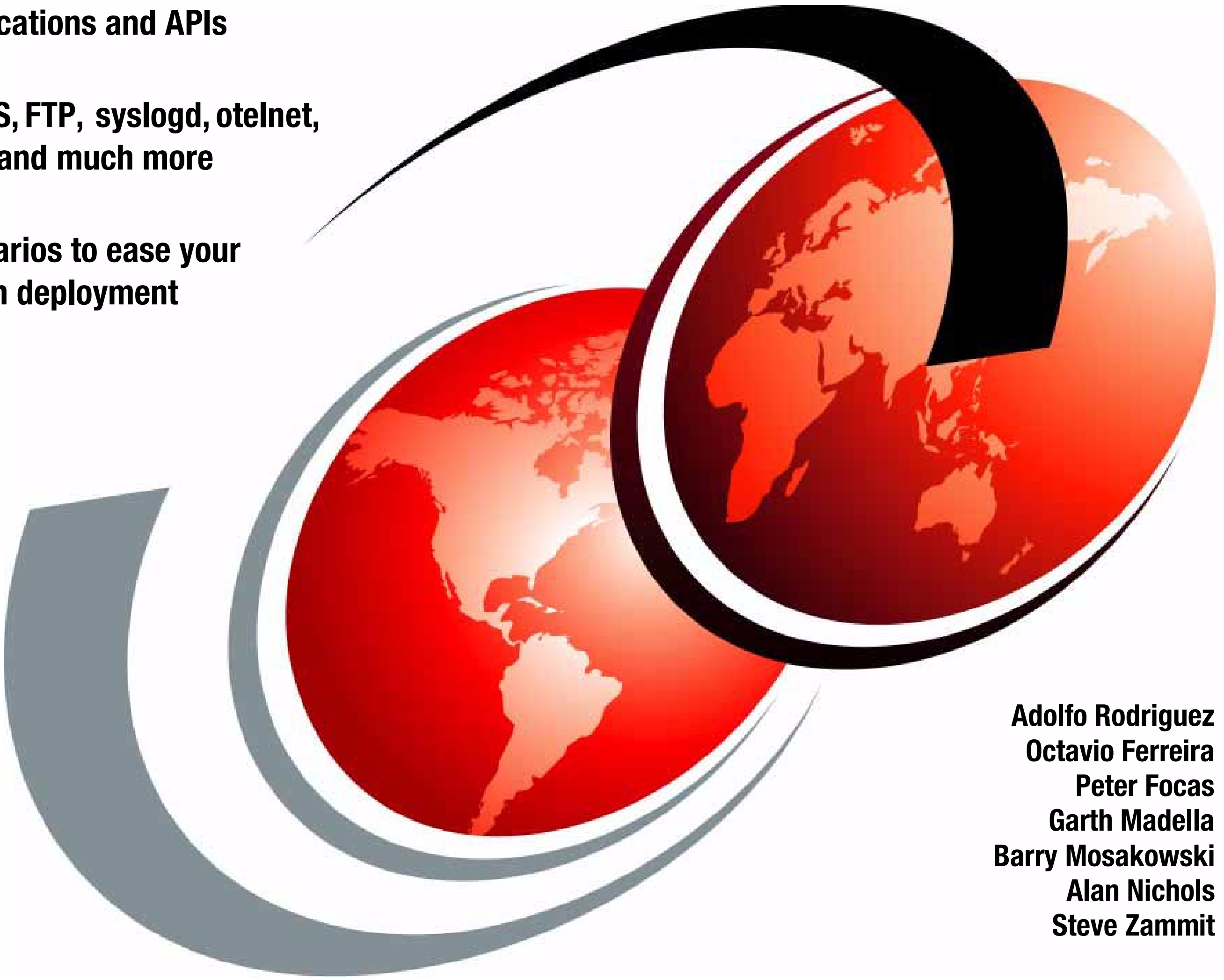
IBM

Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications

Provides a detailed survey of CS for z/OS applications and APIs

Covers DNS, FTP, syslogd, otelnet, sendmail, and much more

Uses scenarios to ease your application deployment



Adolfo Rodriguez
Octavio Ferreira
Peter Focas
Garth Madella
Barry Mosakowski
Alan Nichols
Steve Zammit

Redbooks

ibm.com/redbooks

International Technical Support Organization

Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications

October 2002

SG24-5228-03

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page xi.

Fourth Edition (October 2002)

This edition applies to Version 1 Release 2 of Communications Server for z/OS.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Notices

Trademarks

Preface

The Internet and enterprise-based networks have led to the rapidly increasing reliance upon TCP/IP implementations. The zSeries platform provides an environment in which critical business applications flourish. The demands placed on these systems is ever-increasing and such demands require a solid, scalable, highly available, and high-performance Operating System and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality. The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS.

Volume 2 covers the UNIX applications shipped as part of Communications Server for z/OS IP. In this volume, we classify z/OS applications and provide a detailed survey of the protocols and implementation issues associated with each. These applications provide a rich set of functionality, including remote execution with otelnet and file transfers with FTP and TFTP. In addition, we cover important network functions such as DNS, Dynamic IP, syslogd, and NFS. We provide scenario-based discussions to aid in application deployment.

Because of the varied scope of CS for z/OS, this volume is not intended to cover all aspects of the topic. The main goal of this volume is to provide an insight into the different applications provided by CS for z/OS and, more specifically, into the protocols they use and the mechanisms to deploy them. For more information, including applications available with CS for z/OS IP, please reference the other volumes in the series. These are:

- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227
- *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*, SG24-6516
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Adolfo Rodriguez is a Senior I/T Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of TCP/IP. Before joining the ITSO, Adolfo worked in the design and development of CS for z/OS, in RTP, NC. He holds a B.A. degree in Mathematics and B.S. and M.S. degrees in Computer Science from Duke University. He is currently pursuing the Ph.D. degree in Computer Science at Duke University, with a concentration on Networking Systems.

Octavio Ferreira is a Senior I/T Specialist in IBM Brazil. He has 17 years of experience in IBM software support. His areas of expertise include z/OS, VM, SNA, TCP/IP, LAN and WAN. For the last eight years, he has worked at the Area Program Support Group providing guidance and support to customers and designing networking solutions such as SNA to APPN migration and e-business solutions.

Peter Focas is a Network Systems Programmer in New Zealand. He has 12 years of experience in the SNA and TCP/IP networking field. His areas of expertise include the design and setup of SNA/SNI networks, configuration of secure TCP/IP servers using SSL/TLS, and digital certificate management.

Garth Madella is an Information Technology Specialist with IBM South Africa. He has 17 years of experience in the System/390 networking software field. He has worked with IBM for six years. His areas of expertise include VTAM, SNA, TCP/IP, and sysplex. He has written extensively on TCP/IP, sysplex, and Enterprise Extender issues.

Barry Mosakowski is a Software Engineer working in Raleigh, North Carolina at IBM's RTP site. He has eight years of experience in TCP/IP and SNA networking. He holds an MSE from Rensselaer Polytechnic Institute in Troy, NY. His areas of expertise include design, setup, and debugging of the Communication Server for z/OS TCP/IP to include the Telnet server, device drivers, socket applications, and Policy Agent.

Alan Nichols is an Independent Consultant living in Germany. He has 20 years of experience in MVS and z/OS and 10 years in UNIX and IP. He is currently working with last level IP/USS support in T-Systems.

Steve Zammit is an Advisory I/T specialist with the IBM Software Support Center based in Vancouver, Canada. Steve has 17 years of experience with IBM systems and networking, currently specializing in CS for z/OS TCP/IP. He holds a BSc degree in Applied Physics from Portsmouth Polytechnic, UK.

Thanks to the following people for their contributions to this project:

Bob Haimowitz, Jeanne Tucker, Margaret Ticknor, Tamikia Barrow, Gail Christensen, Linda Robinson
International Technical Support Organization, Raleigh Center

Jeff Haggar, Bebe Isrel, Van Zimmerman, Jerry Stevens, Tom Moore, Robert Perrone, Michael Fitzpatrick, Gus Kassimis, Dinakaran Joseph
Communications Server for z/OS Development, Raleigh, NC

Notice

This publication is intended to help customers implement and configure Communications Server for z/OS IP. The information in this publication is not intended as the specification of any programming interfaces that are provided by Communications Server for z/OS. See the PUBLICATIONS section of the IBM Programming Announcement for Communications Server for z/OS for more information about what publications are considered to be product documentation.

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to the address on page ii.

Part 1

Introduction

Beyond the functionality required of a TCP/IP stack implementation, Communications Server for z/OS IP provides a number of application implementations. These are largely classified into two broad categories:

- MVS applications, described in *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- UNIX applications, the subject of this book

In this book, we further classify the CS for z/OS IP UNIX applications. These include productivity applications such as otelnetd, X-Windows, remote execution, and sendmail. We also describe file related protocols and applications such as FTP, NFS, and TFP, and boostrapping functions such as DHCP and DNS. Finally, we complete our survey of UNIX applications with a discussion of utility applications such InetD and syslogd. While CS for z/OS IP provides a number of applications for policy and network management, we include a discussion of these in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839.

1

z/OS UNIX application programming interfaces

This chapter is a short introduction to the application programming interfaces that are delivered with Communications Server for z/OS IP.

1.1 UNIX System Services APIs

There is one physical file system for the AF_UNIX addressing family and there is another for the AF_INET addressing family. Communications Server for z/OS V1R2 IP has IPv6 API support for IPv6 application development and allows coexistence with IPv4. In a UNIX System Services environment, the following addressing families exist:

Family	Description
AF_INET	The Internet addressing family, also referred to as the Internet domain (IPv4).

This addressing family is used within the TCP/IP domain to identify sockets on IP hosts. A socket address in AF_INET consists of the following:

Family	Half-word binary with a value of 2, which identifies the socket address as belonging to the AF_INET addressing family.
Port	Half-word binary with port number that identifies the process.
IP address	Full-word binary with IP address of IP host in network byte order format.
Reserved	Eight reserved bytes.

The following is an example of an AF_INET address that represents the Telnet Server (port number 23) on an IP host with the IP address of 9.24.104.126:

```
{AF_INET 23 9.24.104.126}
```

AF_INET6	The Internet addressing family, also referred to as the Internet domain (IPv6).

This addressing family is used within the TCP/IP domain to identify sockets on IP hosts. A socket address in AF_INET6 has a very similar structure to AF_INET except for the size of the IP address, which is 128 bits (16 bytes) versus only 32 bits (4 bytes). AF_INET6 sockets communicate over an IPv4 network after a special form of address map processing concludes. The result is called an IPv4-mapped IPv6 address. An IPv4-mapped IPv6 address has 10 bytes of zeros then 'FFFF'X followed by the IPv4 address in the low-order four bytes. The TCP/IP stack does not track AF_INET and AF_INET6 sockets separately. There is one socket table where all sockets are essentially treated the same.

Note: AF_INET6 is not supported in a Common INET environment (CINET). The supported IPv6 address formats are IPv4-mapped IPv6, IN6ADDR_LOOPBACK, and IN6ADDR_ANY. Other IPv6 address formats are not supported.

The AF_INET physical file system does rely on other products to provide the AF_INET transport services. The IBM products that you can use as AF_INET transport providers in the UNIX System Services environment are:

- Communications Server for z/OS IP
- z/OS AnyNet

AF_UNIX The UNIX addressing family; also referred to as the UNIX domain.

You can use AF_UNIX with UNIX System Services sockets, where this addressing family is used for interprocess communication between UNIX System Services processes within one MVS operating system. The syntax of an AF_UNIX address family is shown below:

Family	Half-word binary with a value of 1, which identifies the socket address as belonging to the AF_UNIX addressing family.
Path	108 characters defining a path name (similar to a hierarchical file system path name) by which this local process wants to be known by other local processes.

The following is an example of an address in the AF_UNIX addressing family:

```
AF_UNIX /u/xyz/testsrv
```

The UDS physical file system is used to handle socket requests for the AF_UNIX address family of sockets.

UNIX System Services implements support for a given addressing family through different physical file systems. The UDS physical file system is self-contained within UNIX System Services and does not rely on other products to implement the required functions. AF_UNIX includes security enhancements with CS for OS/390 V2R10 IP. Refer to *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800 for more information. The following statements illustrate coding of BPXPRMxx member in SYS1.PARMLIB for the addressing families:

```
FILESYSTYPE TYPE(UDS)  ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(1000)
        TYPE(UDS)
FILESYSTYPE TYPE(INET)
        ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(64000)
        TYPE(INET)
NETWORK DOMAINNAME(AF_INET6)
        DOMAINNUMBER(19)
        MAXSOCKETS(2000)
        TYPE(INET)
```

Figure 1-1 Address family configurations in BPXPRMxx

Since the AF_INET6 address family is not supported in a Common INET environment, we configured the INET physical file system.

1.2 Remote procedure APIs

Remote procedure APIs are located at a higher level in the protocol stack than socket-based APIs. The socket API is used underneath the RPC interface, but the details of the socket interface are hidden for the application programmer who uses the RPC programming interface.

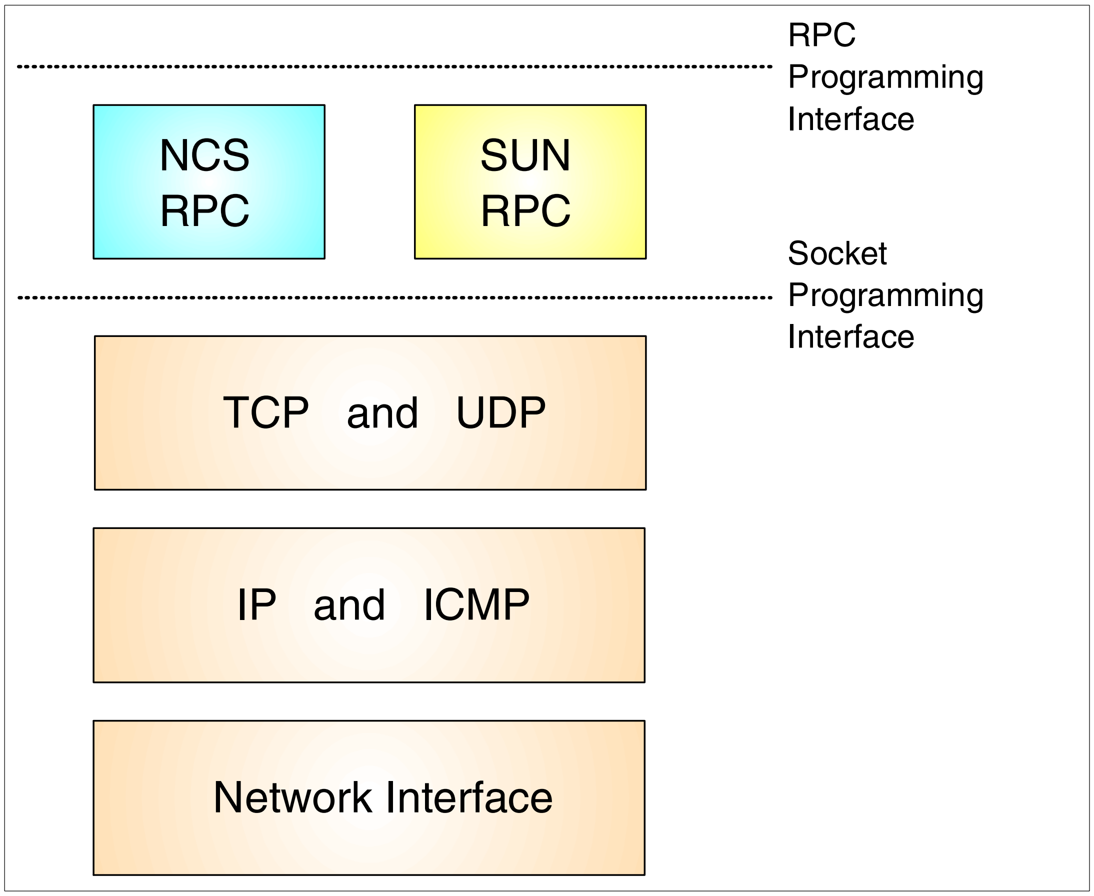


Figure 1-2 RPC programming interface and protocol layers

The RPC programming interfaces offer more functions to the application programmer than the socket programming interfaces do, and make the network programming job somewhat easier to accomplish. The RPC programming interfaces generally deal with such things as different data representation and some kind of state control over the dialog. On the other hand this also implies some restrictions; a dialog is normally limited to one procedure call. Each remote procedure call is stateless and independent of either preceding or succeeding calls. If an RPC client program requires more interactions with the server program, the state data has to be carried back and forth as user data in the parameters passed on each remote procedure call, or the server program has to implement some kind of scratch pad area (SPA) implementation, where state data per client is saved from call to call.

If you develop RPC programs, your only programming language choice is C. The source code for a sample client/server RPC program can be found in hlq.SEZAINST(GENESEND) and hlq.SEZAINST(GENESERV).

In OS/390 V2R7 IP and later, you have two RPC implementations:

- Sun Microsystems Remote Procedure Call (RPC)
- Hewlett Packard Remote Procedure Call, which is called Apollo Network Computing System (NCS)

1.2.1 ONC/RPC files and libraries

The HFS files used by z/OS UNIX system services and their location in the HFS are as follows:

- /usr/include/rpc - all header files are contained here
- orpcgen - ONC RPC protocol compiler
- orpcinfo - utility program for looking at portmaps of networked machines
- oportmap - network service program that maps ONC RPC program and version numbers to transport-specific port numbers.

1.3 X-Windows programming interfaces

If you want to develop distributed presentation programs, where your application program is running in MVS and the user interface is implemented on an X-Windows server in your IP network, you can use the X-Windows application programming interfaces that are supplied with OS/390 V2R7 IP and later to develop X-Windows z/OS client programs.

Two versions of X-windows and the corresponding OSF/Motif were introduced in OS/390 V2R5 IP. For the UNIX System Services environment, OS/390 V2R5 IP and later supports X-Windows System Version 11 Release 6 and OSF/Motif Version 1.2.4 as part of the base IP support. Support for X-Windows System Version 11 Release 4 and OSF/Motif Version 1.1 is available as a separate feature.

The X-Windows System support in UNIX System Services environment includes the following APIs based on the X11.6 specification:

- X11 Core distribution routines (X11)
- Inter-Client Exchange routines (ICE)
- Session Manager routines (SM)
- X-Windows System extended routines (Xert)
- Authentication functions (Xau)
- X10 compatibility routines (oldX)
- X Toolkit (Xt)
- Athena Widget set (Xaw)
- Utility functions used by Xaw (Xmu)
- PEX (PEX5) 3D Graphics
- Standard MIT X clients
- Sample X daemons

The X-Windows System support also provides the APIs based on OSF/Motif Release 1.2.4:

- OSF/Motif-based widget set (Xm library)
- OSF/Motif Resource Manager (Mrm library)
- OSF/Motif User Interface Language (Uil library) and compiler

CS for z/OS IP provides several examples to do a simple installation verification test. They are distributed in several subdirectories that are located under the /usr/lpp/tcpip/X11R6/Xamples/demos/ directory. We used an AIX Version 4.3.3.0 machine as an X-Windows server.

Before starting any X-Windows applications, you need to set your DISPLAY environment variable to point to your X-Windows server host:

```
DISPLAY=9.24.104.208:0
export DISPLAY
```

Then start the sample client application by entering `xsamp3` from the UNIX System Services command line. You will see a new window displayed on the server's screen as shown in Figure 1-3.

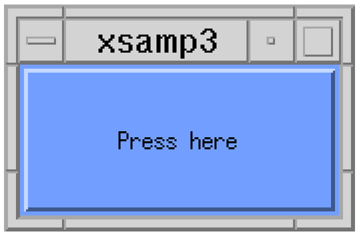


Figure 1-3 X Windows output from xsamp3

You should observe the following output from xsamp3 in your UNIX System Services session.

```
pwd = /usr/lpp/tcpip/X11R6/Xamples/demos/xsamp3: >xsamp3
xsamp3 entered.
XtToolkitInitialize done.
XtCreateApplicationContext done.
Display opened.
XmNwidth set.
XmNheight set.
XmNallowShellResize set.
XtAppCreateShell call done.
XtRealizeWidget call done.
XmCreatePushButton call done.
XtManageChild call done.
XtAddCallback call done.
```

Figure 1-4 Shell output when running xsamp3

1.3.1 X-Windows dynamic linkage libraries

Since OS/390 V2R6 IP applications can use the set of dynamic linkage libraries (DLLs) provided for the X-Windows system and OSF/Motif for z/OS UNIX. The advantage of this support is reduction in the size of application load modules. In addition, the X-Windows system and OSF/Motif archive files contain DLL-enabled modules. Any X-Windows system and OSF/Motif applications linked with these archive files must be compiled with the DLL option. For details on compiling and executing applications with DLL support, refer to *z/OS V1R2.0 CS: IP Application Programming Interface Guide*, SC31-8788.

The X-Windows component uses several library archives and you may have to rebuild one or more of these library archives when you apply maintenance to the X-Windows component. If this is the case, the PTF will include a ++HOLD ACT entry that will identify which commands or shell scripts you need to execute after having run your SMP/E job.

1.4 SNMP agent distributed programming interface (DPI)

This is a special-purpose programming interface that you can use if you want to implement dynamic management information base (MIB) variables. In an SNMP environment the MIB variables are defined in the MIBDESC.DATA data set. If you want to dynamically add, replace or delete MIB variables, you can develop an SNMP subagent program that uses the DPI programming interface to interact with the SNMP agent address space (SNMPD) to perform such functions.

If you develop an SNMP subagent, you can define your own MIB variables and your own SNMP traps.

The connection between the subagent address space and the SNMP agent is established as a TCP socket connection, so the DPI programming interface is again a higher-level programming interface to the socket interface. The DPI programming interface is only supported for programs written in C. For the specifics on the DPI programming interface you can find useful information in *z/OS V1R2.0 CS: IP Application Programming Interface Guide*, SC31-8788 where there is a good example of a C-based SNMP subagent.

Since OS/390 V2R5 IP, two versions of DPI interfaces are supported; they are DPI Version 1 Release 1 and Version 2 Release 0. DPI V2.0 is provided for z/OS UNIX C socket users and DPI V1.1 for traditional C socket users. DPI V2.0 gives you additional function, making it easier to write subagents and simplifying the task of developing and administering your application.

The following RFCs are related to SNMP and will be helpful when you are writing programs using the SNMP DPI:

- RFC1228, SNMP DPI Version 1 Release 1
- RFC1592, SNMP DPI Version 2 Release 0
- RFC 1901 - 1908, SNMP V2

SNMP DPI V2.0 protocol which is specified in RFC1592 provides the ability to connect agents via AF_UNIX or AF_INET sockets.

1.5 Resource Reservation Protocol API (RAPI)

The Resource Reservation Protocol Application Programming Interface (RAPI) is included with the z/OS UNIX Resource Reservation Protocol (RSVP) agent. The RSVP agent is integrated in CS for OS/390 V2R8 IP and later. RSVP is a receiver-oriented signalling Internet protocol that enables applications to request Quality of Service. RSVP requests resources for simple data flow in only one direction to receive or send network data, usually for a multicast or unicast session.

A Quality of Service (QoS) is the overall service that a user of an application receives from a network, in terms of availability, throughput, delay, etc. Refer to *TCP/IP Tutorial and Technical Overview*, GG24-3376 for more details on QoS.

The basic IP protocol stack provides only one QoS which is called *best-effort*. A packet is routed from point to point without any guarantee for a specific bandwidth or minimum time delay. Based on the best-effort traffic model Internet requests are handled on a *first come, first serve* basis. This means that all requests have the same priority and are handled one after the other.

In the next section, we discuss the RSVP API, which allows you to create an application to request networking resources explicitly. Note that applications that were written to use the previous Web version of the RSVP agent have to be rebuilt because the RSVP API (RAPI) has changed from being statically linked to being shipped as a DLL.

1.5.1 RAPI overview

The RAPI interface is one realization of the generic API contained in the RSVP functional specification (refer to RFC 2205). The RAPI interface is a set of C language routines. With these APIs, you can create a custom application that requests enhanced Quality of Service (QoS). The RSVP agent then uses the RSVP protocol to propagate the QoS request through the routers along the paths for the data flow. Each router may accept or deny the request, depending upon the availability of resources. In the case of failure, the RSVP agent will return the decision to the requesting application using RAPI.

RSVP assigns QoS to specific IP data flows which can be either multipoint-to-multipoint or point-to-point data flows, known as sessions. A session is defined by a particular transport protocol, IP destination address, and destination port. To receive data packets for a particular multicast session, an application must join the corresponding IP multicast group.

Under RSVP, QoS requests are made by the data receivers. A QoS request contains a *flowspec*, together with a *filter spec*. The flowspec includes an Rspec (resource specification), which defines the desired QoS and is used to control the packet scheduling mechanism in the router or host, and also a Tspec (traffic specification), which defines the traffic expected by the receiver. The filter spec controls packet classification to determine which sender data packets receive the corresponding QoS.

The Tspec is composed of:

- r: Token bucket rate, which is the average data rate in bytes per second.
- b: Token bucket depth, which is the sending buffer in bytes.
- p: Peak data rate, in bytes per second.
- m: Nominal minimum packet size in bytes.
- M: Maximum packet size (MTU) in bytes.

The Rspec consists of the following values:

- R: The rate in bytes per second.
- S: The slack term in microseconds.

The following API calls are shipped with CS for OS/390 V2R8 IP:

- rapi_session: Establish an API session with the RSVP agent.
- rapi_release: Release an API session.
- rapi_sender: Indicate sender application. This call results in RSVP sending a PATH packet to the destination.
- rapi_reserve: Make a QoS reservation as a data receiver.
- rapi_dispatch, rapi_getfd: Support asynchronous upcall mechanism.

The following routines are standard routines for displaying the contents of RAPI objects.

- rapi_fmt_adspec: Formats a given RAPI Adspec into a buffer.
- rapi_fmt_filtspec: Formats a RAPI filter spec into a buffer.
- rapi_fmt_flowspec: Formats a given RAPI flowspec into a buffer.
- rapi_fmt_tspec: Formats a given RAPI Tspec into a buffer.

One or more RAPI sessions exist for a given RSVP session, each established with the rapi_session() call. Each sender within an RSVP session issues rapi_sender(), primarily to provide the sender Tspec. Each rapi_sender() call results in RSVP sending a PATH packet to the destination. The PATH packet is intercepted at each router along the data path and is used to install path state (the Tspec and other parameters).

When the PATH arrives at the destination, it is presented to the receiver application using the asynchronous upcall mechanism. The receiver uses the Tspec and other information to arrive at a reservation request, which is composed of one or more filter specs (which select one or more senders), and one or more flowspecs (which contain reservation parameters), based on the reservation style. These styles are supported:

- Wildcard style (WF): A single flowspec is shared among all current and future senders.
- Fixed style (FF): A specific flowspec is paired with a specific filter spec for a specific sender. One or more pairs may be specified.
- Shared Explicit style (SE): A single flowspec is shared among a named set of senders.

The reservation parameters take two forms, depending on the service type being used. The following service types are supported:

- Controlled Load (CL): The reservation parameters take the same form as the Tspec (r, b, p, m, M). The goal of CL service is to provide what appears to be unloaded service response characteristics even when one or more hosts or routers along the path are heavily loaded.
- Guaranteed (GUAR): The reservation parameters take the form of the Tspec (r, b, p, m, M) followed by an Rspec (R, S). The additional Rspec parameters provide a requested rate in bytes per second, and a slack term in microseconds. The goal of GUAR service is to use the various parameters in specific equations to produce the maximum delay that will be experienced when sending data. By changing certain parameters, the application can control, to a certain extent, the value of that delay.

The receiver application makes the reservation request using the rapi_reserve() call. This causes RSVP to send an RESV packet hop-by-hop back along the same path traveled by the PATH packet. Each host or router along the path installs the appropriate reservation on the appropriate interface.

When the RESV arrives at the sender, it is presented asynchronously to the sender application. The application may or may not choose to wait for the RESV before commencing data transmission. However, all data sent before the reservation is in place is delivered best effort.

The upcall mechanism requires the application to use a listener thread, which gets a socket descriptor to use via the rapi_getfd() call, then uses this socket descriptor in a select() call. When an RSVP packet arrives, the select() will return, at which point the application uses rapi_dispatch() to cause the asynchronous function registered on the rapi_session() call to receive control. All RAPI/RSVP objects, such as Tspecs and flowspecs, are delivered to this function as parameters.

1.5.2 Compiling and linking RAPI applications

To use the RAPI interface, an application must perform the following

1. Include the <rapi.h> header file, which is available in the /usr/include directory.
2. Compile the application with the DLL compiler option. Refer to the z/OS C/C++ User's Guide for more information on how to specify compiler options.

3. Include the RAPI definition side deck (rapi.x), which is available in the /usr/lib directory, when prelinking or binding the application.
4. If the Binder is used instead of the C Prelinker, specify the Binder DYNAM=DLL option. Refer to DFSMS/MVS Program Management for information on specifying Binder options.

1.5.3 Running RAPI applications

At execution time, the RAPI application must have access to the RAPI DLL (rapi.dll), which is available in the /usr/lib directory. Ensure that the LIBPATH environment variable includes this directory when running the application. The RAPI application must run with superuser authority to use RAPI.

Part 2

Productivity applications

In this part, we introduce the productivity applications included with Communications Server for z/OS IP.

2

z/OS UNIX telnet server

The TelnetD server is used to enable remote telnet clients to log on the UNIX System Services shell environment in either raw mode (also called character mode) or line mode.

This chapter contains the following sections:

2.1 z/OS UNIX telnet server overview

See Figure 2-1 for an overview of how the TelnetD server is implemented in UNIX System Services.

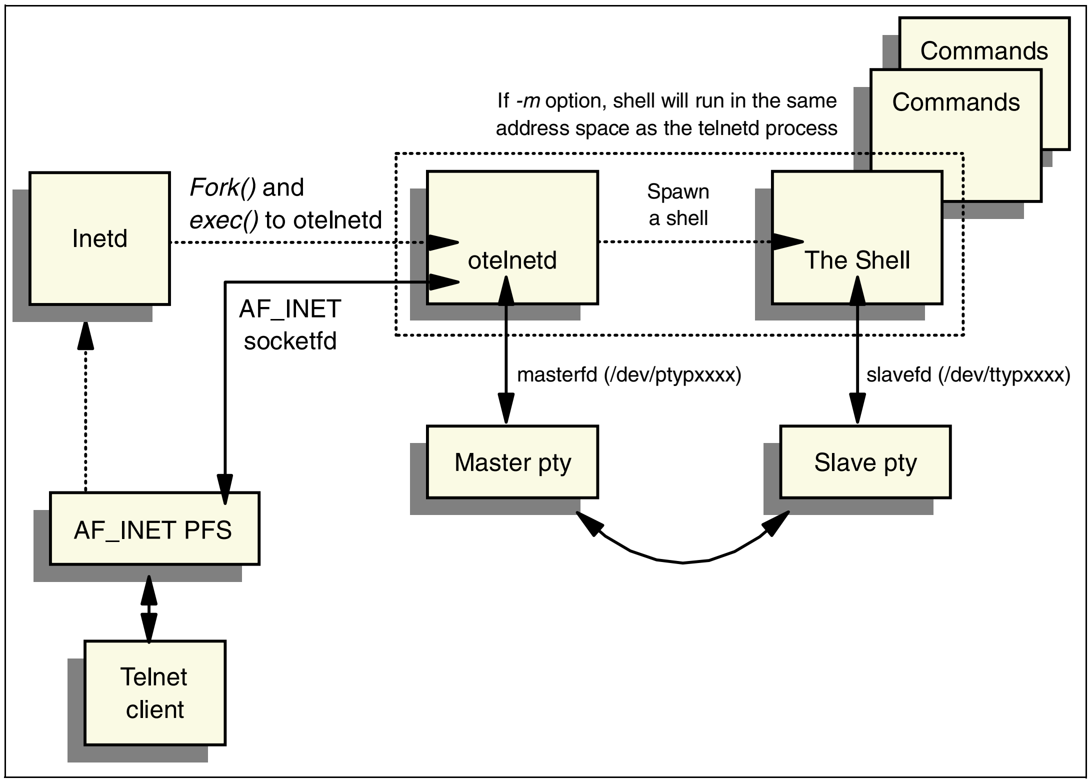


Figure 2-1 TelnetD overview

ASCII/EBCDIC translation is done by the otelnetd process. The z/OS UNIX Telnet Server relies on the **chcp** shell command to provide code page conversions. Therefore, if the Telnet client wants to use a code page other than the default IBM-1047 code page, it has to make use of the **chcp** command.

```
HIRAYAM @ RA39:/u/hirayam>chcp -a IBM-932C -e IBM-939 1
HIRAYAM @ RA39:/u/hirayam>chcp -q 2
 ASCII code page : IBM-932C
EBCDIC code page : IBM-939
HIRAYAM @ RA39:/u/hirayam>export LANG=Ja_JP
```

Figure 2-2 Telnet code page change

1 Sets the ASCII and EBCDIC code pages to be used.

2 Queries the current settings of code pages.

When a **chcp** shell command is executed, the TelnetD process is informed about the code page change via urgent data over the pseudo terminal connection (the master/slave pty interface). A user may select to have a **chcp** command executed as part of the user's login process, for example, via the user's $HOME/.profile or $HOME/.setup shell scripts.

