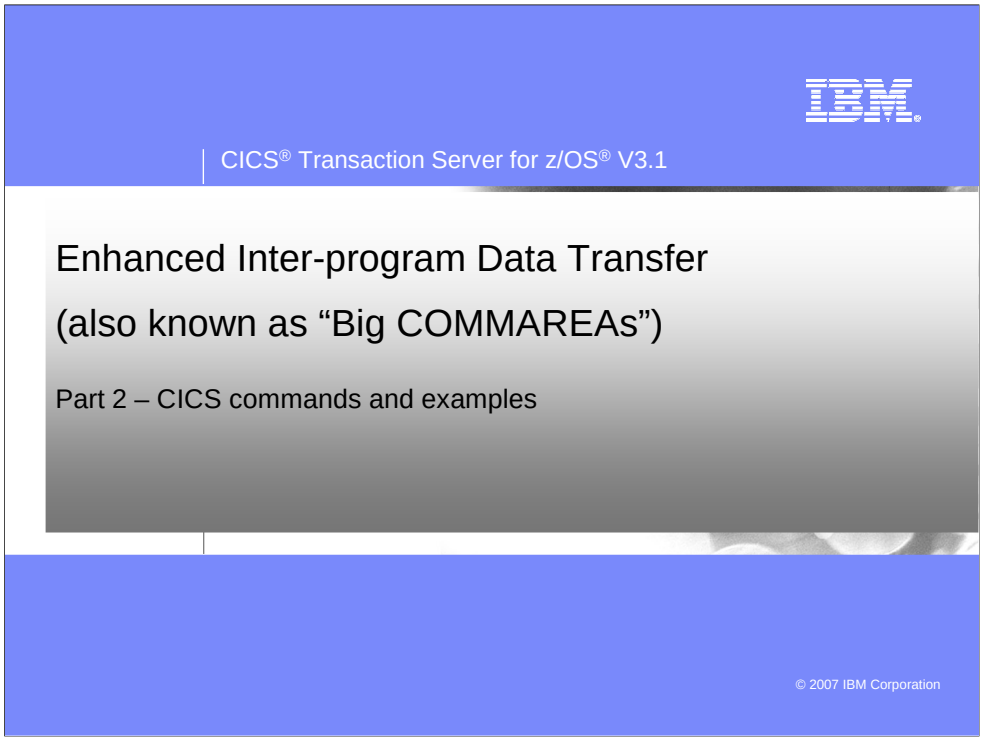
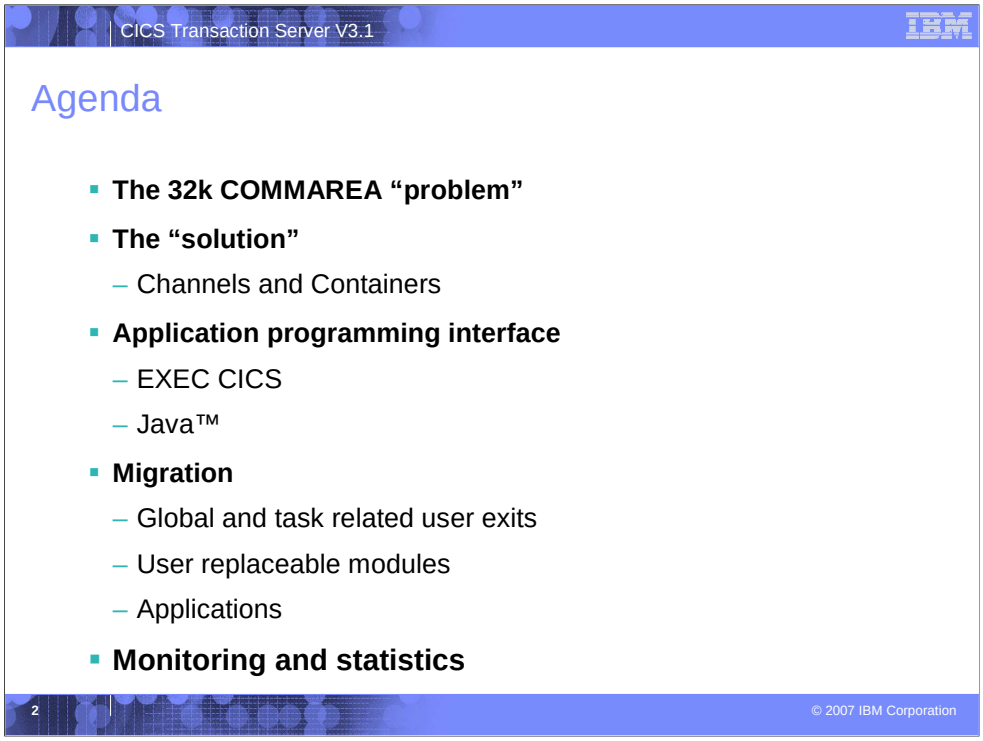


