IBM Network Data Couplers
For MVS/ESA and OS/390

IBM Network Data Couplers User's Guide

GC31-8701-00

IBM Network Data Couplers
For MVS/ESA and OS/390

IBM Network Data Couplers User's Guide

GC31-8701-00

Note!

Before using this document, read the general information under “Appendix C. Notices” on page 155.

First Edition (June 1998)

This document applies to IBM Network Data Couplers for MVS/ESA and OS/390, an IBM licensed program, which runs under the following operating systems:

OS/390

MVS/Enterprise System Architecture

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch serving your locality.

A form for readers’ comments has been provided at the back of this document. If this form has been removed, you may address comments to IBM Corporation, Information Development, Department CGMD / Bldg 500, P.O. Box 12195, Research Triangle Park, NC 27709-9990, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About This Book

This book explains how to install, administer, and use IBM Network Data Couplers for MVS/ESA and OS/390 (referred to as IBM Network Data Couplers for short). IBM Network Data Couplers enable systems and applications programmers to reliably and securely link MVS, UNIX, NetWare**, Windows NT**, or OS/2 data. There are two IBM Network Data Couplers, Network File System Coupler (NFS Coupler) and Remote Shell Coupler (RSH Coupler). See "Chapter 1. What Are IBM Network Data Couplers?" on page 3, for a description of the Network Data Couplers.

Who Should Read This Book

This book is intended primarily for systems and applications programmers who are responsible for tying together existing or new applications in distributed network environments that include MVS or OS/390 and UNIX-based systems or PC network operating systems, such as NetWare, Windows NT, or OS/2. The book also covers information useful to system administrators responsible for installing RSH Coupler and NFS Coupler and for configuring them for optimum system performance.

Before using this book, you should be familiar with the IBM Multiple Virtual Storage (MVS), the TCP/IP, NFS, and RSH protocols.

How This Book Is Organized

This book is organized into three parts:

- **Part 1. An Introduction to IBM Network Data Couplers** contains the chapters listed below:
  - "Chapter 1. What Are IBM Network Data Couplers?" on page 3 describes each of the IBM Network Data Couplers and explains their features and benefits. The chapter also lists examples of applications that can benefit from using IBM Network Data Couplers.
  - "Chapter 2. Installation and Customization" on page 9 describes how to install and customize IBM Network Data Couplers.
  - "Chapter 3. Operations" on page 15 explains the requirements and tasks for starting and operating RSH Coupler and NFS Coupler.
- **Part 2. Using NFS Coupler** contains the chapters listed below:
  - "Chapter 4. Administration" on page 19 explains the steps that MVS and NFS server administrators must follow to ensure NFS Coupler works properly.
  - "Chapter 5. Performance" on page 31 describes how NFS Coupler timing parameters can be changed when performance improvements are desired.
  - "Chapter 6. Application Programmer's Guide" on page 33 explains how programmers can use NFS Coupler to enable their applications to read from and write to files residing on UNIX, OS/2, Netware, Windows NT, or other NFS servers.
- **Part 3. Using RSH Coupler** contains the chapters listed below:
  - "Chapter 7. Administration" on page 53 describes the requirements and tasks related to RSH Coupler administration. The chapter covers the following topics:

- "Security Administration" on page 54 explains how to code access rules necessary to enable PC and UNIX clients to execute RSH Coupler APPC transactions.
- "APPC Administration" on page 58 explains how to code DD statements to remap APPC transaction input and output to standard input, error, and output of the `rsh` command on the client machine.
- "Data Conversion" on page 64 describes data conversion and record mapping for all types of RSH Coupler files. Data conversion and record mapping are required because MVS and UNIX files use different formats and different character sets to store printable characters.
- "Sharing Input and Output" on page 66 explains how input and output RSH Coupler files within the same RSH Coupler APPC transaction are shared. The same type of RSH Coupler file can be opened multiple times, concurrently or otherwise, using the same DD name or different DD names.

– "Chapter 8. Application Programmer's Guide" on page 69 explains how RSH Coupler can be used to enable end users on UNIX and PC workstations to remotely execute existing MVS batch applications as TCP/IP servers. The chapter covers the following topics:

- "Using Existing MVS Batch Programs as TCP/IP Servers" on page 69 explains how to use existing MVS batch applications as TCP/IP servers via RSH Coupler.
- "Using the MULTI-TRAN RSH Coupler Facility" on page 69 describes how to use the RSH Coupler MULTI-TRAN facility by either coding a new RSH Coupler APPC transaction or by modifying an existing MVS batch application.
- "Using the RSH Command From UNIX and PCs" on page 73 explains to use the **rsh** command on the client side in either the UNIX or PC environment.
- "Trouble Shooting" on page 75 explains how to trouble shoot problems with RSH Coupler.
- "RSH Coupler Examples" on page 76 contains six examples that are distributed with RSH Coupler.

– "Chapter 9. Performance" on page 99 describes how RSH Coupler performance could be affected by a lack of MVS resources, insufficient TCP/IP network bandwidth, or a lack of APPC resources on MVS.

IBM Network Data Couplers User's Guide includes the following appendices:

"Appendix A. IBM Network Data Couplers Messages" on page 101 contains messages for IBM Network Data Couplers.

"Appendix B. Prerequisites and Limitations" on page 153 describes prerequisites and limitations of which you should be aware before installing IBM Network Data Couplers for MVS/ESA and OS/390.

"Appendix C. Notices" on page 155 contains general information with which you should be familiar before using this book.

How to Contact IBM

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

For comments and suggestions about *ù*, use the Reader Comment Form located at the back of this book. This form gives instructions for submitting your comments by mail, by fax, or electronic mail.

Where to Find Information on the Internet

You can read more about IBM Network Data Couplers for MVS/ESA and OS/390 by visiting the website shown below:

http://www.software.ibm.com/enetwork/datacouplers/

Part 1. An Introduction to IBM Network Data Couplers

Chapter 1. What Are IBM Network Data Couplers?

IBM Network Data Couplers for MVS/ESA and OS/390 enable applications and systems programmers to reliably and securely link MVS, UNIX, NetWare, Windows NT, or OS/2 data. There are two Network Data Couplers, NFS Coupler and RSH Coupler. The two Network Data Couplers function independently of each other, but they are similar in that they both are MVS-based solutions that rely on standard, open protocols to enable access to information resources across network environments that include UNIX-based systems or PC network operating sytems. The key features and benefits of each product are described in the sections that follow.

Advantages at a Glance

IBM Network Data Couplers offer many advantages to enterprises with distributed network environments that need to link existing applications, develop new applications, or link data across MVS, UNIX, NetWare, Windows NT, or OS/2 systems. IBM Network Data Couplers enable enterprises to quickly and reliably accomplish these objectives with no recoding of existing applications, no retraining of programmers, and minimal installation, security, and administration requirements.

Major advantages of using IBM Network Data Couplers are:

- **Secure access to data:**
  - NFS Coupler uses standard NFS security and provides a mechanism to map MVS and UNIX NFS security differences transparently to the application.
  - RSH Coupler controls which combination of end user and machine (clients) is allowed to execute MVS jobs remotely.
- **Easy single MVS or OS/390 installation:**
  - End user machines (clients) do not require any new software.
- **Open Protocols:**
  - IBM Network Data Couplers use open TCP/IP protocols — Network File System (NFS) and Remote Shell (RSH).
  - The need to retrain programmers to use private network protocols is eliminated because private protocols between each end of the coupling are not required.
- **No changes to MVS applications:**
  - Simple JCL DD statements drive the IBM Network Data Couplers.
- **No passwords needed or transferred:**
  - IBM Network Data Couplers use the security built in to the operating environment. This eliminates the need for new user ids or passwords.
- **No data duplication:**
  - IBM Network Data Couplers enable applications to access data directly. Because applications can access data directly, data does not have to be duplicated. This improves system performance and resource utilization and eliminates the risk of inconsistencies in duplicated data.

NFS Coupler

NFS Coupler uses the Network File System (NFS) protocol to enable MVS and OS/390 applications to read and write to files residing on UNIX, OS/2, NetWare, Windows NT, or other NFS servers. Access to files on remote servers is provided through the standard sequential MVS access methods — Queued Sequential Access Method (QSAM) and Basic Sequential Access Method (BSAM). NFS Coupler services are invoked by coding subsystem parameters in JCL DD statements.

For example, to copy an MVS file `'USER1.SALES.REPORT'` to UNIX file `/home/reports/sales/current` on system `UNIXA`, use IEBGENER as shown in Figure 1:

```
//JOB cards here
//*
//COPY EXEC PGM=IEBGENER
//*
//SYSUT1   DD DSN=USER1.SALES.REPORT,DISP=SHR
//SYSUT2   DD SUBSYS=(IORR,ICDNOPCL,        REQUIRED CONSTANTS
//            UNIXA,                        UNIX SYSTEM NAME
//            ,'/home/reports',             NFS MOUNT POINT
//            'sales/current')              FILE NAME
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
/*
```

Figure 1. IEBGENER Job to Copy an MVS File to a UNIX File

Key Benefits of Using NFS Coupler

Some of the benefits of using NFS Coupler are:

- NFS Coupler gives MVS and OS/390 applications seamless, transparent read/write access to files on remote UNIX or PC network operating systems servers. By eliminating the need to transfer data between MVS and remote servers, NFS Coupler saves time and reduces CPU and I/0 use.
- Robust fault tolerance capabilities and support for MVS Checkpoint /Restart make NFS Coupler suitable for mission critical applications.
- NFS Coupler's full support of REXX allows it to be used from any automation tool currently available on MVS.
- NFS Coupler is easy to implement and point-of-control issues are simplified because installation of NFS Coupler is required only on the MVS or OS/390 system.
- NFS Coupler provides seamless conversion between MVS data formats and UNIX or PC network operating system formats used on remote servers.
- NFS Coupler reduces the risk of security violations by using the standard NFS security protocol. MVS user ids are mapped into the corresponding user ids of the NFS servers where the files being accessed reside. No passwords need to be supplied or transmitted across the network.
- NFS Coupler provides the ability to create directories and files on the NFS servers and to specify their protection mode.
- NFS Coupler also provides the ability to delete a file at closure.

RSH Coupler

RSH Coupler enables end users on UNIX and PC workstations to remotely execute existing MVS batch applications as TCP/IP servers. For example, existing applications, such as database report generators, can be remotely executed from UNIX or PCs, on demand, in real time, and without the need to recode, recompile, or relink the existing MVS batch applications.

RSH Coupler is based on the RSH (Remote SHell) open protocol defined over TCP/IP. End users on client machines can execute MVS jobs by using the `rsh` command, which is part of UNIX and is widely available on PC network operating systems such as NetWare, Windows NT, and OS/2.

When an end user invokes the **rsh** command to request the remote execution of an MVS job, RSH Coupler uses the MVS/APPC services to schedule the job. These jobs are therefore RSH Coupler APPC transactions or simple transactions. RSH Coupler provides the appropriate data and protocol translation so that the input and output of the **rsh** command on the end users machine is mapped to the input and output of the job being executed on the MVS system.

The JCL statements used by APPC to execute an RSH Coupler APPC transaction are like other JCL statements, with the exception of a `SUBSYS=` keyword, which is used to identify the DD names mapped into the **rsh** command input or output. Each read or write request for these DD names is translated into a send or receive request from or to the `rsh` command on the end users (client) system.

Key Benefits of Using RSH Coupler

Some of the benefits of using RSH Coupler are:

- Existing applications can be executed in a distributed environment without the need to recode, recompile, or relink the existing software.
- With RSH Coupler, there is no need to retrain programmers. New servers can be written as batch programs that read and write network data as input and output files, using standard MVS access methods. Debugging requires no experience in network programming.
- Installation is easy because RSH Coupler is installed only on the MVS or OS/390 system. No installation is required on the UNIX or PC client systems because applications are executed using the **rsh** command, which is available as a standard command on any flavor of UNIX and on most TCP/IP packages for DOS, Windows, and OS/2 operating systems.
- RSH Coupler reduces the risk of security violations by using the standard RSH security. No passwords need to be supplied or transmitted across the network. RSH Coupler APPC transactions execute under the correct MVS UID as if they were submitted by a TSO user.

RSH Coupler Example

This example illustrates how RSH Coupler can enable UNIX and PC users to use batch MVS programs without recoding, recompiling, or relinking the existing programs. Consider a PL/1 program, PLIREP, which was written more than 10 years ago. Its function is to extract data from a proprietary data base, according to the selection parameters read from the input file DD=SYSIN. The data extracted is written to the output file DD=SYSDATA. Diagnostic messages are printed to the file DD=SYSPRINT.

What Are IBM Network Data Couplers?

Users on UNIX and PC workstations need to run the PL/1 program several times a day in real time. However the developer who wrote the application is unavailable and no one else is available who is qualified to convert the original code into a TCP/IP server using the socket library. By using RSH Coupler, recoding is unnecessary.

With RSH Coupler, UNIX or PC users can invoke in real time the unmodified PL/1 program with the following command:

```
rsh mvs_host plirep < datain > dataout
```

where local files `datain` and `dataout` are the input and output files to the PL/1 program being remotely executed.

To enable UNIX or PC users to invoke PLIREP using RSH Coupler, the MVS administrator defines, only once, an APPC transaction to MVS with name `RSH_plirep` that executes with the JCL statements shown in Figure 2:

```
//PLIREP  JOB
//*
//PLIREP   EXEC PGM=PLIREP
//*
//SYSIN    DD SUBSYS=(IORR,ICDROPCL,STDIN)    STDIN REMAP
//SYSOUT   DD SUBSYS=(IORR,ICDROPCL,STDOUT)   STDOUT REMAP
//SYSPRINT DD SUBSYS=(IORR,ICDROPCL,STDERR)   STDERR REMAP
//*
//SYSDATA  DD DISP=SHR,DSN=USER1.VSAM.DATABASE
//*
//@ICDRDD@ DD SUBSYS=(IORR,ICDRRMGR)          REQUIRED
```

Figure 2. JCL Member PLIREP

Note: See “RSH Coupler Examples” on page 76 for the following additional examples that are distributed with RSH Coupler:

- “Example 1. Using RSH Coupler for a Remote TSO Session” on page 76
- “Example 2. Using MVS Sort From UNIX and PCs” on page 78
- “Example 3. Using Batch SPUFI to Access DB2 Data” on page 80
- “Example 4. Printing UNIX Files on an MVS Printer” on page 82
- “Example 5. Using IDCAMS Service From UNIX and PCs” on page 84
- “Example 6. Accessing VSAM KSDS Files From UNIX and PCs” on page 90

The RSH Coupler MULTI-TRAN Facility

RSH Coupler initially was designed to extend the life of existing MVS applications in a TCP/IP network environment (see example in “RSH Coupler Example” on page 5). However, functionality was added to RSH Coupler to enable a single RSH Coupler APPC transaction to handle multiple **rsh** commands by using the MVS/APPC MULTI-TRAN facility. The use of a single RSH Coupler APPC transaction to serve multiple `rsh` command requests greatly enhances perfomance, because the overhead to schedule an APPC transaction, open and close data bases, etc., for every `rsh` command request is avoided. The use of MULTI-TRAN APPC requires that existing MVS batch application be slightly modified. Nevertheless, the following advantages are retained:

- The level of modification needed is relatively minor.
- Data coming through the network from the client machine is still read from the input file using standard MVS access methods. The same is true for data going back through the network to the client machine.
- Programmers in charge of modifying the program do not need any experience in data communication because standard file access is used to send and receive data.
- At any time the RSH Coupler APPC transaction will be running under the correct MVS user ID which is transparently changed to the new correct value as a new `rsh` command is being served.

Application Examples

Following are examples of applications that can benefit from the use of IBM Network Data Couplers for MVS/ESA and OS/390.

Typical applications that take advantage of NFS Coupler:

- Report distribution to UNIX machines from MVS.
- Unattended file transfer to and from MVS.
- Database snapshots from the UNIX hosts to the MVS database and back.
- Any batch window data processing that requires Data access across TCP/IP Network.

Typical applications that take advantage of RSH Coupler:

- Remote execution of report generating jobs.
- Invoke MVS jobs such as DB2 utilities, IMS BMPs, VSAM Access Programs.
- Build and receive reports from such things as MVS batch jobs, SAS report generators.
- On-demand access to MVS/ESA and OS/390 applications and data.

Using NFS and RSH Couplers Together

Even though NFS and RSH Couplers can be used separately, their ability to integrate data and applications across a network becomes even more effective when they are used with each other. Essentially, whenever an application or a user needs networked host-to-server data ″on-demand,″ IBM Network Data Couplers are the solution.

Let’s consider the case of a large financial business that is implementing a new workstation loan processing application on a NetWare platform written in C++. All the necessary data resides in a large DB2 database on MVS/ESA. As loan officers process loans, the new NetWare application requires a data extract for each loan. This data is extracted with QMF and written to the NetWare server each night with NFS Coupler.

In this case, the advantages of using the NFS Coupler are:

1. NFS Coupler eliminates the need to download the full database, which would be too large to be transferred to the NetWare server.
2. With NFS Coupler, only the portion of the data required by loan applications is extracted from the database.

If additional data or a unique data request is required, the loan officer could select an additional feature of the new NetWare application that uses RSH Coupler to make a request to the MVS/ESA host for the required data. With RSH Coupler, the request could be accomplished according to the following steps:

1. Write a QMF query on MVS to extract the required information pertaining to the loan under investigation.
2. Define an RSH Coupler APPC transaction so that the QMF query written in step 1 can be invoked using the **rsh** command from the NetWare side.
3. Use the C language ″system ()″ call within the C++ application to invoke the **rsh** command and execute in real time the QMF query written in step 1.
4. Receive the output data into a temporary file to be read by the application.

Note: There is no need to ask the user to enter an MVS password since the RSH Coupler does not use or transmit passwords.

At this point the application is completed. By using IBM Network Data Couplers, there is instant access to the required data on MVS/ESA (or OS/390) with minimal programming effort.

Chapter 2. Installation and Customization

This chapter describes how to install and customize the IBM Network Data Couplers for MVS/ESA and OS/390. Before installing the IBM Network Data Couplers, you also should be familiar with the information in “Appendix B. Prerequisites and Limitations” on page 153.

Installation

IBM Network Data Couplers are distributed on one tape cartridge. Step-by-step installation instructions can be found in the program directory shipped with the tape cartridge.

After completing the SMPE installation, take the following steps to complete the installation:

1. Add ICD.V1R1M0.SICDLINK to the MVS LINKLIST concatenation as an authorized library.
2. Make ICD.V1R1M0.SICDLOAD an authorized library and protect it so that only system administrators have read access to it.
3. Add to member IEFSSNxx in SYS1.PARMLIB the following line:

   ```
   IORR,ICDSINIT
   ```

   If you already have a MVS subsystem whose name is IORR, choose another name which is convenient to your installation.
4. ReIPL the system.
5. Copy member ICDRSTRT from library ICD.V1R1M0.SICDCNTL to one of the libraries in the system SYSPROC concatenation.

Defining the TCP/IP Stack Used

The IBM Network Data Couplers have been designed to support more than one TCP/IP stack. Currently, the Data Couplers support two stacks:

- The IBM TCP/IP stack using the IUCV interface;
- The INTERLINK stack.

Module ICDBTCP in the LINK library contains all the information the Data Couplers need to use a TCP/IP stack. Several sample JCLs are provided in the JCL library showing how to customize module ICDBTCP according to the TCP/IP stack that will be supporting the Network Data Couplers.

Using the IBM TCP/IP Stack With IUCV

If you decide to use the IBM TCP/IP stack with the IUCV interface you should use JOB ICDVZAP to insert the correct information into module ICDBTCP. However, the default values in module ICDBTCP are set to use the IBM TCP/IP stack with the IUCV interface and with the JOB name of the TCP/IP main address space set to ″TCPIP″. Consequently if the IBM TCP/IP main address space executes with JOB name ″TCPIP″, you do not need to run JOB ICDVZAP.

JOB ICDVZAP modifies module ICDBTCP as follows:

- At offset 0 insert 8 characters of the name of the main TCP/IP address space.
- At offset 9 insert character X'E5' (EBCDIC ″V″). This identifies the IBM TCP/IP stack with the IUCV interface. After executing JOB ICDVZA refresh the LLA.

The listing of JOB ICDVZAP is shown in Figure 3:

```
//ICDVZAP  JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*----------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*----------------------------------------------------------------------
//*
//* Use this JOB to customize the Data Coupler to use the
//*   default IUCV TCP/IP driver with a TCP address space name
//*   different from TCPIP
//*
//*----------------------------------------------------------------------
//*
//    EXEC PGM=IMASPZAP,REGION=512K
//*
//SYSLIB   DD DISP=SHR,DSN=ICD.V1R1M0.SICDLINK
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  NAME ICDBTCP ICDBTCP
  REP  000000  E3C3,D7E3,C5E2,E340     EBCDIC FOR "TCPTEST"
  REP  000008  E5                      "V" IDENTIFY INTERLINK DRIVER
```

Figure 3. JCL Member ICDVZAP

Using the INTERLINK Stack

If the INTERLINK stack is used to support IBM Network Data Couplers, use JOB ICDAZAP to insert the correct information into module ICDBTCP. JOB ICDAZAP modifies module ICDBTCP as follows:

- At offset 0 insert 4 characters of the name of the MVS subsystem used by the the INTERLINK stack;
- At offset 9 insert character X'C1' (EBCDIC ″A″). This identifies the INTERLINK stack. After executing JOB ICDAZAP refresh the LLA. The listing of JOB ICDAZAP is shown in Figure 4:

```
//ICDAZAP  JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*--------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*--------------------------------------------------------------------
//*
//* Use this JOB to customize the Data Coupler to use the INTERLINK
//*   TCP/IP driver
//* The SubSystem name used by TCP/IP is "ACSS"
//*
//*--------------------------------------------------------------------
//*
//    EXEC PGM=IMASPZAP,REGION=512K
//*
//SYSLIB   DD DISP=SHR,DSN=ICD.V1R1M0.SICDLINK
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  NAME ICDBTCP ICDBTCP
  REP  000000  C1C3,E2E2               EBCDIC FOR "ACSS"
  REP  000004  4040,4040               UNUSED, SET TO BLANKS
  REP  000008  C1                      "A" IDENTIFY INTERLINK DRIVER
```

Figure 4. Listing of JOB ICDAZAP

Customizing the ICDRSTRT PROC

Member ICDRSTRT must be copied to a library in the SYSPROC concatenation as part of the installation process. The listing of member ICDRSTRT is shown in Figure Figure 5:

```
//ICDRSTRT PROC  LOGCLS=A,TRANPRF='RSH_',MAXTRDS=63
//*
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//*  This is the Data Coupler RSH server address space PROC
//*
//*-------------------------------------------------------------------
//*
//ICDRSTRT EXEC  PGM=ICDRSTRT,
//         PARM='&LOGCLS;,&TRANPRF;,&MAXTRDS;',
//         TIME=1440,REGION=8M
//*
//STEPLIB  DD DISP=SHR,DSN=ICD.V1R1M0.SICDLOAD
//*
//SYSUDUMP DD SYSOUT=*
```

Figure 5. Listing of JCL Member ICDRSTRT

PROC ICDRSTRT has 3 parameters whose defaults can be customized:

&LOGCLS
: Is the SYSOUT class used by RSH Coupler to write the log data set(s). If the SYSOUT class specified is `'*'` then the SYSOUT class used for the started task will be used. The default value is `A`.

&TRANPRF
: Is the APPC transaction prefix. This string is used to prefix ALL commands issued by the users on the client side. This prefix is used to avoid users on PCs and UNIX systems from accidentally (or intentionally) invoking APPC transactions unrelated to RSH Coupler. The default used in ICDRSTRT is **RSH_**.

&MAXTRDS
: Is the maximum number of concurrent connections supported by RSH Coupler. The default is 63. This number should not be higher than 254 or an error will occur.

Reserving Port 514 for RSH Coupler

Some TCP/IP stacks on MVS allow you to reserve a port for a specific started task name (e.g. the IBM TCP/IP stack). In this case port 514, which is used by the RSH Coupler to listen for incoming connection requests, must be reserved for started task name ICDRSTRT. Consult the installation manual of the TCP/IP stack you are using to verify if it supports reserving ports and how the definition of reserved ports can be changed.

Security Customization for RSH Coupler

An MVS user ID must be created to use the RSH Coupler gateway. The RSH Coupler gateway must have a user ID to be able to create security tokens for the RSH Coupler APPC transactions it starts.

If RACF is installed, a user ID ICDRSTRT (same as the started task procedure name) must be created with default group of SYS1. The RACF administrator can create the ICDRSTRT user ID by typing from the TSO prompt the single RACF command:

```
AU ICDRSTRT DFLTGRP(SYS1)
```

If RACF is installed, RSH Coupler customization is complete.

If an equivalent security product such as ACF-2 is installed, the security administrator should make sure that the RSH Coupler gateway is allowed to call the following SAF (RACROUTE) macros:

- TOKENBLD
- TOKENMAP
- TOKENXTR

The RSH Coupler gateway calls the above primitives using the following RACROUTE (SAF) parameters:

- SUBSYS=ICD
- REQSTOR=ICDRSSEC
- DECOUPLE=YES
- RELEASE=1.9

Chapter 3. Operations

This chapter explains the requirements and tasks for starting and operating RSH Coupler and NFS Coupler.

NFS Coupler Operations

The NFS Coupler executes as an extension to the MVS I/O supervisor. Since it does not have its own address space there is no start or modify command to be issued.

RSH Coupler Operations

The following sections discuss RSH Coupler start and stop commands.

RSH Coupler Start Command

RSH Coupler start and stop commands are explained in the sections that follow.

The RSH Coupler gateway address space executes as a started task and it is started by the operator as follows:

```
S ICDRSTRT
```

Any of the default parameters can be overridden on the start command. For example, if the spin log data set must be allocated with output class E, then the operator must type:

```
S ICDRSTRT,LOGCLS=E
```

RSH Coupler Stop Command

To stop RSH Coupler gateway the operator must issue the stop command as follows:

```
P ICDRSTRT
```

If no TCP/IP connections are active at the time the stop command is issued, ICDRSTRT will terminate immediately. If one or more connections are active, ICDRSTRT will refuse to accept new TCP/IP connections and will wait for 60 seconds (grace period) before terminating. This allows any active TCP/IP connections to terminate normally.

The following modify commands are supported by the RSH Coupler gateway:

CLOse
: To switch to a new log data set. This command allows the RSH Coupler administrator to force a close on the current log data set so that it is available for browsing and trouble shooting.

REFresh:
: To reload the security module ICDRMAP. This command allows the RSH Coupler administrator to generate a new security module and have it reloaded without stopping and restarting ICDRSTRT.

Operations

DISplay
To display current TCP/IP connections. For example, to display current RSH Coupler TCP/IP connections the operator will type:

```
F ICDRSTRT,DIS
```

Part 2. Using NFS Coupler

Chapter 4. Administration

This chapter describes the steps that MVS and NFS server administrators must follow to ensure NFS Coupler works properly.

Overview

NFS Coupler uses the NFS protocol. It acts as a client that asks the NFS server on the UNIX host to perform all I/O operations on its behalf.

When an MVS application opens an NFS Coupler file, NFS Coupler sends a *mount* request to the NFS server for the file in question. The server checks that the host issuing the mount request (the MVS, machine in this case) is authorized to do so. After the file has been mounted, NFS Coupler forwards all read and write requests to the NFS server. When the MVS application closes the file, NFS Coupler sends a *umount* request to the NFS server as an indication that no further I/O requests will be sent.

Routing

NFS Coupler communicates with the NFS server by using the NFS server's IP address. The IP address is a four-byte unsigned integer that uniquely identifies a host in the TCP/IP network. IP addresses are specified in character form, by concatenating the decimal representation of each byte separated by a dot. For example, IP address `128.23.1.24` specifies an IP address of `"80170118"x`.

On most operating systems, including MVS and UNIX, hosts are also identified by a host name, which can be mapped through a socket call into an IP address.

NFS Coupler allows users to specify a host in the JCL subparameter only by its name. NFS Coupler uses its own mapping table to map host names into IP addresses.

Security

The NFS protocol provides a mechanism that allows protection across the network of files residing on the NFS server. This mechanism is based on the security scheme used by UNIX, which differs greatly from the security scheme used by MVS.

In UNIX, security is based on these three items:

Login The ASCII character string used to identify the user at logon.

Uid An integer that identifies a user. This number is always used internally by UNIX instead of the login ASCII string.

Gid An integer that identifies a group of Uids. Every login is associated to a Uid and a Gid by the password file whose full path name is `/etc/passwd`.

When a new UNIX file or directory is created, it is associated with a Uid and a Gid. Normally, but not always, the Uid and Gid of a UNIX file is the Uid and Gid of the creator process.

NFS Coupler Administration

Each file has separate security access rules for read, write, and execute, for three different groups of processes. These groups are:

- Processes whose Uid is the same as the file Uid.
- Processes whose Uid differs from the file Uid while their Gid is the same as the file Gid.
- Processes whose Uid and Gid differ from the file Uid and Gid.

When NFS Coupler issues a mount request, it provides to the NFS server a Uid and a Gid. When the NFS server is a UNIX machine, it uses the Uid and Gid for security checking whenever it performs an I/O operation. When the NFS server is run by a DOS-compatible operating system, such as Novell NetWare, Windows NT or OS/2, it emulates the UNIX security scheme, which varies from system to system.

Note: Refer to the operating system documentation for information about the degree of NFS security provided on the NFS server.

Because MVS does not have the concept of a Uid and Gid, NFS Coupler uses a mapping table to map the current MVS USERID to the corresponding Uid and Gid to be used by the NFS server.

MVS Administration

MVS administrators must provide and maintain correct routing and security information for NFS Coupler to work correctly. NFS Coupler needs the following information when a file is opened:

- The IP address of the UNIX host name specified in the JCL DD statement.
- The UNIX Uid and Gid corresponding to the current MVS USERID.
- The default mount point for the current MVS USERID on the UNIX host specified.

NFS Coupler reads the above information from module ICDNMAP, which is a mapping table.

ICDNMAP Generation

Module ICDNMAP is created by running a set of JCL statements provided in member ICDNMAP of the NFS Coupler JCL library.

The input to JOB ICDNMAP is specified via one or more DD statements. Each input file has a DD name in the form of `INPUTxxx`, where `xxx` is a three-digit number. JOB ICDNMAP reads data from the files `INPUT001`, `INPUT002`, `INPUT003`, and so on, until no more sequential DD names are found. For example, if DD names `INPUT001`, `INPUT002`, and `INPUT004` are specified, only `INPUT001` and `INPUT002` will be used because `INPUT003` is missing.

Each `INPUTxxx` file specifies routing, security and default mount directories for UNIX hosts that belong to the same *NFS domain*. In NFS Coupler, an NFS domain is a group of UNIX hosts mapping the same Logins to the same Uids and Gids. For example, if all UNIX hosts share the same password file within a given TCP/IP network, there will be only one NFS domain. However, if there are two LANS, and Logins on one LAN are mapped to Uids and Gids differently from the way they are mapped on the other LAN, there are two NFS domains, which in turn will translate into two input files to JOB ICDNMAP.

In general, UNIX hosts that access each other's files using NFS must belong to the same NFS domain. However, NFS Coupler can mount files from UNIX hosts belonging to different NFS domains, because of its flexibility in mapping Uids and Gids.

Each `INPUTxxx` input file contains three types of statements:

- The **HOST=** statement.
- The **USERID=** statement.
- The **NULL** statement.

Notes:

1. All types of statements can be freely intermixed in any order.
2. The character `';'` indicates that anything following is a comment and should be ignored.
3. A NULL statement starts with the `';'` character.

The HOST= Statement

The Host= Statement

```
                                          ┌─UNIX─┐
►►──HOST=─host_name─,─┬─ip_address────┬─,─┼──────┼──,─┬─────────┬─,──────────►
                      └─(─ip_list─)───┘   └─DOS──┘    └─timeout─┘

►─┬────────────┬─,─┬─────────┬─┬────────────────┬──────────────────────────►◄
  └─retransmit─┘   └─overlap─┘ └─;─┬──────────┬─┘
                                   └─comments─┘
```

Note: Commas are shown separating each parameter to the **HOST=** statement. Because all parameters are positional, commas must be used after a defaulted parameter, unless there are no more parameters following (see "Example" on page 23).

HOST=
This statement maps a **host_name** to an `ip_address` or an **ip_list**.

host_name
The host name as it is used in the JCL DD statement. It is automatically converted to upper case. It can be up to 16 characters long and its characters must belong to the following character set:

`123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@#$_?!.-`

ip_address
The IP address that identifies **host_name** within the TCP/IP network. It must be defined in dotted decimal notation (for example, 128.23.1.24).

ip_list A list of up to four IP addresses in dotted decimal notation, separated by commas. These IP addresses are used by **host_name** when it uses a multiple network interface. All IP addresses that are reachable from MVS must be listed. Those which are not reachable should not be listed. Failure to list all IP addresses used by **host_name** and reachable by MVS could result in timeout errors.

UNIX Indicates that the NFS server is a UNIX system. This is the default.

DOS Indicates that the NFS server is a DOS system.

timeout
A number in the form `t.t`, which specifies the number of seconds NFS Coupler will wait before retransmitting a request to **host_name**. If `timeout` is not specified, the default value is .5 (a half-second). This value should be increased when **host_name** is a particularly slow server or the network line connecting MVS to **host_name** is slow (for example, a telephone line).

retransmit
The number of times NFS Coupler will retransmit a request to `host_name` before issuing a timeout message and returning an error return code to the calling I/O MVS routines. If not specified, the default value is 40.

overlap
The number of overlapping read and write requests that NFS Coupler uses to read and write data from **host_name**. If not specified, the default value is 4. Before overriding the default value, refer to “Chapter 5. Performance” on page 31 .

comments
Any text that is needed to document this statement.

Note: A given **host_name** can be specified only in one input file because a host can belong to only one NFS Coupler domain. If a given **host_name** is coded in more than one input file, an error will result.

The USERID= Statement

```
►►──USERID=──%NODEF─────────────────────────────────────────────────────►◄
```

or

```
►►──USERID=─mvs_userid─,─mount_path─,─uid──,─gid─┬──────────┬───────────►
                                                 └─,─gid.1─┘

►─┬─────────┬─┬─────────┬─┬─────────┬───────────────────────────────────►
  └─,─gid.2─┘ └─,─gid.3─┘ └─,─gid.4─┘

►─┬─────────┬─┬─────────┬─┬─────────┬───────────────────────────────────►
  └─,─gid.5─┘ └─,─gid.6─┘ └─,─gid.7─┘

►─┬─────────┬─┬─────────┬─┬──────────┬─┬────────────────┬──────────────►◄
  └─,─gid.8─┘ └─,─gid.9─┘ └─,─gid.10─┘ └─;─┬──────────┬─┘
                                           └─comments─┘
```

%NODEF
No default ID is used.

USERID=
This statement maps an MVS USERID to UNIX security data.

mvs_userid
The actual MVS USERID that is mapped to UNIX security data. The value

of **mvs_userid** must be a valid seven-character MVS USERID, with the exception of **%DEFUID**, which is also valid.

USERID=%DEFUID
Can be used to specify the default UNIX security data for MVS USERIDs not defined in the current domain. If **USERID=%DEFUID** is not specified in a given domain, NFS Coupler uses the following statement as a default:

```
USERID=%DEFUID,/,65534,65534
```

mount_path
The full path name of the UNIX mount point to be used as the default when it is not explicitly specified as a JCL subparameter.

uid An integer that specifies the UNIX Uid to be used for security authorization.

gid An integer that specifies the UNIX Gid to be used for security authorization.

gid.1-gid.10
A set of integers that specify up to 10 additional UNIX Gids to be used for security authorization in addition to **gid**.

comments
Any data that is needed to document this statement.

Note: Data specified in the **USERID=** statement applies to all UNIX hosts specified in the same input file (same domain).

Example

Member ICDNMAP in the NFS Coupler JCL library contains the JCL statements to generate ICDNMAP. The content of member ICDNMAP is shown in Figure 6:

```
//ICDNMAP  JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*------------------------------------------------------------------
//*
//*  Run this JOB to generate ICDNMAP module for ICD NFS client
//*
//*------------------------------------------------------------------
```

Figure 6. JCL Statements to Generate ICDNMAP (Part 1 of 2)

```
//*
//MAP      EXEC PGM=IRXJCL,PARM='ICDNMAP'
//*
//STEPLIB  DD DISP=SHR,DSN=ICD.V1R1M0.SICDLOAD
//SYSEXEC  DD DISP=SHR,DSN=ICD.V1R1M0.SICDEXEC
//*
//SYSTSPRT DD SYSOUT=*
//*
//*-------------------------------------------------------------------
//* INPUT FILES TO ICDNMAP
//*-------------------------------------------------------------------
//*
//INPUT001 DD DISP=SHR,DSN=ICD.V1R1M0.SICDCNTL(ICDNIN01)
//INPUT002 DD DISP=SHR,DSN=ICD.V1R1M0.SICDCNTL(ICDNIN02)
//*
//OUTPUT   DD DISP=(NEW,PASS),DSN=&&DECK,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//*
//*-------------------------------------------------------------------
//* START ASSEMBLY AND LINK EDIT STEPS
//*-------------------------------------------------------------------
//*
//GEN EXEC SASMCL,COND=(0,NE,MAP),PARM.L='RENT'
//*
//C.SYSIN   DD DISP=(OLD,PASS),DSN=&&DECK
//L.SYSLMOD DD DISP=SHR,DSN=ICD.V1R1M0.SICDLINK(ICDNMAP)
```

Figure 6. JCL Statements to Generate ICDNMAP (Part 2 of 2)

Members ICDNIN01 and ICDNIN02 in the NFS Coupler JCL library contain the two input files for the above JCL statements, one for each NFS domain.