The z/OS UNIX Telnet Server also supports DBCS translation. In this example, IBM-932C and IBM-939 represent Japanese ASCII and EBCDIC DBCS code pages respectively. Since code page setting has been updated, you may change the LANG environment variable to the appropriate one.

An environment-specific banner page can be created in the file /etc/banner using standard edit functions. It will be displayed at the client workstation after the user ID and password have been entered.

```
************************************************************************
*                                                                      *
*  Welcome to OS/390 V2R10 UNIX System Services on ITSO Raleigh RA28  *
*  You accessed this system via CS for OS/390 V2R10                    *
*                                                                      *
*  This is the /etc/banner page for Telnet server use.                 *
*                                                                      *
************************************************************************
```

Figure 2-3 File containing the banner page

With the /etc/banner file above, the following welcome page was displayed on our terminal:

```
************************************************************************
*                                                                      *
*  Welcome to OS/390 V2R10 UNIX System Services on ITSO Raleigh RA28  *
*  You accessed this system via CS for OS/390 V2R10                    *
*                                                                      *
*  This is the /etc/banner page for Telnet server use.                 *
*                                                                      *
************************************************************************

IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 2000
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989..

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

BTHOMPS:/u/bthomps: >
```

Figure 2-4 Telnet welcome page

2.2 Pseudoterminals

In preparing for the TelnetD Server you need to ensure that during UNIX System Services installation and customization you created a sufficient number of pseudoterminal files. The pseudoterminal files are created in the hierarchical file system, in the /dev directory. The file names are ptypnnnn and ttypnnnn, where nnnn is a number from 0000 to 9999. These files are used in pairs by the TelnetD server and the RloginD server.

During the UNIX System Services installation you executed the sample JCL BPXISMKD and internally invoked sample REXX program BPXMKDIR from SYS1.SAMPLIB to create a number of directories and files in your hierarchical file system. One section of this REXX program creates a certain number of pseudoterminal pairs. You may have increased that by customizing the BPXMKDIR REXX program before executing it. To verify how many pseudoterminal pairs you have defined in your system, list the /dev directory and locate the highest numbered ptypnnnn entry you find. If that is ptyp0050, and you want to increase the number of pseudoterminals to, for example, 75, you can either issue individual **mknod** commands (which probably is going to be a bit cumbersome), or you can create a small REXX program like the following:

```
/* REXX */
ptystart = 51    /*First new ptyp and ttyp number*/
ptylast = 75     /*New maximum number            */
call syscalls('ON')
Do pty = ptystart to ptylast
 ptyname = '/dev/ptyp'||right(pty,4,0)
 ttyname = '/dev/ttyp'||right(pty,4,0)
 say 'Creating 'ptyname' and 'ttyname
 address syscall "mknod" ptyname "666 1 0"
 address syscall "mknod" ttyname "666 2 0"
End
exit(0)
```

Figure 2-5 Increasing the number of pseudoterminals

The above REXX program will increase your pseudoterminal pairs from 50 to 75. Please note that the REXX program must be executed from a superuser.

After you have increased the number of pseudoterminals in your hierarchical file system, you need to let UNIX System Services know that more pseudoterminals are now available. You do so by updating the MAXPTYS statement in your BPXPRMxx parmlib member:

```
MAXPTYS(76)
```

The MAXPTYS value is specified as one above the highest pseudoterminal number you created. In the above setup, you have from pseudoterminal 0000 to pseudoterminal 0075, which gives 76 pairs total.

Note: The installation process is different in OS/390 V2R7 UNIX System Services and later. The BPXMKDIR REXX program no longer creates pseudoterminal pairs at installation time, but the nodes will be generated dynamically when you request the connection. You can still use the **mknod** command to add pseudoterminals beforehand.

2.3 Starting the z/OS UNIX telnet server

The TelnetD Server is started via InetD.

The default port number for the TelnetD Server is 23. This is a well-known port number, and you can reserve the port to CS for z/OS IP in the PROFILE data set:

```
23 TCP OMVS ;  UNIX System Services Telnet Server
```

If the default port number 23 is used, a client has to know only the name or IP address of the server to establish a connection and can use a command such as:

```
telnet 9.24.104.43
```

It is also possible to reserve a different port to the UNIX Telnet Server in the PROFILE data set of CS for z/OS IP, for example:

```
2023 TCP OMVS ;  UNIX System Services Telnet Server
```

In this case, a client has to specify both the host name (or IP address) and the port number of the server with the Telnet login command:

```
telnet 9.24.104.43 2023
```

If you assign an alternate port number to your UNIX System Services TelnetD Server, you also need to update your /etc/services configuration file with the chosen port number in order for InetD to listen for Telnet client requests on the chosen port:

```
telnet        2023/tcp
```

If your configuration has more than one CS for z/OS IP stack running on one z/OS image, all of these stacks must have identical port reservations for the UNIX Telnet Server. The chosen port number is a system wide value in the UNIX System Services environment. For more information on running multiple stacks see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

A TelnetD Server process is forked from InetD whenever a telnet client connects to UNIX System Services. See Figure 2-1 for an overview of how TelnetD operates in the UNIX System Services environment.

The TelnetD Server supports the following server options to be specified in the /etc/inetd.conf file:

b	This option forces the server to DO BINARY in the first
c	Specifies a inactivity time-out value for the connection pass during negotiations with the client.
C	Output messages will be in uppercase letters.
D	Various debugging options.
h	Prevent display of host-specific banner page (from /etc/banner).
k	Prevent raw-mode initialization.
l	Set default mode for login to line mode.
m	Run all forked or spawned processes in the same address space, recommended for performance improvement.
n	Disable TCP keep-alive packets.
t	Activate internal tracing information.
U	Refuse connections from any address that cannot be resolved into a host name via a gethostbyaddr() call.

Any of these parameters can be changed dynamically by modifying the configuration file /etc/inetd.conf and issuing a **`kill -SIGHUP`** command to force InetD to re-read the configuration file. All new telnet connections that are established after the configuration file has been re-read will use the new options. The following is a sample /etc/inetd.conf file:

```
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
#
   :
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l
   :
```

Figure 2-6 InetD configuration for TelnetD

All messages from the TelnetD Server are logged by syslogd according to the syslogd configuration file settings. The TelnetD Server uses a facility name of local1 for all its messages.

2.4 Termcap and terminfo

For telnet sessions, you may need some definitions of terminal capabilities in order to let UNIX System Services application programs manipulate the terminal output correctly. These definitions are kept in the terminfo database.

A UNIX system, like other operating systems, must have at least one terminal attached to it. In very early days these were typewriters having keyboard input and a paper output. These terminals were later replaced by video display units (VDU) which actually behave like the typewriter did earlier; the paper is just replaced by the screen. All VDUs understand a data stream, which contains ASCII characters to be printed, including control characters, such as carriage return, or new line. Differences showed up when individual manufacturers started to add specific commands to their terminals. Specific commands may be cursor up, and cursor left, to name a few.

As long as one uses simple printing commands such as **`cat`**, there is no obvious difference among all terminals. But these differences have to be taken into account as soon as one runs a program that uses special commands. This is mostly the case with editors, such as vi, or emacs. These programs need to know, for example, how to move the cursor to a specific location.

As usual, there is more than one approach to solve this problem. The first is to write specific drivers for each terminal. Another solution is to create a database that contains definitions of all capabilities each terminal has and a subroutine library, which gives access to the database.

The first approach would have been a typical UNIX solution. Writing a device driver is not an unusual task in this environment. But, the latter solution was chosen and is today a somewhat agreed-upon standard in the world of UNIX. This is what the termcap database provides. It is actually a readable text file. It has its roots in the Berkley Software Distribution (BSD). In UNIX systems that have their roots in System V UNIX, the equivalent implementation of termcap is the terminfo database. The most important difference is that terminfo is a compiled database. For that reason you need a compiler (tic) to create the terminfo database. While termcap contains all definitions in one file, terminfo organizes the information into a directory structure.

Since OS/390 V2R6, the terminfo database is shipped as part of z/OS. The database is populated with the terminal types defined by ibm.ti, dec.ti, wyse.ti, and dtterm.ti. dtterm.ti is new for OS/390 V2R6 and later. The database is in the directory /usr/share/lib/terminfo and the source files are in /samples.

Because the database exists already, you can comment out the **tic** commands from your customized copy of /etc/rc. If for some reason you need to re-create the terminfo database, run the **tic** utility. Each type of terminal that is defined has a corresponding file with the suffix .ti. For example, to define an IBM terminal for the terminfo database, specify from the shell environment:

```
tic /samples/ibm.ti
```

To define terminal types such as VT100 and VT220, specify from the shell environment:

```
tic /samples/dec.ti
```

Whenever a shell environment is created, the terminal type of the client user is registered in the environment variable TERM. The TSO OMVS command environment handles TERM as if it were set to TERM=dumb. In a telnet or rlogin session the TERM environment variable is set to the terminal type that the telnet client uses or emulates. In our case we used an OS/2 telnet client with a terminal type specification of vt220, which results in a TERM=vt220 setting in the shell. Table 2-1 on page 21 shows the directories and files shipped as part of UNIX System Services or created by the **tic** command.

Directory	Terminal Definition Files
a	aixterm, aixterm-m, aixterm-m-old, aixterm-old
c	cdef
d	dtterm, dumb
h	hft, hft-c-old, hft-m-old, hft-nam-old, hft-c, hft-m, hft-nam, hft-old
i	ibmpc, ibmpcc, ibmpdp, ibm3101, ibm3151, ibm3151-noc, ibm3151-S, ibm3151-132, ibm3151-25, ibm3151-51, ibm3151-61, ibm3152, ibm3152-PS2, ibm3152-132, ibm3152-25, ibm3161, ibm3161-C, ibm3162, ibm3162-132, ibm3163, ibm3164, ibm5081, ibm5081-old, ibm5081-113, ibm5081-56, ibm5151, ibm5154, ibm5550, ibm5570, ibm6153, ibm6153-40, ibm6153-90, ibm6154, ibm6154-40, ibm6154-90, ibm6155, ibm6155-113, ibm6155-56, ibm8503, ibm8507, ibm8512, ibm8513, ibm8514, ibm8515, ibm8604
j	jaixterm, jaixterm-m
l	lft, lft-pc850
L	LFT, LFT-PC850
v	vs100, vs100s, vt100, vt100-am, vt100-nam, vt100x, vt200, vt200-8, vt320, vt320, vt320-8, vt330,vt330-8, vt340
w	wyse100, wyse30, wyse350, wyse50, wyse50-2, wyse60, wyse60-AT, wyse60-PC, wyse60-316X, wy100, wy30, wy350, wy50, wy50-2, wy60, wy60-AT, wy60-PC, wy60-316X
x	xterm, xterms

Table 2-1 Terminfo directory structure and contents

If you are interested in more information on termcap and terminfo, refer to *Termcap & Terminfo*, published by O'Reilly & Associates.

2.4.1 TERMINFO environment variable

In order to use terminal types that are not supported by CS for z/OS IP you can now create a directory to store local terminal definitions and use the TERMINFO environment variable to define the location of that directory. This environment variable was implemented to allow typical UNIX clients with GUI windowed command line prompts access to the UNIX System Services telnet server. This variable should be used if there are installation defined terminfo definitions. See the *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800 for more information on terminal definitions.

The UNIX System Services telnet server is started by inetd, the listener for the server. Since inetd does not pass the environment variables (other than PATH and TZ) to it's child processes, the UNIX System Services telnet server had to be enhanced to support a new environment variable. The "-T" option can be used when otelnetd is started to allow otelnetd to set the TERMINFO environment variable to the specified value prior to searching for the terminfo definitions for the terminal type specified by the telnet client. The following line is a sample from the /etc/inetd.conf file, which starts otelnetd used at ITSO:

```
otelnet stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l -T
```

Using "-T" only tells otelnetd what TERMINFO should be set to. TERMINFO should also be defined in /etc/profile as well so that it is set in the user's login shell. Otherwise, the vi editor and other applications may not be able to find the terminal definitions. We issued a kill -SIGHUP <inetd pid> to have inetd read it's configuration file again. Then displayed the environment variables after logging in UNIX System Services telnet.

```
env
 :
 :
TZ=EST5EDT
RESOLVER_CONFIG=/etc/resolv.conf.28b
MANPATH=/usr/man/%L
NLSPATH=/usr/lib/nls/msg/%L/%N
TERMINFO=/var/terminfo
```

2.4.2 z/OS UNIX otelnetd logging

UNIX System Services telnet also has a start parameter for the -D option that enhances recording to the log file defined in /etc/syslog.conf. The new "-D login" option records log in and logout activities to syslogd facility **`auth`** using message EZYTU36I. To test the new parameter we set up the InetD configuration files as follows:

```
otelnet stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l -D login
```

and the following statement in the syslogd configuration file:

```
local1.*;auth.* /var/telnetlogin.log
```

The otelnetd server uses the local1 and auth facility for recording to syslogd. We chose to let both facilities log in to the same log file to capture all the messages to a single file. The following shows the content of the log file (/etc/telnetlogin.log). It shows normal otelnetd log messages and the new message captured by the `-D login` option.

```
EZYTE52E Couldn't resolve your address into a host name.
IP address is 9.24.106.54
EZYTU34I id 3000F pri 7 call getpwnam(ads) code A3 reason 0B490800 h_errno N/A
EZYTE32W You entered an invalid login name or password.
EZYTE32W You entered an invalid login name or password.
EDC5163I SAF/RACF extract error. rsn = 0B490800
EZYTE52E Couldn't resolve your address into a host name.
IP address is 9.24.106.54
verify_password: __Passwd failed  EDC5111I Permission denied. rsn = 090C0000
EZYTU36I P bthomps 9.24.104.43 2023 9.24.106.54 4917 -
EZYTE32W You entered an invalid login name or password.
EZYTE32W You entered an invalid login name or password.
EDC5111I Permission denied. rsn = 090C0000
EZYTU36I L bthomps 9.24.104.43 2023 9.24.106.54 4917 -   1
EZYTU36I O bthomps 9.24.104.43 2023 9.24.106.54 4917 -   1
EZYTE52E Couldn't resolve your address into a host name.
IP address is 9.24.106.54
EZYTU34I id 3000F pri 7 call getpwnam(thompson) code A3 reason 0B490800 h_errno N/A
EZYTU36I U thompson 9.24.104.43 2023 9.24.106.54 4918 -  1
EZYTE32W You entered an invalid login name or password.
EZYTE32W You entered an invalid login name or password.
EDC5163I SAF/RACF extract error. rsn = 0B490800
EZYTU36I C bthomps 9.24.104.43 2023 9.24.106.54 4918 -   1
```

The letter after the EZYTU36I message has the following meaning:

- **P** - the user provided an invalid password
- L - the user successfully logged in
- **O** - the user logged off
- **U** - the user provided an invalid user ID
- **C** - the user needed to change their password but was unsuccessful

The field after the letter is the user ID specified by the user. The next two fields are the IP address and port to which the user is connected. That is followed by the remote (telnet client) workstation's IP address and port. The telnet client's port number is normally not significant, but it is theoretically possible to determine the user ID on that machine if it is a multiuser system supporting the IDENT protocol. The last field is the DNS hostname of the client machine, or a hyphen 1 if that information is not available.

3

X-Window system

The X-Window system is a network-transparent protocol that supports windowing and graphics. The protocol is communicated between an X client or application and an X server over a TCP/IP network.

In an X-Window system environment, the X server is generally located on the workstation, and distributes user input to and accepts requests from various X client programs located either on the same system or elsewhere on a network. This seemingly reverse terminology sometimes confuses the user since we usually associate anything that runs on the host as the server and any application on our workstation as the client.

One X server may be connected to many X clients, sharing the physical display and input devices among many application programs. The clients may be located on different hosts. If you are planning to use the X-Window system in CS for z/OS IP, be aware that your z/OS system could not be an X-Window Server. You may, however, create and run X-clients in z/OS.

CS for z/OS IP provides two versions of the X-Window system and OSF/Motif which is used for creating X-Window applications:

- X-Window system, Version 11, Release 4 and OSF/Motif 1.1 for Open Sockets and the TSO environment. This version is shipped as a separate feature (FMID HTCP38X).
- X-Window system, Version 11, Release 6 and OSF/Motif 1.2.4 which is installed as part of the CS for z/OS IP base. This is the recommended X-Window system version that you should use.

The following sections are included in this chapter:

- 3.1, “Creating X-Window applications” on page 26
- 3.2, “Running X-Window applications” on page 28

For more information about OSF/Motif and X-Window system libraries, refer to *z/OS V1R2.0 Communications Server IP Programmer’s Reference*, SC31-8787 and *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800.

3.1 Creating X-Window applications

CS for z/OS IP provides you with a set of X-Window system application program interfaces (API) which allows you to write applications in the z/OS UNIX System Services environment. Together with APIs from X-Window system V11R6, CS for z/OS IP also includes APIs based on OSF/Motif Release 1.2.4. OSF/Motif is an X-Window System toolkit defined by the Open Software Foundation Inc. (OSF) and is a set of basic C language routines for developing applications in a variety of application environments.

An X-Window system toolkit is a set of library functions layered on top of the X-Window system functions that allows you to simplify the design of applications by providing an underlying set of common user interface functions. Included are mechanisms for defining and expanding interclient and intracomponent interaction independently, masking implementation details from both the application and component implementer.

An X-Window system toolkit consists of the following:

1. A set of programming mechanisms, called Intrinsics, that are used to build widgets.
2. An architectural model to help programmers design new widgets with enough flexibility to accommodate different application interface layers.
3. A consistent interface, in the form of a coordinated set of widgets and composition policies, some of which are application domain-specific, while others are common across several application domains.

The fundamental data type of the X-Window system toolkit is the widget. A widget is allocated dynamically and contains state information. Every widget belongs to one widget class that is allocated statically and initialized. The widget class contains the operations allowed on widgets of that class.

An X-Window system toolkit manages the following functions:

- Toolkit initialization
- Widgets and widget geometry
- Memory
- Window, data set, and timer events
- Input focus
- Selections
- Resources and resource conversion
- Translation of events
- Graphics contexts
- Pixmaps
- Errors and warnings

Starting with OS/390 V2R6 IP you could use the set of dynamic link libraries (DLLs) provided for the X-Window system and OSF/Motif for z/OS UNIX. A dynamic link library (DLL) is a collection of one or more functions or variables in an executable module that is executable or accessible from a separate application module. This means that your X-Window application does not need to have all of the modules from the X-Window library included with it since all external function and variable references are resolved dynamically at run time. The result is a reduction in the size of the load module.

Since the X-Window system and OSF/Motif archive files contain DLL-enabled modules, any X-Window system and OSF/Motif applications linked with these archive files must be compiled with the DLL option or else the link-edit phase will fail. Figure 3-1 on page 27 shows a sample makefile that has the DLL compile option 1:

```
OBJS            = xclock.o Clock.o
#SYSLIBS      =  -lXaw -lXmu -lXt -lSM -lICE -lXext -lX11
SYSLIBS       =  /usr/lib/Xaw.x /usr/lib/SM.x \
                 /usr/lib/ICE.x /usr/lib/X11.x
CFLAGS        = -D_ALL_SOURCE -W c,dll       1
CC              = c89
PGM             = xclock

.c.o:
        $(CC) -c $(CFLAGS) $<

$(PGM): $(OBJS)
        $(CC) -o $@  $(OBJS) $(SYSLIBS)

clean:
        rm -f $(PGM) *.o core
```

Figure 3-1 Sample makefile for an X-Window application

When we did not compile the application with the DLL option, we received the following error messages:

```
USER1 @ RA03:/u//user1>make
c89 -c -D_ALL_SOURCE xclock.c
c89 -c -D_ALL_SOURCE Clock.c
c89 -o xclock  xclock.o Clock.o /usr/lib/Xaw.x /usr/lib/SM.x /usr/lib/ICE.x
IEW2456E 9207 SYMBOL sessionShellWidgetClass UNRESOLVED.  MEMBER COULD NOT BE
         INCLUDED FROM THE DESIGNATED CALL LIBRARY. NAME SPACE = 3
               ...
IEW2456E 9207 SYMBOL _XtInherit UNRESOLVED.  MEMBER COULD NOT BE
          INCLUDED FROM THE DESIGNATED CALL LIBRARY.

FSUM3065 The LINKEDIT step ended with return code 8.
FSUM8226 make: Error code 3
make: 'xclock' removed.
```

Figure 3-2 Using the makefile

There are several X-Window sample codes in /usr/lpp/tcpip/X11R6/Xamples that you could use as a guide if you are planning to port your X-Window application to the z/OS UNIX Systems Services environment. Since most of the UNIX applications are written for ASCII-based systems, you may also want to visit this Web site:

http://www.s390.ibm.com/products/oe/bpxa1p03.html

for information on how to handle the ASCII-EBCDIC dependencies of your UNIX code. Please remember that the source codes available on the Internet for UNIX are all in ASCII format. You need to make sure that the source codes are all properly converted from ASCII to EBCDIC.

Aside from the demo codes that are shipped with CS for z/OS IP, there are also several standard X-Clients that you may find in /usr/lpp/tcpip/X11R6/Xamples/clients:

Client	Description
appres	Lists application resource database
bitmap	Bitmap editor
editres	Resource editor
iceauth	ICE authority file utility
oclock	Displays time of day
xauth	X authority file utility
xclipboard	Clipboard utility
xclock	Analog/digital clock for X
xdpyinfo	Display information utility for X
xfd	X font display utility
xlogo	Displays X logo
xlsatoms	Lists interned atoms defined on server
xlsclients	Lists client applications running on a display
xmag	Magnifies part of screen
xlsfonts	Lists server fonts
xprop	Property displayer for X
xwininfo	Window information utility for X
xwd	Dumps an image of an X-Window
xwud	Displays dumped image for X

These X-clients will provide you with more examples on creating X-Window applications in the z/OS UNIX Systems Services environment.

3.2 Running X-Window applications

Before you can successfully run your X-Window applications, you need to do the following:

- Ensure that the workstation with an X-Window system server that supports X11R6 is properly configured and reachable by the z/OS system. You can test this by doing a PING to the workstation.
- From the workstation, use telnet to access the z/OS system, and open an z/OS UNIX shell on the z/OS system.
- From the z/OS UNIX shell, export the DISPLAY environment variable using either the network name or the qualified IP address of the workstation as shown in the following example:

  ```
  export DISPLAY=9.24.104.182:0.0
  ```

  In this example, 9.24.104.182 is the IP address of the workstation running the X-Window server. `:0.0` indicates the number of the X server and the number of the screen to be used in that X server. This is usually specified this way in almost all cases unless you have multiple X servers running in your workstation.

- Since we are running an X-Window system DLL-enabled application, we have to ensure that the LIBPATH environment variable contains the value /usr/lib.
- Depending on the operating system of your workstation where you are running your X server, you may need to authorize the z/OS system to access the workstation. This is done by executing the command

  ```
  xhost +
  ```

  where you can specify either the name of the z/OS system or + which will turn off security for this workstation and allow any X client to display at the X server.
- The X-Window system also allows you to optionally modify certain characteristics of an application at run time using application resources. Typically, application resources are set to tailor the appearance and possibly the behavior of an application. The application resources can specify information about an application's window sizes, placement, coloring, font usage, and other functional details. This is specified in the /u/user_id/.Xdefaults file.
- You may now run your X-Window application. As an example, we compiled and linked the XCLOCK program that is included with CS for z/OS IP. The source code is located in /usr/lpp/tcpip/X11R6/Xamples/client/xclock. The application is then invoked and run as a background process by `xclock &` and this is what appeared on our X server screen:

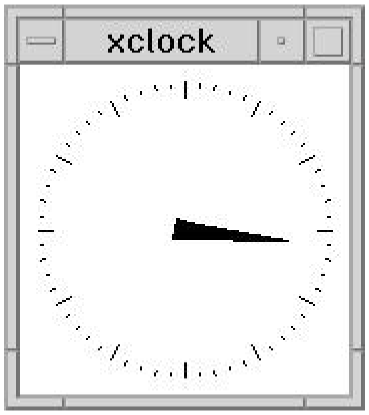


Figure 3-3 X server display for the XCLOCK program

4

z/OS UNIX remote command execution

OS/390 V2R5 IP also provides REXEC and RSH server functions which allow remote clients to run UNIX System Services shell commands.

- UNIX System Services REXECD server

  Provides server functions in the UNIX System Services environment for remote commands based on the remote execution (REXEC) protocol. Commands are executed as OpenEdition shell commands.

- UNIX System Services RSHD server

  Provides server functions in the UNIX System Services environment for remote shell clients based on the remote shell (RSH) protocol. Commands are executed as UNIX System Services shell commands.

The sections in this chapter are:

4.1 z/OS UNIX remote command execution overview

The UNIX System Services REXECD and RSHD server are installed into both the hierarchical file system and OS/390 V2R5 IP and later product libraries. In most installations, these modules are installed into the /usr/lpp/tcpip/sbin directory with the sticky bit, named orexecd and orshd respectively. The corresponding symbolic links are created in /usr/sbin directory for both modules. However, since these modules have the sticky bit, the executable modules are loaded from the SEZALINK library data set. Note that there are options to change the mount point for the TCP/IP filesystem in a hierarchical file system environment, so the modules orexecd and orshd might be installed in another directory.

Both the REXECD server and the RSHD server are used to execute UNIX System Services shell commands from remote users. See Figure 4-1 for an overview of how both the REXECD server and the RSHD server are implemented in the UNIX System Services environment.

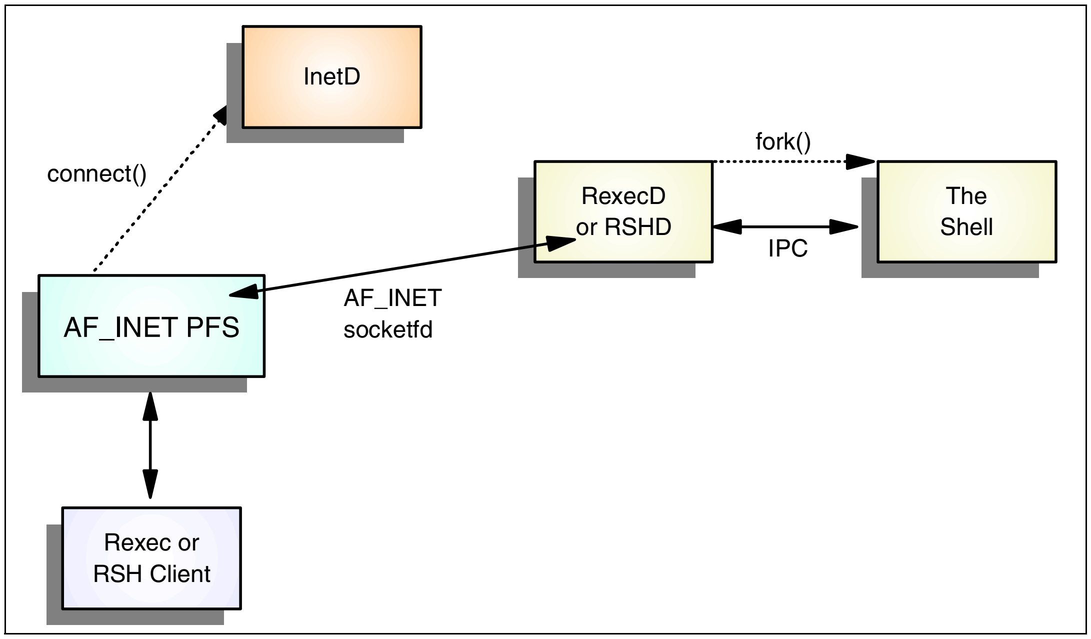


Figure 4-1 z/OS REXECD and RSHD implementation overview

For each remote request, InetD forks a new process with either REXECD or RSHD. The REXECD or RSHD server in turn forks a shell process with which they communicate via pipes.

The REXEC protocol requires a remote user to explicitly enter both a user ID and a password on the REXEC command line invocation:

```
rexec mvs03 -l userid -p password ls -al
```

The RSH protocol does not require an end user to enter a user ID and a password. If the user does not enter a user ID, the local user ID will be sent to the remote RSHD server as the remote user ID, but the protocol and **RSH** command syntax does not support a password parameter, like the **REXEC** command does. Some RSHD servers implement an rhosts.data file on the RSHD server host to authorize certain client users on certain remote hosts to execute commands on the RSHD server without authentication in terms of verifying a password.

However, the *rhosts.data* technique has been determined to be too much out of line with traditional MVS security policy and has *not* been implemented in the UNIX System Services orshd server. This also means that the **RCOPY** command is *not* supported in an UNIX System Services environment.

The RSHD server in UNIX System Services does *not* process any remote commands without a valid MVS user ID and password. So in order to access the RSHD server in UNIX System Services, you have to issue the **RSH** command in the syntax as follows:

```
rsh mvs03 -l userid/password ls -al
```

On the -l parameter, the end user has to enter both an MVS user ID and the password with the user ID and password separated by a slash (/).

The RSH server in OS/390 V2R5 IP and later provides an alternative method for authentication to allow the RSH client without a password to access UNIX System Services resources. This method is provided by the user security exit, named /usr/sbin/ruserok. This is discussed in more detail in “Trusted host concept with RSHD server” on page 35.

4.2 Starting the REXECD and RSHD servers

Both the REXECD and the RSHD servers are started from InetD.

The default port numbers for the two servers are 512 and 514, and both must be reserved in the PROFILE data sets of the TCP/IP stacks that act as UNIX System Services AF_INET transport providers:

```
PORT
   512   TCP   OMVS       ;REMOTE EXECUTION SERVER
   514   TCP   OMVS       ;REMOTE SHELL SERVER
```

Most REXEC and RSH client implementations do not allow the end user to specify an alternate port number, so we do not recommend that you try to start these servers on alternate port numbers.