This presentation describes the capabilities provided by the Enhanced Inter-program Data Transfer function introduced in CICS Transaction Server 3.1. This function will allow programs and transactions to exchange more than 32K of data when using a LINK, XCTL, START or RETURN TRANSID command.

While not technically correct, to help you understand this capability you might think of it as being equivalent to “Big COMMAREAs”.



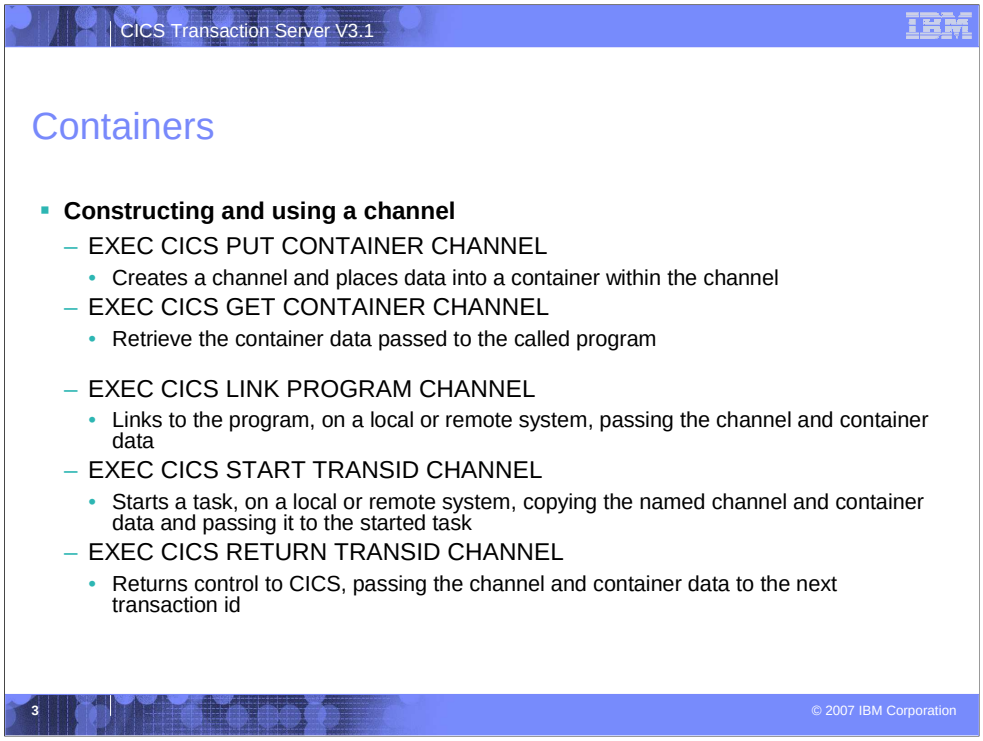
This presentation will discuss the problems that are encountered by programs encountering the 32K COMMAREA limitation, techniques that have been used to circumvent the 32K limitation and then will discuss the CICS solution to the problem.