The first domain, defined in ICDNIN01, has one host and three MVS USERIDs. Each MVS USERID uses the same UNIX Uid and Gid for any of the hosts defined in ICDNIN01. The listing of member ICDNIN01 is shown in Figure 7:

```
;-------------------------------------------------------------------
;       LICENSED MATERIAL - PROPERTY OF IBM
;       5655-A99 (C) COPYRIGHT IBM 1998
;       (C) COPYRIGHT FOLIUM INC. 1994, 1998
;       ALL RIGHTS RESERVED
;       US GOVERNMENT USERS RESTRICTED RIGHTS -
;       USE, DUPLICATION OR DISCLOSURE RESTRICTED
;       BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
;---------------------------------------------------------------------------
;
;  Example of input ICDNMAP for first domain
;
;---------------------------------------------------------------------------
```

Figure 7. Example of Input ICDNMAP for First Domain (Part 1 of 2)

```
;
;--- Start input of UNIX hosts for this NFS domain
;
HOST=FOLIUM_UNIX,(223.1.1.2,224.1.1.2),UNIX,,,10 ;
;                                      UNIX Development
;                                      high overlap since
;                                      it is a fast server
;
;--- Start input of MVS USERIDs for this NFS domain
;
USERID=FLM01,/export,1001,1          ; UID 01 Development
USERID=FLM02,/export,1002,1          ; UID 02 Development
USERID=FLM03,/export,1003,1          ; UID 03 Development
;
;--- End of input
```

Figure 7. Example of Input ICDNMAP for First Domain (Part 2 of 2)

The second domain, defined in ICDNIN02, has two hosts and three MVS USERIDs. Note that MVS USERIDs `FLM01`, `FLM02`, and `FLM03` are specified in both inputs with different default mount paths, Uids, and Gids. The listing of member ICDNIN02 is in Figure 8:

```
;-----------------------------------------------------------------------
;       LICENSED MATERIAL - PROPERTY OF IBM
;       5655-A99 (C) COPYRIGHT IBM 1998
;       (C) COPYRIGHT FOLIUM INC. 1994, 1998
;       ALL RIGHTS RESERVED
;       US GOVERNMENT USERS RESTRICTED RIGHTS -
;       USE, DUPLICATION OR DISCLOSURE RESTRICTED
;       BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
;-------------------------------------------------------------------------------
;
;  Example of input ICDNMAP for second domain
;
;-------------------------------------------------------------------------------
;
;--- Start input of UNIX hosts for this NFS domain
;
HOST=FOLIUM_OS2,223.1.1.3,DOS,,,2    ; OS/2 Development
;                                      use low overlap
;                                      server not so fast (Intel 486)
;
HOST=FOLIUM_WNT,(223.1.1.5,224.1.1.5),DOS ;
;                                      Windows NT Development
;                                      use defaults
;                                      it is on the same ETHERNET
;                                      and faster than OS/2
;
;--- Start input of MVS USERIDs for this NFS domain
;
USERID=FLM01,G:\EXPORT,111,1         ; UID 01 Development
USERID=FLM02,G:\EXPORT,112,1         ; UID 02 Development
USERID=FLM03,G:\EXPORT,113,1         ; UID 03 Development
;
;--- End of input
```

Figure 8. Example of Input ICDNMAP for Second Domain

Where To Get Uids and Gids

UNIX Uids and Gids are stored for all UNIX users in file `/etc/passwd`, also known as the UNIX password file.

NFS Coupler Administration

Each UNIX user is defined as a line in the password file. Each line is composed of fields separated by colons. The first field contains the UNIX Login, which is the ASCII string used to logon to UNIX. The third and fourth fields contain the Uid and Gid associated with the Login. The fifth field contains the home directory for the Login.

For example, if a user has `TSOA0001` as MVS USERID, `johndoe` as UNIX Login, and if the entry in the UNIX password file for Login `johndoe` is:

```
johndoe:jEv234Jh0:10101:10:John Doe login:/users/johndoe:/bin/ksh
```

then the **USERID=** statement to be added to ICDNMAP input for the user is:

```
USERID=TSOA0001,/users/johndoe,10101,10      ; John Doe mapping
```

Note: In this example the default mount path has been set to the UNIX home directory. This should normally be the case unless a different default mount path is needed.

If the NFS server uses a DOS-compatible operating system such as NetWare, Windows NT, or OS/2, ask the NFS server administrator for Uids and Gids used by the NFS server to emulate UNIX security.

Administering NFS Servers

UNIX Administration

UNIX administration follows the standard NFS administration used to allow remote hosts to mount local directories.

The `/etc/exports` file contains an entry for each directory which can be mounted by one or more remote hosts. UNIX manual pages describe in detail how to code entries in the `/etc/exports` file. To review these man pages, at the UNIX prompt type:

```
man exports
```

While modifying `/etc/exports` to allow access from NFS Coupler on the MVS host, keep in mind the following:

- The MVS host name used in `/etc/exports` must be the same as the host name used by the TCP/IP subsystem on MVS. The MVS host name is printed by NFS Coupler whenever an error occurs, as part of the diagnostic messages in the JOB log.
- When MVS can mount a directory, it can also mount any subdirectory of the directory specified. This can be used to reduce the number of entries in `/etc/exports`. For example, if all MVS users need to mount several subdirectories of directory `/home`, one entry in `/etc/exports` for directory `/home` allowing mount from the MVS host satisfies the needs of all users.

Once file `/etc/exports` has been modified, UNIX command `exportfs` should be used to tell the mount daemon to read in the new `/etc/exports` configuration. At the UNIX prompt type:

```
man exportfs
```

to review the man pages for the **exportfs** UNIX command.

Administering DOS-Compatible Operating Systems

The procedures for administrating an NFS server run on a DOS-compatible OS, such as NetWare, OS/2, or Windows NT, vary depending on the specific operating system being used. However, many of the administration concepts are similar to administering an NFS server run on UNIX, particularly in regard to security (see “Security” on page 19).

NFS Coupler Fault Tolerance

Because NFS Coupler uses the connectionless NFS protocol, NFS Coupler can provide a significant level of fault tolerance when the NFS server being accessed has multiple network interfaces. Each network card is a separate network interface with an IP address.

Warning: Fault tolerance does not completely eliminate the possibility of failure; it only decreases the probability. If a mission critical, long running JOB needs to use NFS Coupler services, it should use the MVS Checkpoint/Restart facility (see “MVS Checkpoint/Restart” on page 38).

An NFS server with multiple network interfaces and an equivalent number of IP addresses can use any of the IP addresses to send its requests. Whenever NFS Coupler detects a request timeout it selects the next available NFS server IP address. Consequently, a partial failure of the TCP/IP network, such as a network card hardware failure on the NFS server, will not stop NFS Coupler from providing services. JOBs using NFS Coupler will be able to continue to run and complete successfully.

Note: NFS Coupler does not control the route chosen by the NFS server to send responses back to NFS Coupler. In some cases, this can impede NFS Coupler’s fault tolerance (see Fault Tolerance Examples).

Fault Tolerance Examples

Example 1: Two LANs Connecting MVS and the NFS Server

Figure 9 shows an MVS machine and an NFS server connected by two LANs: an Ethernet and a token ring. The MVS machine has IP address MVS_1 on the token ring and MVS_2 on the Ethernet. The NFS server has IP address UNIX_1 on the token ring and UNIX_2 on the Ethernet.

NFS Coupler begins sending requests to the NFS server using UNIX_1 as the destination IP address. The NFS server sends back the responses using MVS_1 as destination IP address, because MVS_1 is the origin IP address of the NFS Coupler requests. All the traffic is routed over the token ring.

If the token ring stops working, NFS Coupler detects a request timeout and starts using UNIX_2 as the destination IP address of its requests. The NFS server automatically starts using MVS_2 as the destination IP address for its responses because MVS_2 is now the origin IP address of the NFS Coupler requests. As a result, all the traffic is automatically rerouted over the Ethernet. Conversely, if the token ring starts working and the Ethernet stops working, NFS Coupler reroutes all traffic over the token ring.

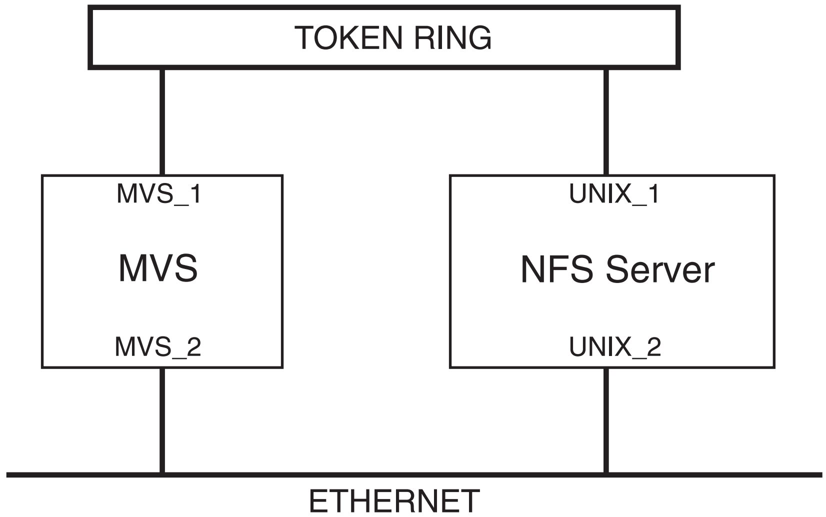

Figure 9. Two LANs Connecting an MVS Host and an NFS Server

In this scenario, the route chosen by the NFS server to send back its responses is fully controlled by the route NFS Coupler uses to send its requests. In this type of situation, NFS Coupler can provide optimum fault-tolerance capabilities.

Example 2: One LAN Connecting MVS and the NFS Server

Figure 10 shows an MVS machine and an NFS server connected by an Ethernet LAN. The MVS machine has IP address MVS_1. The NFS server has two network interfaces and therefore two IP addresses, UNIX_1 and UNIX_2.

NFS Coupler begins sending requests to the NFS server using UNIX_1 as the destination IP address. The NFS server sends back the responses using MVS_1 as destination IP address. However, the TCP/IP driver on the NFS server can choose either network interface (UNIX_1 or UNIX_2) to send the IP packets to MVS_1.

For example, if network interface UNIX_1 malfunctions while the NFS server is using it to send IP packets to MVS_1, NFS Coupler detects a request timeout and starts using UNIX_2 as the destination IP address for requests. At this point, the NFS server will do one of the following:

- It detects that the UNIX_1 interface is malfunctioning and starts using the UNIX_2 interface to send back IP packets to MVS_1.
- It is unable to detect any problem on its network interface, and it continues to use UNIX_1 to send back IP packets to MVS_1. In this case, NFS Coupler will timeout because it never receives responses to its requests, and any MVS JOB accessing files on the NFS server terminates abnormally.

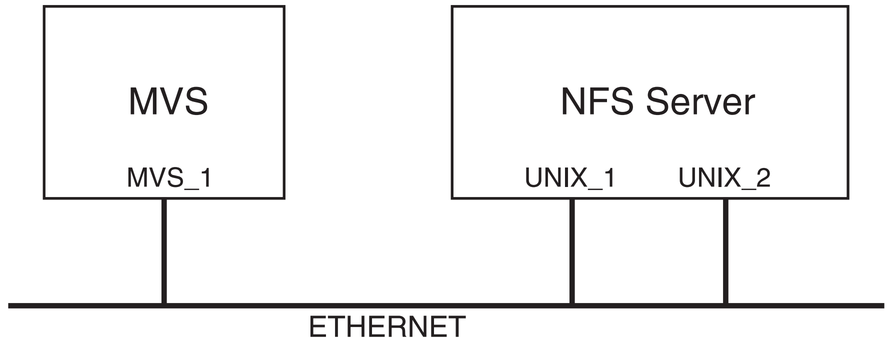

Figure 10. One Ethernet Lan Connecting an MVS Host and an NFS Server

In Figure 10, the MVS machine has only one IP address and its routing is static. In such configurations, NFS Coupler fault tolerance capabilities can be severely limited. If fault tolerance is a requirement, the NFS server manufacturer should be consulted to learn whether the server is capable of detecting problems with its network interfaces.

Example 3: MVS and the NFS Server on Different LANs

Figure 11 shows an MVS machine and an NFS server on different LANs. The MVS machine has one IP address, MVS_1. The NFS server has two network interfaces and two IP addresses, UNIX_1 and UNIX_2. All traffic between MVS and the NFS server must go through either gateway_1 or gateway_2.

NFS Coupler begins sending requests to the NFS server using UNIX_1 as the destination IP address. The NFS sends back the responses using MVS_1. The TCP/IP driver on the NFS server will use either the network interface UNIX_1 or UNIX_2 to send the IP packets to MVS_1, depending on the state of its routing tables.

If the network interface UNIX_1 starts malfunctioning, NFS Coupler detects a request timeout and starts using UNIX_2 as the destination IP address for its requests. At this time, the NFS server does one of the following:

- If routing is static, it keeps using the network interface that the routing tables specify to send IP packets to MVS_1. If that network interface is UNIX_1, NFS Coupler will timeout because it never receives responses to its requests, and any MVS JOB accessing files on the NFS server terminates abnormally.
- If routing is dynamic (for example, Routing Information Protocol), given a few seconds the routing tables will be updated so that UNIX_2 will become the designated network interface to be used to send IP packets to MVS_1. In this case, all MVS JOBs accessing files on the NFS server can continue processing and complete successfully, provided that the number of **retransmits**, specified for the NFS server in the ICDNMAP JOB input, is large enough to allow an update of the server routing tables before NFS Coupler runs out of **retransmits** and time outs.

In configurations in which gateways are between MVS and the NFS server, NFS Coupler's fault-tolerance capabilities are optimal providing that dynamic routing is used and that NFS Coupler retransmits long enough to allow time for the NFS server routing tables to be dynamically updated.

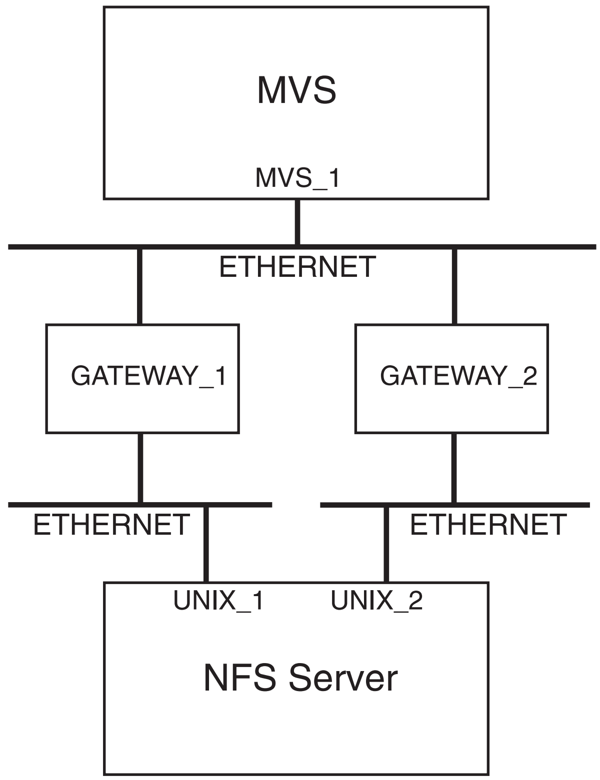

Figure 11. MVS Host and NFS Server on Different LANs

Chapter 5. Performance

NFS Coupler can achieve transfer rates equal or better than those obtained using FTP. Default settings of NFS Coupler timing parameters have been set so that for the most popular NFS servers this result will be achieved automatically. Should this not be the case, additional steps can be taken to significantly improve NFS Coupler performance.

The speed at which NFS Coupler can read or write data from an NFS server is affected by two primary factors:

- Network bandwidth.
- NFS server I/O throughput.

CPU and I/O shortages on MVS also can affect performance. These shortages should be handled the same way they are handled for any other MVS application.

Network Bandwidth

Insufficient network bandwidth can cause NFS Coupler to perform poorly. Improvements to network capacity can solve these performance problems. For instance, if the MVS machine and the NFS server are on different LANs, installing faster bridges or gateways might significantly improve NFS Coupler performance.

If the network connection between MVS and the NFS server is particularly slow (for example, a telephone line connection), the **timeout** parameter for the server should be set to more than 1 second and the **overlap** parameter should be set to 2 to avoid network congestion due to excessive number of retransmits. Both parameters are specified as input to JOB ICDNMAP (see “ICDNMAP Generation” on page 20).

NFS Server I/O Throughput: Overview

NFS server I/O throughput usually is the factor limiting how fast NFS Coupler can read or write data. The I/O throughput of an NFS server is affected by two factors:

- The hardware I/O bandwidth.
- The level of concurrency the NFS server uses to service NFS client requests.

Options for improving hardware I/O bandwidth usually are limited to hardware improvements, such as buying new disks, controllers, or a completely new server. Consequently, the primary way for server administrators to improve data transfer speed when using NFS Coupler is to control the level of concurrency the NFS server uses to service NFS client requests.

As a general rule, a higher level of concurrency results in a higher data transfer speed. The upper limit for data transfer speed improvements is controlled by the following parameters:

- The number of NFS daemons running on the server. Each daemon works synchronously on one client NFS request at a time. The daemon sends a response back to the client only after all I/Os are completed, and only then, will it begin servicing a new NFS client request. Accordingly, the maximum number of client NFS requests a server can concurrently perform is equal to the number of NFS daemons.

NFS Coupler Performance

- The number of concurrent read/write requests NFS Coupler sends to the server. This number is set by the **overlap** parameter, which is specified to NFS Coupler in the input to the JOB ICDNMAP (see “ICDNMAP Generation” on page 20).

NFS Server I/O Throughput: Tuning

If you are experiencing slow transfer speed, the first step to take is to set the **overlap** parameter equal to the number of NFS daemons running on the server. In most cases, this is sufficient to bring NFS Coupler performance to the desired level.

However, if an optimal level of performance is desired, the most favorable combination of values for the **overlap** parameter and the number of daemons on the NFS server can be reached with the following trial-and-error approach:

1. Set the **overlap** parameter and the number of daemons on the NFS server to 2.
2. Run a test JOB that transfers a significant amount of data (100 megabytes should be enough).
3. Increase the **overlap** parameter and the number of daemons by 2 and rerun the test JOB.
4. Check if performance has improved. If it has improved go to step3 and continue with the iteration. Otherwise go to 5.
5. Decrease the **overlap** parameter and the number of NFS daemons by 2. These are the optimal values.

The results obtained with the above procedure can also be used when multiple JOBs will concurrently access data on the same NFS server using NFS Coupler. For such cases, the optimal **overlap** parameter is equal to the **overlap** value previously computed, divided by the number of JOBs to be run concurrently, plus 1. In any case the **overlap** parameter should not be less than 2.

On UNIX machines the number of NFS daemons is controlled with a parameter to the **nfsd** command. For example:

```
nfsd 8 &
```

starts 8 NFS daemons. NFS daemons are normally started at boot time from a command in the boot script.

If the NFS server is run by a DOS compatible operating system such as NetWare, Windows NT, or OS/2, the NFS server administrator should be consulted to know the level of concurrency the server can achieve when servicing NFS client requests.

Chapter 6. Application Programmer's Guide

This chapter describes how application programmers can use NFS Coupler.

NFS Coupler services can be invoked via JCL DD statements or from REXX. Special NFS Coupler REXX functions have been provided to dynamically allocate and deallocate NFS Coupler files.

NFS Coupler JCL Reference

NFS Coupler allows access to UNIX files by coding the proper JCL DD statement parameters.

NFS Coupler services are invoked with the **SUBSYS=** JCL DD statement keyword followed by several subparameters. All subparameters for the **SUBSYS=** JCL keyword are positional.

The syntax of the **SUBSYS=** keyword subparameters for NFS Coupler is as follows:

SUBSYS=

```
►►──SUBSYS=─(─ssname─,─ICDNOPCL─,─host_name─,──────────────────────────────►

                                              ┌─ANY────┐
►─┬────────┬─,─┬────────────┬─,─file_name─,─┼────────┼─,──────────────────►
  └─userid─┘   └─mount_path─┘                ├─DIR────┤
                                              ├─FILE───┤
                                              └─REMOVE─┘

  ┌─CRFILE─┐                ┌─ASCII─┐
►─┼────────┼─,─┬──────┬─,─┼───────┼─)─────────────────────────────────────►◄
  ├─APPEND─┤   └─mode─┘    ├─BIN───┤
  ├─CRPATH─┤               └─NOPRT─┘
  └─NEW────┘
```

Notes:

1. Commas separate each **SUBSYS=** subparameter. Because all subparameters are positional, commas must be used after a defaulted subparameter, unless no more subparameters follow (see "NFS Coupler JCL Examples" on page 35).
2. In the diagram, spaces are shown around commas for clarity. When coding the DD statement, do not use spaces. If you use spaces, MVS will issue an error and the JOB will not run.

ssname
: Subsystem name used by NFS Coupler. It should be **IORR**, unless the subsystem has been installed with a different name.

ICDNOPCL
: Constant indicating the name of the NFS Coupler module that is invoked at open.

host_name
UNIX host name where the file to be read or written resides. The name must be in upper case and it must exist in the routing and security tables created by the MVS administrator.

userid
MVS USERID that is used instead of the current JOB userid. The MVS USERID is mapped to the UNIX Uid and Gid that is used to access the UNIX file. When **userid** is not specified, the current JOB userid will be used. The MVS USERID can be specified only when the current JOB USERID itself maps into superuser (i.e. Uid=0) for the current **host_name**. Specifying **userid** is equivalent to the UNIX command **su** (set user) when used by a process with Uid=0.

mount_path
Full path name to be used as NFS mount point. When not specified, the default mount path defined in the routing and security table for the current **host_name-userid** combination is used. Because UNIX paths are case-sensitive and can contain characters other than alphanumeric, **mount_path** should be enclosed in single quotes.

file_name
UNIX file name to be accessed. **File_name** is appended to the **mount_path** to form the full path name of the file being accessed. Because UNIX paths are case-sensitive and can contain characters other than alphanumeric, `file_name` should be enclosed in single quotes.

ANY
Keyword indicating that if the file is opened for input it can be either a UNIX directory or a UNIX file. **ANY** is the default input disposition.

DIR
Keyword indicating that if the file is opened for input it must be a UNIX directory, as opposed to a UNIX file.

FILE
Keyword indicating that if the file is opened for input it must be a UNIX file as opposed to a UNIX directory.

REMOVE
Keyword indicating that if the file is opened for input it must be a UNIX file and it will be removed at close time. **REMOVE** is bypassed if NFS Coupler detects that there is an ABEND in progress for the task executing the close routine at close time.

APPEND
Keyword indicating that if the file is opened for output, and the file already exists, writing of new data starts at the end of the existing file. If the file does not exist it will be created. The **mode** of the file being created is specified by the mode subparameter. The directory under which the file will be created must exist. If it does not exist, an OPEN error occurs.

CRFILE
Keyword indicating that if the file is opened for output, and it already exists, it will be overwritten. If the file does not exist, it is automatically created. The **mode** of the file being created is specified by the mode subparameter. The directory under which the file will be created must already exist. If it does not exist, an OPEN error occurs. CRFILE is the default output disposition.

CRPATH
Keyword indicating that if the file is opened for output, and if it already exists, the file is overwritten. If the file does not exist, and one or more directories in the full path name do not exist either, the file and the directories are created. The access mode of the directories is the same as

the file being created and is specified by the **mode** subparameter, with the exception that the directories always are marked executable by the owner Uid.

NEW Keyword indicating that if the file is opened for output, it must not exist. The file is created at OPEN time. If the file does exist, an OPEN error occurs.

mode Three-digit octal number that specifies which mode to use when creating a file and/or a directory. Each digit in **mode** indicates access authorization to the file or directory for the file Uid (first digit), the file Gid (second digit) and everybody else (third digit). Within each digit, the first bit (octal 4) indicates read authorization, the second bit (octal 2) indicates write authorization, and the third bit (octal 1) indicates execute authorization. For example a mode of `751` indicates that the file Uid has read, write, and execute authorization, the file Gid has read and execute authorization, while everybody else has execute authorization. The default value for **mode** is `640`, which specifies read and write authorization for the file Uid and read authorization for the file Gid.

ASCII Keyword indicating that the file (input or output) should be treated as a character file. Appropriate character translation and record boundary mapping takes place. **ASCII** is the default for data type.

BIN Keyword indicating that the file (input or output) should be treated as a binary file. Characters are not translated from EBCDIC to ASCII and vice versa.

NOPRT
Keyword indicating that the file (input or output) should be treated as a character file instead of a print file. If the DCB specifies an ANSI or MACHINE carriage control, print file conversion is bypassed. If the DCB does not specify an ANSI or MACHINE carriage control, **NOPRT** is equivalent to **ASCII**.

In general, when using NFS Coupler the only other JCL keywords that optionally can be used are **RECFM=**, **LRECL=**, and **BLKSIZE=** DCB's keywords. Any other JCL keyword that is consistent with the SUBSYS= keyword can be coded, but it will be ignored.

NFS Coupler JCL Examples

Example 1:

For **DD=INPUT**, specify an input ASCII file that resides on host SERVER_1. The file path is `data/phone.listing` under the default mount path. Records are no longer than 256 characters, and record format is variable.

```
//INPUT DD SUBSYS=(IORR,ICDNOPCL,'SERVER_1',,,'data/phone.listing'),
//         DCB=(RECFM=V,LRECL=256)
```

Example 2:

For **DD=SYSPRINT**, specify an output ASCII file that resides on OS/2 host SERVER_2. The full path name of the file is `D:\EXPORT\MVS\TABLES\DATA.MAP`. The mount path is specified as `G:\EXPORT\MVS`. The DCB is specified by the application at OPEN time. If the file does not exist, it is created. However, the directory `D:\EXPORT\MVS\TABLES` must already exist.

```
//SYSPRINT DD SUBSYS=(IORR,ICDNOPCL,'SERVER_2',,
//            'G:\EXPORT\MVS','TABLES\DATA.MAP',,CRFILE)
```

Example 3:

For **DD=DATAIN**, specify an input ASCII file that consists of the directory entries of the default mount directory on host UNIX_01. Record format is variable and record length is 316.

```
//DATAIN   DD SUBSYS=(IORR,ICDNOPCL,'UNIX_01',,,'.',DIR),
//            DCB=(RECFM=V,LRECL=316)
```

Example 4:

For **DD=DATAOUT**, specify an output binary file that contains data preformatted for a UNIX spreadsheet application. The file must be new and it must reside on host UNIX_02. The default mount path is used. The file path is mvs/sales.day/10.13.1994 File must be accessible to everybody in read-only mode. Record format is variable and record length is up to 4096.

```
//DATAOUT DD SUBSYS=(IORR,ICDNOPCL,'UNIX_02',,,
//           'mvs/sales.day/10.13.1994',,NEW,744,BIN),
//           DCB=(RECFM=V,LRECL=4096,BLKSIZE=4100)
```

Data Conversion

MVS and UNIX files use different formats and different characters set to store printable characters.

NFS Coupler converts MVS files into UNIX files and vice versa based on the following:

- The data type specified in the JCL subparameters to NFS Coupler. It can be either **ASCII** or **BIN**.
- The record format of the MVS file specified on the DCB.

Binary Output

When the data type specified is **BIN** and the file is opened for output, NFS Coupler writes each record into the UNIX file without character translation. Data is written to the UNIX file so that the first byte of each record immediately follows the last byte of the previous record. The first byte of the first record is written at file offset zero unless APPEND was specified as NFS Coupler output disposition.

ASCII Output (UNIX)

When the data type specified is **ASCII**, the file is opened for output, and the NFS server is of type UNIX, NFS Coupler writes each record into the file as follows:

- If the output file has Fixed Record Format, the record is first stripped of trailing blanks.
- Each byte in each record is translated from EBCDIC to ASCII.
- An ASCII Line Feed character (Hex X'0A') is added to the end of each record.
- Each record, thus converted, is written to the UNIX file so that its first byte follows the last byte of the previous record. The first byte of the first record is written at file offset zero unless **APPEND** was specified as NFS Coupler output disposition.

ASCII Output (DOS)

When the data type specified is **ASCII**, the file is opened for output, and the NFS server is of type DOS, NFS Coupler writes each record into the file as follows:

- If the output file has fixed record format, the record is first stripped of trailing blanks.
- Each byte in each record is translated from EBCDIC to ASCII.
- An ASCII Carriage Return character followed by an ASCII Line Feed character (Hex X'0D0A') is added to the end of each record.
- Each record, thus converted, is written to the UNIX file so that its first byte follows the last byte of the previous record. The first byte of the first record is written at file offset zero, unless **APPEND** was specified as NFS Coupler output disposition.
- Character X'1A' is added at the end of the file at close as an EOF marker.
- When APPEND is specified, data is added at the last byte of the existing output file and not after the last byte, on the assumption that the last byte itself in the existing output file is an EOF marker (X'1A').

Binary Input

When the data type specified is **BIN** and the file is opened for input, NFS Coupler sequentially reads a stream of bytes from the UNIX file.

For input files with fixed record format, each read record request is handled as follows:

- Data of length equal to the record length is copied from the UNIX file input stream to the user record. No ASCII-to-EBCDIC translation is performed.
- If EOF is encountered and there are not enough bytes to fill the record, the record is padded to the right with null characters (X'00').

For input files with variable or undefined record format, each read record request is handled as follows:

- Data is copied from the UNIX file input stream to the user record. No ASCII-to-EBCDIC translation is performed.
- The length of the data copied is the maximum allowed to fill the record, unless EOF is encountered.

ASCII Input (UNIX)

When the data type specified is **ASCII** and the file is opened for input, NFS Coupler sequentially reads a stream of bytes from the file.

Each read record request is handled as follows:

- Data in the input stream is scanned until an ASCII Carriage Return character followed by an ASCII Line Feed character (X'0D0A') is found as an EOL marker.
- Data up to, but not including, the EOL marker is translated from ASCII to EBCDIC and copied to the user record.
- For fixed record format files only, if the length of data copied is shorter than the record length, the record is padded to the right with blanks.
- If the data length of the data to be copied is larger than the record, an I/O truncation error will occur.

- The last character of the input file is checked to see if it is an EOF (X'1A'). If so, it is dropped.

Support of Print Files

When the record format of an NFS Coupler ASCII input or output file specifies an ANSI or MACHINE carriage control character (for example, record formats FA, FM, VA, VM, FBA, FBM, VBA, VBM), NFS Coupler performs an additional conversion so that the file being written or read is printed on UNIX the same way it is printed on MVS and vice versa.

For output files, NFS Coupler embeds the special characters line feed (X'0A'), carriage return (X'0D') and form feed (X'0C') to support the skip, overstrike, and new page carriage control character functions.

For input files, NFS Coupler detects the special characters line feed (X'0A'), carriage return (X'0D'), and form feed (X'0C') and substitutes them with the appropriate sequence of skip, new page, and overstrike carriage control characters. This allows NFS Coupler to properly read UNIX files formatted using the UNIX **pr** command.

Character Translation

NFS Coupler will translate characters from ASCII to EBCDIC and vice versa whenever the data type specified is **ASCII**.

NFS Coupler translates only printable characters. All nonprintable characters are left unchanged.

The translation rules are described below:

- ASCII characters from hex. ″21″x to hex. X'7E' are converted to their corresponding EBCDIC printable characters, with the exception of '∧' and '|'.
- Character '∧' is converted to EBCDIC ″logicalnot″ (X'5F').
- Character '|' is converted to EBCDIC ″bar″ (X'4F').
- EBCDIC printable characters are translated to ASCII using the exact inverse translation table.
- EBCDIC ″pipe″ (X'6A') is treated as a nonprintable character and left unchanged.

MVS Checkpoint/Restart

MVS Checkpoint/Restart has been provided by IBM to allow a quick recovery, after an abnormal termination, for long-running, mission-critical JOBs.

MVS Checkpoint/Restart allows a JOB to take a checkpoint periodically so that if an ABEND occurs at a later time, the JOB can be restarted from one of the checkpoints taken. When a JOB is restarted, MVS resets its memory and its file positioning to their exact state at the checkpoint specified for restart.

NFS Coupler fully supports MVS Checkpoint/Restart. No special action has to be taken to allow a JOB to use Checkpoint/Restart while accessing NFS server files using NFS Coupler. This support of Checkpoint/Restart is completely transparent.

When a checkpoint is taken, NFS Coupler performs the following steps:

1. Write all output buffers to disk on the NFS server.
2. Save the positioning and other information of the files being read or written.

At restart, NFS Coupler resets the positioning of its input and output files to the state it was in at the checkpoint specified for restart. It also checks whether the following conditions are true:

- Input or output file(s) have the same File System ID and File ID they had at the checkpoint specified for restart.
- Input file(s) have not been updated since the checkpoint specified for restart.
- Output files are not shorter than they were at the the checkpoint specified for restart.

If any of these conditions fail, NFS Coupler will ask the operator whether to continue the restart or terminate. If the operator responds to terminate, the JOB restart fails.

Note: An NFS Coupler input file might have to be updated before restart. For example, the file might contain a bad record, which will trigger an ABEND. In this case, the record should be updated or deleted, and the operator should respond to NFS Coupler to continue with the restart.

Warning:

1. Whenever a NFS Coupler input file is updated before restart, it is very important that no character, including blanks, are added or deleted preceding the record being updated or deleted. NFS Coupler keeps track of file positioning by byte count, not by record count. As a result, MVS Checkpoint/Restart can produce unexpected results if a character is added or deleted in a file location before the last character that was read at the time of the checkpoint specified for restart.
2. Some NFS servers, such as HP servers, allow asynchronous NFS access. Although this feature is a deviation from the NFS protocol, it can significantly improve write access. However, if MVS Checkpoint/Restart is used with an output NFS Coupler file to be written to such a server, data could be lost if the server crashes.

Using IEBGENER With Checkpoint/Restart

To take advantage of Checkpoint/Restart, you must have a JOB that uses its services. However, if you want to use NFS Coupler to copy a very large file (for example, a file containing several GigaBytes) from MVS to the NFS server (or vice versa), NFS Coupler provides module ICDNCHKP to be used as the IEBGENER totaling exit, which will take a checkpoint at constant time intervals.

To use IEBGENER with Checkpoint/Restart, use the JCL statements for IEBGENER, including the DD statement for the input or output of the NFS Coupler file, and make the following modifications:

- Add the checkpoint data set with DD name CHKPTDD. Ask the MVS system administrator which DASDs can be used for checkpoint data sets.
- Add two input lines to SYSIN exactly as follows:

```
GENERATE
EXITS TOTAL=(ICDNCHKP,6)
```

- Optionally, add the following two lines and use nnn to specify in minutes how often a checkpoint should be taken:

  ```
  //SYSPARM DD *
  INTERVAL=nnn
  ```

  where nnn is a number between 1 and 999. If no input is specified to SYSPARM, or its input is invalid, the default value of 10 minutes will be used.

Member ICDNCHKP in the NFS Coupler JCL distribution library contains sample JCL statements to enable the use of IEBGENER with Checkpoint/Restart. Member ICDNCHKP listing is shown in Figure 12:

```
//ICDNCHKP JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//* Before submitting this JOB do the following:
//*
//* 1. Change 'xxxxxx' to the UNIX host name where the output
//*    file will be written
//*
//* 2. Change 'yyyyyy' to the mount path if the default will
//*    not be used. Otherwise null it out.
//*
//* 3. Change 'zzzzzz' to the output file name.
//*
//* 4. Change 'chkpt_dsn' to the DSN name of the CHECKPOINT data set.
//*
//* 5. Change 'chkpt_vol' to the DASD volume to be used for the
//*    CHECKPOINT data set. Ask the MVS administrator which
//*    volumes are eligible to hold CHECKPOINT/RESTART data sets.
//*
```

Figure 12. JCL Statements to Enable the Use of IEBGENER with Checkpoint/Restart (Part 1 of 2)

```
//* 6. Change 'nnn' to number of minutes to be used as CHECKPOINT
//*    interval.
//*
//*-------------------------------------------------------------------
//*
//* Before submitting this JOB for RESTART do the following
//*
//* 1. Uncomment the RESTART= JOB parameter after changing 'cccccccc'
//*    to the checkpoint ID you what to restart with. Add this card
//*    to the JOB statement.
//*
//* 2. Uncomment the DD=SYSCHK statement which is located after the
//*    JOB cards and before the first EXEC statement.
//*
//*-------------------------------------------------------------------
//*
//*         RESTART=(*,cccccccc)
//*SYSCHK   DD DISP=OLD,DSN=chkpt_dsn
//*
//*-------------------------------------------------------------------
//* Start COPY step, CHECKPOINT data set is allocated with this step
//*-------------------------------------------------------------------
//*
//COPY     EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  GENERATE
  EXITS TOTAL=(ICDNCHKP,6)
//SYSPARM  DD *
INTERVAL=nnn
//*
//SYSUT1   DD  DISP=SHR,DSN=input_dsn
//SYSUT2   DD  SUBSYS=(IORR,ICDNOPCL,'xxxxxx',,
//             'yyyyyy','zzzzzz')
//*
//CHKPTDD  DD  DISP=(NEW,CATLG),DSN=chkpt_dsn,
//             VOL=SER=chkpt_vol,SPACE=(CYL,(20,20))
//*
//SYSUDUMP DD  SYSOUT=*
//*
//*-------------------------------------------------------------------
//* CHECKPOINT data set is deleted if IEBGENER completed normally
//*-------------------------------------------------------------------
//*
//DELETE   EXEC PGM=IEFBR14,COND=(4,GT)
//*
//CHKPTDD  DD  DISP=(OLD,DELETE),DSN=chkpt_dsn
```

Figure 12. JCL Statements to Enable the Use of IEBGENER with Checkpoint/Restart (Part 2 of 2)

Reading Content of NFS Server Directories

NFS Coupler allows applications to read the content of an NFS server directory by specifying the keyword **DIR** in the NFS Coupler JCL reference (see "NFS Coupler JCL Reference" on page 33). However, NFS Coupler does not allow applications to directly output data to an NFS server directory.

When an application reads the content of a NFS server directory, NFS Coupler feeds the application one record for each entry in the directory. After the record for the last entry has been read, an EOF condition is encountered.

Each record is formatted using printable characters. The format is similar to the output generated by the UNIX command:

```
ls -an
```

The format for each record is shown in Table 1:

Table 1. NFS Coupler Record Format for Reading NFS Server Directories

Position	Length	Justification	Description
1	10	Left	File mode.
12	5	Right	Number of links.
18	5	Right	File Uid.
24	5	Right	File Gid.
30	10	Right	File size in bytes.
41	10	Left	Date of last update in the form MM/DD/YYYY.
52	8	Left	Time of last update in the form HH:MM:SS.
61	Up to 256	Left	File name.