Note, however, that REXEC and RSH servers in UNIX System Services cannot share a TCP port with the MVS REXECD server. Therefore in most situations, you might have to decide which REXEC and RSH server you use depending on your requirement. If the REXEC and/or RSH server is the first server controlled by InetD, you have to create an InetD configuration file, named */etc/inetd.conf*. The sample file is stored in /samples directory, but you have to modify it to meet your UNIX System Services environment. The following is a part of /etc/inetd.conf file in our test system:

```
#
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -l -v
#
```

Figure 4-2 Sample /etc/inetd.conf for REXECD and RSHD servers

Note: You may use rshd or rexecd as the first entry for the server program argument instead of orshd or orexecd respectively. During the ITSO-Raleigh testing, we could use both uppercase or lowercase characters for some options. Since this is not documented yet, you would have to verify these configuration parameters in your own system.

You will have to modify the sample inetd.conf file, because it points to the invalid module names /usr/sbin/rexecd and /usr/sbin/rshd. Since the module name has to contain the correct hierarchical file system file name to the server modules, you would have to change them to /usr/sbin/orexecd and /usr/sbin/orshd.

Instead of customizing inetd.conf, you may create symbolic links associated with these server modules, named /usr/sbin/rexecd and /usr/sbin/rshd. You can use the **ln** command to make symbolic links as follows:

```
ln -s /usr/lpp/tcpip/sbin/orexecd /usr/sbin/rexecd
```

Note: Depending on your installation option, the executable modules may be installed in a different directory.

The file named /usr/lpp/tcpip/sbin/orexecd is the executable module for REXECD server, and /usr/sbin/orexecd is the symbolic link that was created at the installation phase. If you decide to create a new symbolic link for REXECD server in UNIX System Services environment, you should use /usr/lpp/tcpip/sbin/orexecd as a source file.

However, we recommend that to avoid confusion you update the /etc/inetd.conf configuration file rather than create new links.

If the InetD is started with the sample configuration as it was, when you run REXEC and/or RSH client, these clients do not display any messages and you will see the error message below in the syslogd's message file or MVS console in the z/OS server system. Therefore, when you do not receive proper answers from clients, check on the server side.

```
INETD[pid[: execv /usr/sbin/rexecd: EDC5129I No such file or directory.
```

The REXECD server accepts the following arguments in the /etc/inetd.conf file: c.

-d	Write debug information to syslogd. The facility name is daemon.
-c	Translate all output messages in uppercase.
-l	Write each successful login to syslogd with user ID and command content. The facility name is auth.
-v	Write name and PTF level information to syslogd. The facility name is daemon.

The RSHD server accepts the following arguments in the /etc/inetd.conf file: c.

-d	Write debug information to syslogd. The facility name is daemon.
-c	Translate all output messages in uppercase.
-l	Write each successful login to syslogd with user ID and command content. The facility name is auth.
-v	Write name and PTF level information to syslogd. The facility name is daemon.

-a	Double-check the client IP address and host name correlation. The RSHD server will first do a gethostbyaddr() to obtain the host name of the client host. It will then do a gethostbyname() and verify that the IP address of the client is the same as that returned from the name server.
-r	If a client passes a null password, invoke /usr/sbin/ruserok user exit to authenticate the user ID.

The RSHD server will not execute a command if the client host IP address cannot be resolved to a host name.

There are a few situations where the RSHD server may encounter an error so early in the processing of a command that the server has not established a proper EBCDIC-to-ASCII translation yet. In such a situation, the client end user may see "garbage" data returned to his or her terminal. A packet trace will reveal that the response is in fact returned in EBCDIC, which is the reason for the garbage look on an ASCII workstation. We have seen this happen if the UNIX System Services name resolution has not been configured correctly, so the RSHD server, for example, was not able to resolve IP addresses and host names correctly. If your RSH clients encounter such a problem, please go back and check your name resolution setup. If you are using a local hosts table, make sure that the syntax of the entries in your hosts file is correct.

We also had seen an authentication error during our tests at ITSO-Raleigh. The REXEC server, for example, needs to be associated with a user who has READ authority to the BPX.DAEMON facility class. Otherwise your REXEC client's request will fail. For more information on BPX.DAEMON facility, refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

4.3 Trusted host concept with RSHD server

By APAR PN89720, the support of the *trusted host* concept wad added to the RSH server in UNIX System Services environment. The RSH server can accept the request from the remote client with a null password.

When the -r option for RSH server is enabled, if there is no password specified on the **RSH** command from the client, it will invoke the optional user exit, which is a program named /usr/sbin/ruserok. The name was hardcoded so you have to create your security user exit using exactly the same name. The RSHD will then pass a string to the program as parameters. These are passed in the following order:

1. Program name: /usr/sbin/ruserok
2. Host name: char array
3. Local user's UID: integer
4. Remote user ID: char array
5. Local user ID: char array

If RSHD receives a return code of zero (0) from the exit, RSHD continues. Any non-zero return code from the exit will cause RSHD to issue the message EZYRS25E to the client and terminate all connections.

If the user ID passed to the RSH server is not registered to the RACF database, RSHD does not invoke the user security exit but returns the following EZYRS25E message to the RSH client.

```
Rshd: EZYRS25E Unknown login.
```

The following code can be used as an example to begin building a working /usr/sbin/ruserok user exit. Note that in this sample exit, RSH server checks only the user ID that is used by the RSH client to log on to the server's system. In most situations you would need more logic to provide a secure RSH server on a OS/390 V2R5 and later systems.

```
 /*********************************************************************/
 /* ruserok.c    This is an example RSHD installation exit. Place     */
 /*              executable in /usr/sbin/ruserok.                     */
 /*********************************************************************/
 #include <stdio.h>
 #include <string.h>
 #include <fcntl.h>

 int main(int argc, char** argv)
 {

     char *rhost1;    /* "hostname" or "hostname.domain" of client
                         obtained by caller:
                         gethostbyaddr(getpeername()) */
     int cliuid;      /* uid of user name on this systems */
     char *cliuname;  /* user name on client's system */
     char *servuname; /* user name on this (server's) system */
                          /* the list of trusted users */
     char *u_trusted[] ={ "woza", "gdente", "silviar",
                          "eikens", "kakky" };
     int    cnt;

     rhost1 = *(argv+1);
     cliuid = atoi(*(argv+2));
     cliuname = *(argv+3);
     servuname =*(argv+4);

     {            /* write logging information to a file. */
       int fdes;
       char  buf[80];

       memset(buf,'\0',sizeof(buf));
       fdes=open("/tmp/rulog", O_WRONLY | O_CREAT | O_APPEND, 0666);
       if(fdes!=EOF){
         sprintf(buf,"rhosts = %s, uid=%d, cliname=%s, srvname=%s\n",
                 rhost1, cliuid, cliuname, servuname);
         write(fdes,buf,strlen(buf));
         close(fdes);
       }
     }
           /* check if the userid is trusted. */
     for(cnt=0; cnt<5; cnt++){
            if( (strcmp(servuname,*(u_trusted+cnt))==0))
                   return(0);     /* the userid is trusted. */
     }
    return(1);  /* reject this client */
 }
```

Figure 4-3 Sample exit source code for /usr/sbin/ruserok

To compile the user exit, you can use the Makefile utility which is a very useful tool in a UNIX system for building user-written applications, or just run the **c89** command as follows:

```
c89 -o ruserok ruserok.c
```

The following is the sample Makefile that was used at the ITSO-Raleigh to compile and link the user exit program above.

```
SRCS  = ruserok.c
OBJS  = ruserok.o
SYSLIBS  =
CFLAGS  = -D_ALL_SOURCE
CC              = c89
PGM  = ruserok

.c.o:
 $(CC) -c $(CFLAGS) $<

$(PGM): $(OBJS)
 $(CC) -o $@   $(OBJS) $(SYSLIBS)

clean:
 rm -f $(PGM) *.o core
```

Figure 4-4 Sample makefile

To allow the RSHD server to invoke the exit program, you need to configure the UNIX System Services name resolver to be able to resolve the IP address and host name correctly. If the RSH server cannot get the RSH client's host name from its IP address, you will receive an unreadable message and your request will fail. Therefore, all RSH clients need to be registered to your domain name server or local hosts file. Issuing the **PING** or **oping** command with a client's host name on an z/OS system, you may make sure the name resolution procedure works correctly.

Since the InetD forks a new process with RSHD server for each remote client's request, you can update the user exit dynamically.

The following are the RSHD server's output messages to the syslogd daemon, which is written when the RSHD server invokes the user exit.

```
EZYRS01I MVS OE RSHD PN89720
EZYRS48I Trusted host activated
EZYRS36I kakky@WTR05118.itso.ral.ibm.com as kakky: cmd = 'pwd'
```

Figure 4-5 Message when RSHD server received zero return

```
EZYRS01I MVS OE RSHD PN89720
EZYRS48I Trusted host activated
EZYRS49E Trusted host authentication failed
EZYRS36I alfredc@WTR05118.itso.ral.ibm.com as alfredc: cmd = 'pwd'
```

Figure 4-6 Message when RSHD server received non-zero return

4.4 REXEC client in the z/OS UNIX environment

The REXEC client component of the OS/390 V2R5 IP and later allows you, from the UNIX System Services shell environment, to execute commands on remote hosts.

The syntax of the REXEC client command is as shown in Figure 4-7.

```
Usage: orexec -V -d -n -l <user> -p <pwd>
              -s <port> fhost command
       options: -
                -?       display this message
                -d       turn on debug tracing
                -n       prevents automatic login
                -l <usr>  specifies remote login id
                -p <pwd>  specifies remote password
                -s <port> specifies server port
                -C       Uppercase messages
                -V       display APAR level

Example: orexec -d -l guest -p guest hostname ls -l
```

Figure 4-7 REXEC client command syntax in OS/390 UNIX

The **REXEC** command in UNIX System Services is **orexec**.

A sample invocation of the **orexec** command from UNIX System Services to a REXEC server on an OS/2 host looks like the following:

```
/u/alfredc: >orexec -l alfredc -p nosecret alfred dir config.sys

The volume label in drive C is C_DRIVE.
The Volume Serial Number is E6C3:0014.
Directory of C:\

 5-20-96   1:17p      5138           0  CONFIG.SYS
       1 file(s)       5138 bytes used
                    2591744 bytes free
/u/alfredc: >
```

Note: Since CS for OS/390 V2R8 IP, synonyms for UNIX commands are available to make it easier for users accustomed to a UNIX environment. You can use the **rexec** UNIX command instead of **orexec**.

5

z/OS UNIX sendmail

z/OS UNIX sendmail is a mail program running in an UNIX System Services shell. It replaces the SMTPROC known in previous MVS TCP/IP versions. However, if you intend to use the mail server also as a gateway to an NJE/RSCS network the SMTPROC server has to be used instead of the sendmail daemon because of rewriting the mail header for NJE/RSCS networks or vice versa. The sendmail program Version 8.8.7 is based on the Berkeley UNIX 4.1c BSD code.

This chapter describes briefly how to organize, configure and use a mail system with the z/OS UNIX sendmail program. It will also show the cooperation between the z/OS SMTP server and NJE gateway including the integration of the z/OS UNIX popper as mail server. The popper is a mail server running the Post Office Protocol (POP3).

Note: Because of the complexity of sendmail, we recommend you become familiar with the industry-accepted publication about sendmail: *sendmail* published by O'Reilly & Associates, Inc..

This book is well known as the *bat book*, because of the bat on the cover, by people responsible for mailing questions.

5.1 Overview and terms

The simple mail architecture defines Mail User Agent (MUA), Mail Transfer Agent (MTA) and Mail Delivery Agent (MDA).

- The MUA is any of various offered programs like the original UNIX mail program (/bin/mail) or the Berkeley MAIL program or the Netscape Communicator, etc. which a user runs to compose, dispose, read and reply to e-mail notes.
- The MTA is software which sends the prepared note by the MUA to a remote MTA responsible for the recipient using an SMTP connection.

  The sendmail program is an MTA on the sending and on the receiving side.
- Finally, the sendmail server uses a local mailer program (for example /bin/mail) to deliver the note to a mail spool file by appending the note to this file. **`sendmail`** now has finished its work.
- The user (MUA) may now retrieve his mail from the spool file.
- Another approach, more common today, is that a separate server, a popper server running the POP3 protocol, is used to retrieve notes from the mail spool file. This can be done for example through a Netscape Communicator working as a POP3 client invoking the popper server. A prerequisite however, is that the MUA supports the POP3 protocol.

  The popper is an MDA using the POP3 protocol for the transport between MDA and MUA.

5.1.1 Configuration of our basic tests

The following figure shows the relationship between MUA, MTA and MDA:

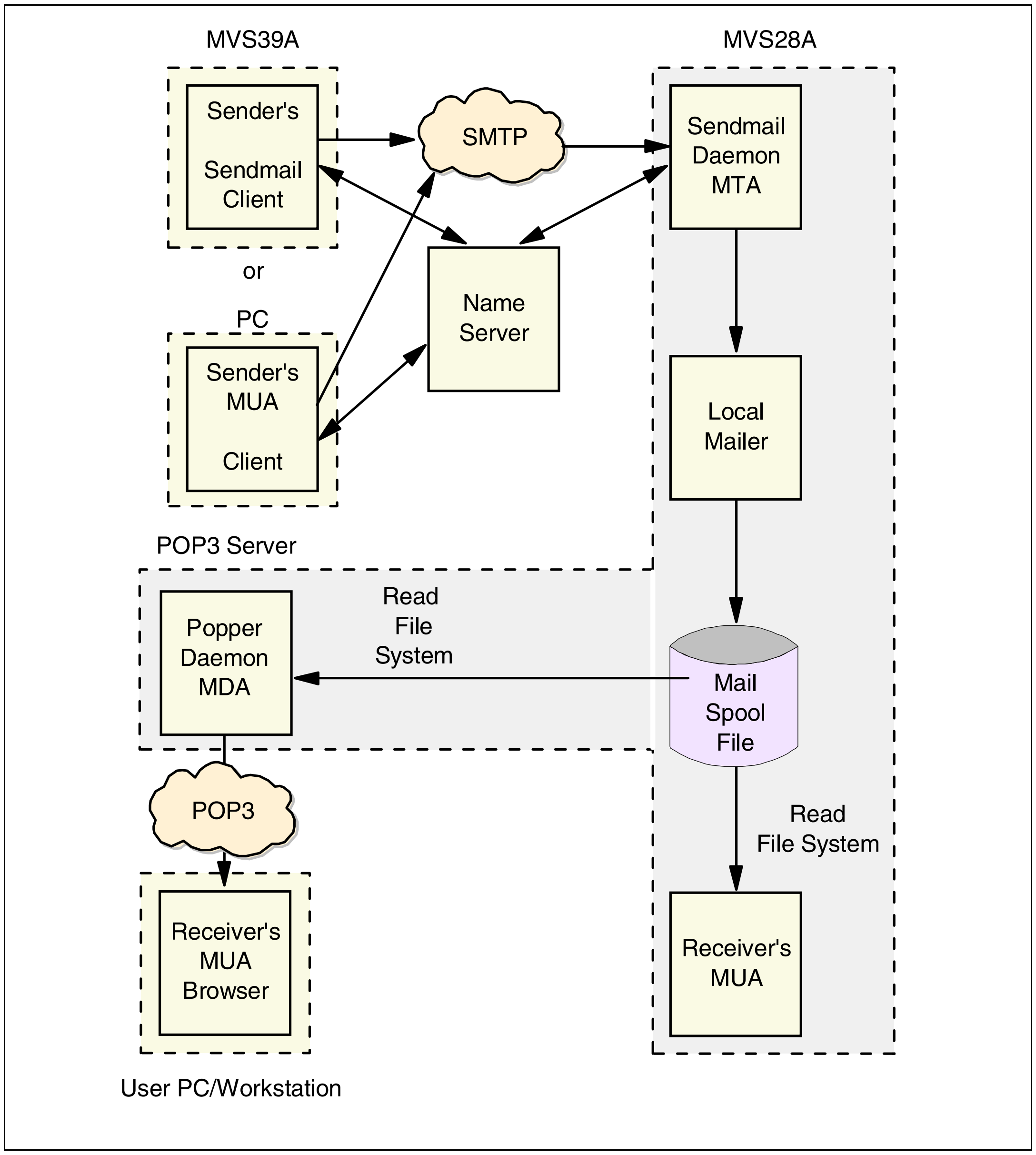


Figure 5-1 Relationship between MUA, MTA and MDA

For our basic tests we had two choices:

1. We used the client function of sendmail in an UNIX System Services system to transmit a file which contained the prepared message. The client function established the SMTP connection to the sendmail daemon at the recipient's machine. In this case we didn't need an MUA.
2. We used an MUA, the Netscape Communicator, on a PC to compose the message and to set up an SMTP connection as client to a MTA acting as mail server.

5.1.2 Configuration of our extended tests

In addition to our basic tests with sendmail and popper we also tested sendmail cooperating with the SMTP gateway transmitting messages to recipients in an NJE network and also to a Lotus Notes partner.

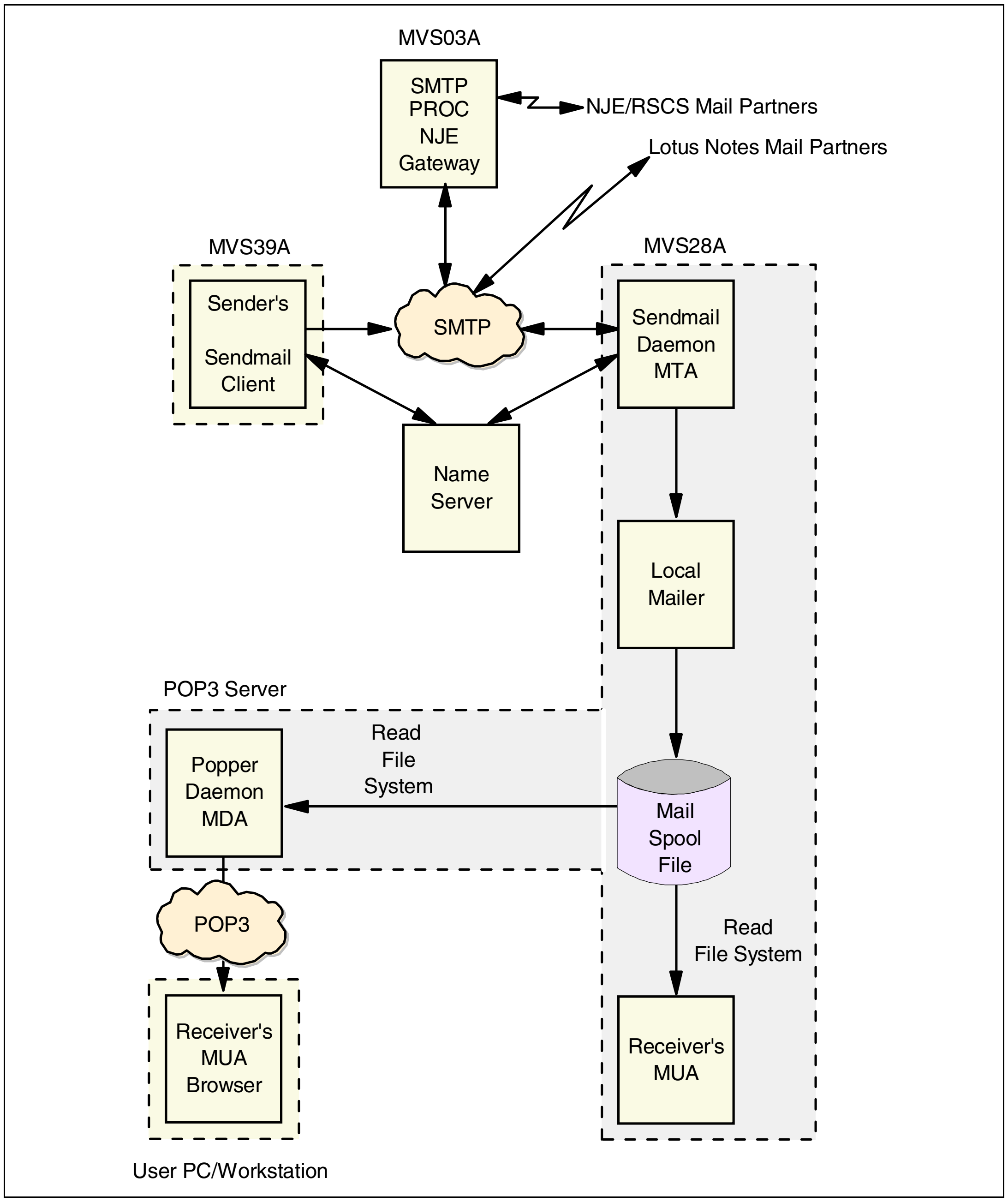


Figure 5-2 Relationship between MUA, MTA and MDA in an extended environment

5.2 Configuration of sendmail

Before we describe the configuration of sendmail we should look at the tasks of sendmail to understand the requirements of the configuration.

sendmail transports mail messages to other machines, listens to the network for incoming mail and hands local mail to a local mailer program for local delivery. It may queue mail for later delivery and alias the recipients' names to other users' names.

For these tasks, sendmail needs information in files that, for example, define the local mailer program, which file system is responsible for deferred delivery and which file contains information about alias names and real names.

When sendmail is run, it first reads the /etc/sendmail.cf configuration file. Among the many items contained in that file are locations of all files and directories which sendmail needs. The following figure shows the files and directory needed by sendmail:

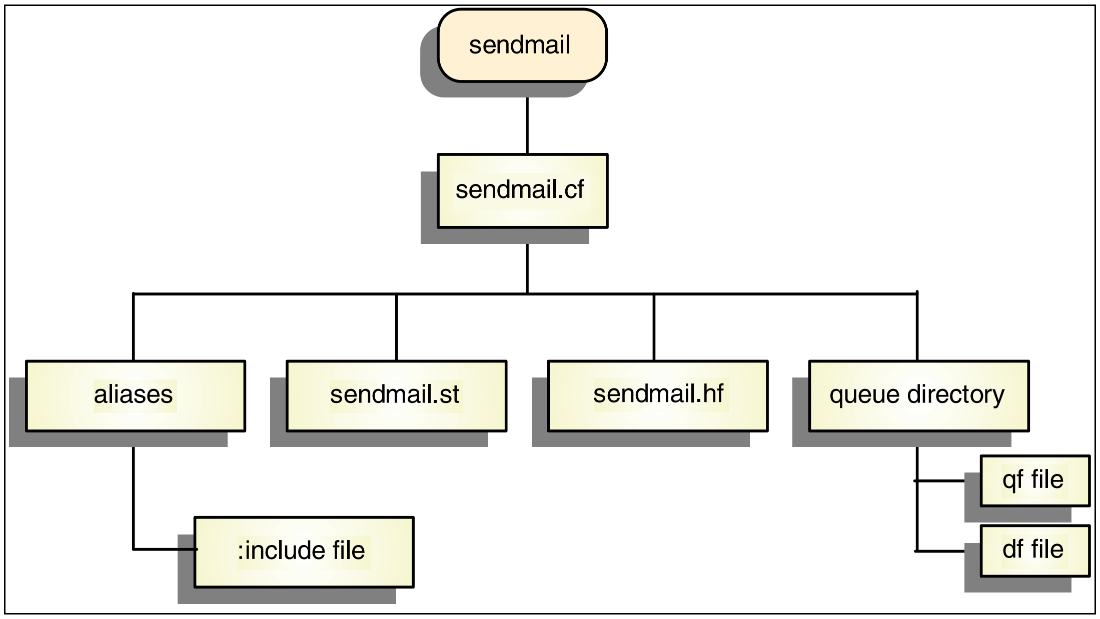


Figure 5-3 sendmail file structure

The sendmail code is located in the directory /usr/lpp/tcpip You will find samples in the directory /usr/lpp/tcpip/samples/sendmail/cf.

```
PESCHKE @ RA28:/>cd /usr/lpp/tcpip/samples/sendmail
PESCHKE @ RA28:/usr/lpp/tcpip/samples/sendmail>ls
README.m4       feature         m4              sh
cf              hack            mailer          siteconfig
domain          inetd.conf.pop  ostype
```

You will miss a sample of the aliases file and the queue directory. The queue file system will be created by the sendmail program itself. Both have to be created before you run sendmail.

The default queue directory entry may be defined under:

```
/usr/spool/mqueue
```

The default alias file may be defined under:

```
/etc/aliases
```

Later we will describe the contents of the most important directories and files.

We will first describe the needed directories and files used by sendmail before we talk about how to create the sendmail.cf configuration file.

The required files are:

- Aliases file for converting recipient's name and to address the postmaster
- Queue directory for deferred messages
- Sendmail statistic file
- Sendmail help file

The **grep** command may help you to check the location of available files. Please see the following example:

```
grep =/ /etc/sendmail.cf
********************************* Top of Data **********************
O AliasFile=/etc/aliases
#O ErrorHeader=/etc/sendmail.oE
O HelpFile=/usr/lib/sendmail.hf
O QueueDirectory=/usr/spool/mqueue
O StatusFile=/etc/sendmail.st
#O UserDatabaseSpec=/etc/userdb
#O ServiceSwitchFile=/etc/service.switch
#O HostsFile=/etc/hosts
#O SafeFileEnvironment=/arch
Mlocal,..P=/usr/lib/tsmail, F=lsDFMAw5:/|@quUbE, S=10/30, R=20/40,
Mprog,..P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/,
******************************** Bottom of Data ********************
```

Figure 5-4 Location of sendmail files as defined in /etc/sendmail.cf

5.2.1 Alias file

The alias file is used to convert the alias name to another recipient's name. There is no sample available in the samples directory. We used the following content for our tests taken from the general publication.

```
PESCHKE @ RA28:/>cat etc/aliases
# Mandatory aliases
postmaster: PESCHKE, /u/peschke/postmaster'
                     "|/usr/local/bin/notify PESCHKE@MVS03A
MAILER-DAEMON: VANDEKE@MVS03A
# Other
Chris03a: VANDEKE@MVS03A
Chris28a: VANDEKE@MVS28A
Chris39a: VANDEKE@MVS39A
Chrisvm:  VANDEKER%WTSCPOK@MVS03A
Rolandvm: PESCHKER%WTSCPOK@MVS03A
Roland3a: PESCHKE@MVS03A
Rolan28a: PESCHKE@MVS28A
rolan39a: PESCHKE@MVS39A
```

Figure 5-5 Alias file

The postmaster entry is for undeliverable mail into the file /u/peschke/postmaster with reference that the user PESCHKE@MVS03A will be notified on another machine.

Aliases database

Because sendmail may have to search through a huge list of names in the aliases file, the use of a database would significantly improve the lookup speed. Therefore, run one of the following commands:

```
/usr/sbin/newaliases
```

or

```
/usr/sbin/sendmail -bi
```

After sendmail has built/rebuilt the database you will see the result on your screen.

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/newaliases
/etc/aliases: 15 aliases, longest 32 bytes, 333 bytes total
```

You may also check a specific entry. For example:

```
PESCHKE @ RA28:/>/usr/sbin/sendmail -bv rolan28a
PESCHKE@MVS28A... deliverable: mailer local, user PESCHKE
```

5.2.2 Queue directory

If a mail message cannot be transmitted because of certain reasons, sendmail stores it in this queue directory until it can be sent successfully. Possible reasons for queuing a message may be:

- The remote machine is down
- Temporary disk problems
- Sendmail on the other machine is not startable
- TCP has problems to set up the connection

Displaying queue information

If you have the permission to look at the queue directory, you may find it empty (all messages are sent) or find some dfxxxxxxxx and qfxxxxxxxx files which still have to be sent.

```
PESCHKE @ RA39:/u/peschke>ls /usr/spool/mqueue
dfIAA402653203   dfPAA100663311   qfJAA1090519058  qfPAA1476395014
dfJAA1090519058  dfPAA1476395014  qfJAA234881032   qfPAA251658251
dfJAA234881032   dfPAA251658251   qfJAA335544328   qfPAA822083602
dfJAA335544328   dfPAA822083602   qfJAA402653198   qfPAA83886093
dfJAA402653198   dfPAA83886093    qfJAA536870926   qfQAA117440537
dfJAA536870926   dfQAA117440537   qfJAA570425351   qfQAA167772175
dfJAA570425351   dfQAA167772175   qfJAA754974734   qfQAA1962934278
dfJAA754974734   dfQAA1962934278  qfJAA989855751   qfQAA637534211
dfJAA989855751   dfQAA637534211   qfLAA1140850706  qfQAA83886105
dfLAA1140850706  dfQAA83886105    qfLAA218103813   qfQAA83886107
dfLAA218103813   dfQAA83886107    qfMAA301989894   qfRAA1979711494
```

Figure 5-6 Files in the queue directory

When a message is queued, it is split into two parts. Each part is saved in a separate file.

1. qf... files contain header information
2. df... files contain the body

A sample of a qf... file with the header part is shown in Figure 5-7 on page 46.

The **obrowse** command was used in the TSO ISHELL. The superuser permission is required to view this file.

```
 BROWSE -- /usr/spool/mqueue/qfIAA402653203 --------- Line 00000000 Col 0
********************************* Top of Data ***************************
V2             1
T918568591     2
K918568681     3
N1             4
P30226         5
IO/1/604111    6
Mhost map: lookup (MVS28A): deferred  7
$_OMVSKERN@localhost                  8
SPESCHKE                              9
C:rolan28a                            10
RPFD:PESCHKE@MVS28A                   11
H?P?Return-Path: <PESCHKE>            12
HReceived: (from OMVSKERN@localhost)  13
 .by MVS39A.itso.ibm.com (8.8.7/8.8.7) id IAA402653203
 .for rolan28a; Tue, 9 Feb 1999 08:56:31 -0500
H?D?Date: Tue, 9 Feb 1999 08:56:31 -0500  14
H?F?From: PESCHKE <PESCHKE>               15
H?x?Full-Name: PESCHKE                    16
H?M?Message-Id: <199902091356.IAA402653203@MVS39A.itso.ibm.com>  A
```

Figure 5-7 Queue file

Following is a description of the header lines:

1 Version of the qf file: V2 means above the V8.8 sendmail version

2 Time created in seconds to limit the message to remain in the queue

3 Determines the time to wait before retry delivery

4 Number of attempts for each delivery

5 Priority when processed from the queue

6 After a system crash the message is stored in lost+found under the referenced number in case the qf file lost its directory entry

7 Reason why the message was stored in the queue (= deferred)

8 Full canonical name of the sender's machine

9 Sender's address

10 For security reasons, this is the real recipient of the message

11 Recipient's address

12 Header information for the return path in case the message cannot be delivered

13 HReceived: where the message came from

14 Header information when the message was received by the MTA

15 Header information about the sender

16 Header information about the full name of the sender

A Header information about the queue file numbers

All header information is based on entries in sendmail's configuration file /etc/sendmail.cf.

A sample of a df... file with the body of the mail is shown in Figure 5-8.

The **obrowse** command was used in the TSO ISHELL. The superuser permission is required to view this file.

```
 BROWSE -- /usr/spool/mqueue/dfIAA402653203 --------- Line 00000000 Col 0
 Command ===>                                                 Scroll ===
********************************* Top of Data ***************************
This message was sent from system RA39 to RA28 using the sendmail
client to a sendmail daemon, started with
                 /usr/sbin/sendmail -bd
The messages was sent with
/usr/sbin/sendmail -v Rolan28a </u/peschke/mailto28
******************************** Bottom of Data *************************
```

Figure 5-8 sendmail data file

Another way to get queue information

You will get queue information for all messages in the queue by running the **sendmail** command with the `-bp` command-line switch (printing the queue). In this case we used the ISHELL instead of OMVS to enter the command.