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.



To create a channel, if it doesn't exist, and to place container data within the channel you will use an EXEC CICS PUT CONTAINER CHANNEL command.

To link or transfer control to another program passing a channel and its associated containers you will use an EXEC CICS LINK PROGRAM CHANNEL or EXEC CICS XCTL PROGRAM CHANNEL command. Remember that you may pass a channel or COMMAREA to a program but not both.

To start a new transaction and pass channel data to the new task you will use an EXEC CICS START TRANSID CHANNEL command. In the case of the START command the channel data is copied from the original channel and passed to the started transaction. At this point there are two separate copies of the channel data. If your starting program continues to make changes to the original container data in the channel it will not be reflected in the copy given to the started task.

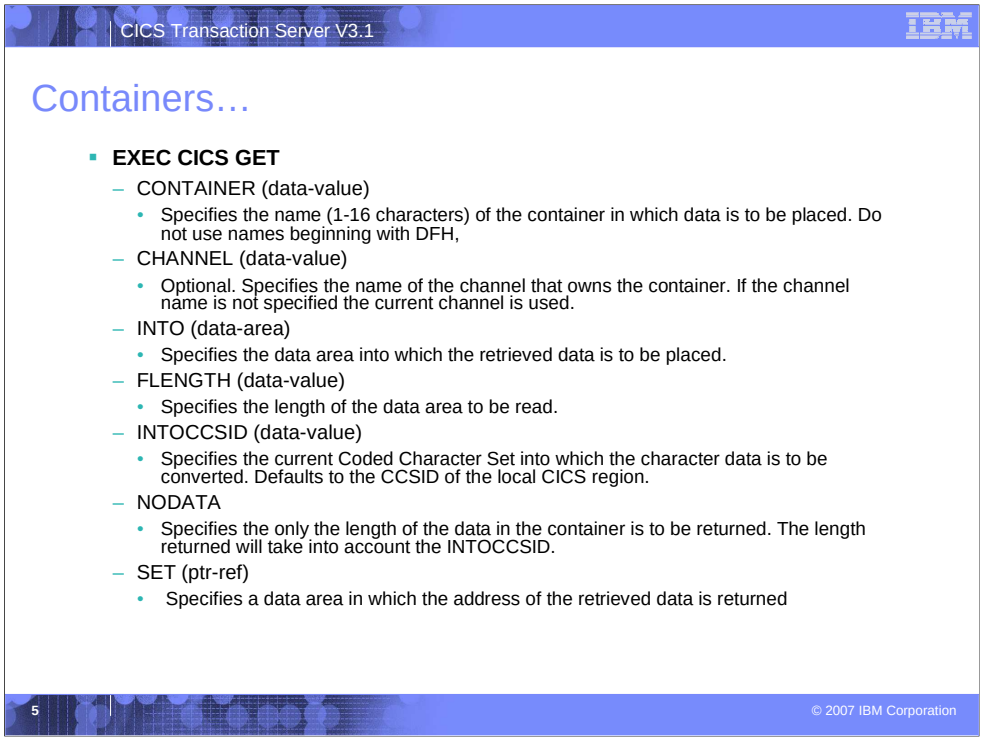
To begin or continue a pseudo-conversational task you will use an EXEC CICS RETURN TRANSID CHANNEL command. This command is only valid at the highest logical level, that is, a program that is returning control to CICS.

To retrieve data passed to your program you will use an EXEC CICS GET CONTAINER CHANNEL command.