A UNIX directory can be read with either variable or fixed-format record type. However, in either case, the record length should be able to accommodate lengths of up to 316 characters.

Note: The file mode might not be meaningful or correct when the NFS server is a DOS-compatible network operating system such as NetWare, OS/2, or Windows NT. If you use a DOS-compatible network OS, check the OS documentation before relying on the file mode field.

Using NFS Coupler With REXX

REXX is the standard MVS command language. NFS Coupler can be used from REXX just as it can be used from any other language, provided that the proper DD names have been allocated to NFS Coupler. The standard REXX command EXECIO can be used to read or write NFS Coupler files sequentially (updating is not supported).

The NFS Coupler EXEC library includes the following REXX functions:

- ICDNDALLC and ICDNFREE are provided to dynamically allocate and deallocate NFS Coupler files. These functions can be used in any environment because they directly invoke SVC99 and do not rely on TSO services. See "ICDNDALLC" on page 43 and "ICDNFREE" on page 44.
- ICDNEMAP is provided to extract the Uid, Gid, and default mount path used by NFS Coupler for a given HOST-USERID combination. See "ICDNEMAP" on page 44 .

To use ICDNDALLC, ICDNEMAP, and ICDNFREE, the NFS Coupler REXX EXECs library must be in the SYSEXEC (or SYSPROC for TSO) concatenation.

Example 3. Reading UNIX Directory Entries Using REXX in the NFS Coupler REXX EXECs library shows how to use ICDNDALLC, ICDNEMAP and ICDNFREE.

ICDNDALLC

```
►►──ICDNDALLC─(─ddname─,─┬───────┬─,─┬───────┬─,─┬─────────┬─,──────────►
                         └─recfm─┘   └─lrecl─┘   └─blksize─┘

►─┬────────┬─,─host_name─,─┬────────┬─┬────────────┬─,─file_name─,──────►
  └─ssname─┘               └─userid─┘ └─mount_path─┘

  ┌─ANY────┐   ┌─CRFILE─┐             ┌─ASCII─┐
►─┼────────┼─,─┼────────┼─,─┬──────┬─,─┼───────┼─)─;──────────────────►◄
  ├─DIR────┤   ├─APPEND─┤   └─mode─┘   ├─BIN───┤
  ├─FILE───┤   ├─CRPATH─┤              └─NOPRT─┘
  └─REMOVE─┘   └─NEW────┘
```

Note: Commas separate each parameter to the **ICDNDALLC** REXX function. Because all parameters are positional, commas must be used after a defaulted parameter, unless no parameters follow (see "ICDNDALLC Example" on page 44).

ICDNDALLC
Allocates **ddname** to NFS Coupler and returns a string with two words separated by a blank space. The first word is the SVC99 error code and the second word is the SVC99 info code. If the error code is zero, the allocation was completed successfully. If the error code is non-zero, the info code gives a detailed reason for the failure. The most common info codes are:

Info Code (Decimal)	Description
1040	DD name is not available. It is currently allocated and it must be freed first.
1192	Subsystem does not support allocation. Ensure that the correct subsystem name was specified.
1196	Subsystem is not operational. Ensure that the correct subsystem name was specified.
1200	Subsystem does not exist. Ensure that the correct subsystem name was specified.

For additional information about Info Codes, please refer to the SVC99 section in *MVS/ESA Application Development Guide: Authorized Assembler Language Programs*.

ddname
DD name to be allocated to NFS Coupler.

lrecl Record length to be used for the file being allocated. When not specified, the default value 32756 is used.

recfm Record format to be used for the file being allocated. When not specified, the default value VB is used.

blksize
Block size to be used for the file being allocated. When not specified, the default value 32760 is used.

ssname
Subsystem name. When not specified, the default value **IORR** is used.

Note: All other ICDNDALLC arguments are the same as those used in the JCL DD statement. Please refer to "NFS Coupler JCL Reference" on page 33 for a complete description.

ICDNDALLC Example

Allocate a NFS Coupler file with DD name DATAOUT, on host UNIX_HOST, using default mount point, with file path **test.mvs** and with **APPEND** output disposition. Default DCB values will be used.

```
CALL ICDNDALLC "DATAOUT",,,,, "UNIX_HOST",,, "test.mvs",, "APPEND"
```

ICDNFREE

ICDNFREE

```
►►──ICDNFREE─(─ddname─)─;──────────────────────────────────────────►◄
```

ICDNFREE
Deallocates **ddname** and returns a string with two words separated by a blank space. The first word is the SVC99 error code and the second word is the SVC99 info code. If the error code is zero the deallocation was completed successfully. If the error code is non zero the info code gives a detailed reason for the failure. The most common info codes are:

Info Code (Decimal)	Description
1056	DD name is still open. File must be closed before being deallocated.
1080	DD name not found. The DD name specified has not been previously allocated.

For additional information about info codes, please refer to the SVC99 section in *MVS/ESA Application Development Guide: Authorized Assembler Language Programs*.

ddname
DD name to be deallocated.

Example:

Deallocate a NFS Coupler file with DD name DATAOUT.

```
CALL ICDNFREE "DATAOUT"
```

ICDNEMAP

```
►►──INTERPRET──ICDNEMAP──(─vroot──,──host_name──┬──────────┬──)─;──────────►◄
                                                └─,─userid─┘
```

ICDNEMAP
Extracts the NFS Coupler mapping information and returns a REXX executable string composed of REXX assignment statements. The string must be executed with the INTERPRET REXX statement. Variables used for assignment are all prefixed by the string **vroot**. Table 2 lists the variables assigned by the string returned by ICDNEMAP:

Table 2. Variables Assigned by the String Returned by ICDNEMAP

Variable	Description
`vroot_rc`	Return code
`vroot_msg`	Feed back message
`vroot_userid`	MVS USERID for which the data is being extracted
`vroot_ip_addr.0`	Number of IP addresses by this host
`vroot_ip_addr.1 - vroot_ip_addr.4`	IP address(es) used by this host
`vroot_rpc_timeout`	Timeout used for RPC calls
`vroot_rpc_retransmit`	Number of retransmits used for RPC calls
`vroot_rpc_overlap`	Number of overlapping RCP calls use to read and write
`vroot_uid`	UNIX Uid
`vroot_gid`	UNIX Gid
`vroot_gid.0`	Number of additional UNIX Gids
`vroot_gid.1 - vroot_gid.10`	Additional UNIX Gids, from 1 to 10

vroot String used to create the variable names to which extracted values are assigned. **vroot** is used as a prefix and it must be a valid REXX variable name. If **vroot** ends with a `'.'` character it indicates a stem.

host_name
Name of the UNIX host for which the extraction is done.

userid MVS USERID for which the extraction is done. If not specified, the caller's USERID is used as default.

ICDNEMAP Example

Extract NFS Coupler mapping information for the caller's MVS USERID for host UNIX_HOST and dump the content.

```
/* REXX */
  INTERPRET IcdnnEmap("map", "UNIX_HOST")

  say "*** ICDNEMAP called"
  say "    Rc........................ "map_rc
  say "    Msg....................... "map_msg
  say "    MVS USERID................ "map_userid
  say "    Number of IP addresses.... "map_ip_addr.0
  DO i= 1 TO map_ip_addr.0
    say "    IP Addr ("right(i, 2)").................. "map_ip_addr.i
    END
  say "    RPC Timeout............... "map_rpc_timeout
  say "    RPC Np. of retransmits.... "map_rpc_retransmit
  say "    RPC No. of overlaps....... "map_rpc_ovelap
  say "    Default mount............. "map_mount
  say "    Uid....................... "map_uid
  say "    Gid....................... "map_gid
    say "    Number of add. Gids....... "map_gid.0
  DO i= 1 TO map_gid.0
    say "  Gid ("right(i, 2)")...................... "map_gid.i
    END
  say "    End of NFS Coupler map extract dump"
  EXIT
```

Examples

NFS Coupler is distributed with three examples.

Example 1. Copying an MVS File to UNIX Using IEBGENER

Member ICDNGENR in the NFS Coupler JCL library contains the first example.

In this example, a JOB uses **IEBGENER** to copy an MVS file to a UNIX host. The file being copied is an in-stream input to DD name SYSUT1. Output DD name SYSUT2 is allocated to NFS Coupler.

Member ICDNGENR listing is shown in Figure 13:

```
//ICDNGENR JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//* Before submitting this JOB do the following:
//*
//* 1. Change 'xxxxxx' to the UNIX host name where the output
//*    file will be written
//*
//* 2. Change 'yyyyyy' to the mount path if the default will
//*    not be used. Otherwise null it out.
//*
//* 3. Change 'zzzzzz' to the output file name
//*
//*-------------------------------------------------------------------
```

Figure 13. JOB Using IEBGENER to Copy an MVS File to a UNIX Host (Part 1 of 2)

```
//*
//IEBGENER EXEC PGM=IEBGENER
//*
//SYSUT2   DD  SUBSYS=(IORR,ICDNOPCL,'xxxxxx',,
//             'yyyyyy','zzzzzz')
//*
//SYSUT1   DD  *
   This is a test ..........................
   This is a test ..........................
   This is a test ..........................
   This is a test ..........................
   This is a test ..........................
   This is a test ..........................
//*
//SYSIN    DD  DUMMY
//SYSPRINT DD  SYSOUT=*
```

Figure 13. JOB Using IEBGENER to Copy an MVS File to a UNIX Host (Part 2 of 2)

Example 2. Reading a UNIX File Using IDCAMS REPRO

Member ICDNREPR in the NFS Coupler JCL library contains the second example.

In this example a JOB lists the content of a UNIX file using the IDCAMS REPRO command. Input DD name DATAIN is allocated to NFS Coupler.

Member ICDNREPR listing is shown in Figure 14:

```
//ICDNREPR JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//* Before submitting this JOB do the following:
//*
//* 1. Change 'xxxxxx' to the UNIX host name where the output
//*    file will be written
```

Figure 14. REPRO Command Reading a UNIX File (Part 1 of 2)

```
//*
//* 2. Change 'yyyyyy' to the mount path if the default will
//*    not be used. Otherwise null it out.
//*
//* 3. Change 'zzzzzz' to the input file name
//*
//*-------------------------------------------------------------------
//*
//REPRO    EXEC PGM=IDCAMS
//*
//DATAIN   DD  SUBSYS=(IORR,ICDNOPCL,'xxxxxx',,
//             'yyyyyy','zzzzzz'),
//         DCB=(RECFM=VB,LRECL=256,BLKSIZE=1024)
//DATAOUT  DD  SYSOUT=*,
//         DCB=(RECFM=VB,LRECL=256,BLKSIZE=1024)
//*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  REPRO  INFILE(DATAIN) OUTFILE(DATAOUT)
```

Figure 14. REPRO Command Reading a UNIX File (Part 2 of 2)

Example 3. Reading UNIX Directory Entries Using REXX

Member ICDNFTRE in the NFS Coupler REXX EXECs library contains the third example.

In this example, a REXX EXEC lists all the files under a given UNIX directory and all its subdirectories.

This EXEC uses the REXX functions **ICDNDALLC** and `ICDNFREE` to dynamically allocate and free NFS Coupler files. It also uses the REXX function **NFSDGMAP** to acquire the default mount point and the UNIX Uid and Gid used to access the files.

EXEC ICDNFTRE can be invoked in batch or under TSO. When ICDNFTRE is invoked from TSO, the target UNIX host and the full path directory name must be passed as parameters. For example, to list all files under `/etc` and all its subdirectories type:

```
EX 'ICD.VIRIM0.SICDEXEC(ICDNFTRE)' 'host_name /etc'
```

where ’xxxxxx’ is the high-level node under which NFS Coupler was installed.

Member ICDNFTRE in the NFS Coupler JCL library can be used to invoke EXEC **ICDNFTRE** in batch. The member ICDNFTRE listing is shown in Figure 15:

```
//ICDNFTRE JOB   (),
//        CLASS=A,
//        MSGCLASS=D,
//        MSGLEVEL=(1,1),
//        NOTIFY=&SYSUID,
//        REGION=1M,
//        TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//* Before submitting this JOB do the following:
//*
//* 1. Change 'yyyyyy' to the UNIX host name whose files you want
//*    to list
//*
//* 2. Change 'zzzzzz' to the full path name of the directory
//*    containing the files you want to list
//*
//*-------------------------------------------------------------------
//*
//LIST     EXEC PGM=IRXJCL,PARM='ICDNFTRE yyyyyy zzzzzz'
//*
//SYSEXEC  DD  DISP=SHR,DSN=ICD.V1R1M0.SICDEXEC
//SYSTSPRT DD  SYSOUT=*
```

Figure 15. Batch ICDNFTRE Used to Invoke EXEC ICDNFTRE

Automated Operations

This section discusses how NFS Coupler can be used to run applications during the batch window in a fully automated environment.

NFS Coupler and Automated Operations

NFS Coupler is particularly useful for automating batch JOBs that need access to UNIX files. NFS Coupler can enable such JOBs to be scheduled by a production control subsystem like any other batch JOB. For example, NFS Coupler can enable a batch JOB to:

- Read data from UNIX and load it to a centralized data base (for example, IMS/DB, DB2, VSAM)
- Do some processing on the data base
- Extract some data from the data base and write it back to a UNIX file

NFS Coupler will allow processes to be scheduled to run on MVS at the most convenient time. This is a significant advantage at sites where the duration of the MVS batch window is becoming smaller.

Synchronizing MVS and UNIX with NFS Coupler

There are two kinds of synchronization between UNIX and MVS:

- An MVS JOB must run after a UNIX process has written into one or more UNIX files the data needed by the MVS JOB.
- A UNIX process must run only after an MVS JOB has written into one or more UNIX files the data needed by the UNIX process.

For example, consider a business that stores data on DB2 and that duplicates part of the data on a UNIX SQL database that is used by a branch office. Each day, updates on the UNIX SQL database must be reflected on the DB2 data base and vice versa. This can be achieved as follows:

1. A UNIX process downloads the UNIX SQL data base into UNIX files.
2. An MVS JOB reads the downloaded UNIX SQL data base and updates the DB2 data base.
3. An MVS JOB extracts from DB2 the data to be used, on the following day, by the UNIX SQL data base, and writes it into UNIX files.
4. A UNIX process loads data from the UNIX files into the UNIX SQL data base.

The MVS JOB in step 2 should not start before the UNIX process in step 1 ends successfully. The UNIX process in step 4 should not start before the MVS JOB in step 3 ends successfully. In either case, correct synchronization between UNIX processes and MVS JOBs can be achieved by using event files.

Event files are short files that are created at the end of processing, after all output data have been written out. An event file contains the following information:

- A time stamp to indicate when the processing completed.
- Whether the processing completed successfully (a return code and/or a message).

The above example can be implemented using event files as follows:

1. A UNIX shell calls a process to download the UNIX SQL data base into UNIX files. Before terminating, the UNIX shell writes into an event file a time stamp and the return code of the process just called.
2. The production control on MVS runs a REXX EXEC every 10 minutes. This EXEC tries to read the event file written in step 1 using NFS Coupler. If the event file is not found, or the time stamp is old, nothing happens. If the return code is bad, the operator is notified. If the return code is good batch processing on MVS is started.
3. An MVS JOB reads the downloaded UNIX SQL data base using NFS Coupler and updates the DB2 data base.
4. An MVS JOB extracts from DB2 the data to be used, on the following day, by the UNIX SQL data base, and writes the data into UNIX files using NFS Coupler.
5. Production control runs a REXX EXEC. The EXEC writes into a new event file a time stamp and the return code of step 4 using NFS Coupler.
6. A UNIX shell tries, in a loop of 10–minute intervals, to read the event file written in step 5. If the event file does not exist or the time stamp is old nothing happens. If the return code is bad, the UNIX operator will be notified. If the return code is good, processing on UNIX restarts.
7. A UNIX process loads data from the UNIX files into the UNIX SQL data base.

Note that no recoding of existing UNIX or MVS applications is required. Only two short REXX EXECs (steps 3 and 5) and two short UNIX shells (steps 2 and 6) need to be coded.

Part 3. Using RSH Coupler

Chapter 7. Administration

This chapter explains requirements and tasks related to RSH Coupler administration. The chapter is divided into the following sections:

- "RSH Coupler Internal Organization" explains RSH Coupler's two major components: the gateway address space and the subsystem I/O interface.
- Security Administration explains how to code access rules necessary to enable PC and UNIX clients to execute RSH Coupler APPC transactions.
- "APPC Administration" on page 58 explains how to code DD statements to remap APPC transaction input and output to standard input, error, and output of the `rsh` command on the client machine.
- "Data Conversion" on page 64 describes data conversion and record mapping for all types of RSH Coupler files. Data conversion and record mapping are required because MVS and UNIX files use different formats and different character sets to store printable characters.
- "Sharing Input and Output" on page 66 explains how input and output RSH Coupler files within the same RSH Coupler APPC transaction are shared. The same type of RSH Coupler file can be opened multiple times, concurrently or otherwise, using the same DD name or different DD names.

RSH Coupler Internal Organization

RSH Coupler has two major components

- The gateway address space, which runs as a started task.
- The subsystem I/O interface, which is used to intercept I/O operations of APPC transactions being executed via RSH Coupler.

RSH Coupler Gateway Address Space

The RSH Coupler gateway address space must be always up and running. It performs the following functions:

- Accepts connection requests from the **rsh** clients.
- Checks **rsh** client's security authorization.
- Starts execution of the APPC transaction as requested by the `rsh` client.
- Coordinates data flow between the **rsh** clients (using TCP/IP) and their respective APPC transactions.

Figure 16 shows the data flow between a `rsh` PC or UNIX client on one side, and the RSH Coupler gateway and the APPC transaction on the other side.

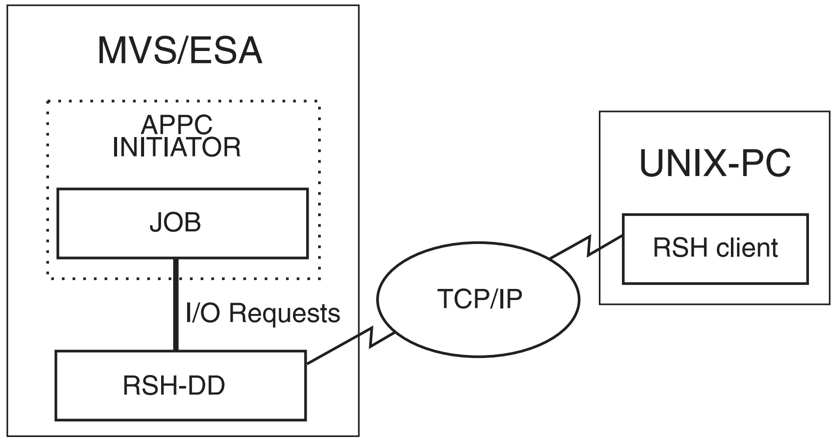

Figure 16. Data Flow Between rsh Client and APPC Transaction

A RSH Coupler gateway address space is required for every MVS copy where RSH Coupler JOBs will be executed. This is true even if it is technically possible for the RSH Coupler gateway to start an APPC transaction on a different MVS copy. The most important reasons for this restriction are:

- Performance considerations.
- The need for the RSH Coupler gateway address space and the APPC transactions to share the same security environment.

RSH Coupler I/O Interface

The RSH Coupler I/O interfaces effectively allow MVS program to read and write to an **rsh** command executing on the client side via the MVS I/O routines. Files opened by the APPC transaction for this purpose are referred in this manual as RSH Coupler files.

Security Administration

To enable PC and UNIX clients to execute RSH Coupler APPC transactions, access rules must be maintained in a partitioned data set called RHOSTS PDS.

Each member in the RHOSTS PDS is equivalent to the UNIX file `.rhosts`, which can be created under the **home** directory for each UNIX login.

UNIX file **.rhosts** contains the access rules for any client that tries to execute a command on a UNIX server using the login associated with the **home** directory.

Each member of the RHOSTS PDS contains the access rules for any client that tries to execute a RSH Coupler APPC transaction on MVS using an MVS USERID equal to the member name itself.

Access rules defined in the RHOSTS PDS are based on the following criteria:

- Host (UNIX or PC) from which the connection was established.
- Username under which the user logged on to the client host (used only for UNIX clients).
- MVS USERID specified by the client.

These are the same criteria used by the standard **rsh** server on UNIX machines. However, the syntax used is slightly different from that used on UNIX.

Coding Access Rules

Each line in a RHOSTS PDS member defines an access rule. Only characters in columns 1-72 are used. Any character after column 72 is ignored.

The syntax used is:

Coding Access Rules

```
►►──client_ip_address──┬────────────────┬──┬─────────────────┬──────────►◄
                       └─user_name_list─┘  └─;─┬──────────┬──┘
                                               └─comments─┘
```

client_ip_address
: Specifies the client IP address in dotted decimal notation. The security rule being defined in this line applies only to this IP address. The IP address can contain the character '*' as a wild card. For example, `129.252.42.*` means any host in the `129.252.42` subnetwork. The IP address `*.*.*.*` means anybody in the TCP/IP network.

user_name_list
: Is a list of client usernames, separated by blanks, that are authorized to use the current MVS USERID (RHOSTS PDS member name) from any of the hosts specified by the **client_ip_address**. If no user name is specified, any username will be able to use the current MVS USERID (RHOSTS PDS member name) from the host(s) specified by the **client_ip_address**.

comments
: Any text that is needed to document this statement.

The character ';' indicates the end of the line. Anything after it is considered a comment. A line starting with the character ';' is a valid line that defines a null security rule.

A sample RHOSTS PDS is distributed with RSH Coupler. The listing for RHOST member `FLM01` is shown in Figure 17:

```
;
;  .rhost input for USERID=FLM01
;
223.1.1.1  flm01                 ; 1 user from MVS (loopback)
;
223.1.1.2  paolo susy            ; 2 users from UNIX devel machine
223.1.1.4  paolo mark            ; 2 users from UNIX devel machine
223.1.1.3                        ; Anybody from the OS2 machine
```

Figure 17. RHOST Member

Notes:

1. The first line specifies that only username `flm01` can use the MVS USERID FLM01 when invoking an RSH Coupler APPC transaction with the `rsh` command from the host with the IP address 223.1.1.1.

Note: The IP address on the first line is the MVS IP address. This allows testing of the RSH Coupler by using the **rsh** command on MVS itself (loopback mode).

2. The second line specifies that only usernames `paola` and `susy` can use the MVS USERID FLM01 when invoking an RSH Coupler APPC transaction with the **rsh** command from the host with the IP address 223.1.1.2.
3. The third line specifies that only usernames `paola` and `mark` can use the MVS USERID FLM01 when invoking an RSH Coupler APPC transaction with the **rsh** command from the host with the IP address 223.1.1.4.
4. The fourth line specifies that any username can use the MVS USERID FLM01 when invoking an RSH Coupler APPC transaction with the **rsh** command from the host with the IP address 223.1.1.3. This is the address of an OS/2 workstation that has no security system installed but is in a secure location to which access is restricted.

ICDRMAP gen

For performance reason, the RSH Coupler gateway does not directly access the RHOSTS PDS. Instead a module named ICDRMAP must be created with input from the RHOSTS PDS.

Warning: If you are using the file `ICD.V1R1M0.SICDRHSD` that is delivered with IBM Network Data Couplers as the RHOST PDS, you ***must*** first delete all its existing members because those members will not correspond to MVS USERIDS for your installation.

Member ICDRMAP in the RSH Coupler JCL library contains the JCL to gen ICDRMAP. The content of member ICDRMAP is shown in Figure 18:

```
//ICDRMAP  JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//*  Run this JOB to generate ICDRMAP module for Data Coupler
//*    RSH server
//*
```

Figure 18. JCL to Create ICDRMAP (Part 1 of 2)

```
//*-------------------------------------------------------------------
//* START EXTRACT STEP
//*-------------------------------------------------------------------
//*
//EXTRACT  EXEC PGM=IEBPTPCH
//*
//SYSIN    DD *
    PUNCH TYPORG=PO,CDSEQ=0,MAXFLDS=1
    RECORD FIELD=(72)
//SYSPRINT DD SYSOUT=*
//*
//SYSUT1   DD DISP=SHR,DSN=ICD.V1R1M0.SICDCRHST
//*
//SYSUT2   DD DISP=(NEW,PASS),DSN=&&RHOSTS,
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//*
//*******************************************************************
//* PRINT &&RHOSTS FOR DEBUG                                        *
//*******************************************************************
//*
//ECHO     EXEC PGM=IEBGENER,COND=(0,NE,EXTRACT)
//*
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=*
//*
//SYSUT1   DD DISP=(OLD,PASS),DSN=&&RHOSTS
//SYSUT2   DD SYSOUT=*
//*
//*******************************************************************
//* START MAP STEP                                                  *
//*******************************************************************
//*
//MAP      EXEC PGM=IRXJCL,PARM='ICDRMAP',COND=(0,NE,EXTRACT)
//*
//SYSEXEC  DD DISP=SHR,DSN=ICD.V1R1M0.SICDEXEC
//*
//SYSTSPRT DD SYSOUT=*
//*
//RHOSTS   DD DISP=(OLD,PASS),DSN=&&RHOSTS
//OUTPUT   DD DISP=(NEW,PASS),DSN=&&DECK,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//            UNIT=SYSDA,SPACE=(CYL,(10,10))
//*
//*******************************************************************
//* START ASSEMBLY AND LINK EDIT STEPS                               *
//*******************************************************************
//*
//GEN EXEC SASMCL,COND=(0,NE,MAP),PARM.L='RENT'
//*
//C.SYSIN   DD DISP=(OLD,PASS),DSN=&&DECK
//L.SYSLMOD DD DISP=SHR,DSN=ICD.V1R1M0.SICDLOAD(ICDRMAP)
```

Figure 18. JCL to Create ICDRMAP (Part 2 of 2)

After the ICDRMAP JOB runs, the RSH Coupler gateway does not need to be stopped and restarted. To notify the RSH Coupler gateway that a new version of module ICDRMAP is available, the operator needs to enter the **REFresh** command (see “Chapter 3. Operations” on page 15).

APPC Administration

This section explains how to code DD statements to remap APPC transaction input and output to standard input, error, and output of the **rsh** command on the client machine. These DD statements must be coded in the JCL section of the input to the APPC TPADD function.

When the RSH Coupler gateway receives a new connection, it first checks security access as described in “Security Administration” on page 54. If the **rsh** client is authorized, the RSH Coupler gateway starts an APPC transaction.

The name of the APPC transaction is formed by concatenating the prefix, as specified in the started task PROC ICDRDD, and the command entered by the user on the **rsh** client.

For example, if the user on the **rsh** client wants to try Example 1 (see “Example 1. Using RSH Coupler for a Remote TSO Session” on page 76) and types:

```
rsh mvs_host_name -l mvs_userid rtso
```

and the APPC transaction prefix is **RSH_**, the resulting APPC transaction name is **RSH_rtso**. APPC transaction names can be up to 64 characters long and are case sensitive.

An APPC transaction must be defined using standard APPC administration tools before it may run. APPC provides both a batch utility and a on-line panel interface to define a new transaction. In batch, the TPADD verb must be used.

The APPC TPADD function requires input for two sections:

- The environment section, which defines the transaction name, the scheduler type (must be standard for RSH Coupler), the transaction class, and accounting tailoring.
- The JCL section, which defines the JCL statements to be used by the scheduler to run the transaction under one of its initiators.

To code the entire JOB to create a RSH Coupler APPC transaction, refer to *MVS/ESA Planning: APPC Management*. Also see the examples distributed with RSH Coupler (see “RSH Coupler Examples” on page 76). These examples can be easily modified to create new RSH Coupler APPC transactions.

JCL Reference

The JCL statements used by an APPC transaction executed as a RSH Coupler command are the same JCL statements used by the corresponding batch JOB, except for the DD statements containing data routed by RSH Coupler to and from the `rsh` client.

There are five types of DD statements:

DD=@ICDRDD@
: Required for every non MULTI-TRAN transaction defined to APPC and it is used by RSH Coupler for internal book-keeping.

STDPARM
Used for input files whose input data is composed of one record. The record itself contains the parameter string specified by the user to the `rsh` command on the client machine.

STDIN Used for input files whose data comes from the **rsh** client.

STDERR
It is used for output files whose data is sent to the **rsh** client as standard error.

STDOUT
used for output files whose data is sent to the **rsh** client as standard output.

In all cases, the DD statement contains the **SUBSYS=** keyword followed, optionally, by the DCB. Any other valid JCL parameter that are specified will be ignored.

DD=@ICDRDD@

```
►►──//DD=@ICDRDD@──DD──SUBSYS=(IORR,ICDRRMGR)──────────────────────────────►◄
```

Each transaction that does not use MULTI-TRAN RSH Coupler APPC must contain the above DD statement coded exactly as shown.

Note: If the IORR subsystem has been installed under a different name, then that name must be used in place of **IORR** in the above DD statement.

STDPARM

```
                                                              ┌─MIXED─┐
►►──SUBSYS─=──(──ssname─,─ICDROPCL─,──STDPARM─,─┬────────┬─,─┼───────┼──)─────►◄
                                                └─prefix─┘   └─UPPER─┘
```

Notes:

1. In the diagram above, spaces are shown around commas for clarity. Do not use spaces when you code the DD statement. If you use spaces, MVS will issue an error and the RSH Coupler APPC transaction will not run.
2. Each **SUBSYS=** subparameter is shown separated by a comma. Because all subparameters are positional, commas must be used after a defaulted subparameter, unless no more subparameters follow (see examples).

ssname
Subsystem name used by RSH Coupler. It should be **IORR**, unless the subsystem has been installed with a different name.

ICDROPCL
Constant indicating the name of the RSH Coupler module that is invoked at open.

STDPARM
Constant indicating that this is a one-record input file. The record contains the parameters string typed by the **rsh** client when invoking this RSH Coupler transaction.

prefix Character string up to 16 characters long. This string, when specified, is prefixed to the parameter string to form the input record.

MIXED
: Parameter string specified by the client is case-sensitive. This is the default.

UPPER
: Parameter string is converted to upper case.

Example:

Specify that input to a PL/1 program is the parameter string of the `rsh` command used on the client side. The DD name is SYSIN. The input must be converted to upper case and the first eight characters must be blanks.

```
//SYSIN DD SUBSYS=(IORR,ICDROPCL,STDPARM,'        ',UPPER)
```

STDIN

```
►►──SUBSYS─=─(─ssname─,─ICDROPCL─,─STDIN─,──────────────────────────────►

                    ┌─ASCII─┐
►─┬───────────┬─,─┼───────┼─,─┬─────────┬─)──────────────────────────►◄
  └─eof_token─┘     ├─BIN───┤     └─timeout─┘
                    └─NOPRT─┘
```

Notes:

1. In the diagram above, spaces are shown around commas for clarity. Do not use spaces when you code the DD statement. If you use spaces, MVS will issue an error and the RSH Coupler APPC transaction will not run.
2. Each **SUBSYS=** subparameter is shown separated by a comma. Because all subparameters are positional, commas must be used after a defaulted subparameter, unless no subparameters follow (see “STDIN Examples” on page 61).

ssname
: Subsystem name used by RSH Coupler. It should be **IORR**, unless the subsystem has been installed with a different name.

ICDROPCL
: Constant indicating the name of the RSH Coupler module that will be invoked at open.

STDIN
: Constant indicating that this is an input file whose data comes from standard input of the **rsh** client.

eof_token
: Character string up to 16 characters long. This string is compared with each input record. If an input record starts with the same string as the **eof_token**, the EOF condition will be returned to the caller of the input request. This EOF triggers a data sensitive EOF condition.

ASCII
: ASCII to EBCDIC translation takes place. This is the default.

BIN
: Data is not translated from ASCII to EBCDIC.

NOPRT
: Input data should be treated as ASCII instead of print data. If the DCB specifies an ANSI or MACHINE carriage control, print file conversion will be bypassed. If the DCB does not specify an ANSI or MACHINE carriage control, `NOPRT` is equivalent to **ASCII**.

timeout
Amount of time (in seconds) the RSH Coupler interface waits for input data to be received from the client before failing the input request. The default value is 180 seconds. If a value of zero is specified, no time-out is used.

The EOF condition on STDIN can be triggered either by using the **eof_token** or invoking the **rsh** command on the client machine, using a file as standard input. To specify a file as standard input to the **rsh** command the redirect character `'<'` must be used (See “Using the RSH Command From UNIX and PCs” on page 73).

When a file is used as standard input to the **rsh** command, the command itself detects the EOF condition while reading standard input, and transmits the same condition to the RSH Coupler. This in turn triggers a permanent EOF condition for STDIN.

STDIN Examples

Example 1:

Specify that input to a COBOL report generator comes from standard input of the **rsh** command used on the client side. The DD name is SYSIN. There is no need to specify the DCB because it is specified at open time by the COBOL program.

```
//SYSIN   DD SUBSYS=(IORR,ICDROPCL,STDIN)
```

Example 2:

Specify input to a REXX program to be executed interactively by a user on a remote client using **rsh**. The DD name is SYSTSIN. The user on the client machine specifies the EOF condition by typing the string `"/*"`. Time-out on input is 30 seconds. The record format is fixed, with a record length of 80.

```
//SYSTSIN DD SUBSYS=(IORR,ICDROPCL,STDIN,'/*',30),
//           DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

STDERR

```
►►──SUBSYS─=─(─ssname─,─ICDROPCL─,──STDERR─,──┬────────┬─,──────────────────►
                                              └─prefix─┘

   ┌─ASCII─┐
►──┼───────┼──,─┬─────────┬─)───────────────────────────────────────────────►◄
   ├─BIN───┤    └─timeout─┘
   └─NOPRT─┘
```

Notes:

1. In the diagram above, spaces are shown around commas for clarity. Do not use spaces when you code the DD statement. If you use spaces, MVS will issue an error and the RSH Coupler APPC transaction will not run.
2. Each **SUBSYS=** subparameter is shown separated by a comma. Because all subparameters are positional, commas must be used after a defaulted subparameter, unless no subparameters follow (see “STDERR Examples” on page 62).

ssname
Subsystem name used by RSH Coupler. It should be **IORR**, unless the subsystem has been installed with a different name.

ICDROPCL
Constant indicating the name of the RSH Coupler module that is invoked at open.

STDERR
Constant indicating that this is an output file whose data is sent to the **rsh** client as standard error.

prefix Character string up to 16 characters long. Each output record is concatenated to this string to form the record sent to the **rsh** client as standard error. This **prefix** can be used to differentiate output when multiple DD names are set to standard error for the same RSH Coupler APPC transaction.

ASCII EBCDIC to ASCII translation takes place. This is the default.

BIN Data is not translated from EBCDIC to ASCII.

NOPRT
Output data should be treated as ASCII instead of print data. If the DCB specifies an ANSI or MACHINE carriage control, print file conversion will be bypassed. If the DCB does not specify an ANSI or MACHINE carriage control, **NOPRT** is equivalent to **ASCII**.

timeout
Amount of time (in seconds) the RSH Coupler interface waits for an available buffer before failing the output request. The default value is 180 seconds. If a value of zero is specified, no time is used.

STDERR Examples

Example 1:

Specify output from a PL/1 program to become standard error of the client **rsh** command. The DD name is SYSPRINT. There is no need to specify the DCB since it is specified at open time by the PL/1 program.

```
//SYSPRINT DD SUBSYS=(IORR,ICDROPCL,STDERR)
```

Example 2:

Specify output from a PL/1 program to become standard error of the client **rsh** command. The DD name is SYSOUT. The record format is fixed with a record length of 130.

```
//SYSOUT   DD SUBSYS=(IORR,ICDROPCL,STDERR),
//         DCB=(RECFM=F,LRECL=130,BLKSIZE=130)
```

STDOUT

```
►►──SUBSYS─=─(─ssname─,──ICDROPCL─,─STDOUT─,─┬────────┬─,──────────────────►
                                             └─prefix─┘
```

```
   ┌─ASCII─┐
►──┼─BIN───┼──,──┬─────────┬─)──────────────────────────────►◄
   └─NOPRT─┘     └─timeout─┘
```

Notes:

1. In the diagram above, spaces are shown around commas for clarity. Do not use spaces when you code the DD statement. If you use spaces, MVS will issue an error and the RSH Coupler APPC transaction will not run.
2. Each **SUBSYS=** subparameter is shown separated by a comma. Because all subparameters are positional, commas must be used after a defaulted subparameter, unless no subparameters follow (see "STDOUT Examples").

ssname
: Subsystem name used by RSH Coupler. It should be **IORR**, unless the subsystem has been installed with a different name.

ICDROPCL
: Constant indicating the name of the RSH Coupler module that is invoked at open.

STDOUT
: Constant indicating that this is an output file whose data is sent to the **rsh** client as standard output.

prefix
: Character string up to 16 characters long. Each output record is concatenated to this string to form the record sent to the **rsh** client as standard output. This **prefix** can be used to differentiate output when multiple DD names are set to standard output for the same RSH Coupler APPC transaction.

ASCII
: EBCDIC to ASCII translation takes place. This is the default.

BIN
: Data is not translated from EBCDIC to ASCII.

NOPRT
: Output data treated as ASCII instead of print data. If the DCB specifies an ANSI or MACHINE carriage control, print file conversion will be bypassed. If the DCB does not specify an ANSI or MACHINE carriage control, **NOPRT** is equivalent to **ASCII**.

timeout
: Amount of time (in seconds) the RSH Coupler interface waits for an available buffer before failing the output request. The default value is 180 seconds. If a value of zero is specified, no time-out is used.

STDOUT Examples

Example 1:

Specify output from a PL/1 program to become standard output of the client `rsh` command. The DD name is SYSDATA. Do not use a prefix string.

```
//SYSDATA  SUBSYS=(IORR,ICDROPCL,STDOUT)
```

Example 2:

Specify outputs from a COBOL program to become standard output of the client **rsh** command. The DD names are SYSREP1 and SYSREP2. The **prefix** strings "1-" and "2-" differentiate the two output files on the client side.