```
 Enter a Shell Command


 Enter a shell command and press Enter.

 Standard output and standard error are redirected to a temporary
 file.  If there is any data in the file when the shell command
 completes, the file is displayed.
    /usr/sbin/sendmail -bp________________________________________
    ______________________________________________________________
    ______________________________________________________________
    ______________________________________________________________






  F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel
```

Figure 5-9 sendmail -bp command from the ISHELL

```
 BROWSE -- /tmp/PESCHKE.11:07:02.633773.ishell ------ Line 00000000 Col 0
 Command ===>                                                  Scroll ===
********************************* Top of Data **************************
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient----------
NAA335544336       51 Fri Feb 19 13:45 VANDEKE
                 (host map: lookup (mvs03a): deferred)
MAA1728053273       51 Tue Feb 23 12:52 VANDEKE
                 (dbm map "Alias0": unsafe map file /etc/aliases)
NAA251658256       51 Fri Feb 19 13:36 VANDEKE
                 (host map: lookup (MVS03A): deferred)
OAA218103822       95 Thu Feb 18 14:50 PESCHKE
                 (Deferred: Connection refused by mvs03a.itso.ral.ibm.com.)
OAA100663310       95 Thu Feb 18 14:11 PESCHKE
                 (host map: lookup (MVS39A): deferred)
OAA117440526       95 Thu Feb 18 14:13 PESCHKE
                 (host map: lookup (MVS39A): deferred)
KAA855638018  (no control file)
******************************** Bottom of Data ***********
```

Figure 5-10 sendmail queue display

Processing the queue

The sendmail program offers two different methods for processing the queue:

1. Process the queue periodically.
   - Process the queue periodically with the -q command line switch:

     A typical invocation for the sendmail daemon looks like this:

     ```
     /usr/sbin/sendmail -bd -q1h
     ```

     which means: process the queue every hour (-q1h) by sendmail running as daemon in the background (-bd) in the listening mode for incoming SMTP connections.
2. Process the queue once and then exit.

   For this case, only the -q command line switch has to be used. For example:

   ```
   /usr/sbin/sendmail -q
   ```

   This command tells sendmail to process the queue only once and then exit. The command can also be used to force processing the queue even if you run the periodically processing mode and you want to get the messages out immediately.

   The -q switch, without an argument, prevents sendmail from running in the background and detaching from its controlling workstation.

   You may also combine -q with -v to view more information about the queue.

   ```
   /usr/sbin/sendmail -qv
   ```

For more detailed information about queueing, we recommend reading *sendmail*, published by O'Reilly & Associates, Inc..

5.2.3 The sendmail.st file

This is a statistics file which will be used by sendmail to record the number and sizes of all incoming and outgoing mail messages handled by each delivery agent. Delivery agents are:

- Local delivery agent
- SMTP delivery agent
- UUCP delivery agent

Statistical information is collected if the option (O) is defined in the sendmail.cf file:

```
# status file
O StatusFile=/etc/sendmail.st
```

Later we will give more details about the creation of the sendmail.cf file in 5.2.5, “The sendmail.cf file” on page 50.

Since sendmail does not create this statistics file, you have to do this first, before using the statistics option. The following example shows how to create the sendmail.st file.

```
PESCHKE @ RA39:/u/peschke>touch /etc/sendmail.st
PESCHKE @ RA39:/u/peschke>ls -l /etc/sendmail.st
-rw-r--r--   1 OMVSKERN DCEGRP         0 Feb 15 16:12 /etc/sendmail.st
```

Be aware that you are a superuser.

The collecting of statistics may be turned off by deleting or renaming this file.

You may view the statistics file using the commands:

```
mailstats -C </etc/sendmail.cf>
```

or

```
mailstats -s </etc/sendmail.st>
```

The configuration file and the statistics file have to be defined explicitly only if other file names in the sendmail.cf file are used than the referenced default file names shown in the previous command sample.

We used the **mailstats** command without arguments using the default names.

```
PESCHKE @ RA39:/u/peschke> /usr/sbin/mailstats
Statistics from Mon Feb 15 16:18:48 1999
 M msgsfr bytes_from  msgsto   bytes_to  Mailer
========================================
 T      0          0K      0          0K
```

After issuing the **touch** command, the sendmail.st file is of course empty; therefore, you don't see any collected data. Now, we sent one message and received five messages. After processing the **mailstats** command, you see the following result.

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/mailstats
Statistics from Tue Feb 16 09:10:49 1999
 M msgsfr bytes_from  msgsto   bytes_to  Mailer
 3      3          3K      0          0K  local
 5      0          0K      1          1K  esmtp
=======================================
 T      3          3K      1          1K
```

After sending and receiving new messages, **mailstats** was issued again. The total counters (see line T) from msgsfr (message from) and bytes_from (received bytes) were updated and also the ones for msgsto (sent messages) and bytes_to (sent bytes).

The lines above the total line show the values for each mailer. The top line gives information from when **mailstats** started collecting data.

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/mailstats
Statistics from Tue Feb 16 09:10:49 1999
 M msgsfr bytes_from  msgsto   bytes_to  Mailer
 3      4          4K      0          0K  local
 5      0          0K      2          2K  esmtp
=======================================
 T      4          4K      2          2K
```

5.2.4 The sendmail.hf file

This is the HELP file implemented for SMTP (and ESMTP) which the sendmail program is looking up to build help messages. You should find this file in the /usr/lpp/tcpip/lib directory.

If you want to view this file, you can issue the following command:

```
obrowse /usr/lpp/tcpip/lib/sendmail.hf
```

We have discussed all the important files the sendmail.cf file points to when sendmail is run. We will now look at the sendmail.cf file itself.

5.2.5 The sendmail.cf file

This file is read and parsed by the sendmail program every time sendmail starts. It lists locations of important files and specifies the default values sendmail is working with. It describes sendmail's behavior and contains rules and rule sets (for example, for rewriting of mailing addresses). This information is defined through configuration commands.

Examples of configuration commands are:

M	Define a mail delivery agent
R	Define rewriting rules
H	Define a header
P	Define delivery priorities
T	Define a trusted user

D	Define a macro
O	Define an option
S	Declare a rule-set start

The definition of a configuration file may be very simple if you use an empty configuration file as a base. You get this empty configuration file by using the following copy command:

```
cp /dev/null/ /etc/client.cf
```

Our new configuration file is called client.cf because we don't want to overwrite the currently existing sendmail.cf file. In the next step, the bat book recommends that you run the sendmail program using the new client.cf file with the switch -bt, which means execution is done in test mode. The `-C` command-line switch tells sendmail to use our new client.cf file. The result was:

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/sendmail -Cclient.cf -bt < /dev/null
EZZ7575I No local mailer defined: EDC5113I Bad file descriptor.
ADDRESS TEST MODE (rule-set 3 NOT automatically invoked)
Enter <rule-set> <address>
```

Chapter 5.2 in the bat book says that this should run with an empty configuration file without any problems.

In our case, the sendmail showed a bad file descriptor. We got the message that a local mailer had to be defined. So we defined a local mailer by copying the Mlocal line of the /etc/sendmail.cf file into our /etc/client.cf file and changed the S=10/10 rule-set parameter. The rerun showed the result as expected:

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/sendmail -Cclient.cf -bt < /dev/null
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
```

Contents of our new configuration file client.cf are as follows:

```
EDIT       /etc/client.cf                                  Columns 00001
****** ***************************** Top of Data ************************
000001 Mlocal,  P=/usr/lib/tsmail, F=lsDFMAw5:/|@quUbE, S=10/10, R=20/40,
000002   T=DNS/RFC822/X-Unix,
000003   A=tsmail $u
****** *************************** Bottom of Data *********************
```

As you remember:

- The `M` command defines a delivery agent; in this case the local mailer using the program P=/usr/lib/tsmail.
- F= Its flags tell sendmail more about the delivery agent.
- S= Tells which rule-set has to be used for the sender for rewriting the sender's address to user@host.domain while other rules for a UUCP agent have to be converted to host!user.
- R= Specifies the rule-set to be used for the receiver for rewriting the address. Rule 20 means rewriting an envelope address, rule 40 means rewriting a header address.

- T= Specifies the lookup of the sender's and receiver's addresses.
- A= Specifies the command-line arguments to be supplied to each mail delivery program.

There are many more definitions in the sendmail.cf file, so this file can be very complex. If you browse through the configuration file found in:

```
/usr/lpp/tcpip/samples/sendmail/cf/sample.cf
```

you will find very complex commands which you have to learn and type in accurately. See the following sample, which is an excerpt only, to show you the syntax of some definitions.

```
######################################################################
######################################################################
#####
#####...REWRITING RULES
#####
######################################################################
######################################################################

############################################
###  Ruleset 3 -- Name Canonicalization  ###
############################################
S3

# handle null input (translate to <@> special case)
R$@...$@ <@>

# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*...$: $1 <@>...mark addresses
R$* < $* > $* <@>.$: $1 < $2 > $3...unmark <addr>
 .............................
 .............................
```

Figure 5-11 Part of the sample sendmail.cf file

In order to make it easier to test sendmail, you only have to copy the /usr/lpp/tcpip/samples/sendmail/cf/sample.cf file as /etc/sendmail.cf and use this. This sample.cf file was also used for all our tests.

Later, when you are more familiar with creating your own configuration file, we recommend you use your own sendmail.cf file. We will discuss this procedure in “M4 preprocessor” on page 53.

The sample.cf file we used was created automatically running an m4 macro preprocessor with a master file as an input file which you will find as sample.mc in the same directory:

```
/usr/lpp/tcpip/samples/sendmail/cf
```

If you browse through this sample.mc file, you will notice that the main information is defined, which helps you run sendmail without the m4 macro preprocessor run. All of the following described tests have been run with the sample.cf configuration file which was created using the following sample.mc file:

```
 BROWSE -- /usr/lpp/tcpip/samples/sendmail/cf/sample.mc
 Command ===>
********************************* Top of Data **********************
divert(-1)
divert(0)dnl
VERSIONID(`OS/390 sample configuration  12/4/97')
OSTYPE(os390)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
******************************** Bottom of Data ********************
```

Figure 5-12 sample.mc file

Note: `dnl` means delete through new line.

If you want to learn this configuration language or need more information about the configuration parameter refer to *sendmail*, published by O'Reilly & Associates, Inc..

In order to give you an overview of the m4 preprocessor run the following section will guide you through the steps needed for the sendmail configuration process with m4.

5.2.6 M4 preprocessor

The m4 macro processor is a front-end processor for any programming language being used in the operating system environment. Besides replacing one string of text with another, the m4 macro processor provides the following features:

- Arithmetic capabilities
- File manipulation
- Conditional macro expansion
- String and substring functions

The m4 macro preprocessor can be given input that will generate a z/OS UNIX sendmail configuration file. It takes as input a user-defined master configuration source file (.mc file) that can define mail delivery mechanisms using files provided in the samples directory.

M4 preprocessor requirements

When you run the m4 preprocessor, you need some files that contain input definitions and an output file which will be your sendmail configuration file. The input files are:

- /m4/cf.m4 - which provides support for different include files such as cfhead.m4 pointing also to proto.m4.
- /cf/sample.mc - which points with its parameters to different files and their locations. We will describe these pointers later.

The cf.m4 file is located in the directory:

```
/usr/lpp/tcpip/samples/sendmail/m4
```

The sample.mc file is located in the directory:

```
/usr/lpp/tcpip/samples/sendmail/cf
```

Building your own sendmail configuration environment

You may use the above referenced input files in the m4 preprocessor run to create your own sendmail.cf output file. You may also customize these input files.

However, we recommend that you create your own configuration directory and copy the later described directories and files to your own directory keeping the same directory structure. This will keep you independent from the license product samples. It will also protect you from any license product updates which might overwrite your definitions in the sample files.

- First step: create a private configuration directory.

  Note: Be aware that you are a superuser.

  ```
  PESCHKE @ RA28:/u/peschke>mkdir /u/peschke/smconfig
  ```

- Second step: list all directories and files to copy to your private configuration directory.

```
PESCHKE @ RA28:/u/peschke/smconfig>ls -la /usr/lpp/tcpip/samples/sendmail
total 480
drwxr-xr-x  11 15069172 2736       8192 Nov 13 18:52 .
drwxr-xr-x   4 15069172 2736       8192 Feb 18 07:46 ..
-rw-r--r--   2 OMVSKERN 2736     150417 Nov 13 18:52 README.m4
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 cf
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 domain
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 feature
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:51 hack
-rw-r--r--   2 OMVSKERN 2736        567 Nov 13 18:51 inetd.conf.pop
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 m4
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 mailer
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 ostype
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:51 sh
drwxr-xr-x   2 15069172 2736       8192 Nov 13 18:52 siteconfig
```

Figure 5-13 Directories containing files needed for M4

Note: The inetd.conf.pop file will not be needed. It will be used only for the dedicated z/OS UNIX System Services popper server implementation. See 5.4, “The popper server” on page 75.

- Third step: copy the listed files and directories to your private configuration directory.

  Below is an example for copying directories with containing files:

  ```
  PESCHKE @ RA28:/u/peschke>cp -R /usr/lpp/tcpip/samples/sendmail/m4 /u/peschke/sm
  config
  ```

After copying the /m4 directory, your private sendmail configuration directory should look like this:

```
PESCHKE @ RA28:/u/peschke>ls -la /u/peschke/smconfig/m4
total 184
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  9 10:42 .
drwxr-xr-x   3 OMVSKERN OMVSGRP     8192 Mar  9 10:42 ..
-rw-r--r--   1 OMVSKERN OMVSGRP     4050 Mar  9 10:42 cf.m4
-rw-r--r--   1 OMVSKERN OMVSGRP     6078 Mar  9 10:42 cfhead.m4
-rw-r--r--   1 OMVSKERN OMVSGRP    10935 Mar  9 10:42 nullrelay.m4
-rw-r--r--   1 OMVSKERN OMVSGRP    31714 Mar  9 10:42 proto.m4
-rw-r--r--   1 OMVSKERN OMVSGRP     3159 Mar  9 10:42 version.m4
PESCHKE @ RA28:/u/peschke>
```

Now copy the /cf directory.

```
PESCHKE @ RA28:/u/peschke>cp -R /usr/lpp/tcpip/samples/sendmail/cf /u/peschke/sm
config
PESCHKE @ RA28:/u/peschke>ls -la /u/peschke/smconfig/cf
total 240
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  9 10:51 .
drwxr-xr-x   4 OMVSKERN OMVSGRP     8192 Mar  9 10:51 ..
-rw-r--r--   1 OMVSKERN OMVSGRP    26542 Mar  9 10:51 sample.cf
-rw-r--r--   1 OMVSKERN OMVSGRP      648 Mar  9 10:51 sample.mc
```

Rename the sample.cf file and use it for your own customization.

```
PESCHKE @ RA28:/u/peschke>cd smconfig/cf
PESCHKE @ RA28:/u/peschke/smconfig/cf>ls
sample.cf  sample.mc
PESCHKE @ RA28:/u/peschke/smconfig/cf>mv sample.mc peschke.mc
PESCHKE @ RA28:/u/peschke/smconfig/cf>ls
peschke.mc  sample.cf
```

Copy the remaining directories and files from the sendmail directory. At the end of this process, your private configuration directory should have the following listed members:

```
PESCHKE @ RA28:/u/peschke/smconfig>ls -la
total 640
drwxrwxrwx  11 PESCHKE  OMVSGRP     8192 Mar  8 12:30 .
drwxr-xr-x   3 PESCHKE  OMVSGRP     8192 Mar  8 14:47 ..
-rw-r--r--   1 OMVSKERN OMVSGRP   150417 Mar  8 12:26 README.m4
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 15:58 cf
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:28 domain
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:29 feature
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:28 hack
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:30 m4
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:28 mailer
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:30 ostype
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:29 sh
drwxr-xr-x   2 OMVSKERN OMVSGRP     8192 Mar  8 12:30 siteconfig
PESCHKE @ RA28:/u/peschke/smconfig>
```

M4 preprocessor run

You now know which directories and files are needed to run the m4 preprocessor. In order to run the m4 program you have to download the compiled code from the Web server and locate it in the directory /usr/sbin or your private directory.

You will find a compiled version of the code using:

```
http://www.s390.ibm.com/oe/bpxa1toy.html
```

This URL leads you to the OS/390 Tools & Toys Web page. Now select **Related Link -> Ported Tools**; search for the m4 package, select and download the compiled code.

We were able to download the m4 code with a PC running under OS/2 and the Netscape Navigator browser. Running the same procedure on a PC under Windows NT 4.0 and Netscape Communicator brought the following error message when clicking the package m4:

```
IMW0254E

Error 406

IMW0326E Not Acceptable - No file exists which can satisfy the accept headers
sent with the request.

Lotus Domino Go Webserver - North American Edition for OS/390 V5R0M0
```

Figure 5-14 Error when using Windows NT 4.0 and Netscape Communicator

Relationship of the files used during the m4 run

The following figure shows an overview of the relationship of the files' statements and what impact they have on the configuration process.

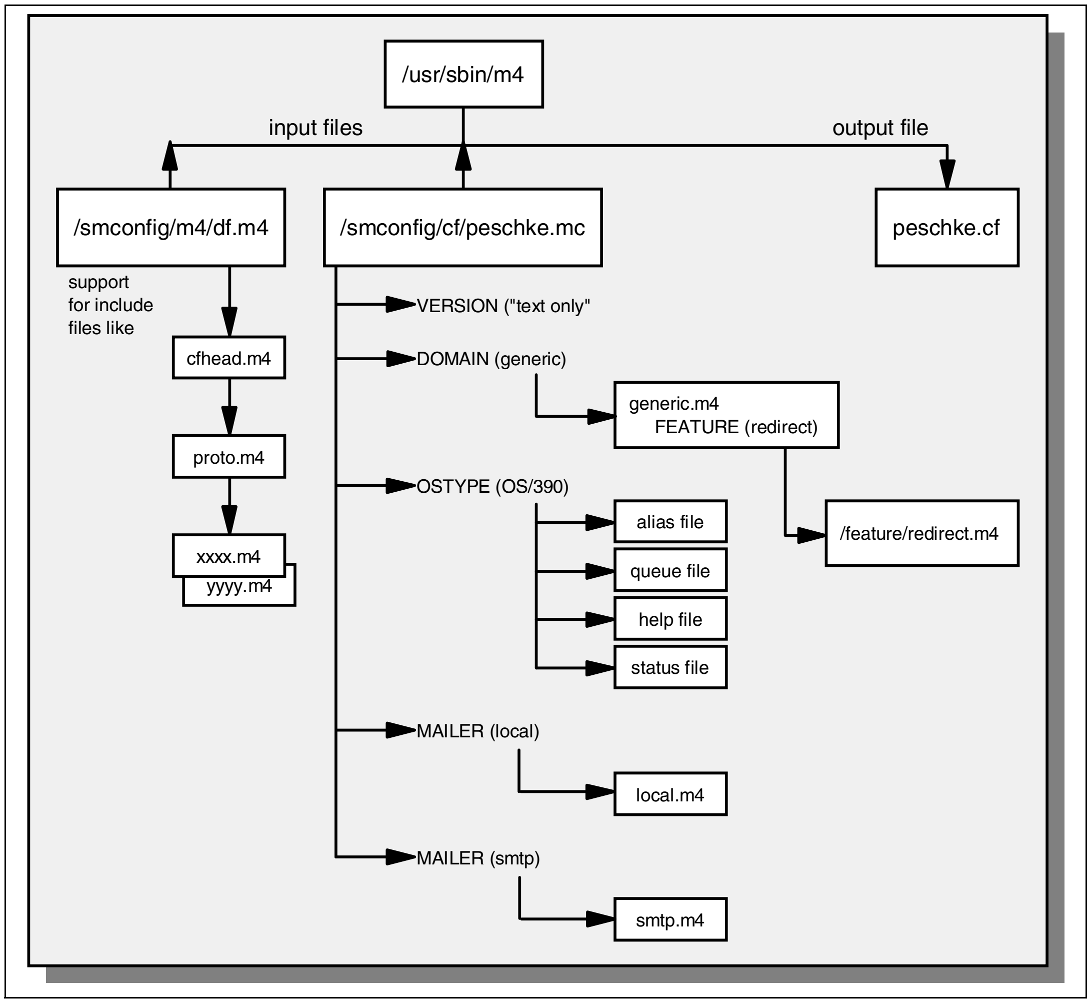


Figure 5-15 Relationship of m4 files/parameters

As you see, there are two main input files which are first read by the m4 program.

- The /m4/cf.m4 file
- Your private configuration .mc file.

The output file will be your private .cf file.

These files have to be defined with their paths for the m4 run.

For example:

```
/u/peschke/smconfig/cf>/usr/sbin/m4 /u/peschke/smconfig/m4/cf.m4 peschke.mc >peschke.cf
```

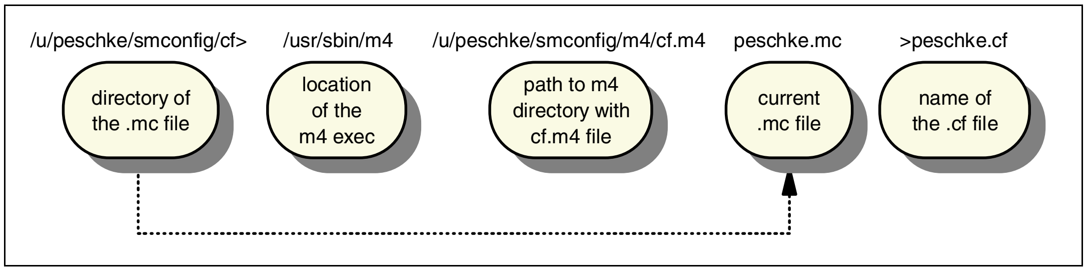


Figure 5-16 m4 run command line

m4 run

Using the command line shown above, m4 reads the .mc file. It also finds, through the definition of the path to the m4 directory, the cf.m4 file which can be seen as an include file for the cfhead.m4 file and other .m4 files like the file proto.m4.

The .mc file is scanned through and the different statements with their parameters lead to other .m4 files which should be customized before you run the m4 preprocessor.

You need to customize only the .m4 files if you don't agree to the provided definitions. For example: if you want to have the alias file in another directory, you have to change the path in the os390.m4 file. The best way is to leave the definitions as they are and take the defaults.

Pointers to .m4 files from the .mc file

When the .mc file is read by the m4 program, most statements point to the .m4 files. Only the VERSION statement doesn't point to an .m4 file.

- DOMAIN which points with its parameter generic to the generic.m4 file. The generic domain may be used by all clients sending this sendmail server their messages to forward to the recipients. If the m4 macro MASQUERADE_AS the server is set, all clients of this server will get the server's host and domain name. The recipients will see the messages coming from the client's server and not from the client's host and domain name.

  The generic.m4 file also has a statement FEATURE with a parameter redirect which points to the file redirect.m4.

  The redirect function refers to an entry in the alias file. It is used, for example, for retired users to redirect messages from the local system to another remote system. Look at the following entry in an alias file.

  ```
  entry of an alias file
  ----------------------
  roland:      rpeschke@csi.com.REDIRECT
  ```

  In this case, the sender would be informed with the following message to redirect e-mail:

  ```
  551 user not local; please try <rpeschke@csi.com>
  ```

- OSTYPE points to the file os390 which has the default addresses for
  - Alias file
  - Queue files
  - Help file
  - Status file
  - Local mailer path (/bin/mail)

- MAILER with the parameter local points to the local.m4 file. This file defines the local mailer program plus envelope sender/recipient rewriting and domain name adding information.

  The line in the .cf file

  ```
  Mlocal,..P=/usr/lib/tsmail
  ```

  points to the local mailer program tsmail.
- MAILER with the parameter smtp points to the smtp.m4 file. This file defines the smtp characteristics between the mail delivery agents.

There are also other MAILERs possible. Please see the directory mailer in your private configuration file.

For more detailed information we recommend to look at *sendmail*, published by O'Reilly & Associates, Inc. and the README.m4 file in your configuration directory.

M4 special file definition rules

If you look at the .mc file, you will discover some commands you are not familiar with, for example, **divert** or **dnl**. You will find these commands in the .mc file.

dnl

Means delete through new line. It is a special method of m4. If you did not use dnl then m4 would create a blank line for each command line. For example in the .mc file:

```
divert(-1)
divert(0)dnl
VERSIONID(`OS/390 sample configuration  03/08/99')
OSTYPE(os390)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

The output of m4 without dnl:

```
<--- blank line
<--- blank line
<--- blank line
<--- blank line
<--- blank line
#####  OS/390 sample configuration  03/08/99  #####
#####  $Id: OS390.m4 0.2 1996/12/20 04:32:47 gmalet Exp $  #####
###############################################
###   Local and Program Mailer specification   ###
###############################################

#####  @(#)local.m4.8.23 (Berkeley) 5/31/96  #####

Mlocal,..P=/usr/lib/tsmail, F=lsDFMAw5:/|@quUbE, S=10/30, R=20/40,
Mprog,..P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/,
##################################
###   SMTP Mailer specification   ###
##################################

#####  @(#)smtp.m4.8.33 (Berkeley) 7/9/96  #####

Msmtp,..P=IPC , F=mDFMuX, S=11/31, R=21, E=\r\n, L=990,
```

```
Mesmtp,..P=IPC , F=mDFMuXa, S=11/31, R=21, E=\r\n, L=990,
Msmtp8,..P=IPC , F=mDFMuX8, S=11/31, R=21, E=\r\n, L=990,
Mrelay,..P=IPC , F=mDFMuXa8, S=11/31, R=61, E=\r\n, L=2040,
```

To suppress this insertion of blank lines use the special m4 command **`dnl`**.

divert

When defining divert, m4 uses a technique to divide its input into different parts in order to later reassemble them in a more logical fashion. For example, all options should be put together for the output.

The following m4 default diversions are used by sendmail:

(-1)	Tells m4 to ignore all lines that follow
(0)	Tells m4 to stop diverting and to output immediately
(1)	Local host detection and resolution
(2)	Rule set 3
(3)	Rule set 0
(4)	Rule set 0 with UUCP (UNIX to UNIX
(5)	....

For more information see *sendmail*, published by O'Reilly & Associates, Inc..

5.3 Running sendmail

Sendmail can run as a client or as a server/daemon.

5.3.1 sendmail's client mode

The simplest way to send messages to other recipients is to create a file and direct this file and the recipient's name to the sendmail program. See the following sample.

```
PESCHKE @ RA39:>/usr/sbin/sendmail rolan28a </u/peschke/mailto28
```

After logging on to the recipient's system RA28 the following information appears:

```
you have mail in /usr/mail/PESCHKE.
```

You may retrieve your mail using the command **`mail`** or **`mailx`** or any other program.

```
PESCHKE @ RA28:/u/peschke>mail

Message  1:
From PESCHKE@MVS39A.itso.ral.ibm.com Tue Feb 16 14:40:04 1999
Received: from MVS39A.itso.ral.ibm.com (mvs39a.itso.ral.ibm.com  9.24.104
        by MVS28A.itso.ral.ibm.com (8.8.7/8.8.7) with ESMTP id OAA5033168
        for <PESCHKE@MVS28A.itso.ral.ibm.com>; Tue, 16 Feb 1999 14:40:03
Received: (from OMVSKERN@localhost)
        by MVS39A.itso.ral.ibm.com (8.8.7/8.8.7) id OAA50331674
        for rolan28a; Tue, 16 Feb 1999 14:39:57 -0500
Date: Tue, 16 Feb 1999 14:39:57 -0500
From: PESCHKE <PESCHKE@MVS39A.itso.ral.ibm.com>
Message-Id: <199902161939.OAA50331674@MVS39A.itso.ral.ibm.com>

This message was sent from system RA39 to RA28 using the sendmail
client to a sendmail daemon, started with
                /usr/sbin/sendmail -bd
The messages was sent with
/usr/sbin/sendmail rolan28a </u/peschke/mailto28
```

Figure 5-17   Output from the mail command

In this sample we didn't use an MUA on the sender's site. However, MUAs, like Netscape Communicator, will provide better support to compose a mail message.

5.3.2 sendmail's server/daemon role

In our case, we invoked the sendmail program in system RA39 to execute the transmission via SMTP to another sendmail program in system RA28 which was listening to incoming SMTP connections. The sending of sendmail in RA39 was not started as a daemon. It was invoked only one time for the sending process and exited after the execution. It acted as a client to establish the connection with the remote sendmail program in RA28 which was started as a daemon running in the background. We used the following command to start the sendmail daemon in RA28 from the OMVS shell.

```
PESCHKE @ RA28:/u/peschke>/usr/sbin/sendmail -bd
PESCHKE @ RA28:/u/peschke>ps -ef
```

A quick check with `ps -ef` will show if the sendmail daemon has been started.

```
 UID        PID       PPID  C    STIME TTY       TIME CMD
OMVSKERN          1         0  -   Feb 15          0:00 BPXPINPR
OMVSKERN  201326594         1  - 11:20:13          0:00 ICADCT
 PESCHKE  150994947         1  - 09:04:55          0:01 OMVS
OMVSKERN  335544324         1  - 18:11:29         12:17 /usr/lpp/tcpip/
proute -t1
OMVSKERN  301989893         1  - 11:20:24          0:00 ICADDCT
OMVSKERN  251658246         1  - 17:13:26          4:16 EZBTMCTL
OMVSKERN  134217735  134217747  - 15:26:18 ttyp0000  0:00 sh -i
OMVSKERN   16777224         1  -   Feb 15          4:45 EZACFALG
OMVSKERN   33554441         1  -   Feb 15          4:45 EZASASUB
OMVSKERN  184549386         1  - 09:21:22          0:00 /usr/sbin/sendmail -bd
OMVSKERN         11         1  -   Feb 15          4:45 EZBTCPIP
 CAMILUC   16777228         1  -   Feb 15          0:29 OMVS
OMVSKERN         13         1  -   Feb 15          0:00 OPORTMAP
```

You will get more information in case of error situations if you look at the syslog or syslog daemon. For example:

```
sendmail 620756996 : EZZ751 I: sendmail starting
sendmail 620756996 : gethostbyaddr(192.168.221.28) failed: 0
sendmail 620756996 : gethostbyaddr(192.168.233.28) failed: 0
sendmail 620756996 : gethostbyaddr(192.168.233.28) failed: 0
sendmail 620756996 : gethostbyaddr(192.168.233.29) failed: 0
sendmail 620756996 : gethostbyaddr(192.168.233.29) failed: 0
sendmail 620756996 : EZZ7511I daemon invoked without full pathname; ill -1 won't work
```

The way we started the sendmail daemon is, for test purposes, a quick solution. If, however, you want to have an automatic start of the sendmail daemon, perhaps together with the TCP/IP start, we recommend:

- Creating a procedure for your PROCLIB
- Adding the procedure's name to the AUTOLOG/ENDAUTOLOG statement in the TCP/IP profile

USER.PROCLIB samples

You have two choices to invoke the sendmail daemon:

1. Via the symbolic link smtpd
2. Via the symbolic link sendmail and the -bd option

Both invocations lead to the same program:

```
../../usr/lpp/tcpip/sbin/sendmail
```

Please compare the following symbolic link attributes:

```
Symbolic Link Attributes

Pathname : /usr/sbin/smtpd

External link  . . : 0
File size  . . . . : 33
File owner . . . . : OMVSKERN(0)
Group owner  . . . : (2736)
Last modified  . . : 11/13/1998 16:24 GMT
Last changed . . . : 11/13/1998 16:24 GMT
Last accessed  . . : 11/13/1998 16:24 GMT
Created  . . . . . : 11/13/1998 16:24 GMT
Link count . . . . : 1
Device number  . . : 1
Inode number . . . : 914B6
Symbolic link contents:

   ../../usr/lpp/tcpip/sbin/sendmail
```

Figure 5-18 Symbolic link attributes of smtpd

```
Symbolic Link Attributes

Pathname : /usr/sbin/sendmail

External link  . . : 0
File size  . . . . : 33
File owner . . . . : OMVSKERN(0)
Group owner  . . . : (2736)
Last modified  . . : 11/13/1998 16:24 GMT
Last changed . . . : 11/13/1998 16:24 GMT
Last accessed  . . : 11/13/1998 16:24 GMT
Created  . . . . . : 11/13/1998 16:24 GMT
Link count . . . . : 1
Device number  . . : 1
Inode number . . . : 914B7
Symbolic link contents:

   ../../usr/lpp/tcpip/sbin/sendmail
```

Figure 5-19 Symbolic link attributes of sendmail

Start procedure with smtpd

When sendmail is invoked with **`smtp`** it will run as a daemon.

```
 BROWSE    USER.PROCLIB(T28ASM) - 01.00               Line 0000
********************************** Top of Data *****************
//T28ASM    PROC MODULE='BPXBATCH'
//SMTPD    EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//     PARM='PGM /usr/sbin/smtpd -v -d'
//STDOUT   DD PATH='/tmp/smtpd.stdout',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR   DD PATH='/tmp/smtpd.stderr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDENV   DD DSN=TCP.TCPPARMS(SM28AENV),DISP=SHR
//SYSERR   DD PATH='/tmp/smtpd.syserr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//CEEDUMP  DD DUMMY
******************************** Bottom of Data ***************
```

Figure 5-20 Start procedure with smtpd

The sendmail daemon was invoked using the smtpd symbolic link:

```
PARM='PGM /usr/sbin/smtpd -v -d'
```

Additional command line switches:

-v	Run in verbose mode
-d	Run in debugging mode using debugging switches where sendmail uses its default category and level which are from category 0 to category 99 level 1 (0-99.1)

The debug expression that provides the maximum debugging output is:

```
-d0-99.127
```

The debugging mode is described in “Debugging mode” on page 67.

