730030,CONTAINER,20731150                                           =000566=
00056 QR    SM 0301 SMGF  ENTRY GETMAIN               1,YES,00,PGCRSET,TASK                                                 =000567=
00056 QR    SM 0302 SMGF  EXIT  GETMAIN/OK            1F606C18                                                              =000568=
00056 QR    PG 19FF PGCR  EVENT DEBUG                                                                                       =000569=
00056 QR    PG 1901 PGCR  EXIT  GET_CONTAINER_SET/OK  ANY,1F606C18 , 00000001,1                                             =000570=
00056 QR    AP F802 EIBAM EXIT  PROCESS_EXEC_ARGUMENTS/OK                                                                   =000571=
00056 QR    AP E110 EISR  ENTRY TRACE_EXIT            1F605D4C                                                              =000572=
00056 QR    AP E161 EXEC  EXIT  GET                   'CONTAINER       ' AT X'1F605C38','CHANNEL         ' AT X'9F605C48',X =000573=
00056 QR    AP E111 EISR  EXIT  TRACE_EXIT/OK                                                                               =000574=
00056 QR    AP 00E1 EIP   EXIT  GET-CONTAINER         OK                          00F4,00000000 ....,00003414 ....          =000575=
```

18

© 2007 IBM Corporation

This is an example of the additional data produced on a CICS Transaction dump when a channels are present in the current linkage stack.



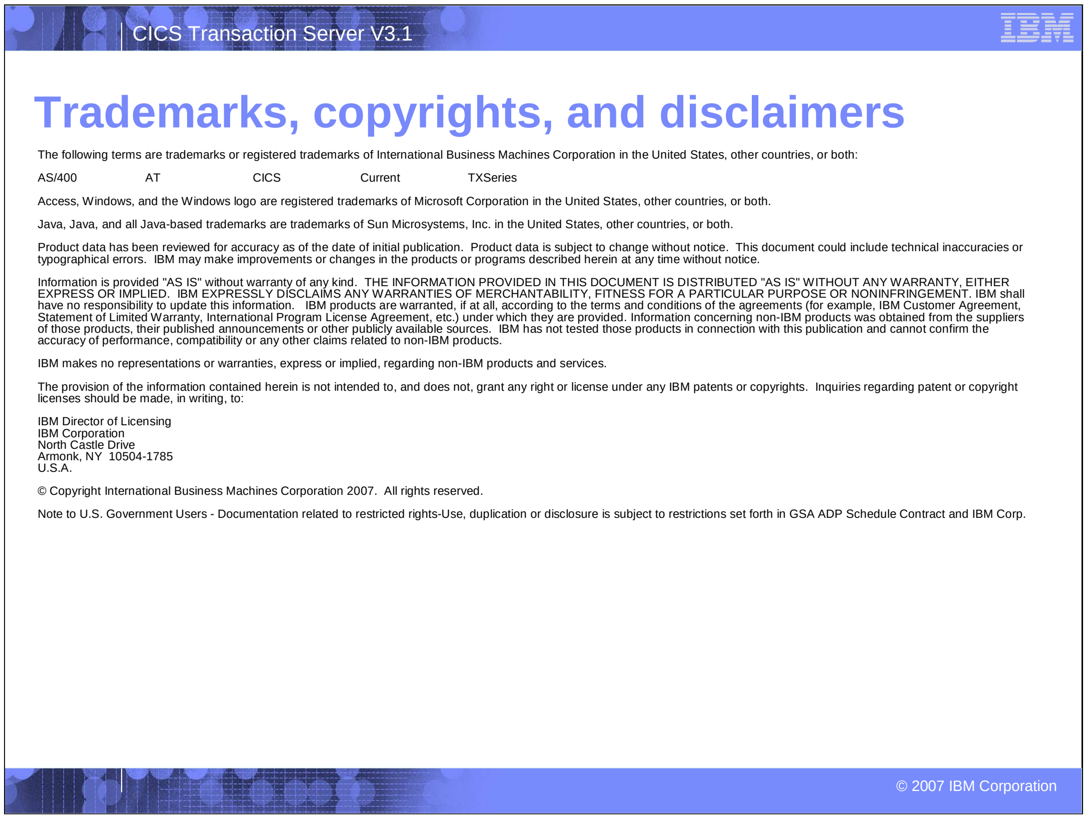
Summary

- **Allows more than 32k of data to be passed between CICS applications**
  - Program to program
    - LINK and XCTL
  - Transaction to transaction
    - START and RETURN
- **Minimal application changes required for exploitation**
- **Allows better structuring of application data**
  - Different containers to prevent overloaded copybooks
- **Allows for data conversion between different code pages**

Channels and containers provide a significant benefit to the application program. The programmer now has the capability to exchange more than 32K of information between application programs and started tasks.

The channel and container construct allows the application suite to be enhanced by adding additional containers to the channel but will not affect programs that do not require the additional data.

The capability to pass multiple containers within a single channel offers the opportunity to simplify the copybook layout making the program easier to understand and future changes simpler to implement.



Trademarks, copyrights, and disclaimers