```
//SYSREP1  SUBSYS=(IORR,ICDROPCL,STDOUT,'1-')
//SYSREP2  SUBSYS=(IORR,ICDROPCL,STDOUT,'2-')
```

Data Conversion

MVS and UNIX files use a different format and a different character set to store printable characters. This section describes data conversion and record mapping for all four types of RSH Coupler files.

STDPARM

File STDPARM is a one-record (or an empty file) formed by concatenating an optional **prefix** (specified in the JCL statements) with the parameter string specified by the client when using the **rsh** command.

If no **prefix** is specified in the JCL statements and the parameter string is null, file STDPARM is empty. This means that the program encounters the EOF condition when it tries to read the first record in the STDPARM file.

When the STDPARM file is not empty, RSH Coupler forms its only record as follows:

- The parameter string is converted from ASCII to EBCDIC.
- If **UPPER** has been specified in the JCL statements, the parameter string is converted to upper case.
- If a **prefix** has been specified in the JCL statements, the record is formed by concatenating the **prefix** with the parameter string. Otherwise, the parameter string alone forms the record.
- If the DCB record format for STDPARM is fixed, the record is padded with blanks to the length specified in the DCB.

For example, if STDPARM is specified for the RSH Coupler APPC transaction `RSH_try` with the following JCL statements:

```
//SYSIN   DD  SUBSYS=(IORR,ICDROPCL,STDPARM,'PROCESS ',UPPER),
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

and the user on the client side types:

```
rsh mvs_host -l mvs_userid try This is the parm string
```

the input record for DD=SYSIN will be the string:

```
"PROCESS THIS IS THE PARM STRING"
```

padded with blanks on the right to make it 80 bytes long.

STDIN

File STDIN is an input file whose records are created from the **rsh** command standard input as follows:

- The stream of bytes coming from the **rsh** command is broken into records using the ASCII line feed command X'0A' as a record separator (this character is not part of the record).

- If the DCB record format for STDIN is fixed, the record is padded with ASCII blanks (X'20') to the length specified in the DCB.
- If **ASCII** is specified in the JCL statements, then the record is translated from ASCII to EBCDIC, otherwise it is kept as is.

If an **eof_token** is specified in the JCL statements, then for each record, after performing the above steps, RSH Coupler will check if the beginning of the record is equal to the **eof_token**. If it is, the record is discarded and the end of file condition is presented to program issuing the I/O request.

Note: If **BIN** has been specified in the JCL statements, then the **eof_token** is translated to ASCII, for consistency, before being compared with the beginning of the record.

STDERR and STDOUT

Files STDERR and STDOUT are output files whose data is processed in the same way before being sent across the network. STDERR data is sent, after processing, as standard error of the **rsh** command. STDOUT data is sent as standard output of the **rsh** command.

RSH Coupler processes output data from STDERR and STDOUT as follows:

- If **ASCII** has been specified in the JCL statements, the record is translated from EBCDIC to ASCII. Oherwise the **prefix**, if any, is translated from EBCDIC to ASCII.
- The **prefix**, if any, is concatenated with the record.
- If the DCB record format is fixed, the record is stripped from trailing ASCII blanks (X'20').
- Character ASCII line feed (X'0A') is added at the end of the record.

Support of Print Files

When record format of STDIN, STDERR, and STDOUT specifies an ANSI or MACHINE carriage control character (i.e. record formats FA, FM, VA, VM, FBA, FBM, VBA, VBM, etc.) and the data type is ASCII, RSH Coupler performs an additional conversion so that the file being written or read is printed on UNIX the same way it is printed on MVS and vice versa.

For STDERR and STDOUT, RSH Coupler embeds the special characters line feed (X'0A'), carriage return (X'0D') and form feed (X'0C') to support the skip, overstrike and new page carriage control character functions.

For STDIN, RSH Coupler detects the special characters line feed (X'0A'), carriage return (X'0D') and form feed (X'0C') and substitutes them with the appropriate sequence of skip, new page and overstrike carriage control characters.

The EOF token for STDIN and the prefix for STDERR or STDOUT are handled within each record the same way as they are handled for normal ASCII files. The only difference is that RSH Coupler uses the three special characters line feed, form feed and carriage return as record separators as opposed to using only line feed for ASCII files.

Character Translation

RSH Coupler will translate characters from ASCII to EBCDIC and vice versa under the conditions previously described.

RSH Coupler translates only printable characters and the **tab** character. All characters which are not printable are left unchanged.

Translation tables are as follows:

- ASCII characters from X'21' to X'7E' are converted to the their corresponding EBCDIC printable characters with the exception of '∧' and '|'.
- Character '∧' is converted to EBCDIC ″logicalnot″ (X'5F').
- Character '|' is converted to EBCDIC ″bar″ (X'4F').
- EBCDIC printable characters are translated to ASCII using the exact inverse translation table.
- ASCII tab character X'09' is converted to EBCDIC tab character X'05' and vice versa.
- EBCDIC ″pipe″ (X'6A') is treated as a non-printable character and it is left unchanged.

Sharing Input and Output

The same type of RSH Coupler file can be opened multiple times, concurrently or otherwise, using the same DD name or different DD names.

In this section, sharing of input and output RSH Coupler files within the same RSH Coupler APPC transaction is described in detail.

Warning: Regardless on how many times the same file type is concurrently open, the RSH Coupler limitation of one RSH Coupler I/O request at a time is still in force. This means that there can be as many RSH Coupler files concurrently open as desired, but that there can be only one I/O request in progress at a time across all open RSH Coupler files within a RSH Coupler APPC transaction (see “APPC Administration” on page 58).

STDPARM

Multiple open DCBs of file STDPARM has no effect on the way data is read for each DCB. RSH Coupler will honor all reads request as if there were only one open DCB for STDPARM and as if STDPARM had not been previously opened.

STDIN

When STDIN is opened multiple times, concurrently or otherwise, the input data is shared among the open DCBs as follows:

- Each time a read request is executed a new record is returned to the caller.
- Once a record has been read, it is discarded, and it cannot be read again by another read request from a different open DCB.
- A subsequent read request, from the same DCB or a different one, will return the next available record.
- Data in the record is converted as indicated by the JCL’s options of the DD name used for each read request.

Special considerations are needed when the **eof_token** is specified in the JCL's options for one or more STDIN DD names.

If a match is detected for a read request between the beginning of the record and the **eof_token**, then the caller (and only the caller) is presented with the end of file condition. All other DCBs for STDIN will be able to continue reading STDIN records.

However all read requests will be presented with the end of file condition once there is no data left in the STDIN buffer and the end of file condition is received by the **rsh** command running on the client side (permanent end of file).

For example, let us consider a RSH Coupler APPC transaction ″RSH_testin″ written in REXX which is invoked using RSH Coupler and does the following:

- Read data from STDIN using DD=DATAIN until it detects end of file.
- Call a PL/1 program that reads the remainder of STDIN using DD=SYSIN until it detects end of file.

If the JCLs for DD=DATAIN and DD=SYSIN are coded as follows:

```
//DATAIN   DD SUBSYS=(IORR,ICDROPCL,STDIN,'/*')
//SYSIN    DD SUBSYS=(IORR,ICDROPCL,STDIN)
```

and the input file `datain` on the client side is:

```
Line read by Rexx #1
Line read by Rexx #2
/*
Data read from the program
Data read from the program
Data read from the program
Data read from the program
```

then if the user on the client side types:

```
rsh mvs_host -l mvs_userid testin < datain
```

the REXX program will read the first 2 lines of **datain** and detect the end of file condition. The PL/1 program will read the remaining lines and detect end of file condition which is transmitted from the **rsh** command when it finishes transmitting file **datain**.

In general, the existence of multiple open DCBs for STDIN within the same RSH Coupler APPC transaction, requires careful analysis and planning to avoid unpredictable results.

STDERR and STDOUT

Sharing of output files STDERR and STDOUT from multiple open DCBs is straightforward. Records are sent to the **rsh** command on the client side in the exact sequence as they are written.

When data is written using more than one DD name, different **prefix** values can be used in the JCLs to distinguish records being written. The use of a different **prefix** for each DD name will allow, at completion, to use filters to separate the data by DD name into different files on the client machine.

Filters to carry out separation based on prefixes are readily available on UNIX and can be easily provided on DOS or OS/2.

RSH Coupler Administration

Member **ICDRFLTR** in the RSH Coupler JCL distribution library contains the source code of a filter written in C, which can handle any output file from RSH Coupler, including print files. Comments at the beginning of the file explain how to compile and use **ICDRFLTR**.

Chapter 8. Application Programmer’s Guide

This chapter is intended for distributed application programmers and covers the following topics:

- How to use existing MVS batch applications as TCP/IP servers via RSH Coupler.
- How to use the RSH Coupler MULTI-TRAN facility by either coding a new RSH Coupler APPC transaction or by modifying an existing MVS batch application.
- How to use the **rsh** command on the client side in either the UNIX or PC environment.
- How to trouble shoot problems with RSH Coupler.

Using Existing MVS Batch Programs as TCP/IP Servers

Using existing MVS batch applications as TCP/IP servers, via RSH Coupler, is a straightforward proposition since no recoding is involved. The only step required is the definition of the RSH Coupler APPC transaction.

The examples described in this manual can be easily modified for the largest majority of cases. However, if this is not the case, the following step by step approach can be used for a successful implementation:

- Identify input and output files which should be mapped into RSH Coupler files (STDPARM, STDIN, STDOUT and STDERR).
- If there are multiple STDOUT or STDERR output files, consider using prefixes to the output records so as to be able to separate output using filters (see “STDERR and STDOUT” on page 65).
- If the input file is one line file, it is always easier to use STDPARM as opposed to STDIN. Also it might be advantageous to specify a prefix for STDPARM input either to simplify the amount of typing the end user has to enter or to limit what the user can do.
- If there are multiple input files, it is very important that these files are read one at the time. When this is the case, the use of an EOF token for STDIN will allow the MVS batch application to read STDIN from multiple DD names, one at a time.
- Give careful consideration to security. Besides the standard security provided by RSH Coupler, you might further restrict access by specifying an APPC security transaction profile for the RSH Coupler APPC transaction and further limit access to the transaction only to a few MVS user IDs.
- Consider the performance issue. The most important decision regarding performance is under which class the APPC transaction will run. The MVS system programmer should be consulted when deciding which APPC class to use.

Using the MULTI-TRAN RSH Coupler Facility

The RSH Coupler MULTI-TRAN facility has been provided to allow for high performance MVS servers based on the Remote SHell protocol. MULTI-TRAN RSH Coupler servers can be new programs coded from scratch, or can be existing MVS batch applications modified to achieve better performance.

The Organization of a RSH Coupler MULTI-TRAN APPC Transaction

An RSH Coupler MULTI-TRAN APPC transaction is a program which is composed of three sections:

- Initialization.
- Processing loop.
- Clean up and termination.

In the initialization section the APPC transaction goes through all the initialization steps such as:

- Allocate all needed storage.
- Open all MVS files needed to access data, such as VSAM data sets.
- Establish connections with other MVS subsystems such as DB2.

All these functions will be performed under the generic MVS user ID specified in the APPC definition of the transaction.

The processing loop section actually executes the **rsh** command requests coming from the client machines. The basic structure of the processing loop section is as follows:

1. Call **ICDRNXT** or **ICDRRNXT** if the transaction is written in REXX. This will block the APPC transaction until an incoming `rsh` command is detected.
2. Upon return from **ICDRNXT** check the return code. If the return code is non-zero start the termination section. Otherwise start processing the incoming **rsh** command.
3. Open input files such as STDPARM and STDIN and read the input data coming from the client machine.
4. Execute the request according to the input data.
5. Write the output back to the end user using output files such as STDOUT and STDERR.
6. Close all RSH Coupler files (STDPARM, STDIN, STDOUT and STDERR). GO TO step 1.

It is important to stress that each time function **ICDRNXT** is called, the APPC transaction MVS user ID is changed to a new value according to the RSH Coupler security information genned into **ICDRMAP**. Any security check that is done thereafter, such as when opening a new MVS file or when establishing a new DB2 thread, will be done under the new MVS user ID.

The clean up and termination section is responsible for freeing all resources and terminating. It is entered when the return code from **ICDRNXT** is nonzero. When this happens the MVS user ID is reset to the generic MVS user ID under which the initialization section was executed.

Routines Used in MULTI-TRAN RSH Coupler APPC Transactions

RSH Coupler provides two routines which can be used to wait for an incoming `rsh` command. The two routines are **ICDRNXT** (to be used from compiled or assembled program using the standard 370 linkage convention) and **ICDRRNXT** (to be used from REXX). RSH Coupler also provides two additional functions to reset the MVS user ID to the generic value in between incoming **rsh** commands so that opening new MVS files or starting new connections to MVS subsystems can be done under a trusted MVS user ID.

ICDRNXT

Wait for a new incoming **rsh** command.

ICDRNXT

```
►►──CALL─ICDRNXT─(─token─,─uid─,─rc─)──────────────────────────────►◄
```

Note: Routine **ICDRNXT** can be called by programs written in Assembler or any compiled language such as PL/1, COBOL, FORTRAN, etc.

token is a 31 bits integers which must be initialized to zero before calling **ICDRNXT** for the first time. **token** is used as internal anchor by the RSH Coupler routines and should not be examined or used except when calling **ICDRNXT** or **ICDRRES**.

uid is a 8 characters fixed length string. Upon return it contains the new MVS user ID under which the RSH Coupler APPC transaction is now running.

rc is a 31 bit integer which, upon return, contains the return code. Possible values are:

Value	Meaning
0	A new **rsh** command request has been received. The MVS user ID has been reset and processing should continue.
8	There are no new incoming **rsh** commands. The wait time for this APPC class has been exceeded. The RSH Coupler APPC transaction has the option to immediately call **ICDRNXT** again and wait, or to terminate. The MVS user ID has been reset to the generic user ID for this RSH Coupler APPC transaction.
12	The APPC transaction scheduler is not active. The RSH Coupler APPC transaction must terminate.
16	The RSH Coupler APPC transaction has not been defined as MULTI-TRAN. The RSH Coupler APPC transaction must terminate.
20	System error. The RSH Coupler APPC transaction must terminate.
24	A call to either **ICDRNXT** or ICDRRES is still in progress. This is a logic error which can happen for multitasking RSH Coupler APPC transaction. The RSH Coupler APPC transaction must terminate.
28	Due to an operator APPC command no more incoming rsh commands will be received. The RSH Coupler APPC transaction must terminate.
98	Unrecoverable error. The RSH Coupler APPC transaction must terminate.
99	Some RSH Coupler files are still open. This is an application logic error. The RSH Coupler APPC transaction must terminate.

REXX function **ICDRRNXT** is a REXX interface to routine **ICDRNXT**. **ICDRRNXT** has no parameters and returns a string which contains the return code. Its possible values and meanings are the same as those returned by **ICDRNXT** and are explained above. After calling **ICDRRNXT** the new MVS user ID can be extracted using the standard REXX function **Userid()**. The typical calling sequence for **ICDRRNXT** is as follows:

```
ret_code= Icdrrnxt()
uid= Userid()
```

ICDRRES

Close the connection with client side of the **rsh** command and reset the MVS user ID to the RSH Coupler APPC transaction generic user ID.

ICDRRES

```
►►──CALL─ICDRRES─(─token─,─rc──)──────────────────────────────────────►◄
```

Note: Routine **ICDRRES** can be called by programs written in Assembler or any compiled language such as PL/1, COBOL, FORTRAN, etc. Function **ICDRRRES** provides the equivalent functionality from REXX.

token Is a 31 bits integers which must be initialized to zero before calling **ICDRRES** for the first time. **token** is used as internal anchor by the RSH Coupler routines and should not be examined or used except when calling **ICDRNXT** or **ICDRRES**.

rc Is a 31 bit integer which, upon return, contains the return code. Possible values are:

Value	Meaning
0	A new **rsh** command request has been received. The MVS user ID has been reset and processing should continue.
4	Generic error. The RSH Coupler APPC transaction must terminate.
12	The APPC transaction scheduler is not active. The RSH Coupler APPC transaction must terminate.
16	The RSH Coupler APPC transaction has not been defined as MULTI-TRAN. The RSH Coupler APPC transaction must terminate.
20	System error. The RSH Coupler APPC transaction must terminate.
24	A call to either **ICDRNXT** or **ICDRRES** is still in progress. This is a logic error which can happen for multitasking RSH Coupler APPC transaction. The RSH Coupler APPC transaction must terminate.
97	Routine **ICDRRES** has been called before ever calling routine **ICDRNXT**. This is an application logic error. The RSH Coupler APPC transaction must terminate.
98	Unrecoverable error. The RSH Coupler APPC transaction must terminate.
99	Some RSH Coupler files are still open. This is an application logic error. The RSH Coupler APPC transaction must terminate.

REXX function **ICDRRRES** is a REXX interface to routine **ICDRRES**. **ICDRRRES** has no parameters and returns a string which contains the return code. Its possible values and meanings are the same as those returned by **ICDRRES** and are explained above. The typical calling sequence for **ICDRRRES** is as follows:

```
ret_code= Icdrrres()
```

Handling Time Out and Connection Lost Conditions

When an RSH Coupler APPC transaction reads or writes to an RSH Coupler file, it uses the MVS I/O routines to communicate with the client side of the **rsh** command.

Under certain conditions, such as the end user on the client machine killing the **rsh** command in the middle of processing, the connection between RSH Coupler and the **rsh** command on the client machine can be lost. This will in turn trigger a time out or connection lost condition.

RSH Coupler will handle these conditions in such a way as to allow the RSH Coupler APPC transaction to recover and continue processing new incoming **rsh** commands. When a time out or connection lost condition will be detected the following will happen in regard to RSH Coupler files:

- If a read or write operation is in progress it will fail.
- Any subsequent read or write operation will also fail.
- Under no circumstances will any OPEN or CLOSE operation fail due to time out or connection lost condition.

Thus a RSH Coupler APPC MULTI-TRAN transaction should always be ready to recover from a failure when reading or writing to an RSH Coupler file. It does not need however to be able to recover from OPEN or CLOSE operations for the same files. When an I/O error is detected for any of the RSH Coupler file, the transaction should close all RSH Coupler files still open and call **ICDRNXT** to wait for the next incoming **rsh** command.

The PL/1 source code for example 6, stored in member **ICDRPHON** of the distribution JCL library, shows clearly how to recover from RSH Coupler files I/O errors.

Using the RSH Command From UNIX and PCs

The **rsh** command is available on all flavors of UNIX and is widely available on PC's network operating systems like NetWare, Windows NT, and OS/2.

The **rsh** command is straightforward and easy to use. However there are a few aspects that need to be understood in order to use it properly and remotely execute a RSH Coupler APPC transaction. These are:

- How to specify an MVS USERID.
- How to redirect standard input, standard output and standard error of the **rsh** command to local files.
- How to execute an RSH Coupler APPC transaction when standard input is the terminal.
- How to terminate **rsh** abnormally.
- How to use special characters in the parm string on UNIX.
- How to use the **rsh** command from other programs such as UNIX SHELLs, C or C++ applications, OS/2 REXX EXECS, and DOS batch scripts.

Specifying an MVS USERID

When using **rsh** to execute a RSH Coupler APPC transaction, the `-l` option should be used to specify the MVS USERID under which the RSH Coupler APPC transaction is executed. There is no need to use the `-l` option if end user UNIX login name is the same as the MVS user ID to be used. In general the `-l` option should always be used when invoking the **rsh** command from a PC.

Specifying Local Files as Input and/or Output

In UNIX, DOS, and OS/2 the characters '<' and '>' are used to redirect standard input and standard output to files. This applies not only to **rsh** but to any command being executed in any of the three environments. For example, to redirect **rsh** standard input and output to files **in** and **out** you can type:

```
rsh mvs_host -l mvs_userid test < in > out
```

If both standard output and standard error must be redirected to files, the redirect strings "1>" for standard output, and "2>" for standard error, followed by their respective file names, can be used on DOS, OS/2, and on UNIX with Korn shell.

For example, to redirect **rsh** standard input, output and error to files in, out, and err, you can type:

```
rsh mvs_host -l mvs_userid test < in 1> out 2> err
```

Using RSH Interactively

When standard input is not redirected from a file the following should be kept in mind:

- Each input line from the terminal becomes a STDIN input record to the RSH Coupler APPC transaction.
- The program reading STDIN should use QSAM. If it uses BSAM, control will not be returned to the program by the MVS access method until the BSAM block is full, which normally requires multiple input lines. This precludes interactive mode.
- The **rsh** command will never detect the end of file condition since it is reading from a terminal. The **eof_token** for the STDIN DD name(s) should be non-null to allow the user on the client side to indicate end of file by typing the string specified as **eof_token.**

Terminating RSH Abnormally

To immediately terminate **rsh** on all environments you can use the CtrlC key. The use of CtrlC key might trigger an ABEND of the RSH Coupler APPC transaction being executed on MVS. In some cases appropriate dump options should be set to avoid excessive dumping.

Using Special Characters on UNIX

On UNIX, if special characters such as '*', '(', ')', etc., need to be specified in the parameter string to the **rsh** command, the parameter string itself must be enclosed in double quotes.

Using RSH From Other Programs

In most instances the **rsh** command will be used from within other programs when remotely executing an RSH Coupler APPC transaction. These programs can range from simple front-end script, to reduce the amount of typing the end user has to enter, to very complex programs such as X window applications written in C or C++. In either case the use of the **rsh** command follows the same sequence of steps:

1. Create the input file to the **rsh** command if one is needed. You should create the file in a memory only file system if possible to eliminate all I/Os.

2. Call the **rsh** command with appropriate parameters. Use the I/O redirect characters `'<'` and `'>'` to redirect standard input, output and error. Pipe standard output and/or error through the **ICDRFLTR** filter if necessary (see "STDERR and STDOUT" on page 65).
3. Check the return code from the **rsh** command. If nonzero call error handling function, otherwise continue with next step.
4. Read in standard error and check for any error message from the command executed remotely.
5. Read in standard output and extract the data needed.

In most cases the above steps will take few lines of code regardless of the language used.

When using the C or C++ language, it is important to remember that the `rsh` command can be invoked via the **system()** C function.

Trouble Shooting

Most problems encountered when using RSH Coupler belong to one of the following groups:

Security problems

Either the end user on the client machine is not specifying the right MVS user ID, or the client login name has not being specified in the **RHOSTS** library for the MVS user ID the end user wants to use.

For security related problems the log files of the RSH Coupler gateway should be inspected. The RSH Coupler gateway always writes a set of messages for each **rsh** command request received. These messages include the IP address of the client machine, the login name of the end user, and the MVS user ID requested. These three values, and only these three, are used to check if the request is valid or not.

Command is not found

For this class of problems either the RSH Coupler APPC transaction has not been defined to APPC or the end user is specifying the wrong transaction name. It should be kept in mind that RSH Coupler APPC transaction names are case sensitive and are prefixed with the prefix specified in the PROC used to start the RSH Coupler gateway started task. The default value is **RSH_**. The RSH Coupler gateway log contains the information needed to solve this type of problems.

Command cannot be executed

This normally happens either when the RSH Coupler APPC transaction has not been defined as active, or when some of the APPC resources used by the RSH Coupler APPC transaction have not been defined and/or are not available. A typical case is when the RSH Coupler APPC transaction has been assigned to an APPC class that does not exist or is not active.

I/O ABENDs

The APPC transaction starts executing but terminates abnormally due to some I/O error for one of the RSH Coupler files. Most of these problems are due to incompatible DCB parameters specified in the JCLs, or the lack of DCB parameters altogether. RSH Coupler does not supply defaults for the DCB parameters if none have been specified. Unless the APPC transaction program specifies internally the DCB parameters, the correct values should be coded in the JCLs. If an I/O error is detected, a complete description of the problem is given by the MVS manuals describing error messages and codes. The description is normally very complete and enough information is provided to fix the problem.

Note: In order to browse the last log file being written by the RSH Coupler gateway, while it is up and running, it may be necessary to tell RSH Coupler to close the current log file and open a new one. See page 10 on how to issue the **CLOse** command to have the current log file closed.

RSH Coupler Examples

RSH Coupler is distributed with the following examples:

Example 1. Using RSH Coupler for a Remote TSO Session

Member ICDRTSOA in the RSH Coupler JCL library contains the JCLs to create the RSH Coupler APPC transaction for Example 1.

The name of the APPC transaction is **RSH_rtso,** which starts a line mode TSO session which is equivalent to a line mode TELNET session. The major advantage of using **RSH_rtso** over TELNET is that it does not require a password. For example it can be used to automatically run TSO script stored on a UNIX or PC file without the need of the end user intervention.

RSH_rtso uses the following files as input and output:

- DD=SYSTSIN of type STDIN is used as input to TSO.
- DD=SYSTSPRT of type STDOUT is used as output from TSO. The EOF token has been specified as `'/*'`

For both files the time out parameter has been set to 60 seconds so that if no traffic will take place in the next 60 seconds the TSO session will be automatically terminated.

To invoke the remote TSO session a UNIX or PC end user has to type:

```
rsh mvs_host -l mvs_userid ICDRTSOA
```

The user will be prompted with the familiar TSO **READY** prompt and can enter as many TSO commands as needed. The output from each command will be shown on the screen. To end the session the end user has to type the EOF token which is `/*` immediately followed by ENTER.

Member ICDRTSOA listing is shown in Figure 19:

```
//ICDRTSOA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//*  Run this JOB to create the Data Coupler RSH server APPC
//*  transaction "RSH_rtso"
//*
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  2. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  3. Add transaction JOB cards
//*
//*-------------------------------------------------------------------
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_rtso)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      KEEP_MESSAGE_LOG(ERROR)
      JCL_DELIMITER(@@)
```

Figure 19. JCL Member ICDRTSOA (Part 1 of 2)

```
//*---> Put APPC JOB cards here (include time limit !!!)
//*
//TSO      EXEC PGM=IKJEFT01,DYNAMNBR=75,TIME=100,REGION=6M
//*
//SYSTSIN  DD SUBSYS=(IORR,ICDROPCL,STDIN,'/*',,60),
//            DCB=(RECFM=F,LRECL=255,BLKSIZE=255,BUFNO=1)
//SYSTSPRT DD SUBSYS=(IORR,ICDROPCL,STDOUT,,,5),
//            DCB=(RECFM=F,LRECL=255,BLKSIZE=255,BUFNO=1)
//*
//@ICDRDD@ DD SUBSYS=(IORR,ICDRRMGR)
@@
##
$$
/*
```

Figure 19. JCL Member ICDRTSOA (Part 2 of 2)

Example 2. Using MVS Sort From UNIX and PCs

Member ICDRSRTA in the RSH Coupler JCL library contains the JCLs to create the RSH Coupler APPC transaction for Example 2.

The APPC transaction name is **RSH_rsort**, which allows UNIX and PC end users to invoke MVS sort utility. It uses the following files as input and output:

- DD=SYSIN of type STDPARM is used as input for sort control statement. Its data is automatically converted to upper case and is prefixed with the string: ″SORT″.
- DD=SORTIN of type STDIN is used as input for the data to be sorted. The DCB has been defined with variable record format of up to 255 data characters. This is the maximum length allowed by the VI editor on UNIX.
- DD=SORTOUT of type STDOUT is used as output for the sorted data. It has the same DCB as DD=SORTIN.
- DD=SYSOUT of type STDERR is used for information and diagnostic message by the sort program.

Option **BIN** has been specified for both STDIN and STDOUT so that data will not be translated from ASCII to EBCDIC and vice versa. This allows to sort character fields according to the ASCII character sort order.

The combination of fixed record length and the **BIN** option for both input and output sort files will force each sort record to be padded on input with ASCII blank characters, up to its DCB record length, and to be stripped on output of trailing ASCII blanks.

RSH_rsort can be used as follows to sort the UNIX or PC file **sortin** in ascending order into file **sortout,** using characters in column 1-8 as a sort field. The PC end user will type:

```
rsh mvs_host -l mvs_userid rsort fields=(5,8,ch,a) <sortin > sortout
```

The UNIX end user will type:

```
rsh mvs_host -l mvs_userid rtso "fields=(5,8,ch,a)" <sortin > sortout
```

This example can be quite useful to sort data residing on PCs where available sort routines are rudimentary at best.

Notes:

1. The starting column for sort fields must be increased by 4, since input record format is variable length. This requires 4 control bytes at the beginning of each record.
2. Double quotes are necessary in UNIX since otherwise characters `'('` and `')'` have special meaning for the shell.

Warning: MVS sort requires that input sort records be long enough to contain all the sort fields specified, since it does not automatically pad records with blanks or binary zeros. Make sure that all records in the input sort file are long enough when using this example.

Member ICDRSRTA listing is shown in Figure 20:

```
//ICDRSRTA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//*  Run this JOB to create the Data Coupler RSH server APPC
//*  transaction "RSH_rsort".
//*
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  2. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  3. Add transaction JOB cards
//*
```

Figure 20. JCL Member ICDRSRTA (Part 1 of 2)

```
//*--------------------------------------------------------------------
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_rsort)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      JCL_DELIMITER(@@)
//*---> Put APPC JOB cards here (include time limit !!!)
//*
//SORT     EXEC PGM=SORT
//*
//SYSIN    DD SUBSYS=(IORR,ICDROPCL,STDPARM,
//            '        SORT ',UPPER),
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSOUT   DD SUBSYS=(IORR,ICDROPCL,STDERR)
//*
//SORTIN   DD SUBSYS=(IORR,ICDROPCL,STDIN,,BIN),
//            DCB=(RECFM=VB,LRECL=259,BLKSIZE=32760)
//SORTOUT  DD SUBSYS=(IORR,ICDROPCL,STDOUT,,BIN),
//            DCB=(RECFM=VB,LRECL=259,BLKSIZE=32760)
//*
//@ICDRDD@ DD SUBSYS=(IORR,ICDRRMGR)
//*
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(20,20))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(20,20))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(20,20))
@@
##
$$
/*
```

Figure 20. JCL Member ICDRSRTA (Part 2 of 2)

Example 3. Using Batch SPUFI to Access DB2 Data

Member ICDRSPFA in the RSH Coupler JCL library contains the JCLs to create the RSH Coupler APPC transaction for Example 3.

The APPC transaction name is **RSH_rspufi,** which allows end users on UNIX and PCs to use the PL/1 program DSNTEP2 which is the batch version of SPUFI distributed with DB2. It uses the following files as input and output:

- DD=SYSIN of type STDPARM is used by DSNTEP2 to read the SQL query. Since the record length is 80, the query itself cannot be longer than 80 characters.
- DD=SYSPRINT of type STDOUT is used to output the result of the SQL query as well as the SQL diagnostic messages.
- DD=SYSTSPRT of type STDERR is used to output TSO diagnostic messages.

Program DSNTEP2 must be run under batch TSO. File DD=SYSTSIN (input to TSO) points to member RSPUFIIN in the RSH Coupler JCL distribution library. It contains TSO commands to execute DSNTEP2 under the proper DB2 environment.

Member RSPUFIIN must be edited to specify the target DB2 subsystem name and, optionally, the plan name, if DSNTEP2 has been bound to DB2 with a plan name different from DSNTEP2.

RSH_rspufi can be used as follows to store the attributes of all DB2 tables in local file `list`. The PC user will type:

```
rsh mvs_host -l mvs_userid rspufi select * from sysibm.systables >list
```

The UNIX user will type:

```
rsh mvs_host -l mvs_userid rspufi \
  "select * from sysibm.systables" >list
```

Note: Double quotes are necessary in UNIX since otherwise character `'*'` has a special meaning for the UNIX shell.

Member ICDRSPFA listing is shown in Figure 21:

```
//ICDRTSOA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*------------------------------------------------------------------
//*
//*  Run this JOB to create the Data Coupler RSH server APPC
//*  transaction "RSH_rspufi"
//*
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  3. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  4. Add transaction JOB cards
//*
//*  5. Change 'dddddd' to the high level node used for DB2 libraries
//*
//*  6. If PL/I run time environment is not in the system LINKLIB
//*     contenation add PL/I or LE libraries to the STEPLIB
//*
//*  7. Modify member ICDRSPFI if the name of the DB2 subsystem
//*     name is not "DB2A"
//*
```

Figure 21. JCL Member ICDRSPFA (Part 1 of 2)

```
//*------------------------------------------------------------------
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_rspufi)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      KEEP_MESSAGE_LOG(ERROR)
      JCL_DELIMITER(@@)
//*---> Put APPC JOB cards here (include time limit !!!)
//*
//SPUFI    EXEC PGM=IKJEFT01,REGION=1M
//*
//STEPLIB  DD DISP=SHR,DSN=dddddd.RUNLIB.LOAD
//         DD DISP=SHR,DSN=dddddd.SDSNEXIT
//         DD DISP=SHR,DSN=dddddd.SDSNLOAD
//*
//SYSTSIN  DD DISP=SHR,DSN=ICD.V1R1M0.SICDCNTL(ICDRSPFI)
//SYSTSPRT DD SUBSYS=(IORR,ICDROPCL,STDERR)
//*
//SYSIN    DD SUBSYS=(IORR,ICDROPCL,STDPARM),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSPRINT DD SUBSYS=(IORR,ICDROPCL,STDOUT)
//*
//@ICDRDD@ DD SUBSYS=(IORR,ICDRRMGR)
@@
##
$$
/*
```

Figure 21. JCL Member ICDRSPFA (Part 2 of 2)

In case there is a need to invoke DSNTEP2 with queries which span multiple lines, member RSHSPUFI can be changed by specifying STDIN instead of STDPARM for DD=SYSIN as follows:

```
//SYSIN DD SUBSYS=(IORR,ICDROPCL,STDIN),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

The end user must first edit the input file (say `spufi.in`) with the correct SQL statements and enter the following command:

```
rsh mvs_host -l mvs_userid spufi <spufi.in > list
```

Upon completion, output from DSNTEP2 will be in file `list`.

Example 4. Printing UNIX Files on an MVS Printer

Member ICDRPRTA in the RSH Coupler JCL library contains the JCLs to create the RSH Coupler APPC transaction for Example 4.

The APPC transaction name is **RSH_rprint,** which allows end users to print local files using MVS high volume printers. **RSH_rprint** uses IEBGENER to spool data to an MVS printer with following input and output files:

- DD=SYSUT1 of type STDIN is used as input to program IEBGENER. The record format of this file has been specified as VBA so that RSH Coupler will create the proper ANSI carriage control character for each record read.
- DD=SYSUT2 is used as SYSOUT output from program IEBGENER whose class specifies an MVS printer.
- DD=SYSPRINT of type STDERR is used by IEBGENER to print diagnostic messages.

RSH_rprint can be used to print the UNIX flat file `source.c`, which contains C source code, as follows:

```
pr source.c | rsh mvs_host -l mvs_userid rprint
```

As shown above, output from the **pr** command, which preformats file `source.c` into a UNIX print file, is piped into the **rsh** command which in turn sends the data to **RSH_rprint.**

Member ICDRPRTA listing is shown in Figure 22:

```
//ICDRPRTA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*------------------------------------------------------------------
//*
//*  Run this JOB to create the Data Coupler RSH server APPC
//*  transaction "RSH_rprint".
//*
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  2. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  3. Add transaction JOB cards
//*
//*  4. Modify DD=SYSUT2 statement to target the correct printer.
//*
```

Figure 22. JCL Member ICDRPRTA (Part 1 of 2)

```
//*-------------------------------------------------------------------
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_rprint)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      JCL_DELIMITER(@@)
//*---> Put APPC JOB cards here (include time limit !!!)
//*
//PRINT    EXEC PGM=IEBGENER
//*
//SYSIN    DD DUMMY
//SYSPRINT DD SUBSYS=(IORR,ICDROPCL,STDERR)
//*
//SYSUT1   DD SUBSYS=(IORR,ICDROPCL,STDIN),
//            DCB=(RECFM=VA,LRECL=135,BLKSIZE=139)
//SYSUT2   DD SYSOUT=P     CLASS USED FOR THE MAIN PRINTER
//*
//@ICDRDD@ DD SUBSYS=(IORR,ICDRRMGR)
@@
##
$$
/*
```

Figure 22. JCL Member ICDRPRTA (Part 2 of 2)

Example 5. Using IDCAMS Service From UNIX and PCs

Member ICDRIDCA in the RSH Coupler JCL library contains the JCLs to create the MULTI-TRAN RSH Coupler APPC transaction for Example 5.

The APPC transaction name is **RSH_ridcams,** which allows end users to remotely execute IDCAMS commands. **RSH_ridcams** is written in REXX and is stored in member ICDRIDCM of the RSH Coupler REXX distribution library. It can be easily modified to create others RSH Coupler APPC MULTI-TRAN transactions.

RSH_ridcams uses the following files as input and output:

- DD=STDPARM of type STDPARM is used to read the IDCAMS command to be executed.
- DD=STDERR of type STDERR is used to print diagnostic messages from the REXX program. The prefix '`Ridcams: `' has been specified to differentiate this output from IDCAMS output.
- DD=SYSPRINT of type STDERR is used by IDCAMS itself to print its output.
- DD=SYSDOUT of type STDOUT is used by IDCAMS to print data. This file is only used when the IDCAMS command contains the keyword **ofile(stdout).**
- DD=SYSDIN of type STDIN is used by IDCAMS to read data. This file is only used when the IDCAMS command contains the keyword **ifile(stdin)**.

A time out of 2 seconds has been specified for all RSH Coupler files to limit the possibility of an end user to hang on the transaction.

The execution logic of **RSH_ridcams** is as follows:

1. Call RSH Coupler function **ICDRRNXT.**
2. If the return code from **ICDRRNXT** is non zero exit, otherwise continue.
3. Read the IDCAMS command from DD=STDPARM.
4. Echo the command to the end user and check that is neither null (i.e. no command specified) nor too long. If either case is true write error message and GO TO step 7
5. Write the IDCAMS command into file DD=SYSIN.
6. Call MVS IDCAMS.
7. Close DD=STDPARM and DD=STDERR.
8. GO TO step 1.

It is important to stress the fact that each time the MVS IDCAMS is called, it executes under a different MVS userid which is reset every time the function **ICDRRNXT** is called.

RSH_ridcams can be used to list all files with high level qualifier `SYS1`. The PC end user will type:

```
rsh mvs_host -l mvs_userid ridcams listc level(sys1)
```

The UNIX end user will type:

```
rsh mvs_host -l mvs_userid ridcams "listc level(sys1)"
```

RSH_ridcams can also be used to copy the content of file `SYS1.PARMLIB(IEFSSN00)` into local file `list`. The PC end user will type:

```
rsh mvs_host -l mvs_userid ridcams repro ids(sys1.parmlib(iefssn00)) ofile(stdout) > list
```

The UNIX end user will type:

```
  rsh mvs_host -l mvs_userid ridcams \
    "repro ids(sys1.parmlib(iefssn00)) ofile(stdout)" > list
```

A similar command can be used to copy files from the end user machine into MVS files.

Note: Double quotes are necessary in UNIX since otherwise characters `'('` and `')'` have a special meaning for the UNIX shell.

Member ICDRIDCA listing is shown in Figure 23:

```
//ICDRIDCA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
```

Figure 23. JCL Member ICDRIDCA (Part 1 of 3)

```
//*
//*------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*------------------------------------------------------------------
//*
//*  Run this JOB to create Data Coupler RSH server APPC
//*  transaction "RSH_ridcams"
//*
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  2. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  3. Change  uuuuuu  to the MVS USERID to be used as generic
//*     USERID
//*
//*  4. Add transaction JOB cards
//*
//*------------------------------------------------------------------
//*
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_ridcams)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      TPSCHED_TYPE(MULTI_TRANS)
      GENERIC_ID(uuuuuu)
      KEEP_MESSAGE_LOG(ERROR)
      JCL_DELIMITER(@@)
```

Figure 23. JCL Member ICDRIDCA (Part 2 of 3)

```
//*---> Put APPC JOB cards here (include time limit !!!)
//*
//IDCAMS   EXEC PGM=IRXJCL,PARM=ICDRIDCM
//*
//SYSEXEC  DD DISP=SHR,DSN=ICD.V1R1M0.SICDEXEC
//*
//SYSIN    DD UNIT=SYSDA,SPACE=(TRK,(1,1)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//SYSPRINT DD SUBSYS=(IORR,ICDROPCL,STDERR,,NOPRT,2)
//*
//STDPARM  DD SUBSYS=(IORR,ICDROPCL,STDPARM,,UPPER),
//            DCB=(RECFM=V,LRECL=259,BLKSIZE=263,BUFNO=1)
//STDERR   DD SUBSYS=(IORR,ICDROPCL,STDERR,'Ridcams: ',,2),
//            DCB=(RECFM=V,LRECL=259,BLKSIZE=263,BUFNO=1)
//SYSTSPRT DD SUBSYS=(IORR,ICDROPCL,STDERR,'Debug..: ',,2),
//            DCB=(RECFM=V,LRECL=259,BLKSIZE=263,BUFNO=1)
//*
//STDIN    DD SUBSYS=(IORR,ICDROPCL,STDIN,,,2)
//STDOUT   DD SUBSYS=(IORR,ICDROPCL,STDOUT,,,2)
@@
##
$$
/*
```

Figure 23. JCL Member ICDRIDCA (Part 3 of 3)

Member **ICDRIDCM** listing is shown in Figure 24:

```
/*-Rexx------------------------------------------------------------*/
/*                                                                 */
/*      Licensed Material - Property of IBM                        */
/*      5655-A99 (C) Copyright IBM 1998                            */
/*      (C) Copyright FOLIUM Inc. 1994, 1998                       */
/*      All Rights Reserved                                        */
/*      US Government Users Restricted Rights -                    */
/*      Use, duplication or disclosure restricted                  */
/*      by GSA ADP Schedule Contract with IBM Corp.                */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*                                                                 */
/*       This EXEC must be invoked with the "rsh" command via the  */
/*         ICD RSH server product                                  */
/*                                                                 */
/*       ICDRIDCM execute an IDCAMS command such as LISTC          */
/*                                                                 */
/*       This EXEC reads input from DD=STDPARM and writes output to */
/*        DD=STDERR.                                               */
```

Figure 24. REXX Member ICDRIDCM (Part 1 of 4)

```
/*                                                                   */
/*      IDCAMS, when invoked by this EXEC, reads input from         */
/*       DD=SYSIN which is of type STDPARM (i.e. 1 record           */
/*       containing the paramters to the rsh command).              */
/*       Output from IDCAMS is written to DD=SYSYPRINT which        */
/*       is of type STDOUT.                                          */
/*                                                                   */
/*      Addtionally IDCAMS can read input from DD=STDIN             */
/*       (type STDIN) and write output to DD=STDOUT                  */
/*       (type STDOUT). This can be done by specifying the          */
/*       the clause "IFILE(STDIN)" or "OFILE(STDOUT)" to            */
/*       a command such as REPRO.                                    */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*      Edit memebr ICDRIDCA in ICD.V1R1M0.BASE.CNTL library        */
/*      to define the APPC transaction which execute this EXEC      */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*     Examples:                                                     */
/*                                                                   */
/*      A) List all files with high level identifier 'SYS1'         */
/*         The user on the client machines types:                   */
/*                                                                   */
/*           rsh mvs_host -l mvs_uid ridcams "listc level(sys1)"    */
/*                                                                   */
/*                                                                   */
/*      B) List content of file 'SYS1.PARMLIB(IEFSSN00)'            */
/*         The user on the client machines types:                   */
/*                                                                   */
/*           rsh mvs_host -l mvs_uid ridcams   \                    */
/*             "repro ids('sys1.parmlib(iefssn00)') ofile(STDOUT)"  */
/*                                                                   */
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/*  Start main loop                                                  */
/*                                                                   */
/*--!! WARNING !!----------------------------------------------------*/
/*                                                                   */
/*  To debug use the "say" or the "trace" statement ONLY in between  */
/*  the Icdrrnxt() function and the Close() function. All           */
/*  Rexx debugging output will be routed to the RSH client          */
/*  terminal.                                                        */
/*                                                                   */
/*-------------------------------------------------------------------*/
```

Figure 24. REXX Member ICDRIDCM (Part 2 of 4)

```
  DO FOREVER
    r= Icdrrnxt()
    IF r ^= 0 THEN                       /* Exit if RC ^= 0            */
      EXIT

    CALL Reset                           /* Reset variables            */
    CALL Add_line "Started, Uid="userid() /* Punch start message out */
    CALL Get_parm                        /* Get RSH paramters          */

    SELECT                               /* Check if parms are valid   */
      WHEN parm = "" THEN                /* Nothing is not valid       */
        CALL Add_line "Error, No input specified"
      WHEN length(parm) > 71 THEN        /* Too long if > 71 characters*/
        CALL Add_line "Error, Command is too long"
      OTHERWISE DO
        CALL Add_line "Cmd= "parm        /* Echo parm                  */
        CALL Flush                       /* Push messages out          */
        queue " "parm                    /* Put parm into DD=SYSIN as  */
        "EXECIO 1 DISKW SYSIN (FINIS)"   /*  input to IDCAMS           */
        ADDRESS LINK "IDCAMS"            /* Call IDCAMS                */
        END
      END

    CALL Close                           /* CLose all files            */
    END
/*------------------------------------------------------------------*/
/*  Reset all global variables                                      */
/*------------------------------------------------------------------*/

Reset: PROCEDURE EXPOSE out.
  out.0= 0
  return

/*------------------------------------------------------------------*/
/*  Get RSH paramters from DD=STDPARM                               */
/*------------------------------------------------------------------*/

Get_parm: PROCEDURE EXPOSE parm
  "EXECIO 1 DISKR STDPARM (STEM PARM. FINIS)"
  IF parm.0 = 0 THEN
    parm= ""
  ELSE
    parm= parm.1
  return

/*------------------------------------------------------------------*/
/*  Add a line to be sent to the RSH user later on                  */
/*------------------------------------------------------------------*/

Add_line: PROCEDURE EXPOSE out.
  i= out.0+1
  out.i= ARG(1)
  out.0= i
  return
```

Figure 24. REXX Member ICDRIDCM (Part 3 of 4)

```
/*-----------------------------------------------------------------*/
/*  Force all lines added with Add_line to the user screen          */
/*   by issuing an EXECIO followed by a CLOSE                       */
/*-----------------------------------------------------------------*/

Flush: PROCEDURE EXPOSE out.
  "EXECIO "out.0" DISKW STDERR (STEM OUT. FINIS)"
  out.0= 0
  return

/*-----------------------------------------------------------------*/
/*  Close all netwrok files                                         */
/*-----------------------------------------------------------------*/

Close: PROCEDURE EXPOSE out.
  IF out.0 > 0 THEN                    /* Write an close if any data */
    "EXECIO "out.0" DISKW STDERR (STEM OUT. FINIS)"
  "EXECIO 0 DISKW SYSTSPRT (FINIS)"    /* Force close on Rexx output */
  return
```

Figure 24. REXX Member ICDRIDCM (Part 4 of 4)

Example 6. Accessing VSAM KSDS Files From UNIX and PCs

Member ICDRPHOA in the RSH Coupler JCL library contains the JCLs to create the MULTI-TRAN RSH Coupler APPC transaction for Example 6.

The APPC transaction name is **RSH_rphone**, which allows end users to remotely access an employee's phone list stored in a VSAM KSDS data set. **RSH_rphone** is written in PL/1 and is stored in member ICDRPHON of the RSH Coupler JCL distribution library. It can be easily modified to create others RSH Coupler APPC MULTI-TRAN transactions, such as DB2 or IMS-BMPs servers.

RSH_rphone uses the following files as input and output:

- DD=STDPARM of type STDPARM is used to read the employee last name to be used as a search key. If the last name specified by the end user ends with character `'*'` the key is treated as generic.
- DD=STDERR of type STDERR is used by the PL/1 program to echo back diagnostic messages.
- DD=STDOUT of type STDOUT is used by the PL/1 program to write out the results of the query.

A time out of 2 seconds has been specified for all RSH Coupler files to limit the possibility of an end user to hang on the transaction.

The execution logic of **RSH_ridcams** is as follows:

1. Open the VSAM KSDS file
2. Call RSH Coupler procedure **ICDRNXT**.
3. If return code from ICDRNXT is non zero close the VSAM data set and terminate.
4. Read the search key from DD=STDPARM.
5. Check if the key is a valid key. If key is not valid echo error message and GO TO step 7.
6. Extract all records with a matching key from the VSAM data set and write them out to DD=STDOUT.

7. Close all RSH Coupler files.
8. GO TO step 2.

Before you can test this example you must do the following:

- Create and load the PHONE data base using member RPHONDEF in the RSH Coupler JCL distribution library. The data to be loaded is part of the JOB as in stream input. You can change it with whatever data you choose.
- Compile the PL/1 program stored in member ICDRPHON of the RSH Coupler JCL distribution library.
- Define the RSH Coupler APPC transaction using member ICDRPHOA.

RSH_rphone can be used to list all employees whose last name starts with character 'W'. The PC end user will type:

```
rsh mvs_host -l mvs_userid rphone w*
```

The UNIX end user will type:

```
rsh mvs_host -l mvs_userid rphone "w*"
```

Note: Double quotes are necessary in UNIX since otherwise character `'*'` has a special meaning for the UNIX shell.

Member ICDRPHOA listing is shown in Figure 25:

```
//ICDRPHOA JOB   (),
//         CLASS=A,
//         MSGCLASS=D,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//         REGION=1M,
//         TIME=(1)
//*
//*-------------------------------------------------------------------
//*     LICENSED MATERIAL - PROPERTY OF IBM
//*     5655-A99 (C) COPYRIGHT IBM 1998
//*     (C) COPYRIGHT FOLIUM INC. 1994, 1998
//*     ALL RIGHTS RESERVED
//*     US GOVERNMENT USERS RESTRICTED RIGHTS -
//*     USE, DUPLICATION OR DISCLOSURE RESTRICTED
//*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*-------------------------------------------------------------------
//*
//*  Run this JOB to create Data Coupler RSH server APPC
//*  transaction "RSH_rphone".
//*
```

Figure 25. JCL Member ICDRPHOA (Part 1 of 3)

```
//*  Before running do the following:
//*
//*  1. Transaction CLASS is set to A. Change it if necessary
//*
//*  2. Account tailoring is set to YES
//*     Change to NO if necessary
//*     If set to NO make sure ACCOUNT number is on transaction JCLs
//*
//*  3. Change  uuuuuu  to the MVS USERID to be used as generic
//*     USERID
//*
//*  4. Add transaction JOB cards
//*
//*  5. Change  llllll  to the name of the load library which
//*     contains the load module RPHONE
//*
//*  6. Change  dddddd  to the VSAM file name used as PHONE data base
//*
//*-------------------------------------------------------------------
//*
//*
//TPADD    EXEC PGM=ATBSDFMU
//*
//SYSSDLIB DD DISP=SHR,DSN=SYS1.APPCTP
//SYSPRINT DD SYSOUT=*
//SYSSDOUT DD SYSOUT=*
//SYSIN    DD DATA,DLM=$$
  TPADD
    TPNAME(RSH_rphone)
    ACTIVE(YES)
    TPSCHED_DELIMITER(##)
      CLASS(A)
      TAILOR_ACCOUNT(YES)
      TPSCHED_TYPE(MULTI_TRANS)
      GENERIC_ID(uuuuuu)
      KEEP_MESSAGE_LOG(ERROR)
      JCL_DELIMITER(@@)

//*---> Put APPC JOB cards here (include time limit !!!)
//*
//RPHONE   EXEC PGM=RPHONE
//*
//STEPLIB  DD DISP=SHR,DSN=llllll
//         DD DISP=SHR,DSN=SYS1.PLILINK
//         DD DISP=SHR,DSN=SYS1.SIBMLINK
//*
```

Figure 25. JCL Member ICDRPHOA (Part 2 of 3)

```
//STDPARM  DD SUBSYS=(IORR,ICDROPCL,STDPARM,,UPPER),
//            DCB=(RECFM=V,LRECL=259,BLKSIZE=263)
//STDERR   DD SUBSYS=(IORR,ICDROPCL,STDERR,'Err: '),
//            DCB=(RECFM=F,LRECL=81,BLKSIZE=81)
//STDOUT   DD SUBSYS=(IORR,ICDROPCL,STDOUT),
//            DCB=(RECFM=F,LRECL=81,BLKSIZE=81)
//*
//PHONLST  DD DISP=SHR,DSN=dddddd
//*
//SYSUDUMP DD SYSOUT=*
@@
##
$$
/*
```

Figure 25. JCL Member ICDRPHOA (Part 3 of 3)

Member ICDRPHON listing is shown in Figure 26:

```
/*-------------------------------------------------------------------*/
/*     LICENSED MATERIAL - PROPERTY OF IBM                           */
/*     5655-A99 (C) COPYRIGHT IBM 1998                               */
/*     (C) COPYRIGHT FOLIUM INC. 1994, 1998                          */
/*     ALL RIGHTS RESERVED                                           */
/*     US GOVERNMENT USERS RESTRICTED RIGHTS -                       */
/*     USE, DUPLICATION OR DISCLOSURE RESTRICTED                     */
/*     BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*  This PL/1 program is a Data Coupler MULTI-TRAN RSH server       */
/*  transaction which allows users on client machines to query      */
/*  the Phone data base by Last Name.                                */
/*                                                                   */
/*  Edit member ICDRPHOD to run the JOB which creates and            */
/*  load the Phone data base and memeber ICDRPHOA to define          */
/*  the MULTI-TRAN APPC transaction.                                 */
/*                                                                   */
/*  Compile and link this program using the PLIXCL proc              */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*  Example: get all entry with last name starting with "W"          */
/*           the user on the UNIX client machine types:              */
/*                                                                   */
/*    rsh mvs_host -l mvs_uid rphone "w*"                            */
/*                                                                   */
/*           the user on one of the DOS network OSes types:          */
/*                                                                   */
/*    rsh mvs_host -l mvs_uid rphone w*                              */
/*                                                                   */
/*-------------------------------------------------------------------*/
```

Figure 26. JCL Member ICDRPHON (Part 1 of 5)

```
Rphone: PROC options(main);

/*------------------------------------------------------------------*/
/*  Declare variables used as paramters to ICDRNXT                  */
/*------------------------------------------------------------------*/

 Dcl    token      fixed bin(31)  init(0);
 Dcl    uid        char(8);
 Dcl    rc         fixed bin(31);

/*------------------------------------------------------------------*/
/*  Declare variables used to read and manipulate user input        */
/*------------------------------------------------------------------*/

 Dcl    l          fixed bin(31);
 Dcl    data_flag  bin(1);
 Dcl    io_flag    bin(1);

 Dcl    parms      char(255) var;
 Dcl    data       char(80);

/*------------------------------------------------------------------*/
/*  Declare INPUT and OUTPUT RSH server files                       */
/*                                                                  */
/*  STDPARM is used to read the user paramter input                 */
/*  STDERR  is used for error messages                              */
/*  STDOUT  is used for data output                                 */
/*                                                                  */
/*  STDIN   is not used                                             */
/*                                                                  */
/*  Note:   see memeber RPHOAPPC for the APPC definition of the     */
/*          MUTLI-TRAN JOB                                          */
/*------------------------------------------------------------------*/

 Dcl    stdparm    file record input;
 Dcl    stdout     file stream output;
 Dcl    stderr     file stream output;

/*------------------------------------------------------------------*/
/*  Declare the VSAM file used as Phone data base                   */
/*------------------------------------------------------------------*/

 Dcl    phonlst    file input sequential keyed
                   env(indexed genkey);
/*------------------------------------------------------------------*/
/*  Declare ICDRNXT as dynamically linked entry                     */
/*------------------------------------------------------------------*/

 Dcl    icdrnxt    entry options(asm inter retcode);

/*------------------------------------------------------------------*/
/*  Set conditions for RSH server files                             */
/*                                                                  */
/*  Whenever an ERROR condition is detected and io_flag is trues    */
/*   the connection has been lost                                   */
```

Figure 26. JCL Member ICDRPHON (Part 2 of 5)

```
 /*                                                                   */
 /*  If EOD for STDPARM is detected no paramters have been specified  */
 /*  by the user on the client machine                                */
 /*-------------------------------------------------------------------*/

  On     error begin;
           IF io_flag THEN
             GOTO Transaction_end;
           END;

  On     endfile(stdparm) begin;
           io_flag= '1'b;
           PUT file(stderr)
             edit('rphone <last_name<*>>') (a);
           io_flag= '0'b;

           GOTO Transaction_end;
           END;

/*-------------------------------------------------------------------*/
 /*  Set conditions for VSAM file                                     */
 /*-------------------------------------------------------------------*/

  On     endfile(phonlst) data_flag= '0'b;
  On     key(phonlst) data_flag= '0'b;

 /*-------------------------------------------------------------------*/
 /*  Do initialization under the GENERIC UID                          */
 /*-------------------------------------------------------------------*/

  display('*==> JOB Rphone started');
  fetch icdrnxt;
  open file(phonlst) sequential keyed;

  DO WHILE('1'B);
 /*-------------------------------------------------------------------*/
 /*  Call ICDRNXT: if rc ∧= 0 time to clean up and terminate          */
 /*-------------------------------------------------------------------*/

    CALL icdrnxt(token, uid, rc);
    IF rc ∧= 0 THEN DO;
      display('*==> Rc from ICDRNXT is:'||rc);
      close file(phonlst);
      exit;
      END;
```

Figure 26. JCL Member ICDRPHON (Part 3 of 5)

```
/*-------------------------------------------------------------------*/
/*  Open RSH server files                                            */
/*-------------------------------------------------------------------*/

   io_flag= '0'b;

   open file(stdparm);
   open file(stderr);
   open file(stdout);

/*-------------------------------------------------------------------*/
/*  Read user paramters                                              */
/*-------------------------------------------------------------------*/

   io_flag= '1'b;
   read file(stdparm) into(parms);
   io_flag= '0'b;

   parms= substr(parms, 1, index(parms||' ', ' ')-1);

/*-------------------------------------------------------------------*/
/*  Strip trailing blanks and check if input terminates with '*'     */
/*  character for generic search as opposed to exact                 */
/*-------------------------------------------------------------------*/

   l= length(parms);
   IF substr(parms, l, 1) = '*' THEN
     l= min(16, l-1);
   ELSE
    l = 16;
   IF length(parms) > l THEN
     parms= substr(parms, 1, l);
   ELSE
     parms= parms||repeat(' ', l-length(parms));

/*-------------------------------------------------------------------*/
/*  Issue first read and go into a loop until no matches are found   */
/*-------------------------------------------------------------------*/

   data_flag= '1'b;
   read file(phonlst) into(data) key(parms);
   IF ∧ data_flag THEN DO;
     put file(stderr) skip
       edit('*** No matching data found') (a);
     GOTO Transaction_end;
     END;

   io_flag= '1'b;
   PUT file(stdout) skip edit(
     '-Last Name -------M -First Name -----'||
     '-Phone -------Info ------>') (a);
   io_flag= '0'b;

   DO until(∧ data_flag);
     IF substr(data, 1, l) = parms THEN DO;
       io_flag= '1'b;
       put file(stdout) skip edit(' '||data) (a);
       io_flag= '0'b;

       read file(phonlst) into(data);
       END;
     ELSE
       data_flag= '0'b;
     END;
```

Figure 26. JCL Member ICDRPHON (Part 4 of 5)

```
   io_flag= '1'b;
   PUT file(stdout) skip edit('-End of Data ---') (a);
   io_flag= '0'b;

/*-------------------------------------------------------------------*/
/*  All done. Close RSH server files and restart the main loop       */
/*-------------------------------------------------------------------*/

Transaction_end:
   close file(stdparm);
   close file(stderr);
   close file(stdout);
   END;

 END;
```

Figure 26. JCL Member ICDRPHON (Part 5 of 5)

Chapter 9. Performance

RSH Coupler performance is affected by the following factors:

- The lack of MVS resources for RSH Coupler gateway.
- The TCP/IP network bandwidth available between the client machine and MVS.
- The availability of APPC resources on MVS when a RSH Coupler APPC transaction is executed.

RSH Coupler Gateway

The RSH Coupler gateway will need to run with a dispatching priority slightly lower than the TCP/IP address space. If the priority is too low, RSH Coupler will suffer CPU shortages. This translates in a slow response time for all RSH Coupler APPC transactions.

Network Bandwidth

Network bandwidth should not be a problem if the client machine and MVS are connected via one or more LANs which use Ethernet or Token Ring.

However if the UNIX or PC are connected via a slow telephone line (equal or less than 28.8 Kbits per second) using SLIP or PPP, problems can arise particularly when input data is very large.

When SLIP and PPP is used, input data for STDIN is transmitted in packets of about 512 bytes or less, as opposed to the typical size of 8 Kbytes or more used for LANs. The RSH Coupler gateway receives these packets one at a time and ships them to the RSH Coupler APPC transaction as they are. Since packets are too small, the CPU overhead for receiving and sending them will dramatically increase the CPU consumption for the RSH Coupler gateway and the RSH Coupler APPC transaction.

The RSH Coupler gateway always prints the average packet size statistic in the log file(s) for every transaction (message ICDRP34I). This message should always be checked first if performance problems appear.

APPC Resources

All RSH Coupler APPC transactions are executed by the standard APPC scheduler. Most performance problems with RSH Coupler will happen for lack of APPC resources.

The most important APPC resources are the APPC initiators. Initiators should be already started when one is needed to execute an RSH Coupler APPC transaction.

Parameters **MIN** and **MAX** in the APPC class definition control the minimum and maximum number of initiators for a given APPC class. The value for **MIN** should *never be set to zero,* since in such a case every time an RSH Coupler APPC transaction is executed, an APPC initiator will be started and stopped.

For other APPC resources please refer to: *MVS/ESA Planning: APPC Management*.

Appendix A. IBM Network Data Couplers Messages

This appendix describes the messages and return codes for IBM Network Data Couplers for MVS/ESA and OS/390. For a consolidated list of messages for IBM Network Data Couplers, see "List of Messages" on page 103.

Data Coupler Messages

All NFS Coupler messages are of the form:

CCCCCNT Text of message

or

CCCCCNNT Text of message

The first five letters `CCCCC` indicate the component issuing the message; `N` or `NN` is a one or two digits id; `T` is the message type.

The component `CCCCC` can have the following values:

Value	Component issuing the message
ICDNA	NFS Coupler SNS/TCPAccess routines (INTERLINK only).
ICDNI	NFS Coupler IUCV routines (IBM only).
ICDNM	NFS Coupler NFS routines.
ICDNR	NFS Coupler RPC routines.
ICDNU	NFS Coupler IEBGENER checkpoint exit
ICDNX	NFS Coupler XDR routines.
ICDRI	RSH Coupler I/O handler routine.
ICDRC	RSH Coupler gateway main data communication routine.
ICDRG	RSH Coupler main gateway routine.
ICDRL	RSH Coupler gateway log writer routine.
ICDRP	RSH Coupler gateway process data communication routine.
ICDRS	RSH Coupler gateway security routine.
ICDRV	RSH Coupler gateway IBM TCP/IP interface routine.
ICDRA	RSH Coupler gateway INTERLINK TCP/IP interface routine.
ICDSA	IORR Allocation routine.
ICDSC	IORR Close routine.
ICDSH	IORR Checkpoint routine.
ICDSI	IORR Subsystem initialization routine.
ICDSO	IORR Open routine.
ICDSR	IORR Restart routine.
ICDSU	IORR modules Add Subsystem and Replace Subsystem.

The message type `T` can have the following values:

Value	Message type
I	Information only.

Value	Message type
W	Warning.
E	Error.

RSH Coupler writes messages in four different logs:

- The MVS system log during subsystem initialization at IPL.
- The RSH Coupler gateway JOB log.
- The RSH Coupler log files (trace messages).
- The RSH Coupler APPC transaction log. These messages are available only when APPC is directed to store the RSH Coupler APPC transaction log into a file (see APPC TPADD parameters).
- Standard error of the `rsh` command invoked by the user on the client machine.

Network Data Coupler Abend Codes

In most cases, when the NFS or RSH Couplers detect an error, they do not issue an abend but returns an error code to the MVS I/O routines after issuing one or more messages. The MVS I/O routine will eventually issue an abend depending on how severe the error is.

However, in some instances, the NFS or RSH Coupler I/O interfaces will abend the program after issuing a message so a dump can be obtained at the right moment. NFS Coupler uses abend user code `3333`. When the RSH Coupler gateway encounters a condition that does not allow it to continue to operate, it also will abend with user code 3333 after issuing one or more operator messages.

NFS Return Codes

In many NFS Coupler diagnostic messages the NFS return code is reported. This code is passed back from the NFS server on the UNIX host to NFS Coupler. When non zero, it indicates an error of some sort. The possible values for the NFS return code are show in Table 3.

Table 3. NFS Return Code Values

Value (Decimal)	Value (Mnemonic)	Meaning
`0`	`NFS_OK`	NFS call completed OK
`1`	`NFSERR_PERM`	The caller does not own the file or directory specified.
`2`	`NFSERR_NOENT`	The file or directory specified does not exist or is not defined in the `/etc/exports` file.
`5`	`NFSERR_IO`	The I/O operation on the NFS server failed (e.g. hard disk failure).
`6`	`NFSERR_NXIO`	Device or address specified does not exist.

Table 3. NFS Return Code Values (continued)

Value (Decimal)	Value (Mnemonic)	Meaning
13	NFSERR_ACCESS	Caller is not authorized for the request. That is the Uid and Gid used by NFS Coupler do not have enough authorization. When the request is mount the mount path is not defined in file /etc/exports on the NFS server or the mount path is defined but the MVS host is not included in the access list.
17	NFSERR_EXIST	File or directory specified already exists.
19	NFSERR_NODEV	Device specified does not exist.
20	NFSERR_NOTDIR	File specified is not a directory.
21	NFSERR_ISDIR	File specified is a directory.
27	NFSERR_FBIG	File specified is too large. The NFS server limits have been exceeded.
28	NFSERR_NOSPC	File specified is too large. There is not enough space on the NFS server file system.
30	NFSERR_ROFS	File system can be accessed as read only. A write has been attempted.
63	NFSERR_NAMETOOLONG	File name or mount path specified is too long.
66	NFSERR_NOTEMPTY	Directory specified is not empty and it cannot be removed.
69	NFSERR_DQOUT	MVS host disk quota on the NFS server has been exceeded.
70	NFSERR_STALE	NFS fhandle passed by NFS Coupler is not valid any more. This can happen if a UNIX file is removed while it is read or written by NFS Coupler.
99	NFSERR_WFLUSH	The NFS server's write cashe used in the WRITECACHE call was flushed to disk (not applicable for NFS Coupler).

List of Messages

ICDNA00E Macro=mmmmmmmm failed, CB Addr=aaaaaaaa Ret. Code=cccccccc, Rec. Code=rrrrrrrr, EWord=xxxxxxxx

Explanation: The TCP ACCESS macro mmmmmmmm returned error code cccccccc and recovery code rrrrrrrr. The related control block is at address aaaaaaaa and the content of its error code word is xxICD. The reason for the error could be an environmental problem or a software error.

User Response: Make sure that the TCP ACCESS subsystem is up and running, and resubmit the JOB. If the problem persists, Call IBM for support.

ICDNA01E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that the TCP ACCESS subsystem is up and running, and and resubmit the JOB. If the error persists, call IBM for support.

ICDNA30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNA31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI00E IUCV CALL ERROR, FUNCTION CODE=ffffffff, RETURN CODE=rrrrrrrr IUCV PARAMETER BLOCK IS:

Additional Message Text:

```
OFFSET   ----- DATA ------
+0000    xxxxxxxx xxxxxxxx
+0008    xxxxxxxx xxxxxxxx
+0010    xxxxxxxx xxxxxxxx
+0018    xxxxxxxx xxxxxxxx
+0020    xxxxxxxx xxxxxxxx
```

Explanation: An IUCV call failed with function code `ffffffff` (hex) and return code `rrrrrrrr` (hex). The content of the parameter block is dumped in hexadecimal format.

User Response: Call IBM for support.

ICDNI01I IUCV INTERRUPT BUFFER IS:

Additional Message Text:

```
OFFSET   ----- DATA ------
+0000    xxxxxxxx xxxxxxxx
+0008    xxxxxxxx xxxxxxxx
+0010    xxxxxxxx xxxxxxxx
+0018    xxxxxxxx xxxxxxxx
+0020    xxxxxxxx xxxxxxxx
```

Explanation: The message is issued to dump the IUCV interrupt buffer. This message is issued after an error message.

User Response: check the previous IUCV error message and follow its directions.

ICDNI02E CONNECTION COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV CONNECTION COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI04E CONNECTION SEVERED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV CONNECTION SEVERED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI05E MESSAGE COMPLETED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI06E MESSAGE COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI07E IPAUDIT FIELD IN INTERRUPT BUFFER SHOWS UNRECOVERABLE ERROR

Explanation: The IPAUDIT field in the IUCV interrupt field is invalid. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI08E INVALID CODE ENCOUNTERED IN IUCV EXIT ROUTINE

Explanation: An invalid code was returned by IUCV. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI10E CONNECTION SEVERED INTERRUPT, USER DATA=dddd

Explanation: The TCP/IP address space severed the IUCV connection with NFS Coupler. User data `dddd` is an EBCDIC string which indicates the reason for severing the connection.

User Response: check that the TCP/IP started task was not terminated. If the TCP/IP started task is up and running and the error reoccurs, call IBM for support.

ICDNI11E RECVFROM CALL FAILED, ERRNO=dddd

Explanation: The TCP/IP RECVFROM call failed with SOCKET return code `dddd` decimal.

User Response: Call IBM for support.

ICDNI12E PERMANENT ERROR FOUND IN IUCV ANCHOR BLOCK: rrrrrrrr

Explanation: The permanent error flag was found in the main IUCV control block. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI13E CRITICAL SECTION EXECUTION NOT COMPLETED: rrrrrrrr

Explanation: Execution of the IUCV critical section was not completed during the last NFS Coupler call. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI14E IUCV INTERFACE ALREADY IN USE BY ANOTHER TASK IN THE ADDRESS SPACE

Explanation: The application is a multitasking application and multiple NFS Coupler files are being opened concurrently by more than one task. NFS Coupler is unable to support concurrent opens of multiple files from more than one task due to limitations of the IBM TCP/IP product.

User Response: Modify your application so that concurrent opens of NFS Coupler files is from one task only (see Limitations on page 51).

ICDNI15E MAXIMUM NUMBER OF IUCV CONNECTIONS EXCEEDED

Explanation: The application is opening concurrently too many NFS Coupler files (more than 256).

User Response: Modify your application so that no more than 256 NFS Coupler files are open at the same time (see Limitations on page 51).

ICDNI16E VMCF SubSystem NOT FOUND

Explanation: The VMCF SubSystem and the IBM TCP/IP product do not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI17E VMCF SubSystem CVT INVALID, BAD EYE CATCHER FOUND

Explanation: The VMCF SubSystem was found but its CVT has a bad eye catcher. Most probably the IBM TCP/IP product does not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI18E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Gethostname failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI19E SOCKET CALL FAILED, ERRNO=dddd

Explanation: Socket call Socket failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI20E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Bind failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI22E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that TCP/IP is working properly and resubmit the JOB. If the error persists, call IBM for support.

ICDNI25E CLOSE CALL FAILED, ERRNO=dddd

Explanation: Socket call Close failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI28E SELECT CALL FAILED, ERRNO=dddd

Explanation: Socket call Select failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI29E SENDTO CALL FAILED, ERRNO=dddd

Explanation: Socket call Sendto failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNM00E DD=dddddddd, Load Module ″ICDNMAP″ was no loaded from an authorized library

Explanation: Module ICDNMAP was not loaded from an authorized library. OPEN of file `dddddddd` will fail.

User Response: make sure module ICDNMAP has been copied into an authorized library which is part of the LINKLIST concatenation. Also make sure that no module with name ICDNMAP exists in either the JOBLIB or the STEPLIB concatenation.

ICDNM00I DD=dddddddd, READ_BIN, Last record padded with ″00″x

Explanation: The last record of a NFS Coupler file with fixed record format has been padded with ″00″x characters, since the length of the file read is not an exact multiple of the record length.

User Response: none.

ICDNM01E DD=dddddddd, READ_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in UNIX format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM02E DD=dddddddd, READ_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of binary characters from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM03E DD=dddddddd, WRITE_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of binary characters to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM05E DD=dddddddd, WRITE_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for UNIX to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM08E DD=dddddddd, LIST_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for directory entries to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM09E DD=dddddddd, READ_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for a UNIX file attribute to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM10E DD=dddddddd, MOUNT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request to mount a directory failed. The MOUNT server returned NFS error code `nnnnnnnn` mnemonics `ccc` decimal. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM11E DD=dddddddd, File= ″ffff″ is not a regular file

Explanation: NFS Coupler will not access UNIX file `ffff` because it is not a regular file. File `dddddddd` was opened for output or for input with an NFS Coupler input disposition of `FILE`. The OPEN for file `dddddddd` will fail.

User Response: for file `dddddddd` either change the UNIX file path to indicate a regular file or change the NFS Coupler input disposition to `DIR` to read the contents of a UNIX directory.

ICDNM12E DD=dddddddd, File= ″ffff″ already exists

Explanation: NFS Coupler refuses to write to UNIX file `ffff` because it already exists. File `dddddddd` was open with an NFS Coupler output disposition of `NEW`. The OPEN for file `dddddddd` will fail.

User Response: either change the NFS Coupler output disposition or change the UNIX file path in the JCL for file `dddddddd`.

ICDNM13E DD=dddddddd, Unable to look up file= ″ffff″, File= ″oooo″ is not a directory

Explanation: NFS Coupler was not able to look up file `ffff` under directory `oooo` because file `oooo` is not a directory. File `ffff` is an entry in the UNIX file name path specified. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd`.

ICDNM14E DD=dddddddd, File= ″ffff″ is not a directory

Explanation: NFS Coupler will not access file `ffff` since it is not a directory. File `dddddddd` was opened for input with NFS Coupler input disposition `DIR`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or change NFS Coupler input disposition to `FILE`.

ICDNM16E DD=dddddddd, LOOKUP failed, File= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX file `ffff`. UNIX file `ffff` is the last entry in the UNIX file path specified in the JCL for file `dddddddd`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX file `ffff` already exists.

ICDNM17E DD=dddddddd, LOOKUP failed, Directory= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX directory `ffff`. UNIX directory `ffff` is an intermediate entry in the UNIX file path specified in the JCL for file `dddddddd`. File `dddddddd` was opened for input. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX directory `ffff` already exists.

ICDNM18E DD=dddddddd, RESET LENGTH failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to reset the length of UNIX file `ffff` to zero. UNIX file `ffff` is the last entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. File `dddddddd` was opened for output. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM19E DD=dddddddd, LOOKUP failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to get a file handle of UNIX file `ffff`. UNIX file `ffff` is an entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM20E DD=dddddddd, CREATE FILE failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX file ffff. UNIX file ffff is the last entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDRC00I Main communication task started

Explanation: This message is issued by the gateway at start up.

ICDRC02W ip_addr:port1, unable to fork process

Explanation: The gateway is unable to start a new thread for a new connection from client with IP address and port ip_addr:port1. The new TCP/IP connection is closed. .note Action: increase the maximum number of threads for the gateway.

ICDRC03I ip_addr:port1, accepted, process ID=nnn

Explanation: A new connection has been accepted from client with IP address and port ip_addr:port1. The thread ID is nnn.

ICDRC04I ip_addr:port1, refused, port is not reserved

Explanation: A new connection from client with IP address and port ip_addr:port1 has been refused (closed) since the client port is not a reserved port. This is a rsh protocol violation.

User Response: Make sure that the program connecting from the client machine uses the rshprotocol.

ICDRC05I Conn= ip_addr:port1[.port2] Cmd=[ccc]

Explanation: This message is issued by the gateway in response to the DISplay command entered by the operator. One message is issued per active TCP/IP connection. The client IP address ip_addr, and ports port1 and port2 are followed by the command string command_string. If port2 and command string are not shown the secondary connection is in progress.

ICDRC06I total number of current connections=nnn

Explanation: This message is issued by the gateway in response to the DISplay command and it summarizes the total number of current TCP/IP connections.

ICDRC07I Communication task terminating now

Explanation: This message is issued by the gateway when it terminates.

ICDRC08I connection(s) still active, waiting for 60 seconds grace period

Explanation: This message is issued by the gateway in response to the STOP command. Since one or more TCP/IP connections are active it will wait for 60 seconds before shutting down.

ICDRC09E Port 514 is already in use.

Explanation: Reserved port 514 is already in use.

User Response: Use the the TCP/IP netstatcommand to find out which JOB or started task is using port 514, stop it and restart the ICDRSTRT gateway.

ICDRG01E One or more tasks terminated when not expected

Explanation: A software error has occurred in the gateway.

User Response: Call IBM for support.

ICDRG02I LOG task ECB=nnnnnn

Explanation: This is a follow up message of RSHG002E. The termination ECB for the logger task is nnnnnn

User Response: Call IBM for support.

ICDRG03I SEC task ECB=nnnnnn

Explanation: This is a follow up message of RSHG002E. The termination ECB for the security task is nnnnnn

User Response: Call IBM for support.

ICDRG04I COM task ECB=nnnnnn

Explanation: This is a follow up message of RSHG002E. The termination ECB for the communication task is `nnnnnn`

User Response: Call IBM for support.

ICDRG05W COM task ECB=nnnnnn

Explanation: An operator entered an invalid command for the gateway.

User Response: Reenter the correct command.

ICDRG06I Termination in progress

Explanation: The gateway address space has started termination processing.

ICDRG07I Termination completed successfully

Explanation: The gateway address space has finished termination processing.

ICDRG08E Trial license has expired. Call IBM for support.

Explanation: Trial license for RSH Coupler has expired.

User Response: Call IBM for support.

ICDRG09E License not available on this machine. Call IBM for support.

Explanation: License for RSH Coupler has not been purchased for the current machine.

User Response: Call IBM for support.

ICDRG10E Trial license key is corrupt. Call IBM for support.

Explanation: Corrupt data has been found in the license key memory location.

User Response: Call IBM for support.

ICDRG11E TCP/IP started task name is invalid

Explanation: The TCP/IP started task name specified to the gateway is not a valid JOB name.

User Response: Change the name and restart the gateway.

ICDRG12E Log data sets SYSOUT class is invalid

Explanation: The SYSOUT class specified for the gateway log files is an invalid character.

User Response: Change the SYSOUT class for the log data sets and restart the gateway.

ICDRG13E Transaction prefix is invalid

Explanation: The RSH Coupler APPC transaction prefix specified to the gateway is either too long, null or from an invalid character set.

User Response: Change the RSH Coupler APPC transaction prefix and restart the gateway.

ICDRG14E Maximum number of THREADS is invalid

Explanation: The maximum number of threads specified to the gateway is either larger than 254 or an invalid integer.

User Response: Change the maximum number of threads and restart the gateway.

ICDRG15E No or invalid CIB found

Explanation: The gateway was unable to locate the CIB or the CIB contains invalid data.

User Response: Call IBM for support.

ICDRG16W Communication task terminated too early

Explanation: A software error was detected during gateway termination.

User Response: Call IBM for support.

ICDRG27I LOG data sets SYSOUT clas is c

Explanation: The SYSOUT class used by the RSH Coupler gateway for its LOG data sets is `c`.

User Response: None.

ICDRG28I APPC transaction prefix is ......ppp

Explanation: The prefix that the RSH Coupler gateway uses to form the APPC transaction names is `ppp`.

User Response: None.

ICDRG29I Maximum number of THREADS is ....nnn

Explanation: The maximum number of concurrent `rsh` requests allowed by the RSH Coupler gateway is `nnn.`

User Response: None.

ICDRI01E DD=ddd, TYPE=ttt, Open of @ICDRDD@ failed

Explanation: The RSH Coupler I/O routines tried to open DD=@ICDRDD@, but open failed. Open for file with DD name `ddd` and type `ttt` will fail.

User Response: Add the DD statement for DD=@ICDRDD@ to the RSH server APPC transaction JCLs.

ICDRI02W DD=ddd, TYPE=STDPARM , Cmd parms too long, truncation occurred

Explanation: The parm string specified by the user on the client machine is longer than the record length for STDPARM. The record has been truncated and the I/O will fail. I/O request for DD name `ddd` and file type STDPARM will fail.

User Response: Use as shorter parm string if possible, or use a longer record length for STDPARM.

ICDRI03W DD=ddd, TYPE=STDIN , One or more records have lengths shorter than the EOF token

Explanation: One or more input records for DD name `ddd` and file STDIN is shorter than the End Of File token specified in the JCLs. These record are not checked for End Of File condition.

User Response: If the end of file is not detected when expected, check input data to make sure the `eof_token`is present.

ICDRI04I DD=ddd, TYPE=STDIN , EOF token detected

Explanation: The record being read has been found to contain the End Of File token. The End Of File condition will be returned to the caller for DD name `ddd` and file type STDIN.

ICDRI05E DD=ddd, TYPE=ftype, Time out while waiting for APPC call=aaaa

Explanation: Time out specified or defaulted in the JCLs for file with DD name `ddd` and type `ttt` was exceeded while waiting for completion of APPC call `aaa.` The I/O operation in progress will fail.

User Response: Check for the reason of the time out. Two of the reasons could be problems with the TCP/IP network or the user on the client machine not entering input within the time allowed.

ICDRI06E DD=ddd, TYPE=ftype, APPC error detected, APPC function=aaa, Rc=ccc

Explanation: A bad error code was returned by APPC function DD name `aaa.` The return code is `ccc.` The I/O request in progress for file with DD name `ddd` and type `ttt` will fail.

User Response: Make sure the RSH Coupler APPC transaction was invoked by the RSH Coupler gateway. If this is the case call IBM for support.

ICDRI07E DD=ddd, TYPE=ftype, Protocol violation

Explanation: A protocol violation between the RSH Coupler gateway and the RSH Coupler APPC transaction was detected. The I/O request in progress for file with DD name `ddd` and type `ttt` will fail.

User Response: Make sure the RSH Coupler APPC transaction was invoked by the RSH Coupler gateway. If this is the case call IBM for support.

ICDRI08E DD=ddd, TYPE=ftype, Unable to lock shared buffer

Explanation: The RSH Coupler I/O routine is unable to lock the RSH Coupler APPC transaction shared buffer. This occur when the application issues multiple concurrent I/O requests for RSH Coupler files. The I/O request in progress for file with DD name `ddd` and type `ttt` will fail.

User Response: Modify the application to avoid multiple concurrent I/O requests for RSH Coupler files (e.g. by using END/DEQ).

ICDRI10E DD=ddd, TYPE=ftype, Exiting due to severe error

Explanation: A severe error was encountered. A previous message indicates the type of error. The I/O request in progress for file with DD name `ddd` and type `ttt.` will fail.

ICDRI11E DD=ddd, TYPE= , Invalid number of arguments

Explanation: The number of arguments for keyword `SUBSYS=` in the JCLs for DD name `ddd` is invalid. Open will fail.

User Response: Fix the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI12E DD=ddd, TYPE= , File type is required

Explanation: File type has not been specified in the JCLs for DD name `ddd`. Open will fail.

User Response: Specify the file type and redefine the RSH Coupler APPC transaction.

ICDRI13E DD=ddd, TYPE= , File type is too long

Explanation: File type specified in the JCLs for DD name `ddd` is too long. Open will fail.

User Response: Correct the file type and redefine the RSH Coupler APPC transaction.

ICDRI14E DD=ddd, TYPE=ttt, File type invalid

Explanation: File type specified in the JCLs for DD name `ddd` is not valid. It must be either STDPARM, STDIN, STDERR or STDOUT. Open will fail.

User Response: Correct the file type and redefine the RSH Coupler APPC transaction.

ICDRI15E DD=ddd, TYPE=ttt, Opened for output, it must be opened for input

Explanation: File with DD name `ddd` and type `ttt` has been opened for output. The file (STDPARM or STDIN) can only be opened for input. Open will fail.

User Response: Either change the JCLs of the RSH Coupler APPC transaction or change the application.

ICDRI16E DD=ddd, TYPE=ttt, Opened for input, it must be opened for output

Explanation: File with DD name `ddd` and file type `ttt` has been opened for input. The file (STDERR or STDOUT) can only be opened for output. Open will fail.

User Response: Either change the JCLs of the RSH Coupler APPC transaction or change the application.

ICDRI17E DD=ddd, TYPE=STDIN , Invalid EOF token

Explanation: The End Of File token specified for file with DD name `ddd` and type STDIN is too long. It cannot be longer than 16 characters. Open will fail.

User Response: Change the EOF token in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI18E DD=ddd, TYPE=ttt, Invalid PREFIX token

Explanation: The PREFIX token specified for file with DD name `ddd` and type `ttt` is too long. It cannot be longer than 16 characters. Open will fail.

User Response: Change the PREFIX token in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI19E DD=ddd, TYPE=STDPARM , Invalid CASE

Explanation: The CASE specified for file with DD name `ddd` and type STDPARM is neither ″UPPER″ nor ″MIXED″. Open will fail.

User Response: Change the CASE in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI20E DD=ddd, TYPE=ttt, Invalid DATA TYPE

Explanation: The data type for file with DD name `ddd` and type `ttt` is neither ″ASCII″ nor ″BIN″ Open will fail.

User Response: Change the data type in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI21E DD=ddd, TYPE=ttt, Time out is too long

Explanation: Time out for file with DD name `ddd` and type `ttt` has more than 4 digits. Open will fail.

User Response: Specify a shorter time out in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI22E DD=ddd, TYPE=ttt, Time out is invalid

Explanation: Time out for file with DD name `ddd` and type `ttt` is not a valid unsigned integer. Open will fail.

User Response: Specify a valid time out in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI23E APPC Deallocate failed, Type=ttt, Rc= cccc

Explanation: A call to APPC deallocate failed with return code `cccc`. The type of deallocation `ttt` can be either `0` (normal deallocate) or `3` (deallocate abend). Processing continues. The user invoking the `rsh` command on UNIX or a PC will probably get message `RSHP019E`.

User Response: Make sure there are no problems outstanding with APPC on your installation. If the problem persists, call IBM for support.

ICDRI24E DD=ddd, TYPE=ttt, Length of print record is larger than 256

Explanation: The data portion of an output print record for file with DD name `ddd` and type `ttt` is longer than 256 bytes. The I/O operation will fail.

User Response: Make sure the file has no record with the data portion (carriage control character excluded) longer than 256 bytes or specify the `NOPRT`data type in the JCLs and redefine the RSH Coupler APPC transaction.

ICDRI24W Connection lost, Appc function=ffffff, Rc=rrr

Explanation: The TCP/IP connection between the JOB executing as server and the client has been lost while attempting APPC function `ffffff`. The return code from `ffffff` is `rrr`. The I/O request being execute, will fail unless it is an OPEN or a CLOSE.

User Response: None.

ICDRI25E DD=ddd, TYPE=ttt, Input record(s) truncated'

Explanation: One or more input record had to be truncated to fit into the maximum record length of `DD=ddd`. The input operation will fail with a standard MVS I/O abend.

User Response: Make sure that the longest line of the input data is not longer than the maximum record length for file `DD=ddd`.

ICDRI26W Stdin input line(s) too long, truncation occurred

Explanation: One or more standard input lines is longer than the maximum record length for DD=STDIN of the RSH server transaction.

User Response: Make sure that the longest line of standard input is not longer than the maximum record length defined for file type STDIN in the RSH server APPC transaction.

ICDRI72E DD=ddd, MVS Checkpoint/Restart not allowed

Explanation: The program invoked by RSH Coupler issued a checkpoint which is not supported by RSH Coupler. The checkpoint will fail.

User Response: Modify the program so that it will not issue a checkpoint.

ICDRI73W DD=ddd, Nested close request

Explanation: A close request was invoked while another I/O request was in progress for the same DCB of file `ddd`. Standard close processing is bypassed. .br If the close was issued in an asynchronous exit, such as timer or VTAM exit, the program issuing the close is in error. If the close was issued in an ESTAE recovery or retry routine after an abend during an I/O request to RSH Coupler, this message should be expected.

User Response: If this message is issued because of programming error, correct the program and submit again.

ICDRI74E DD=ddd, Nested input request detected

Explanation: An input request was invoked while another I/O request was in progress for the same DCB of file `ddd`. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: Correct the program and submit again.

ICDRI75E DD=ddd, Nested output request detected

Explanation: An output request was invoked while another I/O request was in progress for the same DCB of file `ddd`. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: Correct the program and submit again.

ICDRI77E DD=ddd, asynchronous check invoked

Explanation: The MVS SSAM routines requested an asynchronous check to RSH Coupler for file `ddd`. This is an error. The caller is abnormally terminated.

User Response: Call IBM for support.

ICDRL01I Log task started

Explanation: This is a message issued by the RSH Coupler gateway log task at start up.

ICDRL02I Log started

Explanation: This message is printed by the RSH Coupler gateway log task at the beginning of each new log file.

ICDRL03I Log closed, termination in progress

Explanation: This message is printed by the RSH Coupler gateway log task at the end of the current log file, when the RSH Coupler gateway terminates.

ICDRL04I Log task terminating normally

Explanation: This is a message issued by the RSH Coupler gateway log task at termination.

ICDRL05I Log closed, new log will be open

Explanation: This message is printed by the RSH Coupler gateway log task at the end of the current log file, when it switches to a new log file.

ICDRL06E Dynamic allocation of log file=fff has failed with Err.Code=eee, Info Code= iii

Explanation: The log file tried to allocate dynamically a new log file with DD name `fff`. The allocation (SVC 99) failed with Error code `eee` and info code `iii`.

User Response: Check the SVC 99 return codes and take proper action. If the problem persists call IBM for support.

ICDRL07I Log file DD=LOGxxxxx has been closed

Explanation: The log file with DD NAME LOGxxxxx, where `xxxxx` is a sequence number, has been closed. A new log file will be open unless RSH Coupler is terminating.

User Response: None.

ICDRP01W ip_addr:port1.port2, connect failed with errno=nnn

Explanation: The gateway tried to connect back to the client on port `port2`. The attempt failed with error number `nnn`. The client IP address is `ip_addr` and its primary port is `port1`. The primary connection is closed.

User Response: Try again since the problem might be temporary. If the problem persists call IBM for support.

ICDRP02W ip_addr:port1.port2, no reserved port available

Explanation: The gateway is unable to connect back to the `rsh` client with IP address `ip_addr` primary connection port `port1` and secondary connection port `port2`, since no free port is available in the range 512-1023. The primary connection is closed.

User Response: Try again since the problem should be temporary. If the problem persists, call IBM for support.

ICDRP03I ip_addr:port1.port2, remuser=rrr, locuser=lll

Explanation: The `rsh` client with IP address `ip_addr` primary connection port `port1` and secondary connection port `port2`, has a remote username of `rrr` and has requested a local user (MVS USERID) of `lll`. This message is issued to allow security auditing.

ICDRP04I ip_addr:port1.port2, cmd=ccc

Explanation: The `rsh` client with IP address `ip_addr` primary connection port `port1` and secondary connection port `port2`, specified the command and parm string `ccc` (truncated if necessary). This message is issued to assist application programmers in debugging.

ICDRP05W Command required

Explanation: The user on the client machine must provide a command when invoking the `rsh` command.

ICDRP06I ip_addr:port1.port2, command line is missing

Explanation: The `rsh` client with IP address `ip_addr` primary connection port `port1` and secondary connection port `port2`, did not specify a command. Primary and secondary connections are closed.

ICDRP07W Invalid character(s) found in command

Explanation: The user on the client machine specified a remote command with invalid characters.

ICDRP08I ip_addr:port1.port2, invalid character(s) found in command

Explanation: The user on the client machine with ip address `ip_addr`, primary connection port `port1`, secondary connection port `port2`, specified a remote command with invalid characters. Primary and secondary connections are closed.

ICDRP09E Login incorrect

Explanation: The MVS USERID specified by the user on the client machine was not found in the RSH Coupler RSHOSTS PDS or it is not defined to the MVS security subsystem.

ICDRP11E Permission denied

Explanation: The MVS USERID specified by the user on the client machine cannot be used by the client.

ICDRP12I ip_addr:port1.port2, "tttt" returned from security

Explanation: The security task was not able to create a security token for the client with IP address `ip_addr`, primary connection port `port1`, and secondary connection port `port2`. A reason string `tttt` was returned which can have one of the following values:

- `NOID` which indicates the MVS USERID specified by the `rsh` client was not found in the RHOSTS PDS.
- `NTKN` which indicates the security task was not able to create a security (RACROUTE) token.
- `NOUS` which indicates the MVS USERID specified by the `rsh` client was found in the RHOSTS PDS, but cannot be used for the `ip_addr` local username combination of the client.

Primary and secondary connections are closed.

ICDRP13E Invalid state in TCP_FLOW, ID=iiii

Explanation: A software error was detected in the gateway. String `iiii` identifies the caller. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRP14I ip_addr:port1.port2, normal termination

Explanation: Connection with client with IP address `ip_addr`, primary connection port `port1`, and secondary connection port `port2`, terminated normally.

ICDRP16I ip_addr:port1.port2, security not valid, APPC Rc=ccc

Explanation: The gateway was not able to start the command specified by the client with IP address `ip_addr`, primary connection port `port1`, secondary connection port `port2`, due to security reasons. The return code from APPC is `ccc`. Primary and secondary connections are closed.

User Response: Make sure that the MVS USERID defined in the RSHOSTS PDS is an existing and active MVS USERID.

ICDRP17E Command not found

Explanation: The command specified by the `rsh` client does not exist.

ICDRP18I ip_addr:port1.port2, command not found, APPC Rc=ccc

Explanation: The gateway was not able to start the command specified by the client with IP address `ip_addr`, primary connection port `port1`, secondary connection port `port2`, because the command (prefixed with the transaction prefix) is not defined to APPC. The return code from APPC is `ccc`. Primary and secondary connections are closed.

User Response: Make sure that the client specified a correct command or defined the missing command as a new RSH Coupler APPC transaction.

ICDRP19E Command terminated abnormally

Explanation: The command specified by the `rsh` client terminated abnormally (ABENDED).

User Response: Check the RSH Coupler APPC transaction log to have more information about the ABEND.

ICDRP20E ip_addr:port1.port2, command terminated abnormally, APPC Rc=ccc

Explanation: The command invoked by the client with IP address `ip_addr`, primary connection port `port1`, and secondary connection port `port2`, terminated abnormally (ABENDED). The return code from APPC is `ccc`. Primary and secondary connections are closed.

User Response: Check the RSH Coupler APPC transaction log (if available) to find out the reason for the ABEND.

ICDRP21E Communication failure

Explanation: Something went wrong between the RSH Coupler gateway and the RSH Coupler APPC transaction.

User Response: Try again and if the problem persists contact the RSH Coupler administrator for support.

ICDRP22E ip_addr:port1.port2, APPC unrecoverable error, Call=aaa, Rc=ccc

Explanation: APPC returned an unrecoverable error for connection of client with IP address `ip_addr`, primary connection port `port1`, and secondary connection port `port2`. The return code from APPC is `ccc`. Primary and secondary connections are closed.

User Response: If the problem persists, call IBM for support.

ICDRP23E sss too long

Explanation: One of the control string sss sent by the rsh command is too long. The value of sss is one of the following:

Port. This is the port for the secondary connection.

Local username.
This is the user name used on the client machine.

Remote USERID.
This is the MVS USERID requested by the client.

Command line.
This is the command line typed by the user on the client machine.

Command.
This is the command typed by the user on the client machine.

Command parms.
This is the parm string typed by the user on the client machine.

User Response: If sssis portmake sure that the program connecting from the client machine uses the correct rsh protocol. On all other cases retype the command correctly.

ICDRP24I ip_addr:port1[.port2], string too long, ID=sss

Explanation: The rsh command on the client with IP address ip_addr, primary connection port port1 and secondary connection port port2, sent control string sss which is too long. The value of sss can be:

Port. This is the port for the secondary connection.

Local username.
This is the user name used on the client machine.

Remote USERID.
This is the MVS USERID requested by the client.

Command line.
This is the command line typed by the user on the client machine.

Command.
This is the command typed by the user on the client machine.

Command parms.
This is the parm string typed by the user on the client machine. The primary connection is closed.

User Response: If sssis portcheck the correct software is used on the client machine. Otherwise the user must specifies parameters to the rshcommand which are within the limitations of RSH Coupler.

ICDRP25I ip_addr:port1[.port2], time out, ID=sss

Explanation: The rsh command on the client with IP address ip_addr, primary connection port port1, and secondary connection port port2, was closed due to time out on the primary connection. ID sss is a diagnostic string.

User Response: If the problem persists, call IBM for support.

ICDRP26W ip_addr:port1, invalid PORT=port for STDERR

Explanation: The rsh command on the client with IP address ip_addr and primary connection port port1 set an invalid port. The primary connection is closed.

User Response: Check that the right software is used on the client machine.

ICDRP27I ip_addr:port1.port2, signal received

Explanation: A signal byte was received from the rsh command on the client with IP address ip_addr, primary connection port port1 secondary connection port port2. The primary and secondary connections are closed.

ICDRP28I ip_addr:port1[.port2], connection lost

Explanation: A connection with client with IP address ip_addr primary connection port port1, and secondary connection port port2, has been lost. The remaining connection is closed.

User Response: Check that there are no network problems between the client machine and MVS.

ICDRP29E Protocol violation, command is not a valid rsh command

Explanation: The gateway detected a protocol violation with the RSH Coupler APPC transaction. This is normally due to the fact that the command specified kicked an APPC transaction which was not created for RSH Coupler.

User Response: Make sure that the user on the client side does not invoke a command which is not an APPC transaction which was not created for RSH Coupler. The use of a transaction prefix on the RSH Coupler should preclude this from happening.

ICDRP30W ip_addr:port1[.port2], protocol violation, mmm

Explanation: The gateway detected a protocol violation with the RSH Coupler APPC transaction. The transaction was invoked by a user on a client with IP address `ip_addr`, primary connection port `port1`, and secondary connection port `port2`. Message `mmm` gives a detailed explanation. This is normally due to the fact that the command specified kicked an APPC transaction which was not created for RSH Coupler. The primary and secondary connection are closed.

User Response: If the APPC transaction being executed is a valid RSH Coupler APPC transaction, call IBM for support.

ICDRP31E ip_addr:port1.port2, bad return code from APPC deallocate, Rc=ccc

Explanation: The gateway deallocated the APPC conversation with the RSH Coupler APPC transaction which was invoked by a user on a client with IP address `ip_addr` primary connection port `port1`, and secondary connection port `port2`. The APPC return code is `ccc`. for RSH Coupler.

ICDRP32I ip_addr:port1.port2, waiting for APPC event to complete

Explanation: This is a debugging trace record.

ICDRP33I ip_addr:port1.port2, APPC event completed

Explanation: This is a debugging trace record.

ICDRP34I ip_addr:port1.port2, data flow statistics, in=iii (jjj), out=ooo (ppp), err=eee, (fff)

Explanation: Data flow statistics for standard input, output and error across the TCP/IP network are given for a connection with a client with IP address `ip_addr`, primary connection port `port1`, secondary connection port `port2`. This message is issued at connections termination. For standard input, `iii` is the total number of bytes transmitted, `jjj` is the average packet length. For standard output, `ooo` is the total number of bytes transmitted, `ppp` is the average packet length. For standard error `eee` is the total number of bytes transmitted, `fff` is the average packet length.

ICDRP35E Command cannot be executed

Explanation: The command specified by the `rsh` client exists but cannot be executed.

ICDRP36I ip_addr:port1.port2, command cannot be executed, APPC Rc=ccc

Explanation: The gateway was not able to start the command specified by the client with IP address `ip_addr`, primary connection port `port1`, secondary connection port `port2`, because the command (prefixed with the transaction prefix) cannot be started by APPC. The return code from APPC is `ccc`. Primary and secondary connections are closed.

User Response: Make sure that the APPC transaction specified is marked active and that all APPC resources it needs to be started have been defined and are available.

ICDRS01I Security task started

Explanation: This is a gateway message at start up.

ICDRS02I Module ICDRMAP has been reloaded

Explanation: This is a message issued by the gateway when the operator enter the REFresh command.

ICDRS03I Security task terminating normally

Explanation: This is a gateway message at termination.

ICDRS04E Racroute request=ttt failed

Explanation: The RACROUTE request `ttt` failed with a non zero return code. The gateway terminates abnormally.

User Response: If the MVS security subsystem is not RACF check that the proper security customization has been done with the installation of RSH Coupler. If the problem persists, call IBM for support.

ICDRV01E Bad RC or IPRCODE from IUCV SEVER, RC=ccc, IPRCODE=iii

Explanation: The IUCV function SEVERE failed with return code `ccc` and IPR code `iii`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV02E Bad RC from IUCV RTRVBFR, RC=ccc

Explanation: The IUCV function RTRVBFR failed with return code `ccc`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV03I TCP IUCVENV addr=aaa (DEC)

Explanation: This is an internal diagnostic message issued by the gateway at start up.

User Response: None.

ICDRV04E VMCF SubSystem does not exist

Explanation: The gateway was not able to locate the subsystem VMCF.

User Response: Install the IBM TCP/IP product on the MVS copy where the gateway executes.

ICDRV05E Invalid eye catcher found in VMCF subsystem CVT.

Explanation: The gateway found bad data in the VMCF subsystem CVT.

User Response: Make sure that the IBM TCP/IP product is properly installed on the MVS copy where the gateway executes.

ICDRV06E Bad RC or IPRCODE from IUCV DECLARE, RC=ccc, IPRCODE=iii

Explanation: The IUCV function DECLARE failed with return code `ccc` and IPRCODE `iii`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV07E Bad RC or IPRCODE from IUCV CONNECT, RC=ccc, IPRCODE=iii

Explanation: The IUCV function CONNECT failed with return code `ccc` and IPRCODE `iii`. The gateway terminates abnormally.

User Response: Check that the correct name of the TCP/IP started task was passed to the RSH Coupler gateway as a parameter. If the problem persists, call IBM for support.

ICDRV08E Function fff failed with errno=eee

Explanation: The TCP/IP function `fff` failed with error number `eee`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV09E Bad RC or IPRCODE from IUCV PURGE, RC=ccc, IPRCODE=iii, Function=fff

Explanation: The IUCV function PURGE failed with return code `ccc` and IPRCODE `iii`. The calling TCP/IP function is `fff`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV10E Bad RC or IPRCODE from IUCV PURGE, RC=ccc, IPRCODE=iii, Function=fff

Explanation: The IUCV function PURGE failed with return code `ccc` and IPRCODE `iii`. The calling TCP/IP function is `fff`. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV20E IUCV SEVERE or QUIESCE interrupt received, User data=ddd

Explanation: The IUCV interrupt SEVERE and QUIESCE has been received but it is not expected. User data in the interrupt buffer is `ddd`. The content of the entire interrupt buffer is dumped right after with message RSHT023I. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV21E IUCV IPAUDIT shows unrecoverable error

Explanation: The IUCV interrupt exit detected an error in the IPAUDIT field of the interrupt buffer. The content of the entire interrupt buffer is dumped right after with message RSHT023I. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV22E IUCV unexpected interrupt received

Explanation: The IUCV interrupt exit detected a type of interrupt which is unexpected. The content of the entire interrupt buffer is dumped right after with message RSHT023I. The gateway terminates abnormally.

User Response: Call IBM for support.

ICDRV23I IUCV interrupt buffer is:

Additional Message Text:

```
Offset   ----- Data ------
+0000   xxxxxxxx xxxxxxxx
+0008   xxxxxxxx xxxxxxxx
+0010   xxxxxxxx xxxxxxxx
+0018   xxxxxxxx xxxxxxxx
+0020   xxxxxxxx xxxxxxxx
```

Explanation: The IUCV interrupt buffer content is dumped in hexadecimal format due to a previous error.

User Response: Identify the error which triggered this message and take appropriate action.

ICDRV24I IUCV parm data at Loc=aaa

Additional Message Text:

```
Offset   ----- Data ------
+0000   xxxxxxxx xxxxxxxx
+0008   xxxxxxxx xxxxxxxx
+0010   xxxxxxxx xxxxxxxx
+0018   xxxxxxxx xxxxxxxx
+0020   xxxxxxxx xxxxxxxx
```

User Response: Identify the error which triggered this message and take appropriate action. aaais dumped in hexadecimal format due to a previous error.

ICDRV30I TCP TCPAENV addr=aaa (DEC)

Explanation: This is an internal diagnostic message issued by the gateway at start up.

User Response: None.

ICDRV31E Error in ffffff, TPL=nn, Macro=mmmmmm, Type=tttttt

Explanation: An error has been detected when issuing the TPL based macro mmmmmm. The socket function call is ffffff. Type tttttt indicate at which stage of processing it was detected. The gateway will terminate abnormally.

User Response: Call IBM for support.

ICDRV32E Error in ffffff, Macro=mmmmmm

Explanation: An error has been detected when issuing the APCB based macro mmmmmm. The socket function call is ffffff. The gateway will terminate abnormally.

User Response: Call IBM for support.

ICDRV33E Software error in ffffff, Reason=rrrrrr

Explanation: A software error has been detected when executing the socket function ffffff. The reason for failure is given by rrrrrr. The gateway will terminate abnormally.

User Response: Call IBM for support.

ICDRA40I APCB data at Loc=llllllll Return Code=rrrrrrrr, Recovery Code=cccccccc

Additional Message Text:

```
Offset   ----- Data ------
+0000   xxxxxxxx xxxxxxxx
+0008   xxxxxxxx xxxxxxxx
+0010   xxxxxxxx xxxxxxxx
+0018   xxxxxxxx xxxxxxxx
+0020   xxxxxxxx xxxxxxxx
+0028   xxxxxxxx xxxxxxxx
+0030   xxxxxxxx xxxxxxxx
+0038   xxxxxxxx xxxxxxxx
```

Explanation: The APCB content at location llllll is dumped in hexadecimal format due to a previous error.

User Response: Identify the error which triggered this message and take appropriate action.

ICDRA41I TPL data at Loc=llllllll Return Code=rrrrrrrr, Recovery Code=cccccccc

Additional Message Text:

```
Offset   ----- Data ------
+0000    xxxxxxxx xxxxxxxx
+0008    xxxxxxxx xxxxxxxx
+0010    xxxxxxxx xxxxxxxx
+0018    xxxxxxxx xxxxxxxx
+0020    xxxxxxxx xxxxxxxx
+0028    xxxxxxxx xxxxxxxx
+0030    xxxxxxxx xxxxxxxx
+0038    xxxxxxxx xxxxxxxx
```

Explanation: The TPL content at location llllll is dumped in hexadecimal format due to a previous error.

User Response: Identify the error which triggered this message and take appropriate action.

ICDSA0E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME NOT SPECIFIED

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name was not specified as second subparameter of the JCL SUBSYS= keyword. For NFS Coupler the module name must be ICDNOPCL.

User Response: add the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSA1E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME IS INVALID

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name, which is the second subparameter of the JCL SUBSYS= keyword, was not a valid module name. For NFS Coupler the module name must be ICDNOPCL.

User Response: correct the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSC00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CLOSE failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSH00E ssss, DD=dddddddd, CHECKPOINT FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CHECKPOINT failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSI00E - ssss - BLDL FAILED FOR MODULE=IORRCOMM, RC=cccccccc

Explanation: During initialization of SubSystem ssss module IORRCOMM was not found. This message is issued only when using IORR utilities to add or replace an IORR SubSystem.

User Response: make sure that module IORRCOMM is available in the LINKLIB concatenation.

ICDSI00I - ssss - VERSION 1 RELEASE 2, SUBSYSTEM INITIALIZATION STARTED

Explanation: Initialization of SubSystem ssss has started.

User Response: none.

ICDSI01E - ssss - LOAD FAILED, ABEND=cccccccc, R15=rrrrrrrr

Explanation: The initialization routine of SubSystem ssss was not able to load module=IORRCOMM due to an abend with code cccccccc (hex). Register 15 at the time of abend was rrrrrrrr (hex).

User Response: Call IBM for support.

ICDSI01I - ssss - INITIALIZATION COMPLETED

Explanation: Initialization of SubSystem ssss has completed.

User Response: none.

ICDSO00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: OPEN failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: if module name mmmmmmmm is not ICDNOPCL, then specify ICDNOPCL as the second subparameter of the JCL keyword SUBSYS= and resubmit the JOB. Otherwise, call IBM for support.

ICDSR00E ssss, DD=dddddddd, RESTART FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: RESTART failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW is xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSA0E **SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME NOT SPECIFIED**

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name was not specified as second subparameter of the JCL SUBSYS= keyword. For NFS Coupler the module name must be ICDNOPCL.

User Response: add the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSA1E **SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME IS INVALID**

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name, which is the second subparameter of the JCL SUBSYS= keyword, was not a valid module name. For NFS Coupler the module name must be ICDNOPCL.

User Response: correct the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSC00E **ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx**

Explanation: CLOSE failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSH00E **ssss, DD=dddddddd, CHECKPOINT FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx**

Explanation: CHECKPOINT failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSI00E **- ssss - BLDL FAILED FOR MODULE=IORRCOMM, RC=cccccccc**

Explanation: During initialization of SubSystem ssss module IORRCOMM was not found. This message is issued only when using IORR utilities to add or replace an IORR SubSystem.

User Response: make sure that module IORRCOMM is available in the LINKLIB concatenation.

ICDSI00I **- ssss - VERSION 1 RELEASE 2, SUBSYSTEM INITIALIZATION STARTED**

Explanation: Initialization of SubSystem ssss has started.

User Response: none.

ICDSI01E **- ssss - LOAD FAILED, ABEND=cccccccc, R15=rrrrrrrr**

Explanation: The initialization routine of SubSystem ssss was not able to load module=IORRCOMM due to an abend with code cccccccc (hex). Register 15 at the time of abend was rrrrrrrr (hex).

User Response: Call IBM for support.

ICDSI01I **- ssss - INITIALIZATION COMPLETED**

Explanation: Initialization of SubSystem ssss has completed.

User Response: none.

ICDSO00E **ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx**

Explanation: OPEN failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: if module name mmmmmmmm is not ICDNOPCL, then specify ICDNOPCL as the second subparameter of the JCL keyword SUBSYS= and resubmit the JOB. Otherwise, call IBM for support.

ICDSR00E **ssss, DD=dddddddd, RESTART FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx**

Explanation: RESTART failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW is xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDNA00E Macro=mmmmmmmm failed, CB Addr=aaaaaaaa Ret. Code=cccccccc, Rec. Code=rrrrrrrr, EWord=xxxxxxxx

Explanation: The TCP ACCESS macro `mmmmmmmm` returned error code `cccccccc` and recovery code `rrrrrrrr`. The related control block is at address `aaaaaaaa` and the content of its error code word is `xxICD`. The reason for the error could be an environmental problem or a software error.

User Response: Make sure that the TCP ACCESS subsystem is up and running, and resubmit the JOB. If the problem persists, call IBM for support.

ICDNA01E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that the TCP ACCESS subsystem is up and running, and and resubmit the JOB. If the error persists, Call IBM for support.

ICDNA30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNA31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI00E IUCV CALL ERROR, FUNCTION CODE=ffffffff, RETURN CODE=rrrrrrrr IUCV PARAMETER BLOCK IS: OFFSET ----- DATA ------ +0000 xxxxxxxx xxxxxxxx +0008 xxxxxxxx xxxxxxxx +0010 xxxxxxxx xxxxxxxx +0018 xxxxxxxx xxxxxxxx +0020 xxxxxxxx xxxxxxxx

Explanation: An IUCV call failed with function code `ffffffff` (hex) and return code `rrrrrrrr` (hex). The content of the parameter block is dumped in hexadecimal format.

User Response: Call IBM for support.

ICDNI01I IUCV INTERRUPT BUFFER IS: OFFSET ----- DATA ------ +0000 xxxxxxxx xxxxxxxx +0008 xxxxxxxx xxxxxxxx +0010 xxxxxxxx xxxxxxxx +0018 xxxxxxxx xxxxxxxx +0020 xxxxxxxx xxxxxxxx

Explanation: The message is issued to dump the IUCV interrupt buffer. This message is issued after an error message.

User Response: check the previous IUCV error message and follow its directions.

ICDNI02E CONNECTION COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV CONNECTION COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI04E CONNECTION SEVERED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV CONNECTION SEVERED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI05E MESSAGE COMPLETED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI06E MESSAGE COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI07E IPAUDIT FIELD IN INTERRUPT BUFFER SHOWS UNRECOVERABLE ERROR

Explanation: The IPAUDIT field in the IUCV interrupt field is invalid. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI08E INVALID CODE ENCOUNTERED IN IUCV EXIT ROUTINE

Explanation: An invalid code was returned by IUCV. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI10E CONNECTION SEVERED INTERRUPT, USER DATA=dddd

Explanation: The TCP/IP address space severed the IUCV connection with NFS Coupler. User data `dddd` is an EBCDIC string which indicates the reason for severing the connection.

User Response: check that the TCP/IP started task was not terminated. If the TCP/IP started task is up and running and the error reoccurs, call IBM for support.

ICDNI11E RECVFROM CALL FAILED, ERRNO=dddd

Explanation: The TCP/IP RECVFROM call failed with SOCKET return code `dddd` decimal.

User Response: Call IBM for support.

ICDNI12E PERMANENT ERROR FOUND IN IUCV ANCHOR BLOCK: rrrrrrrr

Explanation: The permanent error flag was found in the main IUCV control block. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI13E CRITICAL SECTION EXECUTION NOT COMPLETED: rrrrrrrr

Explanation: Execution of the IUCV critical section was not completed during the last NFS Coupler call. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI14E IUCV INTERFACE ALREADY IN USE BY ANOTHER TASK IN THE ADDRESS SPACE

Explanation: The application is a multitasking application and multiple NFS Coupler files are being opened concurrently by more than one task. NFS Coupler is unable to support concurrent opens of multiple files from more than one task due to limitations of the IBM TCP/IP product.

User Response: Modify your application so that concurrent opens of NFS Coupler files is from one task only (see Limitations on page 51).

ICDNI15E MAXIMUM NUMBER OF IUCV CONNECTIONS EXCEEDED

Explanation: The application is opening concurrently too many NFS Coupler files (more than 256).

User Response: Modify your application so that no more than 256 NFS Coupler files are open at the same time (see Limitations on page 51).

ICDNI16E VMCF SubSystem NOT FOUND

Explanation: The VMCF SubSystem and the IBM TCP/IP product do not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI17E VMCF SubSystem CVT INVALID, BAD EYE CATCHER FOUND

Explanation: The VMCF SubSystem was found but its CVT has a bad eye catcher. Most probably the IBM TCP/IP product does not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI18E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Gethostname failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI19E SOCKET CALL FAILED, ERRNO=dddd

Explanation: Socket call Socket failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI20E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Bind failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI22E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that TCP/IP is working properly and resubmit the JOB. If the error persists, call IBM for support.

ICDNI25E CLOSE CALL FAILED, ERRNO=dddd

Explanation: Socket call Close failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI28E SELECT CALL FAILED, ERRNO=dddd

Explanation: Socket call Select failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI29E SENDTO CALL FAILED, ERRNO=dddd

Explanation: Socket call Sendto failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNM00E DD=dddddddd, Load Module ″ICDNMAP″ was no loaded from an authorized library

Explanation: Module ICDNMAP was not loaded from an authorized library. OPEN of file `dddddddd` will fail.

User Response: make sure module ICDNMAP has been copied into an authorized library which is part of the LINKLIST concatenation. Also make sure that no module with name ICDNMAP exists in either the JOBLIB or the STEPLIB concatenation.

ICDNM00I DD=dddddddd, READ_BIN, Last record padded with ″00″x

Explanation: The last record of a NFS Coupler file with fixed record format has been padded with ″00″x characters, since the length of the file read is not an exact multiple of the record length.

User Response: none.

ICDNM01E DD=dddddddd, READ_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in UNIX format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM02E DD=dddddddd, READ_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of binary characters from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM03E DD=dddddddd, WRITE_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of binary characters to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM05E DD=dddddddd, WRITE_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for UNIX to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM08E DD=dddddddd, LIST_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for directory entries to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM09E DD=dddddddd, READ_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for a UNIX file attribute to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM10E DD=dddddddd, MOUNT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request to mount a directory failed. The MOUNT server returned NFS error code `nnnnnnnn` mnemonics `ccc` decimal. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM11E DD=dddddddd, File= ″ffff″ is not a regular file

Explanation: NFS Coupler will not access UNIX file `ffff` because it is not a regular file. File `dddddddd` was opened for output or for input with an NFS Coupler input disposition of `FILE`. The OPEN for file `dddddddd` will fail.

User Response: for file `dddddddd` either change the UNIX file path to indicate a regular file or change the NFS Coupler input disposition to `DIR` to read the contents of a UNIX directory.

ICDNM12E DD=dddddddd, File= ″ffff″ already exists

Explanation: NFS Coupler refuses to write to UNIX file `ffff` because it already exists. File `dddddddd` was open with an NFS Coupler output disposition of `NEW`. The OPEN for file `dddddddd` will fail.

User Response: either change the NFS Coupler output disposition or change the UNIX file path in the JCL for file `dddddddd`.

ICDNM13E DD=dddddddd, Unable to look up file= ″ffff″, File= ″oooo″ is not a directory

Explanation: NFS Coupler was not able to look up file `ffff` under directory `oooo` because file `oooo` is not a directory. File `ffff` is an entry in the UNIX file name path specified. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd`.

ICDNM14E DD=dddddddd, File= ″ffff″ is not a directory

Explanation: NFS Coupler will not access file `ffff` since it is not a directory. File `dddddddd` was opened for input with NFS Coupler input disposition `DIR`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or change NFS Coupler input disposition to `FILE`.

ICDNM16E DD=dddddddd, LOOKUP failed, File= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX file `ffff`. UNIX file `ffff` is the last entry in the UNIX file path specified in the JCL for file `dddddddd`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX file `ffff` already exists.

ICDNM17E DD=dddddddd, LOOKUP failed, Directory= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX directory `ffff`. UNIX directory `ffff` is an intermediate entry in the UNIX file path specified in the JCL for file `dddddddd`. File `dddddddd` was opened for input. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX directory `ffff` already exists.

ICDNM18E DD=dddddddd, RESET LENGTH failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to reset the length of UNIX file ffff to zero. UNIX file ffff is the last entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM19E DD=dddddddd, LOOKUP failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to get a file handle of UNIX file ffff. UNIX file ffff is an entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM20E DD=dddddddd, CREATE FILE failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX file ffff. UNIX file ffff is the last entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM21E DD=dddddddd, CREATE DIRECTORY failed, Directory= ″oooo″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX directory oooo. UNIX directory oooo is an intermediate entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM23E DD=dddddddd, Invalid number of arguments

Explanation: The number of subparameters specified of the JCL keyword SUBSYS= of file dddddddd is invalid. The OPEN for file dddddddd will fail.

User Response: check the list of NFS Coupler subparameters and correct any error.

ICDNM24E DD=dddddddd, Host name required

Explanation: Host name was not specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd. The OPEN for file dddddddd will fail.

User Response: add the host name to the JCL NFS Coupler subparameter list.

ICDNM25E DD=dddddddd, Host name is too long

Explanation: The host name specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd exceed 16 characters of length. The OPEN for file dddddddd will fail.

User Response: correct the host name in the JCL NFS Coupler subparameter list.

ICDNM26E DD=dddddddd, MVS USERID of JOB is not SUPERUSER for NFS server host, USERID cannot be specified

Explanation: The MVS USERID was specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd, but the JOB USERID is not superuser for the NFS server host. In order to be superuser the JOB USERID must map into Uid=0 for the NFS server host. The OPEN for file dddddddd will fail.

User Response: either run the JOB with APF authorization or delete the MVS USERID from the JCL NFS Coupler subparameter list.

ICDNM27E DD=dddddddd, MVS USERID is too long

Explanation: The MVS USERID was specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd, but its length exceeds 8 characters. The OPEN for file dddddddd will fail.

User Response: change the MVS USERID in the JCL NFS Coupler subparameter list to a correct value.

ICDNM28E DD=dddddddd, File path is required

Explanation: The UNIX file path has not been specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd, but it is required. The OPEN for file dddddddd will fail.

User Response: add the UNIX file path to the JCL NFS Coupler subparameter list.

ICDNM29E DD=dddddddd, Invalid input disposition

Explanation: The input disposition has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. This condition is detected even if the file is opened for output. The OPEN for file dddddddd will fail.

User Response: change the input disposition in the JCL NFS Coupler subparameter list to a correct value.

ICDNM30E DD=dddddddd, Invalid output disposition

Explanation: The output disposition has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. This condition is detected even if the file is opened for input. The OPEN for file dddddddd will fail.

User Response: change the output disposition in the JCL NFS Coupler subparameter list to a correct value.

ICDNM31E DD=dddddddd, Invalid file mode

Explanation: The UNIX file mode has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. The OPEN for file dddddddd will fail.

User Response: change the UNIX file mode in the JCL NFS Coupler subparameter list to a correct value.

ICDNM32E DD=dddddddd, Invalid data type

Explanation: The data type has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. The OPEN for file dddddddd will fail.

User Response: change the data type in the JCL NFS Coupler subparameter list to a correct value.

ICDNM34E DD=dddddddd, Unable to map NFS server host name

Explanation: NFS Coupler is unable to find an entry in its mapping table for the NFS server host name specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd. The OPEN for file dddddddd will fail.

User Response: correct the host name. If the error persists ask the NFS Coupler administrator to add an entry for the host name you are trying to access.

ICDNM35W DD=dddddddd, Unable to map MVS USERID=uuuuuuuu, %DEFUID will be used

Explanation: NFS Coupler is unable to find an entry in its the mapping table for the MVS USERID. The OPEN for file dddddddd will continue. Default MVS USERID ″%DEFUID″ will be used if define in module ICDNMAP for the NFS server being accessed.

User Response: ask the NFS Coupler administrator to add the JOB MVS USERID you are using, to the NFS Coupler mapping table. The JOB MVS USERID must be added for the domain in which the host you are trying to access resides.

ICDNM36E DD=dddddddd, File path has an invalid combination of ″/″ characters

Explanation: The user specified a file path name which is invalid either because it contains two or more consecutive ″/″ or because the first character is ″/″. The OPEN for file dddddddd will fail.

User Response: change the file path to a correct value and resubmit.

ICDNM37I DD=dddddddd, File ″ffff″ has been removed

Explanation: UNIX file with ffff full path name has been removed at close time as specified in the input disposition for file dddddddd.

ICDNM38W DD=dddddddd, File ″ffff″ not removed, ABEND in progress

Explanation: UNIX file with ffff full path name has not been removed as specified in the input disposition for file dddddddd. The reason for bypassing remove is that the close routine detected an ABEND for the task executing the close routine.

ICDNM39W DD=dddddddd, Remove failed, File= "ffff", Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to remove the NFS server file `ffff` as specified by the input disposition for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. Close processing for file `dddddddd` will continue.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM40E DD=dddddddd, READ_PRINT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in PRINT format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM41E DD=dddddddd, WRITE_PRINT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for PRINT to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM42E DD=dddddddd, Input buffer too short, record has been truncated

Explanation: The buffer used for a read operation is too short. The data available from the UNIX file has been truncated. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: use a longer record length to read the UNIX file.

ICDNM43E DD=dddddddd, Unable to write a file larger than 4294967295 bytes

Explanation: NFS protocol limitations have been exceeded. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: break the MVS file being written to UNIX into multiple smaller files.

ICDNM44E DD=dddddddd, READ_DOS failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in DOS format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM45E DD=dddddddd, WRITE_DOS failed, Nfs rc= nnnnnnnn (ccc)

Explanation: An attempt to write characters formatted for DOS to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM46E DD=dddddddd, Checkpoint/Restart not supported for Input Directory files

Explanation: A checkpoint was attempted while file `dddddddd` used to read directory entries is open. Checkpoint/Restart is not supported for this type of files. Checkpoint will fail.

User Response: make sure that files used to read directory entries are closed when a checkpoint is taken.

ICDNM47W DD=dddddddd, RESTART: Input file is different than the one used in last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS input file `dddddddd` has a different file system ID or file ID than those recorded during the last checkpoint. This can happen if the input file on the NFS server was deleted and replaced with a different file before restart was attempted.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM48W DD=dddddddd, RESTART: Input file has been updated since last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS input file `dddddddd` has been updated since the last checkpoint. This will happen, for example, if the input file on the NFS server was updated to remove a bad input record which caused an ABEND.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM49W DD=dddddddd, RESTART: Output file has been changed since last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS output file dddddddd has a different file system ID or file ID than those recorded during the last checkpoint. This can happen if the output file on the NFS server has been deleted and replaced with a different file before restart was attempted. In general this is a problem due to operator error on the NFS server.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM50W DD=dddddddd, RESTART: Output file is shorter than it was at last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS output file dddddddd length is less than it was at the last checkpoint. This can happen if the output file on the NFS server was updated before restart was attempted. In general this is a problem due to operator error on the NFS server.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM51E DD=dddddddd, Checkpoint/Restart not supported for Input Directory files

Explanation: A restart was attempted while file dddddddd used to read directory entries is open. Checkpoint/Restart is not supported for this type of files. Restart will fail. This will happen if the JCL DD statement for the NFS Coupler file dddddddd has been changed so that the input file is now a directory.

User Response: correct the DD statement for the file and resubmit the JOB to restart.

ICDNM52I DD=dddddddd, aaaaaa: terminated due to operator response

Explanation: The operator, when prompted to enter 'Y' to continue processing or 'N' to abnormally terminate, entered 'N'. Processing will be abnormally terminated by aaaaaa. Currently aaaaaa can be CHECKPOINT or RESTART.

User Response: none.

ICDNM53E DD=dddddddd, Default MVS USERID=%DEFUID not defined

Explanation: NFS Coupler is unable to find defualt MVS USERID "%DEFUID". The OPEN for file dddddddd will fail.

User Response: ask the NFS Coupler administrator to add either the JOB MVS USERID or the defualt MVS USERID "%DEFUID" to the NFS Coupler mapping table. Either one must be added for the domain in which the host you are trying to access resides.

ICDNM71W DD=dddddddd, Nested checkpoint request

Explanation: A checkpoint request was invoked while another I/O request was in progress for the same DCB of file dddddddd. Checkpoint will fail.

If checkpoint was issued in an asynchronous exit, such as timer or VTAM exit, or a recovery ESTAE exit the program issuing the close is in error.

User Response: modify the program and make sure checkpoint are only by the mainline task code.

ICDNM72E DD=dddddddd, Close entered by MVS normal termination routine, standard close processing bypassed

Explanation: The close routine for file dddddddd has been entered by the MVS termination routine. Standard close processing is bypassed due to the fact that at this stage the IUCV connection with the TCP/IP address space has been already severed by MVS. For files opened for input, if the REMOVE disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

This message is issued when the terminating program does not close all open files.

User Response: change the program so that all open files are closed before terminanting.

ICDNM73W DD=dddddddd, Nested close request

Explanation: A close request was invoked while another I/O request was in progress for the same DCB of file dddddddd. Standard close processing is bypassed. For files opened for input, if the REMOVE disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

If the close was issued in an asynchronous exit, such as timer or VTAM exit, the program issuing the close is in error. If the close was issued in an ESTAE recovery or retry routine after an abend during an I/O request to NFS Coupler, this message should be expected.

User Response: if this message is issued because of programming error, correct the program and submit again.

ICDNM74W DD=dddddddd, Close entered by MVS abnormal termination routine, standard close processing bypassed

Explanation: The close routine for file dddddddd has been entered by the MVS termination routine due to an abend. Standard close processing is bypassed due to the fact that at this stage the IUCV connection with the TCP/IP address space has been already severed by MVS. For files opened for input, if the REMOVE disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

User Response: correct the problem which triggered the abend and resubmit.

ICDNM75E DD=dddddddd, Nested output request detected

Explanation: An output request was invoked while another I/O request was in progress for the same DCB of file dddddddd. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: correct the program and submit again.

ICDNM76E DD=dddddddd, Nested input request detected

Explanation: An input request was invoked while another I/O request was in progress for the same DCB of file dddddddd. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: correct the program and submit again.

ICDNM77E DD=dddddddd, asynchronous check invoked

Explanation: The MVS SSAM routines requested an asynchronous check to NFS Coupler for file dddddddd. This is an error. The caller is abnormally terminated.

User Response: Call IBM for support.

ICDNM80E Trial license has expired. Call IBM for support.

Explanation: NFS Coupler trial license has expired.

User Response: Call IBM for an extension if you need one.

ICDNM81E License not available on this machine. Please call IBM for support.

Explanation: NFS Coupler product license is not available on the current 370/390 machine.

User Response: Call IBM for for a license extension on the current machine.

ICDNM82E Trial license key is corrupt. Please call IBM for support

Explanation: NFS Coupler trial license key is invalid.

User Response: Call IBM for support.

ICDNM90I DD=dddddddd, MVS USERID= uuuuuuu

Explanation: The MVS userid used by NFS Coupler for file dddddddd, is uuuuuuu. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM91I DD=dddddddd, Local host name= hhhhhh

Explanation: The local (MVS) TCP/IP host name used by NFS Coupler for file dddddddd, is hhhhhh. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM92I DD=dddddddd, NFS server host= iii, jjj, kkk, lll (tttt)

Explanation: The NFS sever TCP/IP host name used by NFS Coupler for file dddddddd, has up to four IP address(es) iii, jjj, kkk, lll. Host type tttt can be UNIX or DOS. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM93I DD=dddddddd, Uid= uuu, Gid= ggg

Explanation: The UNIX Uid and Gid used by NFS Coupler for file dddddddd, are uuu and ggg. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM94I DD=dddddddd, Mount directory= oooooo

Explanation: The NFS mount directory used by NFS Coupler for file dddddddd, is oooooo. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM95I DD=dddddddd, File path= ffffff

Explanation: The UNIX file path used by NFS Coupler for file dddddddd, is ffffff. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM96W DD=dddddddd, Number of RPC retransmits is cc% of total number of RPC calls

Explanation: While closing file dddddddd, NFS Coupler detected the the percentage of RPC calls that was retransmitted was cc percent. This message is issued only if cc is larger than 1.

User Response: Check if this message is issued consistently. If this is the case the network connection between MVS and the NFS server is quite slow and some tuning is needed (see NFS Coupler performance tuning on page 25).

ICDNM97I DD=dddddddd, Timing parameters= t.t, rrr, ooo

Explanation: The timing parameters used by NFS Coupler for file dddddddd, to access data on the NFS servers are `timeout=t.t`, `retransmit=rrr` and `overlap=ooo`. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM99E DD=dddddddd, Exiting due to severe error

Explanation: NFS Coupler terminates execution for file dddddddd due to a severe error. This message is normally preceded by other error messages.

User Response: check previous error messages for file dddddddd.

ICDNR00E DD=dddddddd, PORTMAP, timeout, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed due to timeout. The request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: check that the NFS server host is accessible with the TSO `ping` command. Also make sure that the PORTMAP server on the NFS server host is up and running.

ICDNR01E DD=dddddddd, PORTMAP, RPC mismatch, Host supports= (ll, hh), Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed because the PORTMAP deamon on the NFS server does not support RPC version 2 which is used by NFS Coupler. The lowest RPC version supported on the UNIX host is ll and the highest is hh. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: use a PORTMAP level on the UNIX host which supports RPC version 2.

ICDNR02E DD=dddddddd, PORTMAP, Authorization failed, Auth. Stat= ss, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed because the UNIX portmap deamon could not authorize the request. The authorization stat ss gives a reason for the failure. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR03E DD=dddddddd, PORTMAP, Request rejected, Reject Stat= ss, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request was rejected by the UNIX host. The reject stat is ss. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR04E DD=dddddddd, PORTMAP, Not accepted, Acc. Stat= ss, Prog= pppppp, Vers=vv

Explanation: A PORTMAP request for program `pppppp` and version `vv` was rejected by the UNIX host. The accept stat `ss` gives a reason for the failure. Its possible values can be as follows: .strttbl 100 LR 0 10 * LR 0 0 .cell .bo Value .cell .bo Meaning .cell `1` .cell The PORTMAP program (program 100000) is not available .cell `2` .cell The PORTMAP version (version 2) is not supported .cell `3` .cell The PORTMAP procedure (procedure 3) is not supported .cell `4` .cell PORTMAP received bad parameters .endtbl The PORTMAP request was for program `pppppp` and version `vv`. The OPEN for file `dddddddd` will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR05E DD=dddddddd, PORTMAP, zero port returned, Prog= pppppp, Vers= vv

Explanation: A zero UDP port number was returned to a PORTMAP request. The PORTMAP request was for program `pppppp` and version `vv`. Program `pppppp` (100005 for MOUNT, 100003 for NFS), is not up and running on the UNIX host. The OPEN for file `dddddddd` will fail.

User Response: make sure that both the MOUNT and NFS deamons are up and running on the UNIX host.

ICDNR07E DD=dddddddd, RPCWAIT, timeout, Prog= pppppp, Vers= vv, Proc= rr

Explanation: A timeout was detected while waiting for an RPC response. The original call was for program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: check if the UNIX host is accessible with the TSO `ping` command. Also make sure that both the MOUNT and NFS deamons are up and running on the UNIX host.

ICDNR08E DD=dddddddd, RPCWAIT, RPC mismatch, Host supports= (ll, hh), Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call failed because program `pppppp` does not support RPC version 2 which is used by NFS Coupler. The lowest RPC version supported on the UNIX host is `ll` and the highest is `hh`. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The OPEN for file `dddddddd` will fail.

User Response: use NFS and MOUNT software levels on the UNIX host which support RPC version 2.

ICDNR09E DD=dddddddd, RPCWAIT, Authorization failed, Auth. Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: A RPC call failed because the NFS server did not authorize the call. The authorization stat `ss` gives the reason for the failure. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The I/O request for file `dddddddd` will fail.

User Response: Call IBM for support.

ICDNR10E DD=dddddddd, RPCWAIT, Request rejected, Reject Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call was rejected by the NFS server. The reject stat is `ss`. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: Call IBM for support.

ICDNR11E DD=dddddddd, RPCWAIT, Call not accepted, Acc. Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call was rejected by the NFS server. The accept stat `ss` gives a reason for the failure. Its possible values can be as follows: .strttbl 100 LR 0 10 * LR 0 0 .cell .bo Value .cell .bo Meaning .cell `1` .cell Program `pppppp` is not available (100003 is NFS, 100005 is MOUNT). .cell `2` .cell Version `vv` is not supported .cell `3` .cell Procedure `rr` is not supported .cell `4` .cell program `pppppp` received bad parameters .endtbl The RPC call was to program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: make sure that the MOUNT and NFS deamons are up and running on the UNIX host. Also, on the UNIX host, make sure that the MOUNT software level supports version 1 and that NFS software level supports version 2. If the error persists, call IBM for support.

ICDNR99E DD=dddddddd, RPCCALL, software error, overlapping calls

Explanation: A software error was detected by NFS Coupler. The I/O operation for file `dddddddd`, will fail.

User Response: Call IBM for support.

ICDNU00E Bad return code from CHKPT macro.

Explanation: The IEBGENER totaling ICDNCHKP, which issues checkpoints during copy, detected a bad return code from the checkpoint macro. The JOB is abnormally terminated with code U0000. Registers R15 and R14 will contain the return and reason code from the checkpoint macro.

User Response: check the return and reason codes in registers R15 and R14 to identify the problem. After fixing the problem resubmit the JOB.

ICDNU01I No CHECKPOINT time interval specified, 10 minutes will be used

Explanation: No time interval was specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy. The default value of 10 minutes will be used.

User Response: none.

ICDNU02W Invalid CHECKPOINT time interval specified, 10 minutes will be used

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is invalid. Processing continues, the default value of 10 minutes will be used.

User Response: correct interval value for later runs.

ICDNU03W Zero CHECKPOINT time interval specified, 10 minutes will be used

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is zero. Processing continues, the default value of 10 minutes will be used.

User Response: change the interval value to a positive number for later runs.

ICDNU04I CHECKPOINT time interval specified is″ mmm minute(s)

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is `mmm` minutes. Every `mmm` minutes a check point will be taken by ICDNCHKP.

User Response: none.

ICDNX00E DD=dddddddd, rrrrrr failed, diagnostic string= ssssss, Buffer Loc= llllll

Explanation: XDR routine `rrrrrr` could not encode or decode XDR data. Diagnostic string `ssssss` identifies the caller. Decimal number `llllll` indicates the memory location of the buffer. The I/O operation for file `dddddddd`, will fail.

User Response: Call IBM for support.

ICDSA0E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME NOT SPECIFIED

Explanation: Allocation of DD name `dddddddd` to SubSystem `ssss` failed because OPEN/CLOSE interface module name was not specified as second subparameter of the JCL `SUBSYS=` keyword. For NFS Coupler the module name must be `ICDNOPCL`.

User Response: add the second subparameter to keyword `SUBSYS=` by specifying `ICDNOPCL` and resubmit the JOB.

ICDSA1E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME IS INVALID

Explanation: Allocation of DD name `dddddddd` to SubSystem `ssss` failed because OPEN/CLOSE interface module name, which is the second subparameter of the JCL `SUBSYS=` keyword, was not a valid module name. For NFS Coupler the module name must be ICDNOPCL.

User Response: correct the second subparameter to keyword `SUBSYS=` by specifying `ICDNOPCL` and resubmit the JOB.

ICDSC00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CLOSE failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW was `xxxxxxxx xxxxxxxx` (hex).

User Response: Call IBM for support.

ICDSH00E ssss, DD=dddddddd, CHECKPOINT FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CHECKPOINT failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW was `xxxxxxxx xxxxxxxx` (hex).

User Response: Call IBM for support.

ICDSI00E - ssss - BLDL FAILED FOR MODULE=IORRCOMM, RC=cccccccc

Explanation: During initialization of SubSystem `ssss` module IORRCOMM was not found. This message is issued only when using IORR utilities to add or replace an IORR SubSystem.

User Response: make sure that module IORRCOMM is available in the LINKLIB concatenation.

ICDSI00I - ssss - VERSION 1 RELEASE 2, SUBSYSTEM INITIALIZATION STARTED

Explanation: Initialization of SubSystem `ssss` has started.

User Response: none.

ICDSI01E - ssss - LOAD FAILED, ABEND=cccccccc, R15=rrrrrrrr

Explanation: The initialization routine of SubSystem `ssss` was not able to load module=IORRCOMM due to an abend with code `cccccccc` (hex). Register 15 at the time of abend was `rrrrrrrr` (hex).

User Response: Call IBM for support.

ICDSI01I - ssss - INITIALIZATION COMPLETED

Explanation: Initialization of SubSystem `ssss` has completed.

User Response: none.

ICDSO00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: OPEN failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW was `xxxxxxxx xxxxxxxx` (hex).

User Response: if module name `mmmmmmmm` is not `ICDNOPCL`, then specify `ICDNOPCL` as the second subparameter of the JCL keyword `SUBSYS=` and resubmit the JOB. Otherwise, call IBM for support.

ICDSR00E ssss, DD=dddddddd, RESTART FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: RESTART failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW is `xxxxxxxx xxxxxxxx` (hex).

User Response: Call IBM for support.

ICDNA00E Macro=mmmmmmmm failed, CB Addr=aaaaaaaa Ret. Code=cccccccc, Rec. Code=rrrrrrrr, EWord=xxxxxxxx

Explanation: The TCP ACCESS macro `mmmmmmmm` returned error code `cccccccc` and recovery code `rrrrrrrr`. The related control block is at address `aaaaaaaa` and the content of its error code word is `xxICD`. The reason for the error could be an environmental problem or a software error.

User Response: Make sure that the TCP ACCESS subsystem is up and running, and resubmit the JOB. If the problem persists, Call IBM for support.

ICDNA01E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that the TCP ACCESS subsystem is up and running, and and resubmit the JOB. If the error persists, call IBM for support.

ICDNA30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNA31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI00E IUCV CALL ERROR, FUNCTION CODE=ffffffff, RETURN CODE=rrrrrrrr IUCV PARAMETER BLOCK IS:

Additional Message Text:

```
OFFSET   ----- DATA ------
+0000    xxxxxxxx xxxxxxxx
+0008    xxxxxxxx xxxxxxxx
+0010    xxxxxxxx xxxxxxxx
+0018    xxxxxxxx xxxxxxxx
+0020    xxxxxxxx xxxxxxxx
```

Explanation: An IUCV call failed with function code `ffffffff` (hex) and return code `rrrrrrrr` (hex). The content of the parameter block is dumped in hexadecimal format.

User Response: Call IBM for support.

ICDNI01I IUCV INTERRUPT BUFFER IS:

Additional Message Text:

```
OFFSET   ----- DATA ------
+0000    xxxxxxxx xxxxxxxx
+0008    xxxxxxxx xxxxxxxx
+0010    xxxxxxxx xxxxxxxx
+0018    xxxxxxxx xxxxxxxx
+0020    xxxxxxxx xxxxxxxx
```

Explanation: The message is issued to dump the IUCV interrupt buffer. This message is issued after an error message.

User Response: check the previous IUCV error message and follow its directions.

ICDNI02E CONNECTION COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV CONNECTION COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI04E CONNECTION SEVERED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV CONNECTION SEVERED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI05E MESSAGE COMPLETED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI06E MESSAGE COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI07E IPAUDIT FIELD IN INTERRUPT BUFFER SHOWS UNRECOVERABLE ERROR

Explanation: The IPAUDIT field in the IUCV interrupt field is invalid. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI08E INVALID CODE ENCOUNTERED IN IUCV EXIT ROUTINE

Explanation: An invalid code was returned by IUCV. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI10E CONNECTION SEVERED INTERRUPT, USER DATA=dddd

Explanation: The TCP/IP address space severed the IUCV connection with NFS Coupler. User data `dddd` is an EBCDIC string which indicates the reason for severing the connection.

User Response: check that the TCP/IP started task was not terminated. If the TCP/IP started task is up and running and the error reoccurs, call IBM for support.

ICDNI11E RECVFROM CALL FAILED, ERRNO=dddd

Explanation: The TCP/IP RECVFROM call failed with SOCKET return code `dddd` decimal.

User Response: Call IBM for support.

ICDNI12E PERMANENT ERROR FOUND IN IUCV ANCHOR BLOCK: rrrrrrrr

Explanation: The permanent error flag was found in the main IUCV control block. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI13E CRITICAL SECTION EXECUTION NOT COMPLETED: rrrrrrrr

Explanation: Execution of the IUCV critical section was not completed during the last NFS Coupler call. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI14E IUCV INTERFACE ALREADY IN USE BY ANOTHER TASK IN THE ADDRESS SPACE

Explanation: The application is a multitasking application and multiple NFS Coupler files are being opened concurrently by more than one task. NFS Coupler is unable to support concurrent opens of multiple files from more than one task due to limitations of the IBM TCP/IP product.

User Response: Modify your application so that concurrent opens of NFS Coupler files is from one task only (see Limitations on page 51).

ICDNI15E MAXIMUM NUMBER OF IUCV CONNECTIONS EXCEEDED

Explanation: The application is opening concurrently too many NFS Coupler files (more than 256).

User Response: Modify your application so that no more than 256 NFS Coupler files are open at the same time (see Limitations on page 51).

ICDNI16E VMCF SubSystem NOT FOUND

Explanation: The VMCF SubSystem and the IBM TCP/IP product do not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI17E VMCF SubSystem CVT INVALID, BAD EYE CATCHER FOUND

Explanation: The VMCF SubSystem was found but its CVT has a bad eye catcher. Most probably the IBM TCP/IP product does not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI18E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Gethostname failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI19E SOCKET CALL FAILED, ERRNO=dddd

Explanation: Socket call Socket failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI20E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Bind failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI22E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that TCP/IP is working properly and resubmit the JOB. If the error persists, call IBM for support.

ICDNI25E CLOSE CALL FAILED, ERRNO=dddd

Explanation: Socket call Close failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI28E SELECT CALL FAILED, ERRNO=dddd

Explanation: Socket call Select failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI29E SENDTO CALL FAILED, ERRNO=dddd

Explanation: Socket call Sendto failed with SOCKET error dddd.

User Response: Call IBM for support.

ICDNI30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNM00E DD=dddddddd, Load Module ″ICDNMAP″ was no loaded from an authorized library

Explanation: Module ICDNMAP was not loaded from an authorized library. OPEN of file dddddddd will fail.

User Response: make sure module ICDNMAP has been copied into an authorized library which is part of the LINKLIST concatenation. Also make sure that no module with name ICDNMAP exists in either the JOBLIB or the STEPLIB concatenation.

ICDNM00I DD=dddddddd, READ_BIN, Last record padded with ″00″x

Explanation: The last record of a NFS Coupler file with fixed record format has been padded with ″00″x characters, since the length of the file read is not an exact multiple of the record length.

User Response: none.

ICDNM01E DD=dddddddd, READ_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in UNIX format from the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O GET or READ operation for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM02E DD=dddddddd, READ_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of binary characters from the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O GET or READ operation for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM03E DD=dddddddd, WRITE_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of binary characters to the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O PUT or WRITE operation for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM05E DD=dddddddd, WRITE_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for UNIX to the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O PUT or WRITE operation for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM08E DD=dddddddd, LIST_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for directory entries to the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O GET or READ operation for file dddddddd will fail. File dddddd was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM09E DD=dddddddd, READ_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for a UNIX file attribute to the NFS server failed. The NFS server returned error code nnnnnnnn mnemonics ccc decimal. The I/O GET or READ operation for file dddddddd will fail. File dddddd was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM10E DD=dddddddd, MOUNT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request to mount a directory failed. The MOUNT server returned NFS error code nnnnnnnn mnemonics ccc decimal. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM11E DD=dddddddd, File= ″ffff″ is not a regular file

Explanation: NFS Coupler will not access UNIX file ffff because it is not a regular file. File dddddddd was opened for output or for input with an NFS Coupler input disposition of FILE. The OPEN for file dddddddd will fail.

User Response: for file dddddddd either change the UNIX file path to indicate a regular file or change the NFS Coupler input disposition to DIR to read the contents of a UNIX directory.

ICDNM12E DD=dddddddd, File= ″ffff″ already exists

Explanation: NFS Coupler refuses to write to UNIX file ffff because it already exists. File dddddddd was open with an NFS Coupler output disposition of NEW. The OPEN for file dddddddd will fail.

User Response: either change the NFS Coupler output disposition or change the UNIX file path in the JCL for file dddddddd.

ICDNM13E DD=dddddddd, Unable to look up file= ″ffff″, File= ″oooo″ is not a directory

Explanation: NFS Coupler was not able to look up file ffff under directory oooo because file oooo is not a directory. File ffff is an entry in the UNIX file name path specified. The OPEN for file dddddddd will fail.

User Response: correct the file path specified in the JCL for file dddddddd.

ICDNM14E DD=dddddddd, File= ″ffff″ is not a directory

Explanation: NFS Coupler will not access file ffff since it is not a directory. File dddddddd was opened for input with NFS Coupler input disposition DIR. The OPEN for file dddddddd will fail.

User Response: correct the file path specified in the JCL for file dddddddd or change NFS Coupler input disposition to FILE.

ICDNM16E DD=dddddddd, LOOKUP failed, File= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX file ffff. UNIX file ffff is the last entry in the UNIX file path specified in the JCL for file dddddddd. The OPEN for file dddddddd will fail.

User Response: correct the file path specified in the JCL for file dddddddd or make sure that UNIX file ffff already exists.

ICDNM17E DD=dddddddd, LOOKUP failed, Directory= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX directory ffff. UNIX directory ffff is an intermediate entry in the UNIX file path specified in the JCL for file dddddddd. File dddddddd was opened for input. The OPEN for file dddddddd will fail.

User Response: correct the file path specified in the JCL for file dddddddd or make sure that UNIX directory ffff already exists.

ICDNM18E DD=dddddddd, RESET LENGTH failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to reset the length of UNIX file ffff to zero. UNIX file ffff is the last entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM19E DD=dddddddd, LOOKUP failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to get a file handle of UNIX file ffff. UNIX file ffff is an entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM20E DD=dddddddd, CREATE FILE failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX file ffff. UNIX file ffff is the last entry in the UNIX file path specified for file dddddddd. The request failed with NFS return code nnnnnnnn mnemonics ccc decimal. File dddddddd was opened for output. The OPEN for file dddddddd will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDSA0E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME NOT SPECIFIED

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name was not specified as second subparameter of the JCL SUBSYS= keyword. For NFS Coupler the module name must be ICDNOPCL.

User Response: add the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSA1E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME IS INVALID

Explanation: Allocation of DD name dddddddd to SubSystem ssss failed because OPEN/CLOSE interface module name, which is the second subparameter of the JCL SUBSYS= keyword, was not a valid module name. For NFS Coupler the module name must be ICDNOPCL.

User Response: correct the second subparameter to keyword SUBSYS= by specifying ICDNOPCL and resubmit the JOB.

ICDSC00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CLOSE failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSH00E ssss, DD=dddddddd, CHECKPOINT FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CHECKPOINT failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSI00E - ssss - BLDL FAILED FOR MODULE=IORRCOMM, RC=cccccccc

Explanation: During initialization of SubSystem ssss module IORRCOMM was not found. This message is issued only when using IORR utilities to add or replace an IORR SubSystem.

User Response: make sure that module IORRCOMM is available in the LINKLIB concatenation.

ICDSI00I - ssss - VERSION 1 RELEASE 2, SUBSYSTEM INITIALIZATION STARTED

Explanation: Initialization of SubSystem ssss has started.

User Response: none.

ICDSI01E - ssss - LOAD FAILED, ABEND=cccccccc, R15=rrrrrrrr

Explanation: The initialization routine of SubSystem ssss was not able to load module=IORRCOMM due to an abend with code cccccccc (hex). Register 15 at the time of abend was rrrrrrrr (hex).

User Response: Call IBM for support.

ICDSI01I - ssss - INITIALIZATION COMPLETED

Explanation: Initialization of SubSystem ssss has completed.

User Response: none.

ICDSO00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: OPEN failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: if module name mmmmmmmm is not ICDNOPCL, then specify ICDNOPCL as the second subparameter of the JCL keyword `SUBSYS=` and resubmit the JOB. Otherwise, call IBM for support.

ICDSR00E ssss, DD=dddddddd, RESTART FAILED DUE TO ABEND ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: RESTART failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW is `xxxxxxxx xxxxxxxx` (hex).

User Response: Call IBM for support.

ICDNA00E Macro=mmmmmmmm failed, CB Addr=aaaaaaaa Ret. Code=cccccccc, Rec. Code=rrrrrrrr, EWord=xxxxxxxx

Explanation: The TCP ACCESS macro `mmmmmmmm` returned error code `cccccccc` and recovery code `rrrrrrrr`. The related control block is at address `aaaaaaaa` and the content of its error code word is `xxICD`. The reason for the error could be an environmental problem or a software error.

User Response: Make sure that the TCP ACCESS subsystem is up and running, and resubmit the JOB. If the problem persists, call IBM for support.

ICDNA01E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that the TCP ACCESS subsystem is up and running, and and resubmit the JOB. If the error persists, Call IBM for support.

ICDNA30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNA31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI00E IUCV CALL ERROR, FUNCTION CODE=ffffffff, RETURN CODE=rrrrrrrr IUCV PARAMETER BLOCK IS: OFFSET ----- DATA ------ +0000 xxxxxxxx xxxxxxxx +0008 xxxxxxxx xxxxxxxx +0010 xxxxxxxx xxxxxxxx +0018 xxxxxxxx xxxxxxxx +0020 xxxxxxxx xxxxxxxx

Explanation: An IUCV call failed with function code `ffffffff` (hex) and return code `rrrrrrrr` (hex). The content of the parameter block is dumped in hexadecimal format.

User Response: Call IBM for support.

ICDNI01I IUCV INTERRUPT BUFFER IS: OFFSET ----- DATA ------ +0000 xxxxxxxx xxxxxxxx +0008 xxxxxxxx xxxxxxxx +0010 xxxxxxxx xxxxxxxx +0018 xxxxxxxx xxxxxxxx +0020 xxxxxxxx xxxxxxxx

Explanation: The message is issued to dump the IUCV interrupt buffer. This message is issued after an error message.

User Response: check the previous IUCV error message and follow its directions.

ICDNI02E CONNECTION COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV CONNECTION COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI04E CONNECTION SEVERED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV CONNECTION SEVERED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI05E MESSAGE COMPLETED INTERRUPT RECEIVED FOR UNKNOWN PATHID

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received for a PATHID that does not exist.

User Response: Call IBM for support.

ICDNI06E MESSAGE COMPLETED INTERRUPT RECEIVED WHEN NOT EXPECTED

Explanation: An asynchronous interrupt for IUCV MESSAGE COMPLETED was received when not expected.

User Response: Call IBM for support.

ICDNI07E IPAUDIT FIELD IN INTERRUPT BUFFER SHOWS UNRECOVERABLE ERROR

Explanation: The IPAUDIT field in the IUCV interrupt field is invalid. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI08E INVALID CODE ENCOUNTERED IN IUCV EXIT ROUTINE

Explanation: An invalid code was returned by IUCV. The interrupt buffer will be dumped right after.

User Response: Call IBM for support.

ICDNI10E CONNECTION SEVERED INTERRUPT, USER DATA=dddd

Explanation: The TCP/IP address space severed the IUCV connection with NFS Coupler. User data `dddd` is an EBCDIC string which indicates the reason for severing the connection.

User Response: check that the TCP/IP started task was not terminated. If the TCP/IP started task is up and running and the error reoccurs, call IBM for support.

ICDNI11E RECVFROM CALL FAILED, ERRNO=dddd

Explanation: The TCP/IP RECVFROM call failed with SOCKET return code `dddd` decimal.

User Response: Call IBM for support.

ICDNI12E PERMANENT ERROR FOUND IN IUCV ANCHOR BLOCK: rrrrrrrr

Explanation: The permanent error flag was found in the main IUCV control block. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI13E CRITICAL SECTION EXECUTION NOT COMPLETED: rrrrrrrr

Explanation: Execution of the IUCV critical section was not completed during the last NFS Coupler call. The message was issued by routine `rrrrrrrr`.

User Response: If the application has experienced an ABEND right before, fix the application and resubmit. Otherwise, call IBM for support.

ICDNI14E IUCV INTERFACE ALREADY IN USE BY ANOTHER TASK IN THE ADDRESS SPACE

Explanation: The application is a multitasking application and multiple NFS Coupler files are being opened concurrently by more than one task. NFS Coupler is unable to support concurrent opens of multiple files from more than one task due to limitations of the IBM TCP/IP product.

User Response: Modify your application so that concurrent opens of NFS Coupler files is from one task only (see Limitations on page 51).

ICDNI15E MAXIMUM NUMBER OF IUCV CONNECTIONS EXCEEDED

Explanation: The application is opening concurrently too many NFS Coupler files (more than 256).

User Response: Modify your application so that no more than 256 NFS Coupler files are open at the same time (see Limitations on page 51).

ICDNI16E VMCF SubSystem NOT FOUND

Explanation: The VMCF SubSystem and the IBM TCP/IP product do not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI17E VMCF SubSystem CVT INVALID, BAD EYE CATCHER FOUND

Explanation: The VMCF SubSystem was found but its CVT has a bad eye catcher. Most probably the IBM TCP/IP product does not exist on the current installation.

User Response: Install the IBM TCP/IP product.

ICDNI18E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Gethostname failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI19E SOCKET CALL FAILED, ERRNO=dddd

Explanation: Socket call Socket failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI20E GETHOSTNAME CALL FAILED, ERRNO=dddd

Explanation: Socket call Bind failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI22E UNABLE TO BIND A RESERVED PORT (1023-512)

Explanation: NFS Coupler is unable to find a port available for binding in the range from 512 to 1023.

User Response: Check first that TCP/IP is working properly and resubmit the JOB. If the error persists, call IBM for support.

ICDNI25E CLOSE CALL FAILED, ERRNO=dddd

Explanation: Socket call Close failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI28E SELECT CALL FAILED, ERRNO=dddd

Explanation: Socket call Select failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI29E SENDTO CALL FAILED, ERRNO=dddd

Explanation: Socket call Sendto failed with SOCKET error `dddd`.

User Response: Call IBM for support.

ICDNI30E Software error: RPC queue overflow

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNI31E Software error: RPC queue empty

Explanation: An internal software error has been detected.

User Response: Call IBM for support.

ICDNM00E DD=dddddddd, Load Module ″ICDNMAP″ was no loaded from an authorized library

Explanation: Module ICDNMAP was not loaded from an authorized library. OPEN of file `dddddddd` will fail.

User Response: make sure module ICDNMAP has been copied into an authorized library which is part of the LINKLIST concatenation. Also make sure that no module with name ICDNMAP exists in either the JOBLIB or the STEPLIB concatenation.

ICDNM00I DD=dddddddd, READ_BIN, Last record padded with ″00″x

Explanation: The last record of a NFS Coupler file with fixed record format has been padded with ″00″x characters, since the length of the file read is not an exact multiple of the record length.

User Response: none.

ICDNM01E DD=dddddddd, READ_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in UNIX format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM02E DD=dddddddd, READ_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of binary characters from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM03E DD=dddddddd, WRITE_BIN failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of binary characters to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM05E DD=dddddddd, WRITE_UNIX failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for UNIX to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM08E DD=dddddddd, LIST_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for directory entries to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM09E DD=dddddddd, READ_DIR failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request for a UNIX file attribute to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail. File `dddddd` was opened to read the content of a UNIX directory.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM10E DD=dddddddd, MOUNT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: A request to mount a directory failed. The MOUNT server returned NFS error code `nnnnnnnn` mnemonics `ccc` decimal. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM11E DD=dddddddd, File= ″ffff″ is not a regular file

Explanation: NFS Coupler will not access UNIX file `ffff` because it is not a regular file. File `dddddddd` was opened for output or for input with an NFS Coupler input disposition of `FILE`. The OPEN for file `dddddddd` will fail.

User Response: for file `dddddddd` either change the UNIX file path to indicate a regular file or change the NFS Coupler input disposition to `DIR` to read the contents of a UNIX directory.

ICDNM12E DD=dddddddd, File= ″ffff″ already exists

Explanation: NFS Coupler refuses to write to UNIX file `ffff` because it already exists. File `dddddddd` was open with an NFS Coupler output disposition of `NEW`. The OPEN for file `dddddddd` will fail.

User Response: either change the NFS Coupler output disposition or change the UNIX file path in the JCL for file `dddddddd`.

ICDNM13E DD=dddddddd, Unable to look up file= ″ffff″, File= ″oooo″ is not a directory

Explanation: NFS Coupler was not able to look up file `ffff` under directory `oooo` because file `oooo` is not a directory. File `ffff` is an entry in the UNIX file name path specified. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd`.

ICDNM14E DD=dddddddd, File= ″ffff″ is not a directory

Explanation: NFS Coupler will not access file `ffff` since it is not a directory. File `dddddddd` was opened for input with NFS Coupler input disposition `DIR`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or change NFS Coupler input disposition to `FILE`.

ICDNM16E DD=dddddddd, LOOKUP failed, File= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX file `ffff`. UNIX file `ffff` is the last entry in the UNIX file path specified in the JCL for file `dddddddd`. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX file `ffff` already exists.

ICDNM17E DD=dddddddd, LOOKUP failed, Directory= ″ffff″ not found

Explanation: NFS Coupler did not find UNIX directory `ffff`. UNIX directory `ffff` is an intermediate entry in the UNIX file path specified in the JCL for file `dddddddd`. File `dddddddd` was opened for input. The OPEN for file `dddddddd` will fail.

User Response: correct the file path specified in the JCL for file `dddddddd` or make sure that UNIX directory `ffff` already exists.

ICDNM18E DD=dddddddd, RESET LENGTH failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to reset the length of UNIX file `ffff` to zero. UNIX file `ffff` is the last entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. File `dddddddd` was opened for output. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM19E DD=dddddddd, LOOKUP failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to get a file handle of UNIX file `ffff`. UNIX file `ffff` is an entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM20E DD=dddddddd, CREATE FILE failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX file `ffff`. UNIX file `ffff` is the last entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. File `dddddddd` was opened for output. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM21E DD=dddddddd, CREATE DIRECTORY failed, Directory= ″oooo″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to create UNIX directory `oooo`. UNIX directory `oooo` is an intermediate entry in the UNIX file path specified for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. File `dddddddd` was opened for output. The OPEN for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM23E DD=dddddddd, Invalid number of arguments

Explanation: The number of subparameters specified of the JCL keyword `SUBSYS=` of file `dddddddd` is invalid. The OPEN for file `dddddddd` will fail.

User Response: check the list of NFS Coupler subparameters and correct any error.

ICDNM24E DD=dddddddd, Host name required

Explanation: Host name was not specified as a subparameter of the JCL keyword `SUBSYS=` of file `dddddddd`. The OPEN for file `dddddddd` will fail.

User Response: add the host name to the JCL NFS Coupler subparameter list.

ICDNM25E DD=dddddddd, Host name is too long

Explanation: The host name specified as a subparameter of the JCL keyword `SUBSYS=` of file `dddddddd` exceed 16 characters of length. The OPEN for file `dddddddd` will fail.

User Response: correct the host name in the JCL NFS Coupler subparameter list.

ICDNM26E DD=dddddddd, MVS USERID of JOB is not SUPERUSER for NFS server host, USERID cannot be specified

Explanation: The MVS USERID was specified as a subparameter of the JCL keyword `SUBSYS=` of file `dddddddd`, but the JOB USERID is not superuser for the NFS server host. In order to be superuser the JOB USERID must map into Uid=0 for the NFS server host. The OPEN for file `dddddddd` will fail.

User Response: either run the JOB with APF authorization or delete the MVS USERID from the JCL NFS Coupler subparameter list.

ICDNM27E DD=dddddddd, MVS USERID is too long

Explanation: The MVS USERID was specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd, but its length exceeds 8 characters. The OPEN for file dddddddd will fail.

User Response: change the MVS USERID in the JCL NFS Coupler subparameter list to a correct value.

ICDNM28E DD=dddddddd, File path is required

Explanation: The UNIX file path has not been specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd, but it is required. The OPEN for file dddddddd will fail.

User Response: add the UNIX file path to the JCL NFS Coupler subparameter list.

ICDNM29E DD=dddddddd, Invalid input disposition

Explanation: The input disposition has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. This condition is detected even if the file is opened for output. The OPEN for file dddddddd will fail.

User Response: change the input disposition in the JCL NFS Coupler subparameter list to a correct value.

ICDNM30E DD=dddddddd, Invalid output disposition

Explanation: The output disposition has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. This condition is detected even if the file is opened for input. The OPEN for file dddddddd will fail.

User Response: change the output disposition in the JCL NFS Coupler subparameter list to a correct value.

ICDNM31E DD=dddddddd, Invalid file mode

Explanation: The UNIX file mode has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. The OPEN for file dddddddd will fail.

User Response: change the UNIX file mode in the JCL NFS Coupler subparameter list to a correct value.

ICDNM32E DD=dddddddd, Invalid data type

Explanation: The data type has been specified as a subparameter for the JCL keyword SUBSYS= of file dddddddd, and it is invalid. The OPEN for file dddddddd will fail.

User Response: change the data type in the JCL NFS Coupler subparameter list to a correct value.

ICDNM34E DD=dddddddd, Unable to map NFS server host name

Explanation: NFS Coupler is unable to find an entry in its mapping table for the NFS server host name specified as a subparameter of the JCL keyword SUBSYS= of file dddddddd. The OPEN for file dddddddd will fail.

User Response: correct the host name. If the error persists ask the NFS Coupler administrator to add an entry for the host name you are trying to access.

ICDNM35W DD=dddddddd, Unable to map MVS USERID=uuuuuuuu, %DEFUID will be used

Explanation: NFS Coupler is unable to find an entry in its the mapping table for the MVS USERID. The OPEN for file dddddddd will continue. Default MVS USERID ″%DEFUID″ will be used if define in module ICDNMAP for the NFS server being accessed.

User Response: ask the NFS Coupler administrator to add the JOB MVS USERID you are using, to the NFS Coupler mapping table. The JOB MVS USERID must be added for the domain in which the host you are trying to access resides.

ICDNM36E DD=dddddddd, File path has an invalid combination of ″/″ characters

Explanation: The user specified a file path name which is invalid either because it contains two or more consecutive ″/″ or because the first character is ″/″. The OPEN for file dddddddd will fail.

User Response: change the file path to a correct value and resubmit.

ICDNM37I DD=dddddddd, File ″ffff″ has been removed

Explanation: UNIX file with ffff full path name has been removed at close time as specified in the input disposition for file dddddddd.

ICDNM38W DD=dddddddd, File ″ffff″ not removed, ABEND in progress

Explanation: UNIX file with `ffff` full path name has not been removed as specified in the input disposition for file `dddddddd`. The reason for bypassing remove is that the close routine detected an ABEND for the task executing the close routine.

ICDNM39W DD=dddddddd, Remove failed, File= ″ffff″, Nfs rc= nnnnnnnn (ccc)

Explanation: NFS Coupler tried to remove the NFS server file `ffff` as specified by the input disposition for file `dddddddd`. The request failed with NFS return code `nnnnnnnn` mnemonics `ccc` decimal. Close processing for file `dddddddd` will continue.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM40E DD=dddddddd, READ_PRINT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in PRINT format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM41E DD=dddddddd, WRITE_PRINT failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Write of characters formatted for PRINT to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM42E DD=dddddddd, Input buffer too short, record has been truncated

Explanation: The buffer used for a read operation is too short. The data available from the UNIX file has been truncated. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: use a longer record length to read the UNIX file.

ICDNM43E DD=dddddddd, Unable to write a file larger than 4294967295 bytes

Explanation: NFS protocol limitations have been exceeded. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: break the MVS file being written to UNIX into multiple smaller files.

ICDNM44E DD=dddddddd, READ_DOS failed, Nfs rc= nnnnnnnn (ccc)

Explanation: Read of characters in DOS format from the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O GET or READ operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM45E DD=dddddddd, WRITE_DOS failed, Nfs rc= nnnnnnnn (ccc)

Explanation: An attempt to write characters formatted for DOS to the NFS server failed. The NFS server returned error code `nnnnnnnn` mnemonics `ccc` decimal. The I/O PUT or WRITE operation for file `dddddddd` will fail.

User Response: check the NFS error code with the list given at the beginning of this chapter, and take action accordingly.

ICDNM46E DD=dddddddd, Checkpoint/Restart not supported for Input Directory files

Explanation: A checkpoint was attempted while file `dddddddd` used to read directory entries is open. Checkpoint/Restart is not supported for this type of files. Checkpoint will fail.

User Response: make sure that files used to read directory entries are closed when a checkpoint is taken.

ICDNM47W DD=dddddddd, RESTART: Input file is different than the one used in last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS input file `dddddddd` has a different file system ID or file ID than those recorded during the last checkpoint. This can happen if the input file on the NFS server was deleted and replaced with a different file before restart was attempted.

User Response: operator must respond ’Y’ to continue restart or ’N’ to abnormally terminate.

ICDNM48W DD=dddddddd, RESTART: Input file has been updated since last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS input file `dddddddd` has been updated since the last checkpoint. This will happen, for example, if the input file on the NFS server was updated to remove a bad input record which caused an ABEND.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM49W DD=dddddddd, RESTART: Output file has been changed since last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS output file `dddddddd` has a different file system ID or file ID than those recorded during the last checkpoint. This can happen if the output file on the NFS server has been deleted and replaced with a different file before restart was attempted. In general this is a problem due to operator error on the NFS server.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM50W DD=dddddddd, RESTART: Output file is shorter than it was at last Checkpoint, OK to continue (Y/N) ?

Explanation: During restart, NFS Coupler detected that the NFS output file `dddddddd` length is less than it was at the last checkpoint. This can happen if the output file on the NFS server was updated before restart was attempted. In general this is a problem due to operator error on the NFS server.

User Response: operator must respond 'Y' to continue restart or 'N' to abnormally terminate.

ICDNM51E DD=dddddddd, Checkpoint/Restart not supported for Input Directory files

Explanation: A restart was attempted while file `dddddddd` used to read directory entries is open. Checkpoint/Restart is not supported for this type of files. Restart will fail. This will happen if the JCL DD statement for the NFS Coupler file `dddddddd` has been changed so that the input file is now a directory.

User Response: correct the DD statement for the file and resubmit the JOB to restart.

ICDNM52I DD=dddddddd, aaaaaa: terminated due to operator response

Explanation: The operator, when prompted to enter 'Y' to continue processing or 'N' to abnormally terminate, entered 'N'. Processing will be abnormally terminated by `aaaaaa.` Currently `aaaaaa` can be `CHECKPOINT` or `RESTART.`

User Response: none.

ICDNM53E DD=dddddddd, Default MVS USERID=%DEFUID not defined

Explanation: NFS Coupler is unable to find defualt MVS USERID ″%DEFUID″. The OPEN for file `dddddddd` will fail.

User Response: ask the NFS Coupler administrator to add either the JOB MVS USERID or the defualt MVS USERID ″%DEFUID″ to the NFS Coupler mapping table. Either one must be added for the domain in which the host you are trying to access resides.

ICDNM71W DD=dddddddd, Nested checkpoint request

Explanation: A checkpoint request was invoked while another I/O request was in progress for the same DCB of file `dddddddd.` Checkpoint will fail.

If checkpoint was issued in an asynchronous exit, such as timer or VTAM exit, or a recovery ESTAE exit the program issuing the close is in error.

User Response: modify the program and make sure checkpoint are only by the mainline task code.

ICDNM72E DD=dddddddd, Close entered by MVS normal termination routine, standard close processing bypassed

Explanation: The close routine for file `dddddddd` has been entered by the MVS termination routine. Standard close processing is bypassed due to the fact that at this stage the IUCV connection with the TCP/IP address space has been already severed by MVS. For files opened for input, if the `REMOVE` disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

This message is issued when the terminating program does not close all open files.

User Response: change the program so that all open files are closed before terminanting.

ICDNM73W DD=dddddddd, Nested close request

Explanation: A close request was invoked while another I/O request was in progress for the same DCB of file `dddddddd`. Standard close processing is bypassed. For files opened for input, if the `REMOVE` disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

If the close was issued in an asynchronous exit, such as timer or VTAM exit, the program issuing the close is in error. If the close was issued in an ESTAE recovery or retry routine after an abend during an I/O request to NFS Coupler, this message should be expected.

User Response: if this message is issued because of programming error, correct the program and submit again.

ICDNM74W DD=dddddddd, Close entered by MVS abnormal termination routine, standard close processing bypassed

Explanation: The close routine for file `dddddddd` has been entered by the MVS termination routine due to an abend. Standard close processing is bypassed due to the fact that at this stage the IUCV connection with the TCP/IP address space has been already severed by MVS. For files opened for input, if the `REMOVE` disposition was specified, the UNIX file will not be removed. For files opened for output, the last few records written will be lost.

User Response: correct the problem which triggered the abend and resubmit.

ICDNM75E DD=dddddddd, Nested output request detected

Explanation: An output request was invoked while another I/O request was in progress for the same DCB of file `dddddddd`. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: correct the program and submit again.

ICDNM76E DD=dddddddd, Nested input request detected

Explanation: An input request was invoked while another I/O request was in progress for the same DCB of file `dddddddd`. This is an error and it can happen when I/O requests are issued in asynchronous exits such as timer or VTAM exit.

User Response: correct the program and submit again.

ICDNM77E DD=dddddddd, asynchronous check invoked

Explanation: The MVS SSAM routines requested an asynchronous check to NFS Coupler for file `dddddddd`. This is an error. The caller is abnormally terminated.

User Response: Call IBM for support.

ICDNM80E Trial license has expired. Call IBM for support.

Explanation: NFS Coupler trial license has expired.

User Response: Call IBM for an extension if you need one.

ICDNM81E License not available on this machine. Please call IBM for support.

Explanation: NFS Coupler product license is not available on the current 370/390 machine.

User Response: Call IBM for for a license extension on the current machine.

ICDNM82E Trial license key is corrupt. Please call IBM for support

Explanation: NFS Coupler trial license key is invalid.

User Response: Call IBM for support.

ICDNM90I DD=dddddddd, MVS USERID= uuuuuuu

Explanation: The MVS userid used by NFS Coupler for file `dddddddd`, is `uuuuuuu`. This message is issued when a previous error has been detected for file `dddddddd`.

User Response: none.

ICDNM91I DD=dddddddd, Local host name= hhhhhh

Explanation: The local (MVS) TCP/IP host name used by NFS Coupler for file `dddddddd`, is `hhhhhh`. This message is issued when a previous error has been detected for file `dddddddd`.

User Response: none.

ICDNM92I DD=dddddddd, NFS server host= iii, jjj, kkk, lll (tttt)

Explanation: The NFS sever TCP/IP host name used by NFS Coupler for file `dddddddd`, has up to four IP address(es) `iii`, `jjj`, `kkk`, `lll`. Host type `tttt` can be UNIX or DOS. This message is issued when a previous error has been detected for file `dddddddd`.

User Response: none.

ICDNM93I DD=dddddddd, Uid= uuu, Gid= ggg

Explanation: The UNIX Uid and Gid used by NFS Coupler for file dddddddd, are uuu and ggg. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM94I DD=dddddddd, Mount directory= oooooo

Explanation: The NFS mount directory used by NFS Coupler for file dddddddd, is oooooo. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM95I DD=dddddddd, File path= ffffff

Explanation: The UNIX file path used by NFS Coupler for file dddddddd, is ffffff. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM96W DD=dddddddd, Number of RPC retransmits is cc% of total number of RPC calls

Explanation: While closing file dddddddd, NFS Coupler detected the the percentage of RPC calls that was retransmitted was cc percent. This message is issued only if cc is larger than 1.

User Response: Check if this message is issued consistently. If this is the case the network connection between MVS and the NFS server is quite slow and some tuning is needed (see NFS Coupler performance tuning on page 25).

ICDNM97I DD=dddddddd, Timing parameters= t.t, rrr, ooo

Explanation: The timing parameters used by NFS Coupler for file dddddddd, to access data on the NFS servers are timeout=t.t, retransmit=rrr and overlap=ooo. This message is issued when a previous error has been detected for file dddddddd.

User Response: none.

ICDNM99E DD=dddddddd, Exiting due to severe error

Explanation: NFS Coupler terminates execution for file dddddddd due to a severe error. This message is normally preceded by other error messages.

User Response: check previous error messages for file dddddddd.

ICDNR00E DD=dddddddd, PORTMAP, timeout, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed due to timeout. The request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: check that the NFS server host is accessible with the TSO ping command. Also make sure that the PORTMAP server on the NFS server host is up and running.

ICDNR01E DD=dddddddd, PORTMAP, RPC mismatch, Host supports= (ll, hh), Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed because the PORTMAP deamon on the NFS server does not support RPC version 2 which is used by NFS Coupler. The lowest RPC version supported on the UNIX host is ll and the highest is hh. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: use a PORTMAP level on the UNIX host which supports RPC version 2.

ICDNR02E DD=dddddddd, PORTMAP, Authorization failed, Auth. Stat= ss, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request failed because the UNIX portmap deamon could not authorize the request. The authorization stat ss gives a reason for the failure. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR03E DD=dddddddd, PORTMAP, Request rejected, Reject Stat= ss, Prog= pppppp, Vers= vv

Explanation: A PORTMAP request was rejected by the UNIX host. The reject stat is ss. The PORTMAP request was for program pppppp and version vv. The OPEN for file dddddddd will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR04E DD=dddddddd, PORTMAP, Not accepted, Acc. Stat= ss, Prog= pppppp, Vers=vv

Explanation: A PORTMAP request for program `pppppp` and version `vv` was rejected by the UNIX host. The accept stat `ss` gives a reason for the failure. Its possible values can be as follows: .strttbl 100 LR 0 10 * LR 0 0 .cell .bo Value .cell .bo Meaning .cell `1` .cell The PORTMAP program (program 100000) is not available .cell `2` .cell The PORTMAP version (version 2) is not supported .cell `3` .cell The PORTMAP procedure (procedure 3) is not supported .cell `4` .cell PORTMAP received bad parameters .endtbl The PORTMAP request was for program `pppppp` and version `vv`. The OPEN for file `dddddddd` will fail.

User Response: make sure that UDP port 111 on the UNIX host is used by PORTMAP and not by another deamon. If the error persists, call IBM for support.

ICDNR05E DD=dddddddd, PORTMAP, zero port returned, Prog= pppppp, Vers= vv

Explanation: A zero UDP port number was returned to a PORTMAP request. The PORTMAP request was for program `pppppp` and version `vv`. Program `pppppp` (100005 for MOUNT, 100003 for NFS), is not up and running on the UNIX host. The OPEN for file `dddddddd` will fail.

User Response: make sure that both the MOUNT and NFS deamons are up and running on the UNIX host.

ICDNR07E DD=dddddddd, RPCWAIT, timeout, Prog= pppppp, Vers= vv, Proc= rr

Explanation: A timeout was detected while waiting for an RPC response. The original call was for program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: check if the UNIX host is accessible with the TSO `ping` command. Also make sure that both the MOUNT and NFS deamons are up and running on the UNIX host.

ICDNR08E DD=dddddddd, RPCWAIT, RPC mismatch, Host supports= (ll, hh), Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call failed because program `pppppp` does not support RPC version 2 which is used by NFS Coupler. The lowest RPC version supported on the UNIX host is `ll` and the highest is `hh`. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The OPEN for file `dddddddd` will fail.

User Response: use NFS and MOUNT software levels on the UNIX host which support RPC version 2.

ICDNR09E DD=dddddddd, RPCWAIT, Authorization failed, Auth. Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: A RPC call failed because the NFS server did not authorize the call. The authorization stat `ss` gives the reason for the failure. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The I/O request for file `dddddddd` will fail.

User Response: Call IBM for support.

ICDNR10E DD=dddddddd, RPCWAIT, Request rejected, Reject Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call was rejected by the NFS server. The reject stat is `ss`. The RPC call was for program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: Call IBM for support.

ICDNR11E DD=dddddddd, RPCWAIT, Call not accepted, Acc. Stat= ss, Prog= pppppp, Vers= vv, Proc= rr

Explanation: An RPC call was rejected by the NFS server. The accept stat `ss` gives a reason for the failure. Its possible values can be as follows: .strttbl 100 LR 0 10 * LR 0 0 .cell .bo Value .cell .bo Meaning .cell `1` .cell Program `pppppp` is not available (100003 is NFS, 100005 is MOUNT). .cell `2` .cell Version `vv` is not supported .cell `3` .cell Procedure `rr` is not supported .cell `4` .cell program `pppppp` received bad parameters .endtbl The RPC call was to program `pppppp`, version `vv` and procedure `rr`. The I/O operation for file `dddddddd` will fail.

User Response: make sure that the MOUNT and NFS deamons are up and running on the UNIX host. Also, on the UNIX host, make sure that the MOUNT software level supports version 1 and that NFS software level supports version 2. If the error persists, call IBM for support.

ICDNR99E DD=dddddddd, RPCCALL, software error, overlapping calls

Explanation: A software error was detected by NFS Coupler. The I/O operation for file `dddddddd`, will fail.

User Response: Call IBM for support.

ICDNU00E Bad return code from CHKPT macro.

Explanation: The IEBGENER totaling ICDNCHKP, which issues checkpoints during copy, detected a bad return code from the checkpoint macro. The JOB is abnormally terminated with code U0000. Registers R15 and R14 will contain the return and reason code from the checkpoint macro.

User Response: check the return and reason codes in registers R15 and R14 to identify the problem. After fixing the problem resubmit the JOB.

ICDNU01I No CHECKPOINT time interval specified, 10 minutes will be used

Explanation: No time interval was specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy. The default value of 10 minutes will be used.

User Response: none.

ICDNU02W Invalid CHECKPOINT time interval specified, 10 minutes will be used

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is invalid. Processing continues, the default value of 10 minutes will be used.

User Response: correct interval value for later runs.

ICDNU03W Zero CHECKPOINT time interval specified, 10 minutes will be used

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is zero. Processing continues, the default value of 10 minutes will be used.

User Response: change the interval value to a positive number for later runs.

ICDNU04I CHECKPOINT time interval specified is″ mmm minute(s)

Explanation: The time interval specified to the IEBGENER totaling exit ICDNCHKP, which issues checkpoints during copy, is `mmm` minutes. Every `mmm` minutes a check point will be taken by ICDNCHKP.

User Response: none.

ICDNX00E DD=dddddddd, rrrrrr failed, diagnostic string= ssssss, Buffer Loc= llllll

Explanation: XDR routine `rrrrrr` could not encode or decode XDR data. Diagnostic string `ssssss` identifies the caller. Decimal number `llllll` indicates the memory location of the buffer. The I/O operation for file `dddddddd`, will fail.

User Response: Call IBM for support.

ICDSA0E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME NOT SPECIFIED

Explanation: Allocation of DD name `dddddddd` to SubSystem `ssss` failed because OPEN/CLOSE interface module name was not specified as second subparameter of the JCL `SUBSYS=` keyword. For NFS Coupler, the module name must be `ICDNOPCL`.

User Response: Add the second subparameter to keyword `SUBSYS=` by specifying `ICDNOPCL`and resubmit the JOB.

ICDSA1E SS=ssss, DD=dddddddd, ALLOCATION FAILED, OPEN/CLOSE MODULE NAME IS INVALID

Explanation: Allocation of DD name `dddddddd` to SubSystem `ssss` failed because OPEN/CLOSE interface module name, which is the second subparameter of the JCL `SUBSYS=` keyword, was not a valid module name. For NFS Coupler the module name must be ICDNOPCL.

User Response: Correct the second subparameter to keyword `SUBSYS=`by specifying `ICDNOPCL`and resubmit the JOB.

ICDSC00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND, ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CLOSE failed due to abend in module `mmmmmmmm`. The abend code is `ccccc`. The content of register 15 at the time of abend was `rrrrrrrr` (hex) and the PSW was `xxxxxxxx xxxxxxxx` (hex).

User Response: Call IBM for support.

ICDSH00E ssss, DD=dddddddd, CHECKPOINT FAILED DUE TO ABEND, ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: CHECKPOINT failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSI00E - ssss - BLDL FAILED FOR MODULE=IORRCOMM, RC=cccccccc

Explanation: During initialization of SubSystem ssss module IORRCOMM was not found. This message is issued only when using IORR utilities to add or replace an IORR SubSystem.

User Response: Make sure that module IORRCOMM is available in the LINKLIB concatenation.

ICDSI00I - ssss - VERSION 1 RELEASE 2, SUBSYSTEM INITIALIZATION STARTED * (C) COPYRIGHT FOLIUM INC. 1994, 1995 *** ALL RIGHTS RESERVED**

Explanation: Initialization of SubSystem ssss has started.

User Response: None.

ICDSI01E - ssss - LOAD FAILED, ABEND=cccccccc, R15=rrrrrrrr

Explanation: The initialization routine of SubSystem ssss was not able to load module=IORRCOMM due to an abend with code cccccccc (hex). Register 15 at the time of abend was rrrrrrrr (hex).

User Response: Call IBM for support.

ICDSI01I - ssss - INITIALIZATION COMPLETED

Explanation: Initialization of SubSystem ssss has completed.

User Response: None.

ICDSO00E ssss, DD=dddddddd, CLOSE FAILED DUE TO ABEND, ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: OPEN failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW was xxxxxxxx xxxxxxxx (hex).

User Response: If module name mmmmmmmmis not ICDNOPCL,then specify ICDNOPCLas the second subparameter of the JCL keyword SUBSYS=and resubmit the JOB. Otherwise, call IBM for support.

ICDSR00E ssss, DD=dddddddd, RESTART FAILED DUE TO ABEND, ACCESS MODULE NAME=mmmmmmmm ABEND CODE=ccccc, R15=rrrrrrrr, PSW=xxxxxxxx xxxxxxxx

Explanation: RESTART failed due to abend in module mmmmmmmm. The abend code is ccccc. The content of register 15 at the time of abend was rrrrrrrr (hex) and the PSW is xxxxxxxx xxxxxxxx (hex).

User Response: Call IBM for support.

ICDSU00E SS SPECIFIED ALREADY EXISTS

Explanation: Module IORRSSAD cannot add a new IORR SubSystem since the name specified in the parm field identifies a SubSystem which already exists. The new SubSystem is not installed.

User Response: Change the name specified in the parmfield and resubmit the JOB.

ICDSU01E SS NAME SPECIFIED IS INVALID

Explanation: Module IORRSSAD detected an invalid SubSystem name specified in the parm field. The specified SubSystem is not installed.

User Response: Change the name specified in the parmfield to a valid SubSystem name and resubmit the JOB.

ICDSU02E INVALID MVS CONTROL BLOCKS ENCOUNTERED

Explanation: Module IORRSSAD detected invalid control blocks in memory. The reason most likely is a release of MVS which is too old. The existing SubSystem is not replaced.

User Response: Wait until the next IPL to install the SubSystem using SYS1.PARMLIB(IEFSSNxx).

ICDSU03E SS SPECIFIED DOES NOT EXIST

Explanation: Module IORRSSRP did not find the SubSystem specified in the parm field. The SubSystem specified is not replaced.

User Response: Specify the name of an existing IORR SubSystem in the parmfield and resubmit the JOB.

ICDSU04E SS SPECIFIED EXISTS BUT IT HAS NOT BEEN INITIALIZED BY IORRINIT

Explanation: Module IORRSSRP found the SubSystem specified in the `parm` field, but it detected that the SubSystem had not been initialized by module IORRINIT. The SubSystem is not replaced.

User Response: Specify the name of an existing IORR SubSystem in the `parm`field and resubmit the JOB.

ICDSU05E SS NAME SPECIFIED IS INVALID

Explanation: Module IORRSSRP detected an invalid SubSystem name specified in the `parm` field. The SubSystem specified is not replaced.

User Response: Change the name specified in the `parm`field to a valid SubSystem name and resubmit the JOB.

Appendix B. Prerequisites and Limitations

This appendix describes prerequisites and limitations with which you should be familiar before installing IBM Network Data Couplers for MVS/ESA and OS/390.

Prerequisites

The prerequisites for using IBM Network Data Couplers for MVS/ESA and OS/390 are listed below:

MVS environment:

- MVS/ESA V4 R2.2 or later
- IBM TCP/IP V2 RI (base only) or INTERLINK TCP/IP V3 RI (base only)
- RACF V1 R9 or later or any equivalent security system

OS/390 environment:

- Any release

NFS Coupler Limitations

NFS Coupler has the following limitations when used with the IBM TCP/IP product:

- A maximum of 256 files may be open concurrently by the same JOB.
- When a JOB uses multitasking, all NFS Coupler files opened concurrently, must be opened by the same task.
- NFS Coupler might not work with JOBs that directly invoke sockets functions.
- Two started tasks with the same JOB name cannot concurrently use NFS Coupler.
- Checkpoint/Restart is not supported for input files that are directories.
- UNIX symbolic links are not supported.

NFS Coupler has the following limitations when used with the INTERLINK TCP/IP product:

- A maximum of 16 NFS Coupler files can be open concurrently from one task. There is no limit on the total number of NFS Coupler files that can be open concurrently when using multitasking.
- Checkpoint/Restart is not supported for input files that are directories.
- UNIX symbolic links are not supported.

RSH Coupler Limitations

RSH Coupler has the following limitations:

- The maximum number of RSH Coupler APPC transactions being executed concurrently is 254.
- Within a RSH Coupler APPC transaction, only one I/O request at a time for a RSH Coupler file can be in progress. If more that one I/O request is in progress an error could occur. This is not a problem if the RSH Coupler APPC transaction is a standard single task application.
- When a RSH Coupler APPC transaction uses multitasking and opens RSH Coupler files from different tasks, *all* RSH Coupler files must be closed by the time the task which opened the *first* RSH Coupler file terminates.

Prerequisites and Limitations

- Maximum record length supported for files of type STDPARM is 256 bytes for fixed length and 260 for varying length.
- The **rsh** protocol with only one connection is not supported. This is relevant only to programs, on the client machine, which directly call the `rcmd()` C functions. It has no effect on users of the standard `rsh` command.
- Use of the **rsh** command without specifying a remote command is not supported.
- Maximum length of data within each record for print STDERR and STDOUT files cannot exceed 256. For fixed record files the length is checked after stripping trailing blanks.

Appendix C. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10584

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including warranties of merchantability and fitness for a particular purpose.

Trademarks and Service Marks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM Network Data Couplers

MVS

MVS/ESA

OS/390

OS/2

MVS MULTI-TRAN

MVS Checkpoint/Restart

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Bibliography

Related Publications

The following IBM publications are referenced in *IBM Network Data Couplers User's Guide*:

MVS/ESA JCL Reference, GC28-1654

MVS/ESA JCL User's Guide, C28-1653

MVS/ESA Planning: APPC Management, GC28-1503

MVS/ESA System Programming Library: Application Development Guide, GC28-1852

MVS/ESA Application Development Guide: Authorized Assembler Language Programs, GC28-1467

DFSMS/MVS Checkpoint/Restart, SC26-4907

TSO Extension Version 2: Procedures Language MVS/REXX Reference, SC28-1883

TSO Extension Version 2: Procedures Language MVS/REXX User's Guide, SC28-1882

Glossary

Note: Definitions come from either the IBM Dictionary of Computing or the Microsoft Press Computer Dictionary, 3rd edition.

data definition (DD) statement. A job control statement describing a data set associated with a specific job step.

daemon. A program associated with UNIX systems that performs a housekeeping or maintenance utility function without being called by the user. A daemon sits in the background and is activated only when needed, for example, to correct an error from which another program cannot recover.

event files. A file that stores notifications indicating irregularities or occurrences in the operation of physical elements of a network.

filter. A program or set of features within a program that reads its standard or designated input, transforms the input in some desired way, and then writes the output to its standard or designated output destination.

host name. The name of a specific server on a specific network within the Internet.

Internet Protocol address (IP address). A 32-bit (4-byte) binary number that uniquely identifies a host (computer) connected to the Internet to other Internet hosts, for the purposes of communication through the transfer of packets.

mount. To make a physical disk or tape accessible to a computer's file system. The term is most commonly used to describe accessing disks in UNIX-based computers.

NFS domain. Network File System domain. A group of UNIX hosts that map the same logins into the same user Ids (Uids) and group Ids (Gids).

password file. In a UNIX system, the file where password information resides. The full path name of the file is /etc/passwd.

unmount. To remove a disk or tape from active use.

Index

Readers' Comments — We'd Like to Hear from You

IBM Network Data Couplers
For MVS/ESA and OS/390
IBM Network Data Couplers User's Guide

Publication No. GC31-8701-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
GC31-8701-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC
27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

GC31-8701-00

Cut or Fold
Along Line

Program Number: 5655-A99

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC31-8701-00