Start procedure with sendmail

```
 BROWSE    USER.PROCLIB(T28ASM) - 01.05               Line 00
********************************* Top of Data ***************
//T28ASM    PROC MODULE='BPXBATCH'
//SMTPD    EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//    PARM='PGM /usr/sbin/sendmail -bd'
//STDOUT   DD PATH='/tmp/smtpd.stdout',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR   DD PATH='/tmp/smtpd.stderr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDENV   DD DSN=TCP.TCPPARMS(SM28AENV),DISP=SHR
//SYSERR   DD PATH='/tmp/smtpd.syserr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//CEEDUMP  DD DUMMY
******************************** Bottom of Data *************
```

Figure 5-21 Start procedure with sendmail

The sendmail daemon was invoked using the sendmail symbolic link:

```
PARM='PGM /usr/sbin/sendmail -bd'
```

and the command line switch `-bd` to run sendmail as daemon in listening mode and in the background.

5.3.3 Some considerations about sendmail client and server roles

What happens when we try to send mail but the sendmail daemon is not started? For example, a user tries to send mail from system RA28 to RA39 under the previously introduced configuration. Will it work? Of course not. The following message on the console is the proof:

```
PESCHKE @ RA28:/u/peschke>/usr/sbin/sendmail -v rolan39a </u/peschke/mailo
rolan39a... aliased to PESCHKE@MVS39A
PESCHKE@MVS39A... Connecting to mvs39a.itso.ral.ibm.com. via esmtp...
PESCHKE@MVS39A... Deferred: Connection refused by mvs39a.itso.ral.ibm.com.
```

Why was the message not sent? RA28 started processing the message with aliasing the name. But when sendmail, acting in this case now as a client, tried to set up the TCP connection to the sendmail in RA39 which should act as a server, it found no server listening to receive connection requests. Therefore, the message was queued for future delivery.

So be aware of what state sendmail should run in. The client program always does the setup for the TCP connection while the server or daemon is the program listening for incoming connection requests.

5.3.4 sendmail's tasks to get mail transmitted

As you saw in our first sample:

1. The sendmail client searches the aliases database for the alias of rolan28a and converted it into the address PESCHKE@MVS28A.
2. The domain itso.ral.ibm.com is added to the address.
3. The resolver program now searches for the name server with its defined IP address in the nsinteraddr statement in the TCP.TCPPARMS data set for the member TDATA or in the /etc/resolv.conf HFS file depending on the current valid OpenEdition environment variable. You will find the value of this variable in the PROCLIB member of the TCP/IP start procedure.

   Example for pointing to the environment variable:

   ```
   //T03ATCP  PROC PRM='CTRACE(CTIEZB01)'
   //TCPIP    EXEC PGM=EZBTCPIP,REGION=7500K,TIME=1440,
   //  PARM=(&PRM,
   //  'ENVAR("RESOLVER_CONFIG=//''TCP.TCPPARMS(TDATA03A)''")',
   //  '/')
   ```

   Note: Be aware that your data set or file names may be different.

4. The resolver program asks for the IP addresses of the sender and the receiver. It also requests for IP addresses of alternate mailboxes on other machines if the MX records are defined for that user in the name server's forward file.
5. If the name server is able to provide the requested IP addresses - in our case for host MVS28A in the domain itso.ral.ibm.com and for MVS39A within the same domain - it returns this information to the resolver.
6. The resolver hands the IP addresses to the sendmail program.
7. Now the sendmail program has all the information to forward to the TCP layer to set up the TCP header and to forward to the IP layer to set up the IP header which ends up trying to set up the TCP connection.
8. If the connection setup was successful the two sendmail programs exchange a certain command flow, like 220 with the receiver's domain name followed by an EHLO propagating that the sender is able to understand extended SMTP (ESTMP), and a 250 from MVS28A with a HELO to MVS39A which says that it is pleased to meet you.

   The following 250 messages tell the MVS39A what the receiver is able to support. For example:

   `250-EXPN`, which indicates support of extended mailing lists.

   `250-8BITMIME`, which indicates MIME transport.

   `250-DSN`, which indicates delivery status notification.

   The mail header information follow with the data until the closing of the connection.

9. Now the remote sendmail program on system MVS28A starts the local mailer program which appends the message to the recipient's mail spool file /usr/mail/PESCHKE.
10. The recipient is informed after logon to his machine that mail is in his mail file. The recipient will issue a command such as **`mail`** or **`mailx`** to retrieve the message.

5.3.5 sendmail extended modes

The `sendmail` command-line switch allows sendmail to run three groups of mode:

`sendmail -v`	Run in verbose mode
`sendmail -d`	Run in debugging mode
`sendmail -b`	Set operation mode

Sample of verbose mode

As you can see, it shows the process and the command flow between client and daemon.

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/sendmail -v rolan28a </u/pesch e/mailto28
rolan28a... aliased to PESCHKE@MVS28A
PESCHKE@MVS28A... Connecting to mvs28a.itso.ral.ibm.com. via esmtp...
220 MVS28A.itso.ral.ibm.com ESMTP Sendmail 8.8.7/8.8.7; Tue, 16 Feb 1999 15:18:13 -0500
>>> EHLO MVS39A.itso.ral.ibm.com
250-MVS28A.itso.ral.ibm.com Hello mvs39a.itso.ral.ibm.com 9.24.104.149 , pleased
to meet you
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<PESCHKE@MVS39A.itso.ral.ibm.com> SIZE=226
250 <PESCHKE@MVS39A.itso.ral.ibm.com>... Sender ok
>>> RCPT To:<PESCHKE@MVS28A.itso.ral.ibm.com>
250 <PESCHKE@MVS28A.itso.ral.ibm.com>... Recipient ok
>>> DATA
35  Enter mail, end with "." on a line by itself
>>> .
250 PAA33554472 Message accepted for delivery
PESCHKE@MVS28A... Sent (PAA33554472 Message accepted for delivery)
Closing connection to mvs28a.itso.ral.ibm.com.
>>> QUIT
221 MVS28A.itso.ral.ibm.com closing connection
PESCHKE @ RA39:/u/peschke>
```

Figure 5-22 Sendmail client output in verbose mode

Debugging mode

There are many debug flags that allow you very good error analysis. Each debug flag has a number, also called a category, and a level. A higher level means to print out more detailed information. Debug flags are set using the -d option.

The syntax is:

`debug-flag:`	-d debug-list
`debug-list:`	debug-option {,debug-option}

`debug-option`	debug-range {.debug-level}
`debug-range`	integer | integer-integer
`debug-level`	integer

For example:

`-d12`	Set flag 12 to level 1
`-d12.3`	Set flag 12 to level 3
`-d3-17`	Set flags 3 through 17 to level 1
`-d3-17.4`	Set flags 3 through 17 to level 4

Do not run the debugging mode without defining the category; for example:

```
/usr/sbin/sendmail -d rolan28a </u/peschke/mailto28
```

You will get a huge output of information because this `-d` command line switch defaults to -d0-99.1. Therefore, we recommend selecting from the more than 180 debugging options (the appropriate category and level for the `-d` debugging command-line switch). For example, to show sender information, use:

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/sendmail -d1.1 rolan28a </u/peschke/mailto28
From person = "PESCHKE"
PESCHKE @ RA39:/u/peschke>
```

or if you want to know which host names have been tried, use:

```
PESCHKE @ RA39:/u/peschke>/usr/sbin/sendmail -d8.5 rolan28a </u/peschke/mailto28
dns_getcanonname(MVS28A, trymx=1)
dns_getcanonname: trying MVS28A.itso.ral.ibm.com (ANY)
dns_getcanonname: MVS28A.itso.ral.ibm.com
PESCHKE @ RA39:/u/peschke>
```

Also, you may combine categories and determine your desired level. Regard the following sample where we also added the verbose mode in order to get process information about the proper execution of the sendmail task.

```
PESCHKE @ RA03:/u/peschke>/usr/sbin/sendmail -v -d0-1.5 rolan28a </u/peschke/mail
Version 8.8.7   1s
 Compiled with: LOG MIME7TO8 MIME8TO7 NAMED_BIND NDBM NETINET NETUNIX
                QUEUE SCANF SMTP XDEBUG
canonical name: MVS03A.itso.ral.ibm.com   3
        a.k.a.: MVS03A
        a.k.a.: MVS03A.itso
        a.k.a.: MVS03A.itso.ral
 UUCP nodename: RA03
        a.k.a.: 9.24.104.113
        a.k.a.: 192.168.250.3
        a.k.a.: 192.168.233.3
        a.k.a.: 192.168.20.3
        a.k.a.: 192.168.210.1
        a.k.a.: 192.168.213.1
        a.k.a.: 192.168.221.3
        a.k.a.: 192.168.235.3
        a.k.a.: 192.168.251.4
        a.k.a.: mvs03c.itso.ral.ibm.com
        a.k.a.: 9.24.104.33
        a.k.a.: 192.168.233.4

============ SYSTEM IDENTITY (after readcf) ====
      (short domain name) $w = MVS03A
  (canonical domain name) $j = MVS03A.itso.ral.ibm.com
         (subdomain name) $m = itso.ral.ibm.com
              (node name) $k = RA03       1e
========================================================
```
Figure 5-23 sendmail client output in debugging mode (part 1 of 2)

Screen output is continued on the following page.

```
rolan28a... aliased to PESCHKE@MVS28A  2
From person = "PESCHKE"                4
main: QDONTSEND 152c2df8=PESCHKE:      5s
        mailer 3 (local), host `'
        user `PESCHKE', ruser `<null>'
        next=0, alias 0, uid 4029, gid 1
        flags=6005<QDONTSEND,QGOODUID,QPINGONFAILURE,QPINGONDELAY>
        owner=(none), home="/u/peschke", fullname="(none)"
        orcpt="(none)", statmta=(none), status=(none)
        rstatus="(none)"
        specificity=0, statdate=Wed Dec 31 19:00:00 1969             5e
PESCHKE@MVS28A... Connecting to mvs28a.itso.ral.ibm.com. via esmtp... 2s
220 MVS28A.itso.ral.ibm.com ESMTP Sendmail 8.8.7/8.8.7; Thu, 25 Feb 1999 21:12:1
3 GMT
>>> EHLO MVS03A.itso.ral.ibm.com
250-MVS28A.itso.ral.ibm.com Hello mvs03a.itso.ral.ibm.com  9.24.104.113 ,please
d to meet you
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<PESCHKE@MVS03A.itso.ral.ibm.com> SIZE=80
250 <PESCHKE@MVS03A.itso.ral.ibm.com>... Sender ok
>>> RCPT To:<PESCHKE@MVS28A.itso.ral.ibm.com>
250 <PESCHKE@MVS28A.itso.ral.ibm.com>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 VAA352321560 Message accepted for delivery
PESCHKE@MVS28A... Sent (VAA352321560 Message accepted for delivery)
Closing connection to mvs28a.itso.ral.ibm.com.
>>> QUIT
221 MVS28A.itso.ral.ibm.com closing connection    2e
```

Figure 5-24 sendmail client output in debugging mode (part 2 of 2)

The following is an explanation of the provided debugging and verbose mode information shown above:

Explanation of the following remarks:

- Remark valid for one line only 3.
- Remarks valid for a group of lines.

  1s = start of the group

  1e = end of the group

Description of the remarks:

1s/1e	Provided through -d0.1 = print version information
2s/2e	Provided through -v = verbose switch
3	Provided through -d0.4 = our name and aliases

4	Provided through -d1.1 = show sender information
5s/5e	Provided through -d1.5 = dump the sender address

Become a sendmail -b mode

You will notice that the following modes have been used in previous sections:

-bd	Run sendmail as daemon
-bi	Run initialize alias database
-bp	Print the message queue of deferred mail
-bt	Rule testing mode

5.3.6 Logging

sendmail is able to provide the system administrator with a variety of information concerning mail delivery and forwarding. This is done through the syslog facility. Based on information in the /etc/syslog.conf file, a warning is issued to a device /dev/console, appended to a file, forwarded to another host, or displayed on a logged-in user's screen. We used for our tests the facility to append a file and forward the information to another host. This is described in detail in Chapter 14, “syslogd” on page 435.

Log level

Syslogd uses two items of information to determine how to handle messages:

1. Facility (syslogd is able to handle many categories, but sendmail uses only one, which is called mail)
2. Level (describes the degree of severity of warnings)

Just before sendmail issues a warning, it looks at the logging level defined by its LogLevel in the /etc/sendmail.cf configuration file. We used the log level pre-built through the m4 process, which shows the entry O LogLevel=9.

You may define a log level from 0 to 98. Following is an overview of the different values:

0	Minimal logging
1	Serious system failures and security problems
2	Communications failures (for example, lost connection)
3	Malformed addresses
4	Malformed qf file names and minor errors
5	A record of each message received
6	SMTP VRFY attempts and messages returned to the original sender
7	Delivery failures, excluding mail deferred because of a lack of resources
8	Successful delivery
9	Mail deferred because of lack of resources
10	Each key is looked up in a database
11	All nis errors logged
12	SMTP connections
13	Bad user

14	Connection refusals
15	All incoming and outgoing SMTP commands
16-98	Debugging information

If the severity of the warning is greater than the logging level, nothing is directed to the output. If the severity of the warning to issue is less or equal to the logging level (lower is more serious), sendmail issues a message like this:

```
   syslog(pri, message)
```

Where `pri` means syslog logging priority, and `msg` is the text of the warning message.

Note: You have to distinguish between log level and syslogd priority.

The following list shows the relationship between log level and syslog priority:

Level	***Priority***
1	LOG_CRIT and LOG_ALERT
2-8	LOG_NOTICE
9-10	LOG_INFO
11+	LOG_DEBUG

Tuning syslog.conf

The syslog.conf file is composed of lines of text that each have the form:

```
   facility.level       target
```

The facility is the type of program that may produce a message. The facility called mail is the one that sendmail uses.

The level indicates the severity at or above which messages should be handled. These levels correspond to the LogLevel option levels shown in the previous table. A complete list of syslog.conf levels used by sendmail follows:

Level	***Severity (highest to lowest)***
alert	Conditions requiring immediate correction
crit	Critical conditions for which action may be deferred
err0	Other errors
warning	Warning messages
notice	Non-errors that may require special handling
info	Statistical and informational messages
debug	Messages used only in a debugging program

The target is one of the four possibilities shown below:

Target	***Description***
@host	Forward message to the named host
/file	Append message to the named file
user,user	Write to user's screens; if logged in
*****	Write to all logged-in user's screens

You have to define in the syslog.conf configuration file

1. What facility has to be logged
2. What level should be logged
3. Where the logged information have to be directed to view

We used the following example in our configuration file:

```
 BROWSE -- /etc/syslog.conf.peschke ----------------- Line 00000000
 Command ===>                                                  Scrol
********************************* Top of Data *********************
# (C) COPYRIGHT International Business Machines Corp. 1995
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# /etc/syslog.conf - control output of syslogd
#
# A # sign denotes a comment.
# A blank line is ignored.
#
# Each line of this file must contain at least two parts:
#
# A message source list and a destination
#
#  The message source list consists of two parts:
#
#   Part 1.  The facility argument
#            The acceptable entries for facility are:
#            kern
#            user
#            mail
#            news
#            uucp
#            daemon
#            auth/authpriv
        .............. continued .......................
#   Part 2.  A priority to determine the levels the line applies to
#            The acceptable entries for priority are:
#            emerg/panic
#            alert
#            crit
#            err(or)
#            warn(ing)
#            notice
#            info
#            debug
#            none
#            *                 (All of the above)
#
        .............. continued .......................
mail.info               /u/peschke/log
mail.notice             peschke
mail.debug              /u/peschke/deb
******************************* Bottom of Data *******
```

Figure 5-25 Syslogd configuration for sendmail

We selected the mail facility.

In the first line we selected the level **info** and as the target the file /u/peschke/log.

In the second line we selected the level **notice** and as the target we chose the user **peschke** to be informed.

In the third line we selected the level **debug** and as the target we chose the file **/u/peschke/deb**.

The configuration file is defined in the syslogd start procedure. See the following sample:

```
 BROWSE -- /etc/syslogd.start.peschke --------------- Li
 Command ===>
********************************* Top of Data **********
# Start the syslog Daemon
export _BPX_JOBNAME='SYSLOGD'
/usr/sbin/syslogd -f /etc/syslog.conf.peschke &
echo -- /etc/syslogd.start script executed
******************************** Bottom of Data ********
```

Figure 5-26 Syslogd start with special configuration file for sendmail

Syslog output

We used the following **sendmail** command to produce the contents of the debugging file:

```
/usr/sbin/sendmail  -v -d0-1.5 rolan28a </u/peschke/mailo
```

```
 BROWSE -- /u/peschke/deb --------------------------- Line 00000000 Col 0
 Command ===>                                                  Scroll ===
********************************* Top of Data ***************************
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.221.28)
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.233.28)
Mar  1 16:47:51 MVS28A last message repeated 2 times
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.233.29)
Mar  1 16:47:51 MVS28A last message repeated 2 times
Mar  1 16:47:57 MVS28A sendmail 318767108 : QAA318767108: from=PESCHKE, s
Mar  1 16:47:57 MVS28A sendmail 318767108 : QAA318767108: to=PESCHKE@MVS2
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.221.28)
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.233.28)
Mar  1 16:48:11 MVS28A last message repeated 2 times
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.233.29)
Mar  1 16:48:11 MVS28A last message repeated 2 times
Mar  1 16:48:16 MVS28A sendmail 402653188 : QAA402653188: from=PESCHKE, s
Mar  1 16:48:16 MVS28A sendmail 402653188 : QAA402653188: to=PESCHKE@MVS2
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.221.28)
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.233.28)
Mar  1 16:48:38 MVS28A last message repeated 2 times
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.233.29)
Mar  1 16:48:38 MVS28A last message repeated 2 times
Mar  1 16:48:44 MVS28A sendmail 486539268 : QAA486539268: from=PESCHKE, s
Mar  1 16:48:44 MVS28A sendmail 486539268 : QAA486539268: to=PESCHKE@MVS2
******************************** Bottom of Data *************************
```

Figure 5-27 Truncated debugging recording sample

Since this file has a record length of 194 bytes, you don't see the truncated end. We therefore have prepared the file with its entire contents in multiple lines per entry.

```
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.221.28)
failed: 0
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.233.28)
failed: 0
Mar  1 16:47:51 MVS28A last message repeated 2 times
Mar  1 16:47:51 MVS28A sendmail 318767108 : gethostbyaddr(192.168.233.29)
failed: 0
Mar  1 16:47:51 MVS28A last message repeated 2 times
Mar  1 16:47:57 MVS28A sendmail 318767108 : QAA318767108: from=PESCHKE,
size=95, class=0, pri=30095, nrcpts=1, msgid=<199903012147.QAA318767108@
MVS28A.itso.ral.ibm.com>, relay=PESCHKE@localhost
Mar  1 16:47:57 MVS28A sendmail 318767108 : QAA318767108: to=PESCHKE@MVS
28A, delay=00:00:06, mailer=local, stat=queued
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.221.28)
failed: 0
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.233.28)
failed: 0
Mar  1 16:48:11 MVS28A last message repeated 2 times
Mar  1 16:48:11 MVS28A sendmail 402653188 : gethostbyaddr(192.168.233.29)
failed: 0
Mar  1 16:48:11 MVS28A last message repeated 2 times
Mar  1 16:48:16 MVS28A sendmail 402653188 : QAA402653188: from=PESCHKE,
size=95, class=0, pri=30095, nrcpts=1, msgid=<199903012148.QAA402653188@
MVS28A.itso.ral.ibm.com>, relay=PESCHKE@localhost
Mar  1 16:48:16 MVS28A sendmail 402653188 : QAA402653188:
to=PESCHKE@MVS28A, delay=00:00:05, mailer=local, stat=queued
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.221.28)
failed: 0
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.233.28)
failed: 0
Mar  1 16:48:38 MVS28A last message repeated 2 times
Mar  1 16:48:38 MVS28A sendmail 486539268 : gethostbyaddr(192.168.233.29)
failed: 0
Mar  1 16:48:38 MVS28A last message repeated 2 times
Mar  1 16:48:44 MVS28A sendmail 486539268 : QAA486539268: from=PESCHKE,
size=95, class=0, pri=30095, nrcpts=1, msgid=<199903012148.QAA486539268@
MVS28A.itso.ral.ibm.com>, relay=PESCHKE@localhost
Mar  1 16:48:44 MVS28A sendmail 486539268 : QAA486539268:
to=PESCHKE@MVS28A, delay=00:00:06, mailer=local, stat=queued
**************************** Bottom of Data ****************************
```

Figure 5-28 Edited debugging recording sample

5.4 The popper server

The Qualcomm popper is a mail-delivery agent that uses Post Office Protocol Version 3 (POP3).

5.4.1 Introduction

We discussed the receiving sendmail daemon that appends receiving mail to the user's spool file. For example:

```
/usr/mail/PESCHKE
```

Note: The directory and file names are case-sensitive.

This kind of technology allows access to the mail file system for local users only. But today, many users with laptops need access to their mailing system from remote sites. They also use such mail programs as Netscape Communicator or Microsoft's Outlook Express instead of the simple mail program we used in our tests. This means that another technology has to be used to access the mail spool file from the remote site.

The z/OS popper will allow remote users to access the mail spool file. The z/OS popper is a server for remote users running POP3 (Post Office Protocol). This also means for the user's MUA that POP3 is required. Netscape Navigator/Communicator does support POP3.

The popper acts as a mail delivery agent (MDA) which retrieves mail on request from the MUA from its mail spool file.

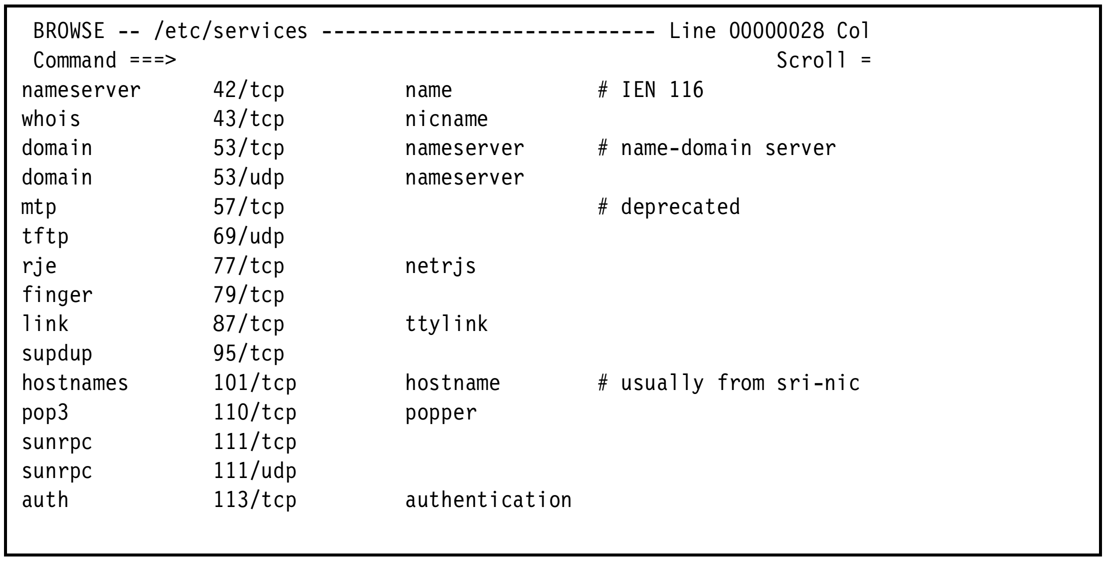
5.4.2 z/OS popper implementation

The popper implementation can be divided into four steps:

1. Update /etc/services file.
2. Update /etc/inetd.conf file.
3. Create the directory for the temporary maildrop file.
4. Start inetd.

Update /etc/services file

You need a port for the POP3 defined in /etc/services. We used for our tests the well-known port 110.



```
 BROWSE -- /etc/services ---------------------------- Line 00000028 Col
 Command ===>                                                  Scroll =
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
tftp            69/udp
rje             77/tcp          netrjs
finger          79/tcp
link            87/tcp          ttylink
supdup          95/tcp
hostnames       101/tcp         hostname        # usually from sri-nic
pop3            110/tcp         popper
sunrpc          111/tcp
sunrpc          111/udp
auth            113/tcp         authentication
```

Figure 5-29 /etc/services updated for popper

Update /etc/inetd.conf file

Since the popper will be invoked by INETD you need to add the following information to your /etc/inetd.conf file:

```
pop3     stream tcp nowait bpxroot  /usr/sbin/popper popper -d
```

The description of this entry is:

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
-d means running popper in debugging mode.
```

Create the directory for the temporary maildrop file

When the popper is invoked through an MUA client request, for example, GET NEW MESSAGES, popper starts a read of the user's mail file system (/usr/mail/username) where sendmail has stored the data and puts this data (if there are messages) to a temporary maildrop file /usr/mail/popper/.username.pop. The contents of this file are transmitted to the remote client using the existing POP3 TCP connection.

Since the directory doesn't exist, you have to create it, as follows:

```
   /usr/mail/popper/
```

The popper uses this directory to create and fill the maildrop file.

If you do not specify this directory, you will get an error message:

```
EZZ7605I: Unable to open temporary maildrop '/usr/mail/popper/.PESCHKE.pop': m
PESCHKE@ 9.24.106.64 : -ERR System error, can't open temporary file, do you own
Sending line "-ERR System error, can't open temporary file, do you own it?"
+OK Pop server at MVS28A.itso.ral.ibm.com signing off.
Sending line "+OK Pop server at MVS28A.itso.ral.ibm.com signing off."
(v2.3) Ending request from "PESCHKE" at (9.24.106.64) 9.24.106.64
```

Start INETD

After updating the inetd.conf file, INETD has to be started:

```
   _BPX_JOBNAME='INETD' /usr/sbin/inetd/ /etc/inetd.conf &
```

You will get the following message:

```
PESCHKE @ RA28:/u/peschke> _BPX_JOBNAME='INETD' /usr/sbin/inetd
/etc/inetd.conf &
 1      452984837
PESCHKE @ RA28:/u/peschke>
 1  + Done  _BPX_JOBNAME='INETD' /usr/sbin/inetd /etc/inetd.conf &
```

A look into the system shows that INETD is running. You also identify the name of the inetd.conf file.

```
OMVSKERN 318767126      1  -   Feb 15          8:58 EZBTCPIP
PESCHKE  402653208      1  - 09:48:51          0:03 OMVS
OMVSKERN   50331673     1  -   Feb 15 /usr/lpp/tcpip/sbin/omproute -t1
OMVSKERN  369098778  - 14:09:16  0:00 /usr/sbin/inetd /etc/inetd.conf
OMVSKERN 201326620      1  -   Feb 15          8:58 EZASASUB
```

You may also check with onetstat -a if INETD is running. onetstat will show you the ports used and the state LISTEN. Port 110 is used for the popper.

```
PESCHKE @ RA28:/u/peschke>onetstat -a
MVS TCP/IP onetstat CS V2R7       TCPIP Name: T28ATCP          14:22:41
User Id  Conn  Local Socket           Foreign Socket         State
-------  ----  ------------           --------------         -----
ICAPCFGS 079EE 0.0.0.0..1014          0.0.0.0..0             Listen
PORTMAP1 046D6 0.0.0.0..111           0.0.0.0..0             Listen
T28AFTP1 046F5 0.0.0.0..21            0.0.0.0..0             Listen
T28ANFSS 09A22 0.0.0.0..4007          0.0.0.0..0             Listen
T28ANFSS 09A19 0.0.0.0..4004          0.0.0.0..0             Listen
T28ANFSS 09A25 0.0.0.0..2049          0.0.0.0..0             Listen
T28ANFSS 09A1C 0.0.0.0..4005          0.0.0.0..0             Listen
T28ANFSS 09A1F 0.0.0.0..4006          0.0.0.0..0             Listen
T28ATCP  08C94 192.168.230.1..1074    192.168.230.1..1075    Establsh
T28ATCP  08C92 0.0.0.0..1074          0.0.0.0..0             Listen
T28ATCP  08C93 192.168.230.1..1075    192.168.230.1..1074    Establsh
T28ATCP  046DB 0.0.0.0..23            0.0.0.0..0             Listen
T28INETD 09C85 9.24.104.42..110       9.24.104.111..3862     TimeWait
T28INETD 04704 0.0.0.0..2023          0.0.0.0..0             Listen
T28INETD 04703 0.0.0.0..110           0.0.0.0..0             Listen
T28INETD 04702 0.0.0.0..109           0.0.0.0..0             Listen
VANDEKE9 05E42 0.0.0.0..25            0.0.0.0..0             Listen
ICAPSLOG 079EC 0.0.0.0..514           *..*                   UDP
PORTMAP1 046D5 0.0.0.0..111           *..*                   UDP
T28ANFSS 09A0C 0.0.0.0..4006          *..*                   UDP
T28ANFSS 09A11 0.0.0.0..4007          *..*                   UDP
T28ANFSS 09A14 0.0.0.0..2049          *..*                   UDP
T28ANFSS 09A07 0.0.0.0..4005          *..*                   UDP
T28ANFSS 09A02 0.0.0.0..4004          *..*                   UDP
PESCHKE @ RA28:/u/peschke>
```

Figure 5-30 onetstat -a output

Another way to start INETD is through a BPXBATCH job.