Containers…

- **EXEC CICS PUT**
  - CONTAINER (data-value)
    - Specifies the name (1-16 characters) of the container in which data is to be placed. Do not use names beginning with DFH,
  - CHANNEL (data-value)
    - Optional. Specifies the name of the channel that owns the container. If the channel name is not specified the current channel is used.
  - FROM (data-area)
    - Specifies the data area from where the data to be saved is read.
  - FLENGTH (data-value)
    - Specifies the length of the data area to be saved.
  - FROMCCSID (data-value)
    - Specifies the current Coded Character Set of the character data to be put into the container. Defaults to the CCSID of the local CICS region.
  - DATATYPE (CVDA)
    - BIT
      - Default value. Bit data. The data in the container cannot be converted.
    - CHAR
      - Character data. The data is in the code page specified by FROMCCSID and will be converted to an internal CICS code page.

The format and options of the EXEC CICS PUT CONTAINER command.



The format and options of the EXEC CICS GET CONTAINER command.

If you use the SET (ptr-ref) parameter the data area returned will be valid until:

A subsequent GET CONTAINER command is issued for the same container in the same channel by any program that can access the channel or until the channel goes out of scope.

The container is deleted by a DELETE CONTAINER command.

The container is moved by a MOVE CONTAINER command

This is CICS managed storage – do NOT issue a FREEMAIN against the storage area.

Should you need to ensure the data is kept, move the data to your own application storage.

Containers…

- **EXEC CICS MOVE**
  - CONTAINER (data-value)
    - Specifies the name (1-16 characters) of the source container that is to be moved.
  - CHANNEL (data-value)
    - Optional. Specifies the name of the channel that owns the source container. If the channel name is not specified the current channel is used.
  - TOCHANNEL (data-value)
    - Specifies the name of the channel that will own the target container
  - AS (data-value)
    - Specifies the name of the target container

The format and options of the EXEC CICS MOVE CONTAINER command.

Containers…

- **EXEC CICS DELETE**
  - CONTAINER (data-value)
    - Specifies the name (1-16 characters) of the container that is to be deleted.
  - CHANNEL (data-value)
    - Optional. Specifies the name of the channel that owns the container. If the channel name is not specified the current channel is used.

The format and options of the EXEC CICS DELETE CONTAINER command.

Containers…

- **Discovering what containers have been passed**
  - Examples:
    - Output containers
    - Error Containers
- **EXEC CICS Commands**
  - STARTBROWSE CONTAINER
    - CHANNEL option added
  - GETNEXT CONTAINER (data-area)
  - ENDBROWSE CONTAINER

When your application program may be passed a channel with a varying number of containers it may use the container browse commands to determine all the containers present in the channel. If your program is only attempting to determine if a specific container has been passed, such as an error container, it is more efficient to issue a GET CONTAINER command against the single container name (for example ERROR).

The CICS commands comprising the browse interface for containers are STARTBROWSE CONTAINER, GETNEXT CONTAINER and ENDBROWSE CONTAINER.

The order in which the containers are returned is not guaranteed.

COBOL Calling Program Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CLIENT1.

WORKING-STORAGE SECTION.

  COPY INQINTC
* copybook INQINTC
* Channel name
* 01 INQUIRY-CHANNEL PIC X(16) VALUE 'inqcustrec'.

* Container names
* 01 CUSTOMER-NO     PIC X(16) VALUE 'custno'.
* 01 BRANCH-NO       PIC X(16) VALUE 'branchno'.
* 01 CUSTOMER-RECORD PIC X(16) VALUE 'custrec'.

* Define the data fields used by the program
* 01 CUSTNO PIC X(8).
* 01 BRANCHNO PIC X(5).
* 01 CREC.
*   02 CUSTNAME  PIC X(80).
*   02 CUSTADDR1 PIC X(80).
*   02 CUSTADDR2 PIC X(80).
*   02 CUSTADDR3 PIC X(80).
```


A sample COBOL program using channels and containers,

COBOL Calling Program Example…

```
PROCEDURE DIVISION.
 MAIN-PROCESSING SECTION.