```
 BROWSE    USER.PROCLIB(T28INETD) - 01.04            Line 0000
********************************* Top of Data *****************
//T28INETD EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//    PARM='PGM /usr/sbin/inetd'
//STDENV   DD DSN=TCP.TCPPARMS(IN28AENV),DISP=SHR
******************************** Bottom of Data ***************
```

All preparations are done on the server side. The popper can be invoked now.

5.4.3 POP3 definitions for the MUA

In our sample, we used Netscape Communicator as the POP3 client to create mail and retrieve mail from the MVS28A popper Server. The Netscape Communicator was running under Windows NT 4.0 on a PC which was connected via a token-ring to the system z/OS MVS28A.

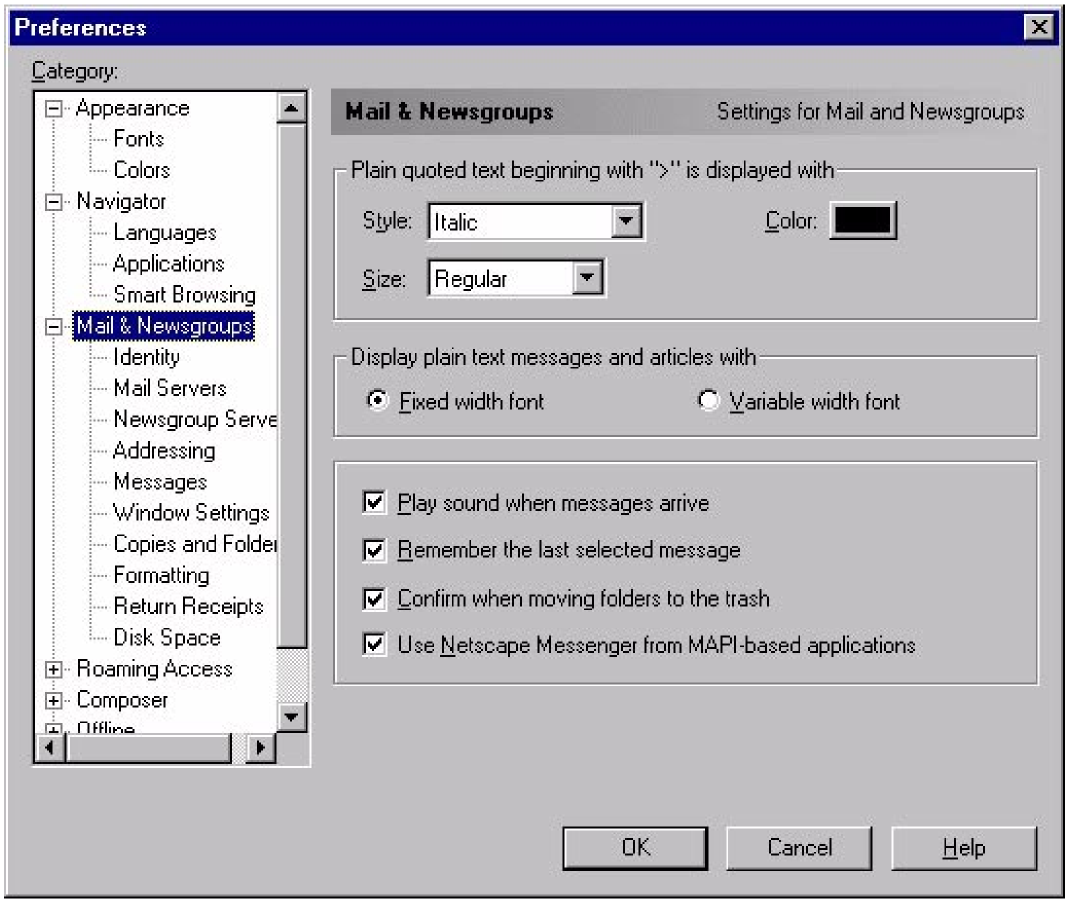
The MUA needs definitions:

1. To identify yourself to the mail world
2. To select the required mail servers and the server type
   a. For incoming mail
   b. For outgoing mail

All definitions are implemented under the Netscape pop-up list Mail & Newsgroups.

The way to find the Mail & Newsgroups pop-up list

1. Select **Edit** on the Netscape invitation panel.
2. Select **Preferences** on Edit's pop-up list.
3. The Preferences will now be shown.
4. Select **Mails & Newgroups**.
5. The following two lines are important to you:
   a. Identity
   b. Mail Servers



Mail & Newsgroups Settings for Mail and Newsgroups

Plain quoted text beginning with ">" is displayed with

Style: Italic Color: ■

Size: Regular

Display plain text messages and articles with

[● Fixed width font] ○ Variable width font

[✓] Play sound when messages arrive

[✓] Remember the last selected message

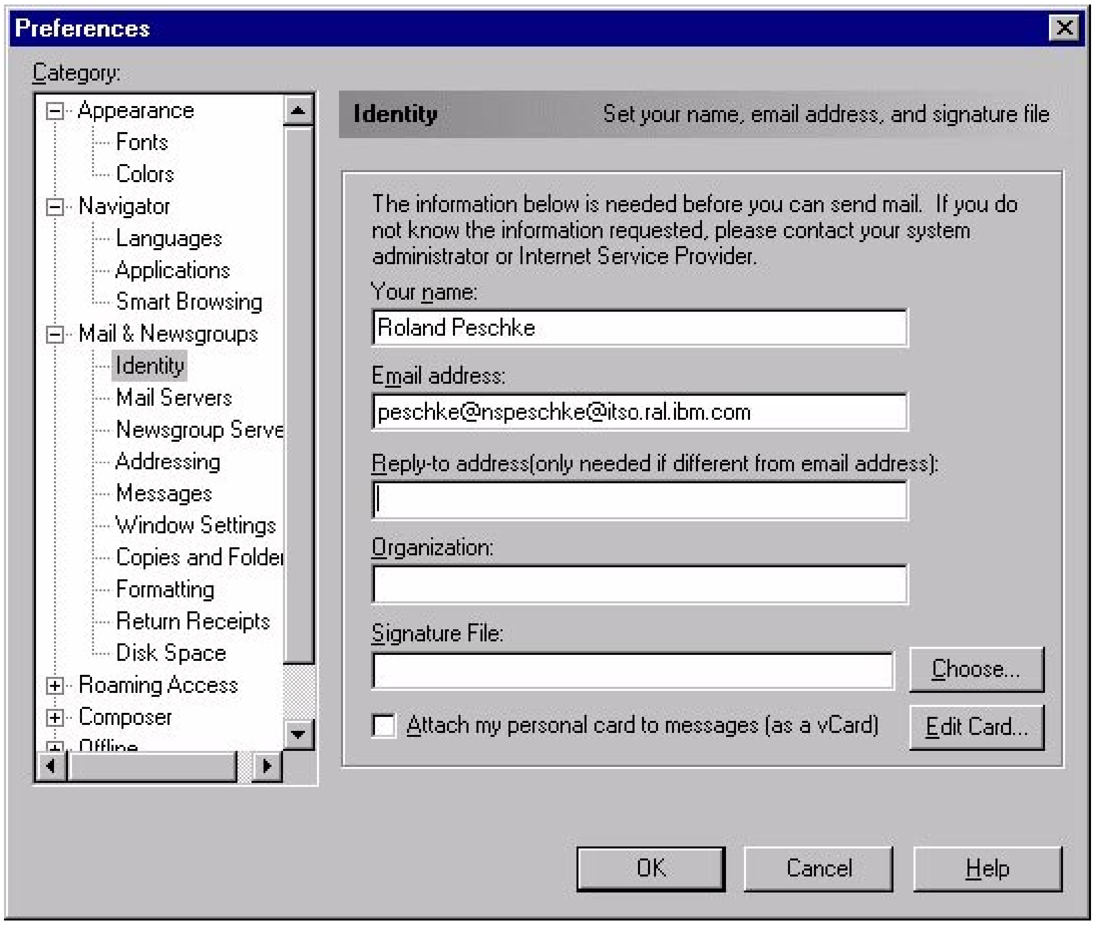
[✓] Confirm when moving folders to the trash

[✓] Use Netscape Messenger from MAPI-based applications

Figure 5-31 Preferences: Mail & Newsgroups

Identity

You define here your identification to the mail world.



Identity Set your name, email address, and signature file

The information below is needed before you can send mail. If you do not know the information requested, please contact your system administrator or Internet Service Provider.

Your name: Roland Peschke

Email address: peschke@nspeschke@itso.ral.ibm.com

Reply-to address(only needed if different from email address):

Organization:

Signature File:

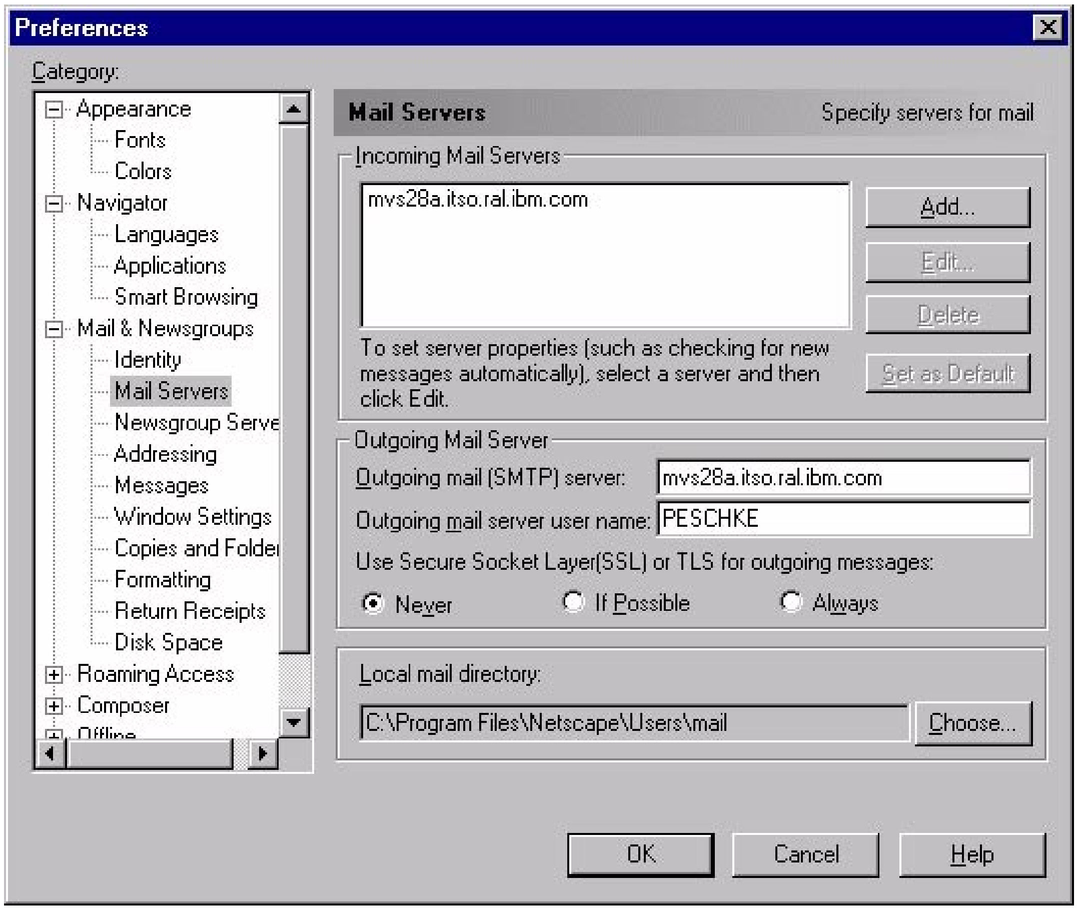
☐ Attach my personal card to messages (as a vCard)

Figure 5-32 Preferences: Identity

The minimum requirement is the definition of your name and your e-mail address so you can be reached from the network. In our case, peschke is the user name and nspeschke is the host name of the PC which is located in the domain itso.ral.ibm.com.

Mail servers

You have to specify the Incoming Mail Server and the Outgoing Mail Server.


Incoming Mail Servers: mvs28a.itso.ral.ibm.com

To set server properties (such as checking for new messages automatically), select a server and then click Edit.

Outgoing Mail Server

Outgoing mail (SMTP) server: mvs28a.itso.ral.ibm.com

Outgoing mail server user name: PESCHKE

Use Secure Socket Layer(SSL) or TLS for outgoing messages: [✓ Never] If Possible Always

Local mail directory: C:\Program Files\Netscape\Users\mail

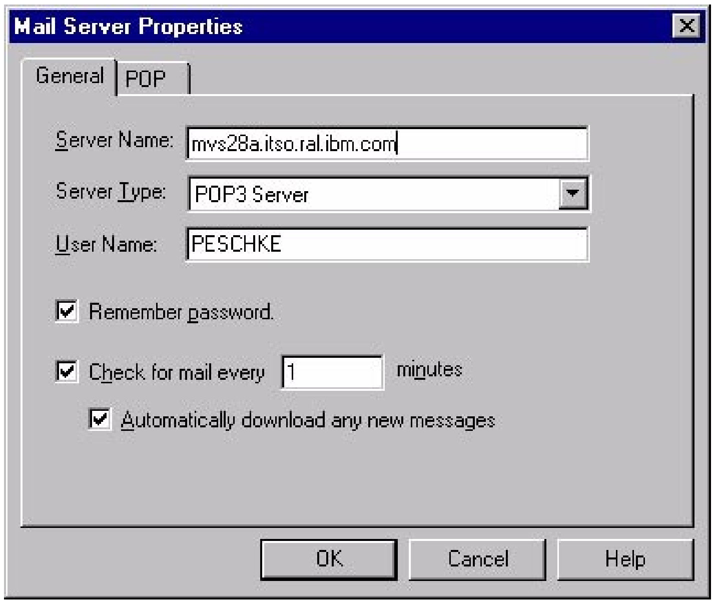
Figure 5-33 Preferences: Mail Servers

Incoming mail server

You define the following on the Mail Server Properties Panel (see Figure 5-34 on page 82):

- Incoming server
- Server type
- User name

In our sample user PESCHKE will receive mail from the server on system MVS28A. There is only one mail server defined. It is the popper which runs on MVS28A. The server type is POP3. The user name is PESCHKE, known as the local user in the system MVS28A. This name is used by the popper to retrieve messages from PESCHKE's mail spool file and to create the temporary maildrop file with the user name PESCHKE. If there is no incoming mail server defined you have to select the **Add** button or if already defined select the **Edit** button.


Mail Server Properties

General | POP

Server Name: mvs28a.itso.ral.ibm.com

Server Type: POP3 Server

User Name: PESCHKE

[✓] Remember password.

[✓] Check for mail every 1 minutes

[✓] Automatically download any new messages

Figure 5-34 Mail Server Properties (General)

This screen allows you to define the:

- Server name
- Server type
- User name

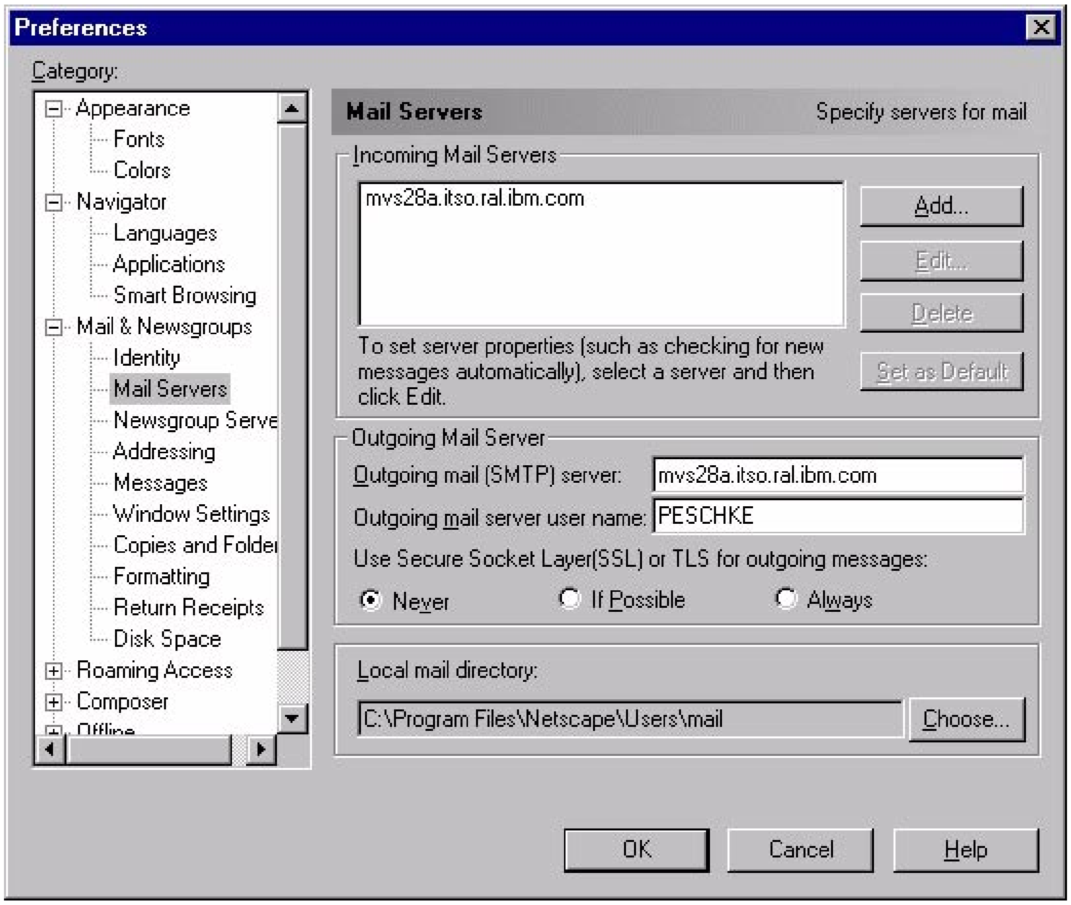
You may define only one POP3 server.

If you don't use the OS/390 popper server you may use the IMAP server type for incoming mail. For example, mailboxes are used from Internet service providers. In this case multiple IMAP servers may be defined. IMAP doesn't use the POP3 protocol; it uses SMTP.

On the POP panel, which is not shown here, you can define when to delete the message on the server.

Outgoing mail server

The client function of your PC establishes an SMTP connection to the defined outgoing mail server, which is defined here as MVS28A.itso.ral.ibm.com. Within the system MVS28A the user PESCHKE has to be defined also as local user. Therefore, you need the user's name in the Netscape definition to identify to the user in the server.



Mail Servers Specify servers for mail

Incoming Mail Servers

mvs28a.itso.ral.ibm.com

To set server properties (such as checking for new messages automatically), select a server and then click Edit.

Outgoing Mail Server

Outgoing mail (SMTP) server: mvs28a.itso.ral.ibm.com

Outgoing mail server user name: PESCHKE

Use Secure Socket Layer(SSL) or TLS for outgoing messages:

[✓ Never] If Possible Always

Local mail directory:

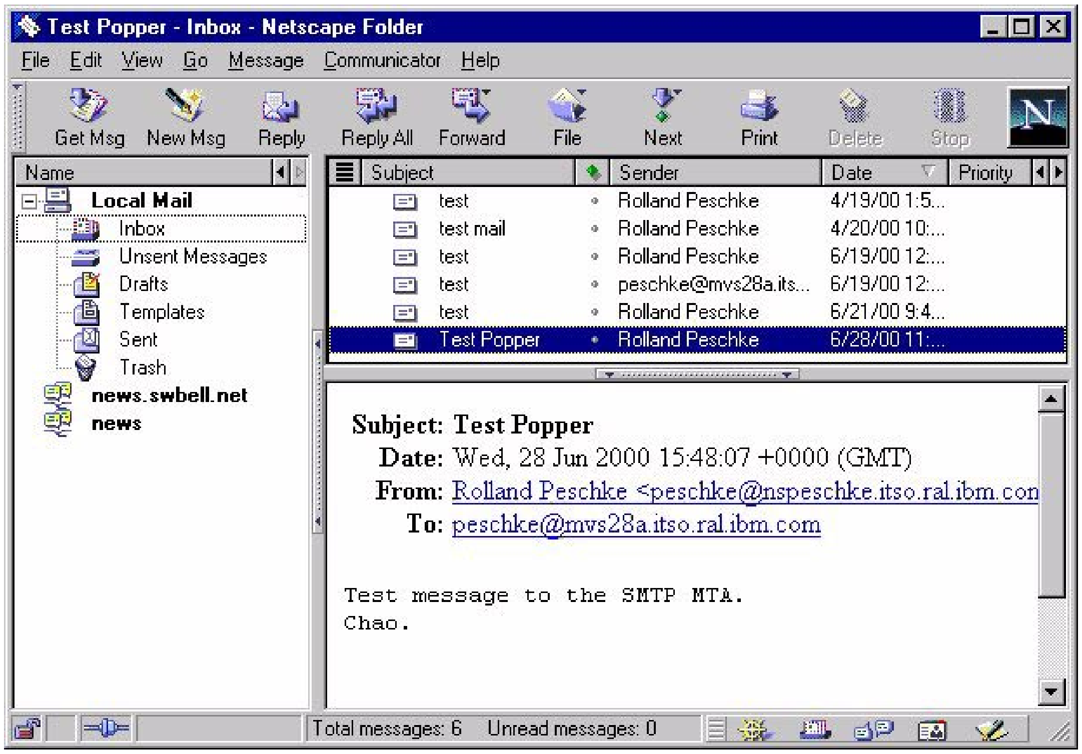
C:\Program Files\Netscape\Users\mail

Figure 5-35 Preferences: Outgoing mail

The minimum definitions are done now. You may use the popper.

5.4.4 Using the popper

Since the popper runs the POP3 protocol you have to use an MUA application working as a POP3 client. In our case we used the Netscape Communicator. The Netscape Communicator Messenger allows all notes for the user PESCHKE from the server on MVS28A to be retrieved by clicking the **Get Message** button.



Subject	Sender	Date	Priority
test	Rolland Peschke	4/19/00 1:5...	
test mail	Rolland Peschke	4/20/00 10:...	
test	Rolland Peschke	6/19/00 12:...	
test	peschke@mvs28a.its...	6/19/00 12:...	
test	Rolland Peschke	6/21/00 9:4...	
Test Popper	Rolland Peschke	6/28/00 11:...	

Subject: **Test Popper**
Date: Wed, 28 Jun 2000 15:48:07 +0000 (GMT)
From: Rolland Peschke <peschke@nspeschke.itso.ral.ibm.con
To: peschke@mvs28a.itso.ral.ibm.com

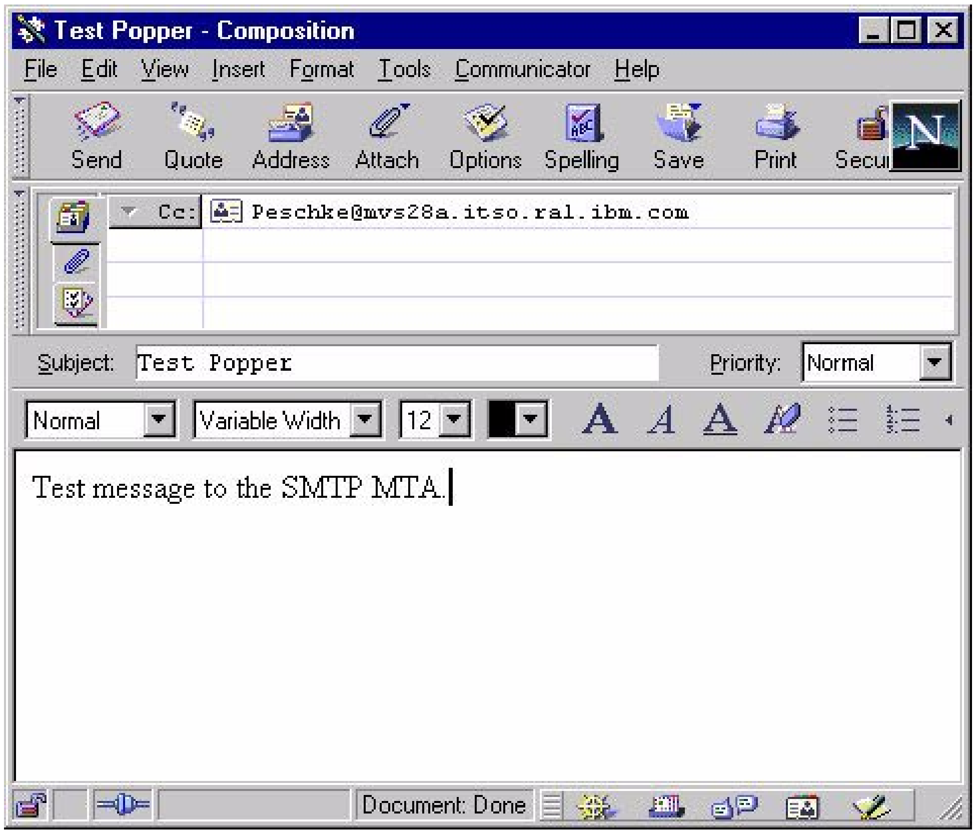
```
Test message to the SMTP MTA.
Chao.
```

Figure 5-36 Inbox Netscape folder

In our sample you may discover incoming mail with or without attachments.

5.4.5 Using Netscape to send files

While the popper is for incoming mail only you may also use a mail server on another system (may be installed on another system than the popper, for example on system MVS03A) for outgoing mail. You invoke the Netscape Communicator Messenger as done before and select the New Message button and finally create your message. When you have finished composing the mail, click the **Send** button. The message will now be transmitted to the server defined in Preferences for outgoing mail.


Test Popper - Composition
File Edit View Insert Format Tools Communicator Help
Send Quote Address Attach Options Spelling Save Print Secu
Normal Variable Width 12
Document: Done

Cc: Peschke@mvs28a.itso.ral.ibm.com

Subject: Test Popper

Priority: Normal

Test message to the SMTP MTA.

Figure 5-37 Creating a new mail message

5.4.6 popper debugging samples

During our tests we ran into several error situations.

Failed connection to the popper server

The user name was not defined correctly to the Netscape Communicator.

```
Debugging turned on
EZZ7608 Unable to get canonical name of client: m
(v2.3) Servicing request from "9.24.106.64" at 9.24.106.64
+OK QPOP (version 2.3) at MVS28A.itso.ral.ibm.com starting.
Sending line "+OK QPOP (version 2.3) at MVS28A.itso.ral.ibm.com starting.
Received: "USER default"
@ 9.24.106.64 : -ERR Unknown username: default
Sending line "-ERR Unknown username: default"
@ 9.24.106.64 : -ERR POP EOF received
Sending line "-ERR POP EOF received"
+OK Pop server at MVS28A.itso.ral.ibm.com signing off.
Sending line "+OK Pop server at MVS28A.itso.ral.ibm.com signing off."
(v2.3) Ending request from "" at (9.24.106.64) 9.24.106.64
```

Figure 5-38 Failed connection to the popper server

Correct connection

```
Debugging turned on
(v2.3) Servicing request from "nspeschke.itso.ral.ibm.com" at 9.24.104.11
1
+OK QPOP (version 2.3) at MVS28A.itso.ral.ibm.com starting.
Sending line "+OK QPOP (version 2.3) at MVS28A.itso.ral.ibm.com starting.  "
Received: "USER PESCHKE"
+OK Password required for PESCHKE.
Sending line "+OK Password required for PESCHKE."
Received: "pass xxxxxxxxx"
Creating temporary maildrop '/usr/mail/popper/.PESCHKE.pop'
uid = 4029, gid = 1
Checking for old .PESCHKE.pop file
Old .PESCHKE.pop file not found, errno (0)
Msg 1 being added to list
Msg 1 uidl c313e341d7a5167b833153eb4bbfea25  at offset 0 is 700 octets long and h
Msg 2 being added to list
Msg 2 uidl c50b65c95f934fb6c22dc23573be88a1  at offset 748 is 2072 octets long an
Msg 3 being added to list
Msg 3 uidl 91c0636d1513127489f49995e6d8f1e5  at offset 2842 is 3998 octets long a
Msg 4 uidl 61021c4ea6471f83f518d8c64d8c1740  at offset 6772 is 453814 octets long
Msg 5 being added to list
Msg 5 uidl da701c81e2b2df3a60121a5ca1cdd76b  at offset 454502 is 928 octets long
Msg 6 being added to list
Msg 1 uidl c313e341d7a5167b833153eb4bbfea25  at offset 0 is 700 octets long and h
Msg 2 uidl c50b65c95f934fb6c22dc23573be88a1  at offset 748 is 2072 octets long an
Msg 3 uidl 91c0636d1513127489f49995e6d8f1e5  at offset 2842 is 3998 octets long a
Msg 4 uidl 61021c4ea6471f83f518d8c64d8c1740  at offset 6772 is 453814 octets long
Msg 5 uidl da701c81e2b2df3a60121a5ca1cdd76b  at offset 454502 is 928 octets long
Msg 6 uidl 0a48082f727723c8f47b306e26b49652  at offset 455469 is 691 octets long
+OK PESCHKE has 6 messages (462203 octets).
Sending line "+OK PESCHKE has 6 messages (462203 octets)."
Received: "STAT"
6 message(s) (462203 octets).
+OK 6 462203
Sending line "+OK 6 462203"
Received: "LIST"
+OK 6 messages (462203 octets)
Received: "UIDL"
+OK uidl command accepted.
Sending line "+OK uidl command accepted."
Sending line "1 c313e341d7a5167b833153eb4bbfea25"
Sending line "2 c50b65c95f934fb6c22dc23573be88a1"
Sending line "3 91c0636d1513127489f49995e6d8f1e5"
Sending line "4 61021c4ea6471f83f518d8c64d8c1740"
Sending line "5 da701c81e2b2df3a60121a5ca1cdd76b"
Sending line "6 0a48082f727723c8f47b306e26b49652"
Received: "QUIT"
Performing maildrop update...
Checking to see if all messages were deleted
Opening mail drop "/usr/mail/PESCHKE"
Creating new maildrop "/usr/mail/PESCHKE" from "/usr/mail/popper/.PESCHKE.pop"
Copying message 1.
Copying message 2.
Copying message 3.
Copying message 4.
Copying message 5.
Copying message 6.
+OK Pop server at MVS28A.itso.ral.ibm.com signing off.
Sending line "+OK Pop server at MVS28A.itso.ral.ibm.com signing off."
(v2.3) Ending request from "PESCHKE" at (nspeschke.itso.ral.ibm.com) 9.24.104.111
******************************** Bottom of Data *********************************
```

Figure 5-39 Working connection to the popper server

Temporary maildrop file cannot be opened

The /usr/mail/popper directory was not defined.

```
Debugging turned on
EZZ7608 Unable to get canonical name of client: m
(v2.3) Servicing request from "9.24.104.96" at 9.24.104.96
+OK QPOP (version 2.3) at MVS03C.itso.ral.ibm.com starting.
Sending line "+OK QPOP (version 2.3) at MVS03C.itso.ral.ibm.com starting.  "
Received: "USER karl"
+OK Password required for KARL.
Sending line "+OK Password required for KARL."
Received: "pass xxxxxxxxx"
Creating temporary maildrop '/usr/mail/popper/.KARL.pop'
uid = 4019, gid = 1
EZZ7605I: Unable to open temporary maildrop '/usr/mail/popper/.KARL.pop':
KARL@ 9.24.104.96 : -ERR System error, can't open temporary file, do you own it?
Sending line "-ERR System error, can't open temporary file, do you own it?
+OK Pop server at MVS03C.itso.ral.ibm.com signing off.
Sending line "+OK Pop server at MVS03C.itso.ral.ibm.com signing off."
(v2.3) Ending request from "KARL" at (9.24.104.96) 9.24.104.96
: gethostbyaddr(192.168.233.4) failed: 0
```

Figure 5-40 Missing /usr/mail/popper directory

5.5 Bind-based Domain Name Server and sendmail

TCCP/IP applications refer to host computers by their IP addresses, but human beings find it easier to use and remember better host names. Host tables can be used to translate the host name into an IP address.

The file etc/hosts contains the IP addresses and their correspondent host names. In MVS TCP/IP, the host tables are built with the hlq.HOSTS.LOCAL data set. After **makesite** is run the data sets hlq.HOSTS.ADDRINFO and hlq.HOSTS.SITEINFO are created and contain the host tables. But when testing the sendmail we disabled the route to the DNSD server. When the DNS server is unreachable, the name of the destination host cannot be resolved even if that host name is in the /etc/hosts or in the hql.HOSTS.ADDRINFO and hql.HOST.SITEINFO and the mail to be sent is queued.

The domain name server resolves the names of the hosts but also supports an advanced mail routing. Backup hosts can be specified to handle the mail for a destination host that cannot be reached at a certain time for whatever reason. They can assume mail handling responsibilities for other hosts.

The complete name of a host, also known as the fully qualified domain name (FQDN), is a series of labels separated by dots or periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to one of the larger networks generally has more than one subdomain as shown in these examples:

```
host.subdomain1.subdomain2.subdomain3.rootdomain
mvs03a.itso.ral.ibm.com
```

The resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

The domain name server also provides IP-to-host name mapping using a special domain called in-addr.arpa. This kind of mapping is useful for producing output (host names) that is easy to read. The format of an in-addr.arpa name is the reverse octet of an IP address concatenated with the in-addr.arpa. For example the address 9.24.104.113 has an in-addr.arpa name of 113.104.24.9 in-addr.arpa.

5.5.1 MX records

The basic idea behind MX records is to send the mail as closely as possible to the final destination. The domain name server uses this single type of record to provide advanced mail routing. Originally two records were needed. The MD record (mail destination) and the MF record (mail forwarder) used in case the destination was unreachable.

The mail server had to do two queries, one for the MD and one for the MF data. The overhead of running the mail server was higher than running other servers.

The two records were merged into a single record type: MX.

The general format of a resource record in the domain name server is

```
{name}{ttl}{address_class} record_type record_data
```

name	Specifies the name of the zone or the host system associated with the record. This field is optional. If the field is blank, the name server uses the name of the zone or host system from the preceding resource record. You can use the special character @ in place of a zone name if the zone name is the same as the zone defined in the boot file.
ttl	Specifies the time-to-live value, in number of seconds, which is the amount of time this record is valid in a cache. This field is optional. The default is the time-to-live value contained in the minimum field in the file's start of authority (SOA) record.
address_class	Specifies the address class of the entry. The allowable values are HS (specifies HESIOD class), IN (specifies the TCP/IP-based Internet). This field is optional. The default is IN.
record_type	Should be MX for a record defining a mail exchanger, which identifies a host capable of acting as a mail exchanger for the domain specified in the name field.
record_data	Contains the preference and the exchanger name. The MX record type is followed by the mail preference, which is the priority number used to rank the mail exchangers. Mailers attempt delivery first to the mail exchangers with the lowest preference value. If delivery fails, the host with the next to the highest value is tried. The highest possible preference value is 0, the next highest preference value is 1, and so on. Hosts with identical preference values are selected randomly. The exchanger name is the host name running the mail server. Create an MX record for every host that receives mail.

SeeFigure 5-41 on page 89 for an example of MX record entries.

```
mvs03a        IN         A    9.24.104.113

              IN         MX    0   mvs03a   1

              IN         MX    5   mvs28a   2

              IN         MX    10  mvs39a   3
```

Figure 5-41 MX records in the DNS

1	The host with the name MVS03A has the highest preference. Mailers should attempt to deliver mail first to this mail exchanger.
2	The host with the name MVS28A has the second highest preference. Mailers should deliver mail to this mail exchanger if the first is not reachable.
3	The host with name MVS39A has the third priority and mailers should deliver to this mail exchanger if the two previous servers are not reachable.

In the figure above we see preferences specified of 0, 5 and 10. Preferences of 0, 50, 100 do exactly the same as 0, 5, 10. The preferences specified in the MX records also prevent mail routing loops.

5.5.2 Configuration

The configuration we are using is a Sysplex with three systems running in it, System RA03, System RA28 and System RA39.

See the configuration with token-ring in Figure 5-42. The other links, 2216, 3746-MAE, 3172, Ethernet via OSA and dynamic XCF are not shown here. The VIPA address has for now been put in the HOME statement just after the token-ring addresses. The token-ring addresses are independent from the VIPA.

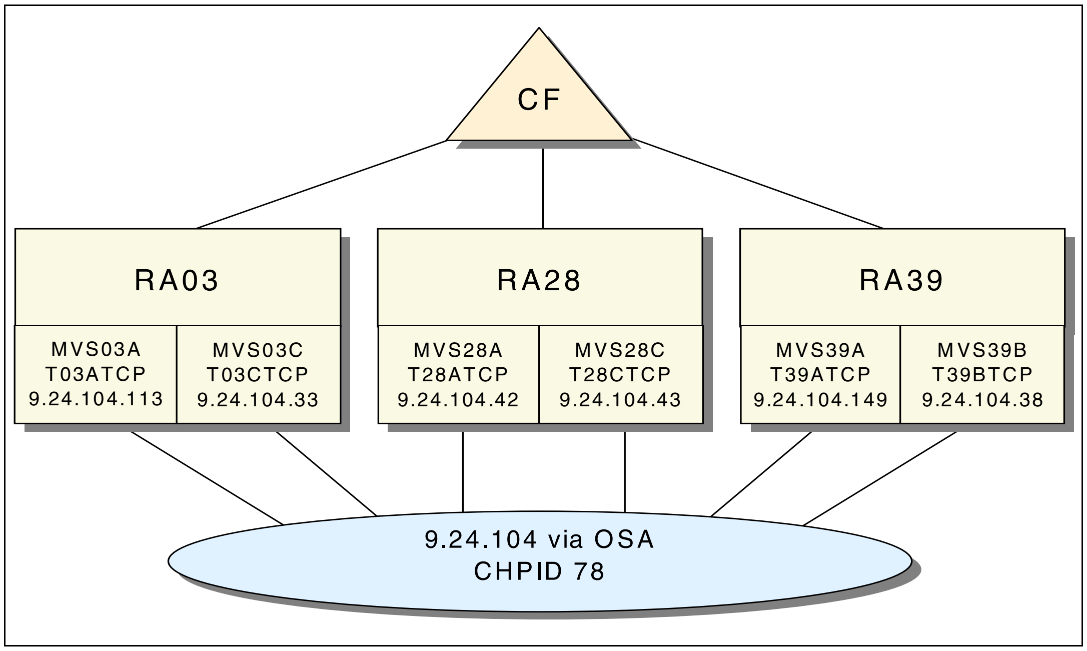


Figure 5-42 Configuration with token-ring

Each system runs two TCP/IP stacks:

- System RA03:

```
First TCP/IP stack:
      TCP/IP jobname : T03ATCP
      host name      : MVS03A
      ip address     : 9.24.104.113
Second TCP/IP stack:
      TCP/IP jobname : T03CTCP
      host name      : MVS03C
      ip address     : 9.24.104.33
```

- System RA28:

```
First TCP/IP stack:
      TCP/IP jobname : T28ATCP
      host name      : MVS28A
      ip address     : 9.24.104.42
Second TCP/IP stack:
      TCP/IP jobname : T28CTCP
      host name      : MVS28C
      ip address     : 9.24.104.43
```

- System RA39:

```
First TCP/IP stack:
      TCP/IP jobname : T39ATCP
      host name      : MVS39A
      ip address     : 9.24.104.149
Second TCP/IP stack:
      TCP/IP jobname : T39BTCP
      host name      : MVS39B
      ip address     : 9.24.104.38
```

Tests have been run with DNS server, SMTPPROC and sendmail on the different systems.

- DNS server on T03ATCP
- SMTPPROC on T03ATCP
- sendmail on T03CTCP
- sendmail on T28ATCP
- sendmail on T28CTCP
- sendmail on T39ATCP
- sendmail on T39BTCP

5.5.3 Files in the BIND-based DNS server

In this section we discuss the files necessary to configure the domain name server. For detailed information about the Domain Name System you may refer to Chapter 10, "BIND Domain Name System (DNS)" on page 315. The files are listed below:

- The boot file
- The forward file
- The reverse file
- The loopback file
- The Cache file

The boot file

The default boot file is named.boot and can be found in the /etc directory. We changed the name to /etc/t03dns.boot. This file is referenced in the start procedure used from the MVS console or the autolog definition in the TCP/IP profile or in the z/OS UNIX command used to start the domain name server.

```
;
;    /etc/named.boot for T03ATCP
;
;  TYPE      DOMAIN                    HOST        FILE
;
directory    /etc/dnsdata  1
;
primary    itso.ral.ibm.com          2  t03dns.for
primary    104.24.9.in-addr.arpa     3  t03dns.rev
primary    105.24.9.in-addr.arpa     4  t03dns.rev.105.24.9
primary    168.192.in-addr.arpa      5  t03dns.rev.168.192
primary    0.0.127.in-addr.arpa      6  t03dns.lbk
cache      .                         7  t03dns.ca
forwarders 9.24.104.108
options query-log
```

Figure 5-43 DNS boot file /etc/t03dns.boot

1 This entry indicates the directory in which the other files reside; /etc/dnsdata in this case.

2 This entry tells us where the forward file can be found in the /etc/dnsdata directory: t03dns.for.

3 The reverse file is located in /etc/dnsdata/t03dns.rev for the addresses in the shape of 104.24.9.in-addr.arpa. This means for the address range from 9.24.104.1 to 9.24.104.254, the domain name server should scan this file.

4 The reverse file /etc/dsndata/t03dns.rev.105.24.9 should be scanned for the address range from 9.24.105.1 to 9.24.105.254.

5 The reverse file /etc/dnsdata/t03dns.rev.168.192 should be scanned for the address range from 192.168.0.1 to 192.168.255.254.

6 The file /etc/dnsdata/t03dns.lbk should be consulted for loopback addresses in the range of 127.0.0.1 to shape 127.0.0.254.

7 Points to the cache file /etc/dnsdata/t03dns.ca.

The forward file

The forward file contains the host names the domain name server will be able to translate in IP addresses and the MX records indicating the names and IP addresses of the mail exchangers and their preferences. See Figure 5-44 on page 93 for an example.

```
 ;
 ;  /etc/dnsdata/t03dns.for for T03ATCP
 ;
 $ORIGIN itso.ral.ibm.com.
 @ IN      SOA     mvs03a.itso.ral.ibm.com. vandeke@mvs03a  (
             1998051904  ; Serial
             7200        ; Refresh time after 2 hours
             3600        ; Retry after 1 hour
             604800      ; Expire after 1 week
             3600    )   ; Minimum TTL of 1 hour
                 IN      NS      mvs03a
localhost       IN      A       127.0.0.1
mvs03a          IN      A       192.168.250.3
mvs03a          IN      A       9.24.104.113
                IN      MX      0 mvs03a
                IN      MX      5 mvs28a
                IN      MX      10 mvs39a
ra3anje         IN      CNAME   mvs03a
mvs03c          IN      A       9.24.104.33
mvs03c          IN      A       192.168.251.4
mvs03e          IN      A       9.24.105.76
mvs28a          IN      A       192.168.252.28
mvs28a          IN      A       9.24.104.42
                IN      MX      0 mvs28a
                IN      MX      5 mvs03a
                IN      MX      10 mvs39a
mvs28c          IN      A       9.24.104.43
mvs28c          IN      A       192.168.253.29
mvs28e          IN      A       9.24.105.77
mvs39a          IN      A       192.168.232.39
mvs39a          IN      A       9.24.104.149
                IN      MX      0 mvs39a
                IN      MX      5 mvs03a
                IN      MX      10 mvs28a
mvs39b          IN      A       9.24.104.38
mvs39b          IN      A       192.168.233.40
mvs39e          IN      A       9.24.105.73
wtr05246        IN      A       9.24.104.183
ourmvs        IN     CNAME   mvs03a
```

Figure 5-44 Forward file /etc/dnsdata/t03dns.for

The reverse file

```
File  /etc/dnsdata/t03dns.rev

@ IN SOA mvs03a.itso.ral.ibm.com. vande e@mvs03a (
     199805190 ; Serial
     7200 ; Refresh time after 2 hours
     3600 ; Retry after 1 hour
     604800 ; Expire after 1 wee
     3600 ) ; Minimum TTL of 1 hour
      IN NS mvs03a.itso.ral.ibm.com.
113  8  IN PTR mvs03a.itso.ral.ibm.com.
33      IN PTR mvs03c.itso.ral.ibm.com.
2       IN PTR mvs28a.itso.ral.ibm.com.
3       IN PTR mvs28c.itso.ral.ibm.com.
149     IN PTR mvs39a.itso.ral.ibm.com.
38      IN PTR mvs39b.itso.ral.ibm.com.
183     IN PTR wtr05246.itso.ral.ibm.com.

File /etc/dnsdata/t03dns.rev.105.24.9

@ IN SOA mvs03a.itso.ral.ibm.com. vande e@mvs03a (
     199805190 ; Serial
     7200 ; Refresh time after 2 hours
     3600 ; Retry after 1 hour
     604800 ; Expire after 1 wee
     3600 ) ; Minimum TTL of 1 hour
      IN NS mvs03a.itso.ral.ibm.com.
76  9   IN PTR mvs03e.itso.ral.ibm.com.
77      IN PTR mvs28e.itso.ral.ibm.com.
73      IN PTR mvs39e.itso.ral.ibm.com.

File /etc/dnsdata/t03dns.rev.168.192

@ IN SOA mvs03a.itso.ral.ibm.com. vande e@mvs03a (
     199805190 ; Serial
     7200 ; Refresh time after 2 hours
     3600 ; Retry after 1 hour
     604800 ; Expire after 1 wee
     3600 ) ; Minimum TTL of 1 hour
      IN NS mvs03a.itso.ral.ibm.com.
3.250  10  IN PTR mvs03a.itso.ral.ibm.com.
4.251      IN PTR mvs03c.itso.ral.ibm.com.
4.221      IN PTR mvs03c.itso.ral.ibm.com.
28.252     IN PTR mvs28a.itso.ral.ibm.com.
29.253     IN PTR mvs28c.itso.ral.ibm.com.
39.232     IN PTR mvs39a.itso.ral.ibm.com.
40.233     IN PTR mvs39b.itso.ral.ibm.com.
```

Figure 5-45 Reverse files

8 When the domain name server has to translate an IP address, for example 9.24.104.113, it looks in the boot file and finds the entry 3 in Figure 5-43 on page 91 104.24.9.in-addr.arpa is to be found in file /etc/dnsdata/t03dns.rev where the entry 113 is the address we are looking for and the name is mvs03a.itso.ral.ibm.com.

9 When the domain name server has to translate an ip address, for example 9.24.105.76, it looks in the boot file and finds the entry 4 in Figure 5-43 105.24.9.in-addr.arpa is found in file /etc/dnsdata/t03dns.rev.105.24.9 where the entry 76 is the address we are looking for and the name is mvs03e.itso.ral.ibm.com.

10 When the domain name server has to translate an ip address for example 192.168.250.3 it looks in the boot file and finds the entry 5 in Figure 5-43 168.192.in-addr.arpa is found in file /etc/dnsdata/t03dns.rev.168.192 where the entry 3.250 is the address we are looking for and the name is mvs03a.itso.ral.ibm.com.

The loopback file

The loopback file contains the loopback address. This is the address that the host uses to route queries to itself. The default is 127.0.0.1, but you can configure additional loopback addresses. In Figure 5-46 the loopback address contains only a 1. This has to added in reverse order to 0.0.127.in-addr.arpa.

```
@   IN      SOA     mvs03a.itso.ral.ibm.com vandeke@mvs03a  (
            1998051901  ; Serial
            7200        ; Refresh time after 2 hours
            3600        ; Retry after 1 hour
            604800      ; Expire after 1 week
            3600    )   ; Minimum TTL of 1 hour
                IN      NS      mvs03a.itso.ral.ibm.com.
1               IN      PTR     loopback.
```

Figure 5-46   The loopback file /etc/dnsdata/t03dns.lbk

The cache file

Figure 5-47 shows the cache or hints file containing the IP addresses of the authoritative root domain server. The name servers contain the names of name servers in the top-level such as com, edu,and mil.

```
 .              IN      NS A.ROOT-SERVERS.NET.
 A.ROOT-SERVERS.NET.  IN  A  9.24.104.109
```

Figure 5-47   The cache file /etc/dnsdata/t03dns.ca

5.5.4 Startup

In the profile of TCP/IP we have put the autolog on for the domain name server.

```
 AUTOLOG 1
    T03DNS  JOBNAME T03DNS1    ; Domain Name Server
```

The start procedure USER.PROCLIB(T03DNS) in Figure 5-48 on page 96. points to the /etc/t03dns.boot file and requests port 53 to be used for the domain name server.

```
//T03DNS PROC B='/etc/t03dns.boot',P='53'
//*************************************************
//T03DNS    EXEC PGM=EZANSNMD,REGION=0K,TIME=NOLIMIT,
//      PARM='POSIX(ON) ALL31(ON)/ -b &B -p &P'
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSTCPD DD DISP=SHR,DSN=TCP.TCPPARMS(TDATA03A)
```

Figure 5-48 Procedure USER.PROCLIB(T03DNS)

The TCPIP.DATA files for the TCP/IP stacks that can be seen in Figure 5-42 contain all the NSINTERADDR of the TCPIP stack with host name MVS03A and IP address 9.24.104.113. See Figure 5-49 for the TCPIP.DATA file for TCP/IP stack MVS03A.

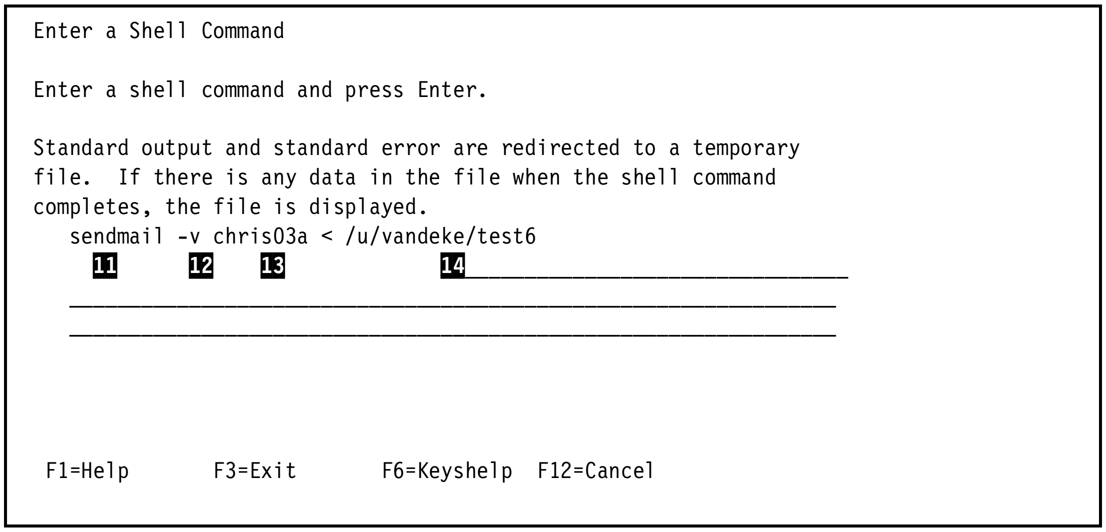
```
TCPIPJOBNAME T03ATCP
RA3ANJE: HOSTNAME  MVS03A
DOMAINORIGIN  itso.ral.ibm.com
NSINTERADDR  9.24.104.113   ;RA03 nameserver..
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 2
TRACE RESOLVER
DATASETPREFIX TCP
MESSAGECASE MIXED
```

Figure 5-49 TCPIP.DATA file in MVS03A

The entry TRACE RESOLVER will produce messages of question and answers from, in our case, the sendmail to the domain name server.

5.5.5 Operation of DNS server with sendmail

Once the domain name server is running, the sendmail on any of the MVS TCP/IP stacks can request the translation of a host name or the vice versa. This means the IP address is sent to the domain name server and the host name is returned. In Figure 5-50 we issued the **sendmail** command.



```
Enter a Shell Command

Enter a shell command and press Enter.

Standard output and standard error are redirected to a temporary
file.  If there is any data in the file when the shell command
completes, the file is displayed.
   sendmail -v chris03a < /u/vandeke/test6
     11      12    13             14

 F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel
```

Figure 5-50   sendmail command in OS/390 UNIX ISHELL

11 `Sendmail` command is in fact the client. The user wants to send a mail, issues the command and uses the running daemon.

12 -v activates the verbose mode and creates a lot of output.

13 chris03a is an entry in the alias file of the sendmail daemon of the system where a user wants to send mail. Here the alias is translated to vandeke@MVS03A.

14 The file /u/vandeke/test6, due to the < sign will be put in a queue, ready to transmit.

```
res_querydomain(MVS03A, itso.ral.ibm.com, 1, 1)
res_query(MVS03A.itso.ral.ibm.com, 1, 1)
res_m query(0, MVS03A.itso.ral.ibm.com, 1, 1)
res_send()
HEADER:
.opcode = QUERY, id = 1, rcode = NOERROR
.header flags: rd
.qdcount = 1, ancount = 0, nscount = 0, arcount = 0
QUESTIONS:
.MVS03A.itso.ral.ibm.com, type = A, class = IN
Querying server (# 1) address = 9.24.104.113
got answer:
HEADER:
.opcode = QUERY, id = 1, rcode = NOERROR
.header flags: qr aa rd ra
.qdcount = 1, ancount = 2, nscount = 1, arcount = 2
QUESTIONS:
.MVS03A.itso.ral.ibm.com, type = A, class = IN
ANSWERS:
.MVS03A.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 9.24.104.113
.MVS03A.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 192.168.250.3
NAME SERVERS:
.itso.ral.ibm.com
.type = NS, class = IN, ttl = 1 hour, dlen = 9
.domain name = mvs03a.itso.ral.ibm.com
ADDITIONAL RECORDS:
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 9.24.104.113
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 192.168.250.3
```

Figure 5-51 Forward resolution of sendmail's host name

After the forward resolution comes the reverse resolution.

```
res_query(113.10 .2 .9.in-addr.arpa, 1, 12)
res_m query(0, 113.104.24.9.in-addr.arpa, 1, 12)
res_send()
HEADER:
.opcode = QUERY, id = 2, rcode = NOERROR
.header flags: rd
.qdcount = 1, ancount = 0, nscount = 0, arcount = 0
QUESTIONS:
.113.104.24.9.in-addr.arpa, type = PTR, class = IN
Querying server (# 1) address = 9.24.104.113
got answer:
HEADER:
.opcode = QUERY, id = 2, rcode = NOERROR
.header flags: qr aa rd ra
.qdcount = 1, ancount = 1, nscount = 1, arcount = 2
QUESTIONS:
.113.104.24.9.in-addr.arpa, type = PTR, class = IN
ANSWERS:
.113.104.24.9.in-addr.arpa
.type = PTR, class = IN, ttl = 1 hour, dlen = 25
.domain name = mvs03a.itso.ral.ibm.com
NAME SERVERS:
.104.24.9.in-addr.arpa
.type = NS, class = IN, ttl = 1 hour, dlen = 2
.domain name = mvs03a.itso.ral.ibm.com
ADDITIONAL RECORDS:
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 9.24.104.113
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 192.168.250.3
```

Figure 5-52 Reverse IP address resolution of primary interface

Several IP addresses are defined in the domain name server for MVS03A. There is an information exchange between the sendmail and the domain name server.

```
res_query(3.250.168.192.in-addr.arpa, 1, 12)
res_m query(0, 3.250.168.192.in-addr.arpa, 1, 12)
res_send()
HEADER:
.opcode = QUERY, id = 3, rcode = NOERROR
.header flags: rd
.qdcount = 1, ancount = 0, nscount = 0, arcount = 0
QUESTIONS:
.3.250.168.192.in-addr.arpa, type = PTR, class = IN
Querying server (# 1) address = 9.24.104.113
got answer:
HEADER:
.opcode = QUERY, id = 3, rcode = NOERROR
.header flags: qr aa rd ra
.qdcount = 1, ancount = 1, nscount = 1, arcount = 2
QUESTIONS:
.3.250.168.192.in-addr.arpa, type = PTR, class = IN
ANSWERS:
.3.250.168.192.in-addr.arpa
.type = PTR, class = IN, ttl = 1 hour, dlen = 25
.domain name = mvs03a.itso.ral.ibm.com
NAME SERVERS:
.168.192.in-addr.arpa
.type = NS, class = IN, ttl = 1 hour, dlen = 2
.domain name = mvs03a.itso.ral.ibm.com
ADDITIONAL RECORDS:
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 9.24.104.113
.mvs03a.itso.ral.ibm.com
.type = A, class = IN, ttl = 1 hour, dlen = 4
.internet address = 192.168.250.3
res_query(3.20.168.192.in-addr.arpa, 1, 12)
res_m query(0, 3.20.168.192.in-addr.arpa, 1, 12)
res_send()
HEADER:
.opcode = QUERY, id = 4, rcode = NOERROR
.header flags: rd
.qdcount = 1, ancount = 0, nscount = 0, arcount = 0
QUESTIONS:
.3.20.168.192.in-addr.arpa, type = PTR, class = IN
Querying server (# 1) address = 9.24.104.113
got answer:
HEADER:
.opcode = QUERY, id = 4, rcode = NXDOMAIN
.header flags: qr aa rd ra
.qdcount = 1, ancount = 0, nscount = 1, arcount = 0
QUESTIONS:
.3.20.168.192.in-addr.arpa, type = PTR, class = IN
NAME SERVERS:
.168.192.in-addr.arpa
.type = SOA, class = IN, ttl = 1 hour, dlen = 62
.origin = mvs03a.itso.ral.ibm.com
.mail addr = vande e@mvs03a.168.192.in-addr.arpa
.serial=199805190 , refresh=2 days 17 hours 41 mins 29 secs, retry=2 d
rcode = 3, ancount=0
\es_search failed, errno = 1105
. . .
. . .
```

Figure 5-53 Resolution of all IP addresses of the sendmail host

About 400 entries follow with questions and answers between the sendmail daemon and the domain name server

After the domain name server has resolved the name of the sendmail daemon with a large number of exchanges containing questions and answers, the resolution of the alias is started.

```
chris03a... aliased to VANDEKE@MVS03A
;; res_querydomain(MVS03A, itso.ral.ibm.com, 1, 255)
;; res_query(MVS03A.itso.ral.ibm.com, 1, 255)
;; res_mkquery(0, MVS03A.itso.ral.ibm.com, 1, 255)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65425
;; flags: rd; Ques: 1, Ans: 0, Auth: 0, Addit: 0
;; QUESTIONS:
;;.MVS03A.itso.ral.ibm.com, type = ANY, class = IN


;; ...truncated
;; Querying server (# 1) address = 9.24.104.113
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65425
;; flags: qr aa rd ra; Ques: 1, Ans: 5, Auth: 1, Addit: 6
;; QUESTIONS:
;;.MVS03A.itso.ral.ibm.com, type = ANY, class = IN

;; ANSWERS:
MVS03A.itso.ral.ibm.com..3600.IN.MX.0 mvs03a.itso.ral.ibm.com.
mvs03a.itso.ral.ibm.com..3600.IN.MX.5 mvs28a.itso.ral.ibm.com.  15
mvs03a.itso.ral.ibm.com..3600.IN.MX.10 mvs39a.itso.ral.ibm.com
mvs03a.itso.ral.ibm.com..3600.IN.A.9.24.104.113
mvs03a.itso.ral.ibm.com..3600.IN.A.192.168.250.3

;; AUTHORITY RECORDS:
itso.ral.ibm.com..3600.IN.NS.mvs03a.itso.ral.ibm.com.

;; AUTHORITY RECORDS:
itso.ral.ibm.com..3600.IN.NS.mvs03a.itso.ral.ibm.com.

;; ADDITIONAL RECORDS:
mvs03a.itso.ral.ibm.com..3600.IN.A.9.24.104.113
mvs03a.itso.ral.ibm.com..3600.IN.A.192.168.250.3
mvs28a.itso.ral.ibm.com..3600.IN.A.9.24.104.42
mvs28a.itso.ral.ibm.com..3600.IN.A.192.168.252.28
mvs39a.itso.ral.ibm.com..3600.IN.A.192.168.232.39
mvs39a.itso.ral.ibm.com..3600.IN.A.9.24.104.149

VANDEKE@MVS03A... Connecting to local...
VANDEKE@MVS03A... Sent    16
```

Figure 5-54   Alias resolution

15 The MX records in the domain name server are sent to sendmail.

16 Mail is delivered.

5.6 sendmail and SMTP

In this section we describe the tests executed with several sendmail servers and an SMTP server that has the gateway parameter defined and allows mail to be sent via NJE/RSCS to our VM system.

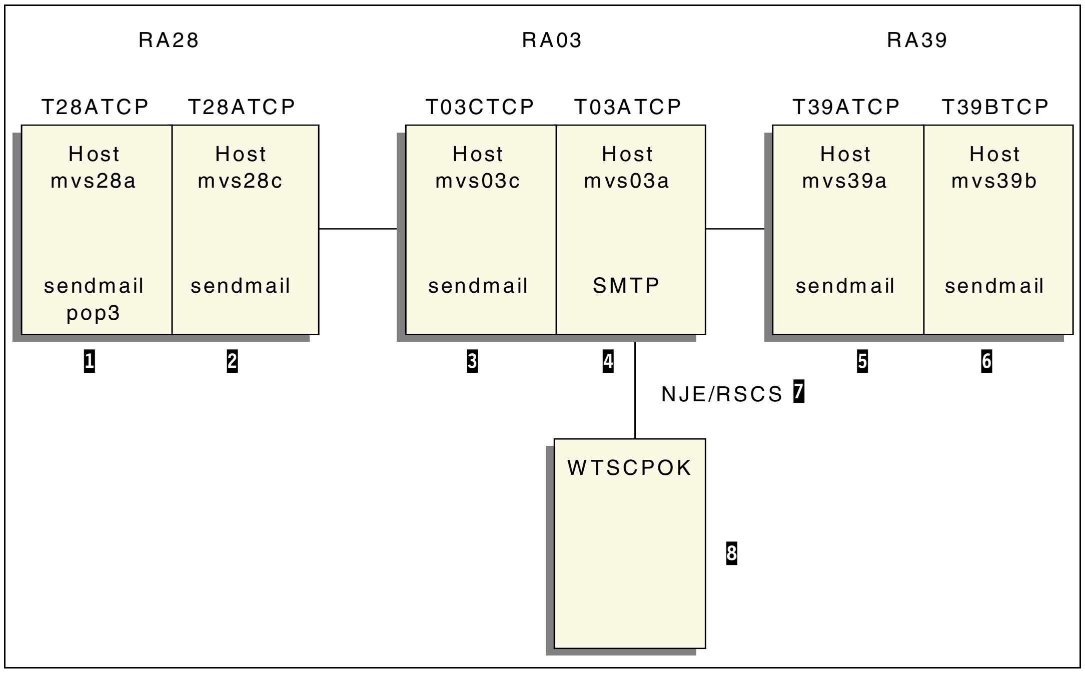


Figure 5-55 Configuration of the TCP/IP and NJE/RSCS network

Figure 5-55 shows the configuration during the tests. Three systems are used; systems RA03, RA28 and RA39. Each system runs two TCP/IP stacks and all are interconnected via several adapters as explained in the routing chapter in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

1	In System RA28 the TCP/IP stack T28ATCP runs sendmail and the POP3.
2	In System RA28 the TCP/IP stack T28CTCP runs only sendmail.
3	In System RA03 the TCP/IP stack T03CTCP runs only sendmail.
4	Only system RA03 with TCP/IP stack T03ATCP runs SMTPPROC.

SMTP can also use MX records, in conjunction with the name server, to direct the server to deliver mail to an alternate host.

With the mail header customization, users can change the rules used to rewrite header addresses and specify desired header address transformations.

The SMTP server accepts SMSG commands from TSO users. This allows the querying of SMTP mail delivery queues and statistics, and provides a set of privileged commands for system administration tasks.

5	System RA39 runs TCP/IP stack T39ATCP with sendmail.
6	System RA39 runs TCP/IP stack T39BTCP with sendmail.
7	JES2 in the RA03 system is using a NJE connection to an RSCS of a VM system.

The SMTP process can be configured to be a mail server and at the same time to be a mail gateway between TCP/IP network sites and NJE/RSCS sites.

8 The remote VM system, WTSCPOK, is connected to the RA03 system.

5.6.1 Configuration of SMTPPROC

The following actions were taken in order to run SMTP as a gateway and to be able to send mail to and from all systems:

1. Autolog and port definitions.
2. Update the SMTPPROC procedure.
3. Update the configuration file.
4. Run the **SMTPNJE** command.
5. Update the secure tables.
6. Update the alias files in the sendmail servers.

Autolog and port definitions

If we want the SMTP to start automatically at the time the TCP/IP address space starts, the name of the member of the catalogued procedure should be added after the AUTOLOG statement in the PROFILE of TCP/IP.