* ... Create CUSTNO and BRANCHNO

* ... Create Channel and Container data

EXEC CICS PUT CONTAINER(CUSTOMER-NO) CHANNEL(INQUIRY-CHANNEL) FROM(CUSTNO)
FLENGTH(LENGTH OF CUSTNO) END-EXEC

EXEC CICS PUT CONTAINER(BRANCH-NO) CHANNEL(INQUIRY-CHANNEL) FROM(BRANCHNO)
FLENGTH(LENGTH OF BRANCHNO) END-EXEC

EXEC CICS LINK PROGRAM('SERVER1') CHANNEL(INQUIRY-CHANNEL) END-EXEC

EXEC CICS GET CONTAINER(CUSTOMER-RECORD) CHANNEL(INQUIRY-CHANNEL) INTO(CREC)
END-EXEC

* ...Process customer record

EXEC CICS RETURN END-EXEC

EXIT.
```


A sample COBOL program using channels and containers,

COBOL Server Program Example

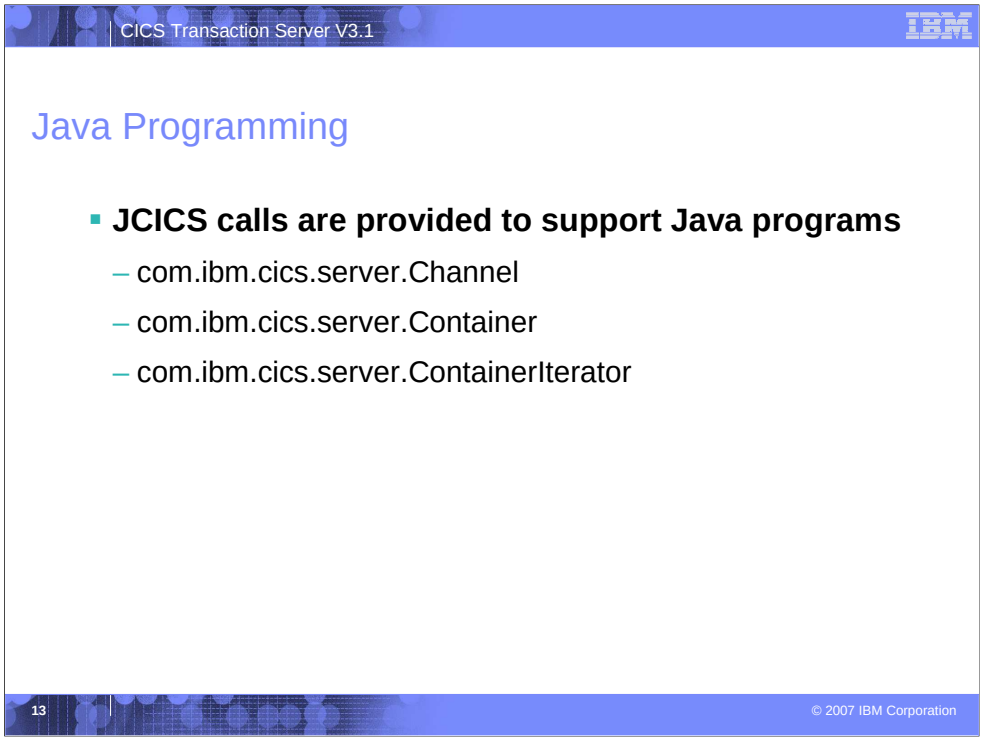
```
IDENTIFICATION DIVISION.
PROGRAM-ID. SERVER1.

WORKING-STORAGE SECTION.

  COPY INQINTC
* copybook INQINTC

* Channel name
* 01 INQUIRY-CHANNEL PIC X(16) VALUE 'inqcustrec'.

* Container names
* 01 CUSTOMER-NO     PIC X(16) VALUE 'custno'.
* 01 BRANCH-NO       PIC X(16) VALUE 'branchno'.
* 01 CUSTOMER-RECORD PIC X(16) VALUE 'custrec'.

* Define the data fields used by the program
* 01 CUSTNO PIC X(8).
* 01 BRANCHNO PIC X(5).
* 01 CREC.
*   02 CUSTNAME PIC X(80).
*   02 CUSTADDR1 PIC X(80).
*   02 CUSTADDR2 PIC X(80).
*   02 CUSTADDR3 PIC X(80).
```


A sample COBOL program using channels and containers,

COBOL Server Program Example…

```
PROCEDURE DIVISION.
MAIN-PROCESSING SECTION.

EXEC CICS GET CONTAINER(CUSTOMER-NO) INTO(CUSTNO) END-EXEC

EXEC CICS GET CONTAINER(BRANCH-NO) INTO(BRANCHNO) END-EXEC

... USE CUSTNO AND BRANCHNO TO FIND CREC IN A DATABASE

EXEC CICS PUT CONTAINER(CUSTOMER-RECORD) FROM(CREC) FLENGTH(LENGTH OF CREC)
END-EXEC

EXEC CICS RETURN END-EXEC

EXIT.
```


A sample COBOL program using channels and containers,



There are three new classes to support channels and containers in the Java programming language.

A Channel class used to create new containers in a channel.

A Container class used to place data in a container.

A ContainerInterator class used to browse the current channel.

Java Programming…

- **Creating a new channel**
  - Create a channel using the createChannel method of the task class

```
Task t = Task.getTask();
Channel custData = t.createChannel("Customer_Data");
```


An example of creating a new channel using the ChannelFactory class.

Java Programming…

- **Creating a new container**
  - Create the container using the createContainer method of the Channel class

```
Container custRec = custData.createContainer("Customer_Record");
```

- **Putting data into the container**
  - Use the Container.put() method
    - Data can be added as a byte array or string

```
String custNo = "00054321";
byte[] custRecIn = custNo.getBytes();
custRec.put(custRecIn);
```

  - Or

```
custRec.put("00054321");
```


An example of creating a new container and placing data in the newly created container.

Java Programming…

- **Passing a channel to another program**
  - Use the link() and xctl() methods of the Program class

```
programX.link(custData);

programY.xctl(custData);
```

- **Passing a channel to another task**
  - Use the issue method of the StartRequest class

```
StartRequest.issue(custData);
```

  - Use the setNextChannel() method of the TerminalPrincipalFacility class:

```
terminalPF.setNextChannel(custData);
```


An example of passing a channel to another program or another task.

Java Programming…

- **Receiving the current channel**
  - Use the getCurrentChannel() method of the Task class

```
Task t = Task.getTask();
Channel custData = t.getCurrentChannel();
Container custRec = custData.getContainer("Customer_Record");
```

- **Getting data from a container**
  - Use the Container.get() method to read the data in a container into a byte array:

```
byte[] custInfo = custRec.get();
```


An example of getting the current channel and retrieving data from a container within the channel.

Java Programming…

- **Browsing the current channel**
  - Use the ContainerIterator object

```
Task t = Task.getTask();
ContainerIterator ci = t.containerIterator();
While (ci.hasNext()) {
   Container custRec = ci.next();
    // Process the container... }
```


An example of determining what containers are present in a channel using the ContainerIterator class.

Java Programming Example

```
import com.ibm.cics.server.*;
public class Payroll {
...
// create the payroll_2005 channel
Task t = Task.getTask();
Channel payroll_2005 = t.createChannel("payroll-2005");

// create the employee container Container employee =
payroll_2005.createContainer("employee");

// put the employee name into the container
employee.put("John Doe");

// create the wage container Container wage =
payroll_2005.createContainer("wage");

// put the wage into the container
wage.put("2000");
```


A sample program using channels and containers written in Java.

Java Programming Example…

```
// Link to the PAYROLL program, passing the payroll_2005 channel
Program p = new Program();
p.setName("PAYR");
p.link(payroll_2005);

// Get the status container which has been returned
Container status = payroll_2005.getContainer("status");

// Get the status information
byte[] payrollStatus = status.get(); ... }
```


A sample program using channels and containers written in Java.

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:

CICS

Java, Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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.

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.

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.