```
AUTOLOG 1
   T03ASMTP                  ; SMTP Server
ENDAUTOlOG
```

Port 25 for SMTP is reserved in the hlq.PROFILE.TCPIP

```
PORT
    25 TCP T03ASMTP           ; SMTP Server
```

Update the SMTP catalogued procedure

The SMTP cataloged procedure is copied from the samples and updated.

```
//SMTP PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE/', SYSERR=SYSERR
//SETSMSG EXEC PGM=SETSMSG,PARM=ON
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SMTP    EXEC PGM=MVPMAIN,
//           PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS',
//           REGION=6144K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSMDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//LOGFILE  DD SYSOUT=*
//SMTPNJE  DD DSN=TCP.SMTPNJE.HOSTINFO,DISP=SHR    1
//CONFIG   DD DSN=TCP.TCPPARMS(SMTP03CF),DISP=SHR  2
//SECTABLE DD DSN=TCP.SMTP.SECTABLE,DISP=SHR       3
//*SMTPRULE DD DSN=SMTP.SMTP.RULE,DISP=SHR         4
//SYSTCPD  DD DSN=TCP.TCPPARMS(TDATA03A),DISP=SHR
```

Figure 5-56 SMTP cataloged procedure

1 Data set created by the **SMTPNJE** command

2 SMTP configuration file

3 The secure tables

4 Data set containing the SMTP rules. It is commented out here and the defaults are used.

Update the configuration file

The important parameters in the configuration file for the tests with a NJE/RSCS connection are:

```
; Configuration for a typical NJE to TCP/IP mail gateway.
GATEWAY  5      ; Accept mail from and deliver mail to NJE host
NJEDOMAIN BITNET ; Pseudo domain name of associated NJE network
NJEFORMAT PUNCH  ; NJE recipients receive mail in punch format
NJECLASS B       ; spool class for mail delivered by
                 ; NJE network
LOCALFORMAT NETDATA ; Local recipients get mail in ne
                    ; Netdata allows TSO receive to b
LOCALCLASS B        ; Spool class for local mail deli
;
REWRITE822HEADER YES NOPRINT
 ;
SECURE   6
 ;
```

Figure 5-57 Part of configuration file

5 To be able to use the NJE/RSCS connection to transfer mail from a sendmail to a VM or MVS system, the gateway parameter must be specified. Sending mail from SMTP to an NJE or RSCS node via another SMTP server, the destination is composed of 'userid'%'nodeid'@'hostname', where the user ID is the VM/MVS user ID, the node ID is the NJE/RSCS node ID and the host name is the TCP/IP host name of the gateway. Sending mail from sendmail to NJE or RSCS, a destination like 'userid'%'nodeid'@'hostname' is not understood by sendmail and an error message is issued. The mail stays in the gateway.

A solution is to change the configuration file in the sendmail /etc/sendmail.cf using the /etc/sendmail.mc and the M4 macro preprocessor and adapt the rule sets.

6 Another solution is to specify the secure option. The secure option forces the SMTP to scan the secure table and to use the nickname.

sendmail has no problem with a destination in the shape of 'nickname'@'hostname'. The nicknames can be defined in the /etc/aliases.

Run the SMTPNJE command

SMTP must know the name of the NJE/RSCS node where it can send mail. The command **`SMTPNJE 'datasetname' (JES2`** creates a data set hlq.SMTPNJE.HOSTINFO. and contains the host names known in the NJE/RSCS network. Following is a small example of the file:

```
WTSCPOK
RASANJE

RALVSMV6
RA3ANJE


RABANJE
```

Figure 5-58 SMTPNJE.HOSTINFO file

Update the secure tables

When the secure option is added in the SMTP.CONFIG data set the gateway will run in secure mode and only the NJE/RSCS addresses in the SMTP security table (the SMTP.SECTABLE data set) are authorized to send and receive mail.

The format of the file is:

```
<userid> <nodeid> <nickname> <primary_nick?> <primary_mbox? >
```

Following is an example of the TCP.SMTP.SECTABLE:

```
VANDEKE     MVS03A
VANDEKE     MVS28A
VANDEKE     MVS39A
PESCHKE     MVS03A
PESCHKE     MVS28A
PESCHKE     MVS39A
TUTOR       MVS03A
TUTOR       MVS28A
TUTOR       MVS39A
VANDEKER    WTSCPOK  CHRISVM   Y Y
WOZABAL     WTSCPOK  KARLVM    Y Y
PESCHKER    WTSCPOK  ROLANDVM  Y Y
```

Figure 5-59 Sample SMTP.SECTABLE file

The nicknames CHRISVM, KARLVM and ROLANDVM are the three users that can receive mail from the gateway.

When sendmail provides mail with a destination of CHRISVM@MVS03A and MVS03A runs a gateway SMTP with this secure table CHRISVM@MVS03A is translated into VANDEKER at WTSCPOK.

The other names, without the nicknames specified, are users defined in the TCP/IP network and have a format of userid@hostname.

These nicknames are used to build the standard format of the destination in the sendmail and may also be used in the /etc/aliases.

Update alias files in the sendmail servers

The following example of /etc/aliases is a copy of the /etc/aliases in TCP/IP mvs39a. All the sendmail servers in our configuration have the /etc/aliases specified.

```
#sendmail alias file

#mandatory aliases
postmaster: root
mailer-daemon: postmaster
#other aliases

Chris03a: VANDEKE@MVS03A
Chris28a: VANDEKE@MVS28A
Chris28c: VANDEKE@MVS28C
Chris39a: VANDEKE@MVS39A
Chrisvm: CHRISVM@MVS03A  7
Rolandvm: ROLANDVM@MVS03A
Roland3a: PESCHKE@MVS03A
Rolan28a: PESCHKE@MVS28A
Rolan28c: PESCHKE@MVS28C
rolan39a: pesch e@MVS39A
root: woza
karl: karl@wtr05113.buddha.ral.ibm.com
nobody: /dev/null
```

Figure 5-60 sendmail alias file /etc/aliases

7 Is the nickname specified in the /etc/aliases in TCP/IP of MVS03C, MVS28A, MVS28C, MVS39A and MVS39B. Refer to the configuration in Figure 5-55. These TCP/IP stacks are running sendmail. This nickname is also specified in the secure table of TCP/IP stack MVS03A in the above section and is translated in the SMTP of MVS03A into VANDEKER at WTSCPOK.

5.6.2 Mail from sendmail to SMTPPROC

To issue the **`sendmail`** command we use the TSO ISHELL and in the TOOLS option we choose **2. Run shell command(SH)...** and entered the **`sendmail`** command with the options:

- -v = verbose
- Chris03a = The alias used for VANDEKE@MVS03A

```
Enter a Shell Command

Enter a shell command and press Enter.

Standard output and standard error are redirected to a temporary
file. If there is any data in the file when the shell command
completes, the file is displayed.
      sendmail -v chris03a < /u/vande e/fr39vdk
```

After the command has been issued, we receive the messages of the execution of the **`sendmail`** command due to the -v option.

In Figure 5-61 you can see that the alias chris03a is translated into VANDEKE@MVS03A. MVS39A is contacting MVS03A and delivers the mail.

```
. BROWSE -- /tmp/VANDEKE.11: 5:36.1 3171.ishell ------------- Line 00000000 Col 001 087 .
. Command ===> Scroll ===> CSR .
. ************************************ Top of Data ************************************** .
. chris03a... aliased to VANDEKE@MVS03A
. vandeke@mvs03a... Connecting to mvs03a.itso.ral.ibm.com. via esmtp... .
. 220 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 on Wed, 03 Mar 99 11: 6:12 EST .
. >>> EHLO MVS39A.itso.ral.ibm.com .
. 500 Unknown command, 'EHLO'.
. >>> HELO MVS39A.itso.ral.ibm.com .
. 250 MVS03A.itso.ral.ibm.com is my domain name. HELO from IP (9.24.104.149) .
. >>> MAIL From:<VANDEKE@MVS39A.itso.ral.ibm.com> .
. 250 OK .
. >>> RCPT To:<vandeke@mvs03a.itso.ral.ibm.com> .
. 250 OK .
. >>> DATA .
. 35  Enter mail body. End by new line with just a '.'.
. >>> . .
. 250 Mail Delivered .
. vandeke@mvs03a... Sent (Mail Delivered) .
. Closing connection to mvs03a.itso.ral.ibm.com. .
. >>> QUIT .
. 221 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 closing connection .
. ********************************** Bottom of Data ************************************ .
```

Figure 5-61 sendmail command

In the TCP/IP stack MVS03A the SMTP is contacted and the log is displayed in Figure 5-62.

```
EZA5125I IBM MVS SMTP CS V2R7 on Wed, 03 Mar 99 11:42:29 EST
EZA5461I 03/03/99 11:46:12 TCP (0) Helo Domain: MVS39A.itso.ral.ibm.com 9.24.104.149
EZA5475I 03/03/99 11:46:20 Received Note 00000001 via TCP (0) From
<VANDEKE@MVS39A.itso.ral.ibm.com> 501 Bytes
EZA5476I 03/03/99 11:46:25 Delivered Note 00000001 to VANDEKE at RA3ANJE
```

Figure 5-62 Log of T03ASMTP

The extended log in Figure 5-63 is caused by the debug option being turned on, which results in extra messages send to the log about the SMTP process.

```
EZA5547I 03/03/99 11:46:12 Processing Path String:
    <VANDEKE@MVS39A.itso.ral.ibm.com> and length = 33
EZA5503I 03/03/99 11:46:12    Processing Sender Address:
    <VANDEKE@MVS39A.itso.ral.ibm.com>
EZA5547I 03/03/99 11:46:12 Processing Path String:
    <@MVS03A.itso.ral.ibm.com:vandeke@MVS39A.itso.ral.ibm.com> and length = 58
EZA5504I 03/03/99 11:46:12    Sender Converted to:
    <@MVS03A.itso.ral.ibm.com:vandeke@MVS39A.itso.ral.ibm.com>
EZA5547I 03/03/99 11:46:13 Processing Path String:
    <vandeke@mvs03a.itso.ral.ibm.com> and length = 33
EZA5506I 03/03/99 11:46:13    Resolving Recipient Address:
    <vandeke@mvs03a.itso.ral.ibm.com>
EZA5547I 03/03/99 11:46:20 Processing Path String:
    <VANDEKE@MVS39A.itso.ral.ibm.com> and length = 33
EZA5126I ==================================================================
```

Figure 5-63 Entries in the SMTP log due to the debug option

In MVS System RA03, on the TSO command line we issued the **RECEIVE** command causing the mail to be received and added in the dataset VANDEKE.LOG.MISC. The following screen shows the received mail.

```
INMR901I Dataset ** MESSAGE ** from T03ASMTP on RA3ANJE
 Received: from MVS39A.itso.ral.ibm.com (9.24.104.149) by MVS03A.itso.ral.ibm.com
    (IBM MVS SMTP CS V2R7) with TCP; Wed, 03 Mar 99 11:46:13 EST
 Received: (from VANDEKE@localhost)
 :by MVS39A.itso.ral.ibm.com (8.8.7/8.8.7) id LAA67108881
 :for vandeke@mvs03a; Wed, 3 Mar 1999 11:46:12 -0500
 Date: Wed, 3 Mar 1999 11:46:12 -0500
 From: VANDEKE <VANDEKE@MVS39A.itso.ral.ibm.com>
 Message-Id: <199903031646.LAA67108881@MVS39A.itso.ral.ibm.com>

 testing from vandeke @ mvs39a/mvs39b to somewhere
```

5.6.3 Mail from SMTP to sendmail

Using SMTPNOTE a note is sent from a TCP/IP stack running SMTP to a user in a TCP/IP stack running sendmail, in this case from user ID VANDEKE@RA3ANJE to vandeke@mvs39a.

```
EZA5547I 03/03/99 15:23:56 Processing Path String: <VANDEKE@RA3ANJE> and length = 17
EZA5547I 03/03/99 15:24:02 Processing Path String: <vandeke@mvs39a> and length = 16
2
EZA5506I 03/03/99 15:24:02    Resolving Recipient Address: <vandeke@mvs39a>
EZA9554I * * * * * Beginning of Message * * * * *
EZA9555I Query Id:                1
EZA9556I Flags:                   0000 0001 0000 0000
EZA9516I Number of Question RRs:  1
EZA9517I Question   1: mvs39a.itso.ral.ibm.com MX (9500) IN (9507)
EZA9516I Number of Answer RRs:    0
EZA9516I Number of Authority RRs: 0
EZA9516I Number of Additional RRs: 0
EZA9557I * * * * * End of Message * * * * *
EZA5520I 03/03/99 15:24:02#  1 UDP Query Sent, Try: 1 to NS(.1.) := 9.24.104.113
EZA5521I 03/03/99 15:24:02#  1 Adding Request to Wait Queue
EZA5522I 03/03/99 15:24:02#  1 Setting Wait Timer: 10 seconds
EZA5491I 03/03/99 15:24:02    UDP packet arrived from: 9.24.104.113 229/229 bytes.
EZA9554I * * * * * Beginning of Message * * * * *
EZA9555I Query Id:                1
EZA9556I Flags:                   1000 0101 1000 0000
EZA9516I Number of Question RRs:  1
EZA9517I Question   1: mvs39a.itso.ral.ibm.com MX (9500) IN (9507)
EZA9516I Number of Answer RRs:    3
EZA9517I Answer     1: mvs39a.itso.ral.ibm.com 3600 MX (9500) IN (9507) 10 mvs28a.itso.ral.ibm.com
EZA9517I Answer     2: mvs39a.itso.ral.ibm.com 3600 MX (9500) IN (9507) 0 mvs39a.itso.ral.ibm.com
EZA9517I Answer     3: mvs39a.itso.ral.ibm.com 3600 MX (9500) IN (9507) 5 mvs03a.itso.ral.ibm.com
EZA9516I Number of Authority RRs: 1
EZA9517I Authority  1: itso.ral.ibm.com 3600 NS (9487) IN (9507) mvs03a.itso.ral.ibm.com
EZA9516I Number of Additional RRs: 6
EZA9517I Additional 1: mvs28a.itso.ral.ibm.com 3600 A (9486) IN (9507) 9.24.104.42
EZA9517I Additional 2: mvs28a.itso.ral.ibm.com 3600 A (9486) IN (9507) 192.168.252.28
EZA9517I Additional 3: mvs39a.itso.ral.ibm.com 3600 A (9486) IN (9507) 192.168.232.39
EZA9517I Additional 4: mvs39a.itso.ral.ibm.com 3600 A (9486) IN (9507) 9.24.104.149
EZA9517I Additional 5: mvs03a.itso.ral.ibm.com 3600 A (9486) IN (9507) 192.168.250.3
EZA9517I Additional 6: mvs03a.itso.ral.ibm.com 3600 A (9486) IN (9507) 9.24.104.113

EZA9557I * * * * * End of Message * * * * *
EZA5507I 03/03/99 15:24:02    MX records left: 96 mvs39a.itso.ral.ibm.com
3
EZA5547I 03/03/99 15:24:02 Processing Path String: <vandeke@mvs39a.itso.ral.ibm.com> and length = 33
EZA5502I 03/03/99 15:24:02    Enqueuing file 00000001 recipient 1 on  192.168.232.39
EZA5126I ===============================================================
EZA5125I IBM MVS SMTP CS V2R7 on Wed, 03 Mar 99 15:18:50 EST
EZA5460I 03/03/99 15:23:54 BSMTP Helo Domain: RA3ANJE You check out okay!
EZA5474I 03/03/99 15:23:56 Received Note 00000001 via BSMTP From <VANDEKE@RA3ANJE> 263 Bytes
EZA5476I 03/03/99 15:24:09 Delivered Note 00000001 to <vandeke@mvs39a.itso.ral.ibm.com>
4
```

Figure 5-64 SMTP log

In Figure 5-64 the SMTP log contains three parts:

From **1** to **2** the log contains the messages captured by the debug option.

From **2** to **3** the log shows the resolution of the addresses by the domain name server running in the TCP/IP stack MVS03A.

From **3** to **4** the log of the SMTP address space.

The mail is received by sendmail on TCP/IP stack MVS39A and a **mail** command on a TSO ISHELL shows the following screen:

```
 BROWSE -- /tmp/VANDEKE.15:24:17.782004.ishell ------ Line 00000000 Col 001 072
 Command ===>                                                  Scroll ===> CSR
******************************** Top of Data ***********************************
Message  1:
From RA3ANJE@MVS03A.itso.ral.ibm.com Wed Mar  3 15:24:09 1999
Received: from MVS03A.itso.ral.ibm.com ([192.168.233.3])
 Wed, 3 Mar 1999 15:24:08 -0500
Message-Id: <199903032024.PAA436207633@MVS39A.itso.ral.ibm.com>
Received: from RA3ANJE by MVS03A.itso.ral.ibm.com (IBM MVS SMTP CS V2R7)
   with BSMTP id 3173; Wed, 03 Mar 99 15:23:54 EST
Date:     3 Mar 99 15:22:16 LCL
From: VANDEKE@MVS03A.itso.ral.ibm.com
To: vandeke@mvs39a.itso.ral.ibm.com
Subject:  smtpnote

 send note to myself at mvs39a

Saved 1 message in "/u/vandeke/mbox"
******************************** Bottom of Data ********************************
```

Figure 5-65 SMTPNOTE received with the mail command

5.6.4 Mail from sendmail to NJE/RSCS

As mentioned earlier an alias is used to specify the destination. sendmail in our system is not set up to support the format userid%nodeid@hostname. We are using the nickname in sendmail to bypass this.

Following is the output of a **sendmail** command in a TSO ISHELL screen to send a note to a VM system via gateway SMTP:

```
1 chrisvm... aliased to 2 CHRISVM@MVS03A
CHRISVM@MVS03A... Connecting to mvs03a.itso.ral.ibm.com. via esmtp...

220 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 on Wed, 03 Mar 99 17:21:02 EST
>>> EHLO MVS39A.itso.ral.ibm.com
500 Unknown command, 'EHLO'
>>> HELO MVS39A.itso.ral.ibm.com
250 MVS03A.itso.ral.ibm.com is my domain name.   HELO from IP  (9.24.104.149)
>>> MAIL From:<VANDEKE@MVS39A.itso.ral.ibm.com>
250 OK
>>> RCPT To:<CHRISVM@MVS03A.itso.ral.ibm.com>
250 OK
>>> DATA
354 Enter mail body.  End by new line with just a '.'
>>> .
250 Mail Delivered

CHRISVM@MVS03A... Sent (Mail Delivered)   3
Closing connection to mvs03a.itso.ral.ibm.com.
>>> QUIT
221 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 closing connection
```

Figure 5-66   Mail sent from sendmail to VM

1 chrisvm is the alias specified in the /etc/aliases file of mva39a.

2 CHRISVM is a nickname in the SMTP, which will translate it in SMTP to VANDEKER at WTSCPOK.

3 Mail is delivered to the gateway SMTP, which should forward it to the NJE/RSCS network.

SMTP receives the mail and changes the destination according to the secure tables as seen in Figure 5-67.

```
03/03/99 17:21:02 Processing Path String: <VANDEKE@MVS39A.itso.ral.ibm.com> and length = 33
03/03/99 17:21:02    Processing Sender Address: <VANDEKE@MVS39A.itso.ral.ibm.com>
03/03/99 17:21:02 Processing Path String:
         <@MVS03A.itso.ral.ibm.com:vandeke@MVS39A.itso.ral.ibm.com> and length = 58
03/03/99 17:21:02    Sender Converted to:
         <@MVS03A.itso.ral.ibm.com:vandeke@MVS39A.itso.ral.ibm.com>
03/03/99 17:21:02 Processing Path String: <CHRISVM@MVS03A.itso.ral.ibm.com> and length = 33
03/03/99 17:21:02    Resolving Recipient Address: <CHRISVM@MVS03A.itso.ral.ibm.com>
03/03/99 17:21:10 Processing Path String: <VANDEKE@MVS39A.itso.ral.ibm.com> and length = 33
===================================================================
IBM MVS SMTP CS V2R7 on Wed, 03 Mar 99 15:18:50 EST
03/03/99 17:21:02 TCP (0) Helo Domain: MVS39A.itso.ral.ibm.com 9.24.104.149
03/03/99 17:21:10 Received Note 00000002 via TCP (0) From <VANDEKE@MVS39A.itso.ral.ibm.com> 494 Bytes
03/03/99 17:21:15 Delivered Note 00000002 to VANDEKER at WTSCPOK    4
```

Figure 5-67   SMTP log

4 The alias CHRISVM has been translated into VANDEKER at WTSCPOK.

Again referring to the configuration in Figure 5-55, the VM system WTSCPOK is connected via RSCS/NJE to MVS System RA03. The NJENODE name is RA3ANJE.

A RDR LIST displays the following screen:

```
TO3ASMTP OUTPUT   PUN B TO3ASMTP RA3ANJE  NONE       10 03/03    17:19:24
```

Executing the **PEEK** command for the received file in the VM reader list gives the following display:

```
 0226     PEEK     A0  V 80  Trunc=80 Size=10 Line=0 Col=1 Alt=0
File TO3ASMTP OUTPUT from TO3ASMTP at RA3ANJE Format is PUNCH.
* * * Top of File * * *
Received: from MVS39A.itso.ral.ibm.com (9.24.104.149) by MVS03A.itso.ral.ibm.com
   (IBM MVS SMTP CS V2R7) with TCP; Wed, 03 Mar 99 17:21:02 EST
Received: (from VANDEKE@localhost)
;by MVS39A.itso.ral.ibm.com (8.8.7/8.8.7) id RAA134217749
;for chrisvm; Wed, 3 Mar 1999 17:21:02 -0500
Date: Wed, 3 Mar 1999 17:21:02 -0500
From: VANDEKE <VANDEKE@MVS39A.itso.ral.ibm.com>
Message-Id: <199903032221.RAA134217749@MVS39A.itso.ral.ibm.com>

testing from vandeke @ mvs39a/mvs39b to WTSCPOK
* * * End of File * * *
```

Figure 5-68 Mail from sendmail received at VM

Sendmail to transfer MVS data sets

The files we were sending until now were files on the HFS.

An easy way to transfer sequential files or members of a partitioned data set using sendmail, is to submit the following job:

```
//VANDEKE1 JOB (SWCE),'VANDEKE',
//             NOTIFY=VANDEKE,
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*******************************************************************
//S1     EXEC PGM=IKJEFT01,DYNAMNBR=30
//OUTPUT DD PATH='/tmp/vandeke.mail.file',   5
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//INPUT  DD DISP=SHR,DSN=VANDEKE.JCL(SENDMAIL)  6
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD DATA,DLM='*/'
  OCOPY INDD(INPUT) OUTDD(OUTPUT) TEXT
*/
//********************************************************************
//S1       EXEC PGM=BPXBATCH,REGION= 4096K,TIME=NOLIMIT,
//    PARM='PGM /usr/lpp/tcpip/sbin/sendmail -v Chrisvm@mvs03a'   7
//STDOUT   DD PATH='/u/vandeke/send.report',      8
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDIN    DD PATH='/tmp/vandeke.mail.file',      9
//         PATHOPTS=(ORDONLY),
//         PATHDISP=(DELETE,DELETE)
```

Figure 5-69 Sample batch job to send MVS data sets with sendmail

5 A file /tmp/vandeke.mail.file is created.

6 The file we want to transfer with sendmail is copied into the HFS file /tmp/vandeke/mail.file.

7 sendmail is invoked and sends the contents of /tm/vandeke.mail.file to the alias chrisvm@mvs03a, who is the user VANDEKER at WTSCPOK in the VM system as seen above.

8 The log will be appended to file /u/vandeke/send.report.

```
BROWSE -- /u/vandeke/send.report -------------------------- Line 00000000 Col 001 087
 Command ===> Scroll ===> PAGE
************************************ Top of Data *************************************
Chrisvm@mvs03a... Connecting to mvs03a.itso.ral.ibm.com. via esmtp...
220 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 on Wed, 2  Feb 99 17:51:27 EST
>>> EHLO MVS28A.itso.ral.ibm.com
500 Unknown command, 'HELO'
>>> HELO MVS28A.itso.ral.ibm.com
250 MVS03A.itso.ral.ibm.com is my domain name. HELO from IP (9.24.104.42)
>>> MAIL From:<VANDEKE@MVS28A.itso.ral.ibm.com>
250 OK
>>> RCPT To:<Chrisvm@mvs03a.itso.ral.ibm.com>
250 OK
>>> DATA
35  Enter mail body. End by new line with just a '.'
>>> .
250 Mail Delivered
Chrisvm@mvs03a... Sent (Mail Delivered)
Closing connection to mvs03a.itso.ral.ibm.com.
>>> QUIT
221 MVS03A.itso.ral.ibm.com running IBM MVS SMTP CS V2R7 closing connection
*********************************** Bottom of Data ************************************
```

Figure 5-70 sendmail log of the batch job

9 The file /tmp/vandeke.mail.file created in 5 will be deleted after execution of the job.

5.6.5 Mail from NJE/RSCS nodes to sendmail

To send a note via an SMTP gateway to a TCP/IP host from an NJE/RSCS node you should create a sequential data set on MVS or VM with the proper SMTP-defined content and transmit it to the SMTP gateway.

This example was used to send a note from the WTSCPOK VM system to a user in the TCP/IP network:

```
helo mvs03a
mail from :<VANDEKER@WTSCPOK>
rcpt to:<vandeke@mvs39a>
data
Date:   March 1999, 16:19:53 ETW
From: VANDEKER@WTSCPOK
To: vandeke@mvs39a.itso.rl.ibm.com
Subject: mail from VM to sendmail
This mail is coming from WTSCPOK
Best regards
```

Figure 5-71   File to be sent from VM to sendmail

You send the file created in VM with the normal **sendfile** command:

```
Sendfile VANDEKE NOTE A to T03ASMTP 1 AT RA3ANJE 2
```

1 is the name of the SMTP gateway

2 is the name of the NJENODE

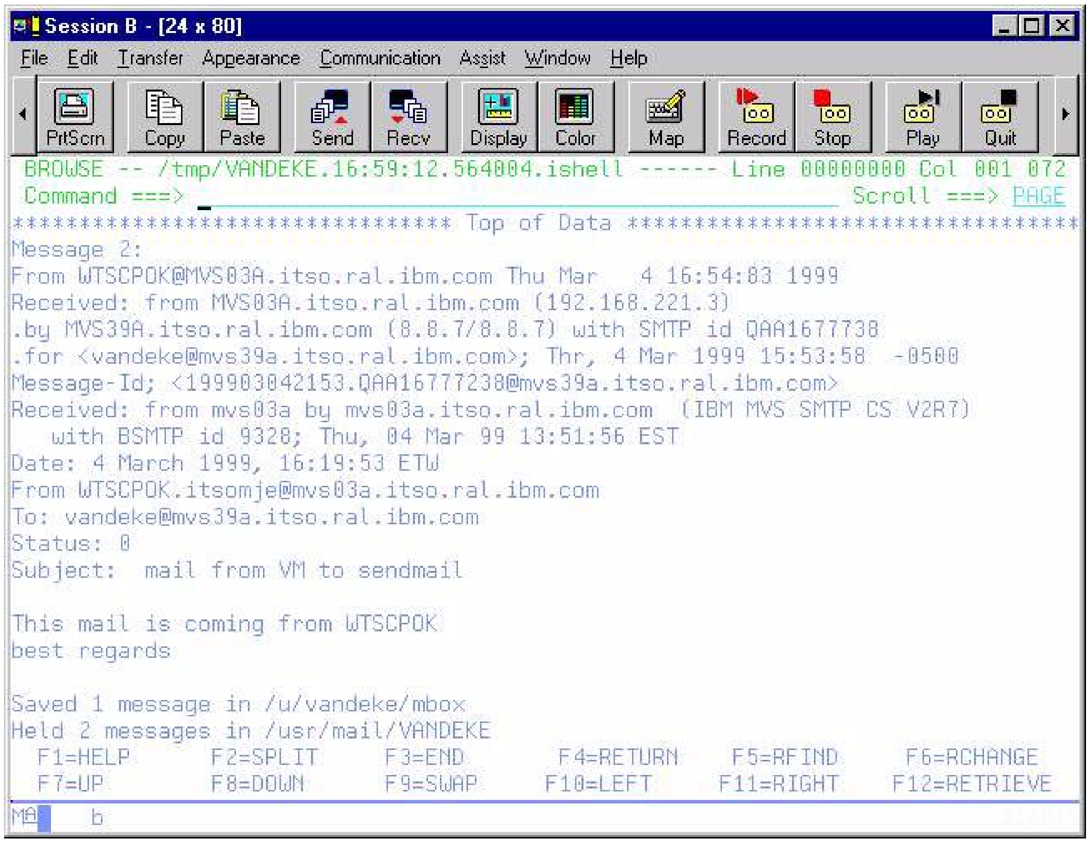
The receiver in this scenario is a sendmail in TCP/IP stack MVS28A .

In the SMTP log, the messages tell us the mail is forwarded to the destination host.

```
EZA5460I 03/04/99 10:33:26 BSMTP Helo Domain:
           mvs03a I've never heard of you!
EZA5474I 03/04/99 10:33:29 Received Note 00000002 via BSMTP From
           <VANDEKER@WTSCPOK> 301 Bytes
EZA5476I 03/04/99 10:33:39 Delivered Note 00000002 to
           <vandeke@mvs39a.itso.ral.ibm.com>
```

Figure 5-72   SMTP log of transfer from NJE to sendmail

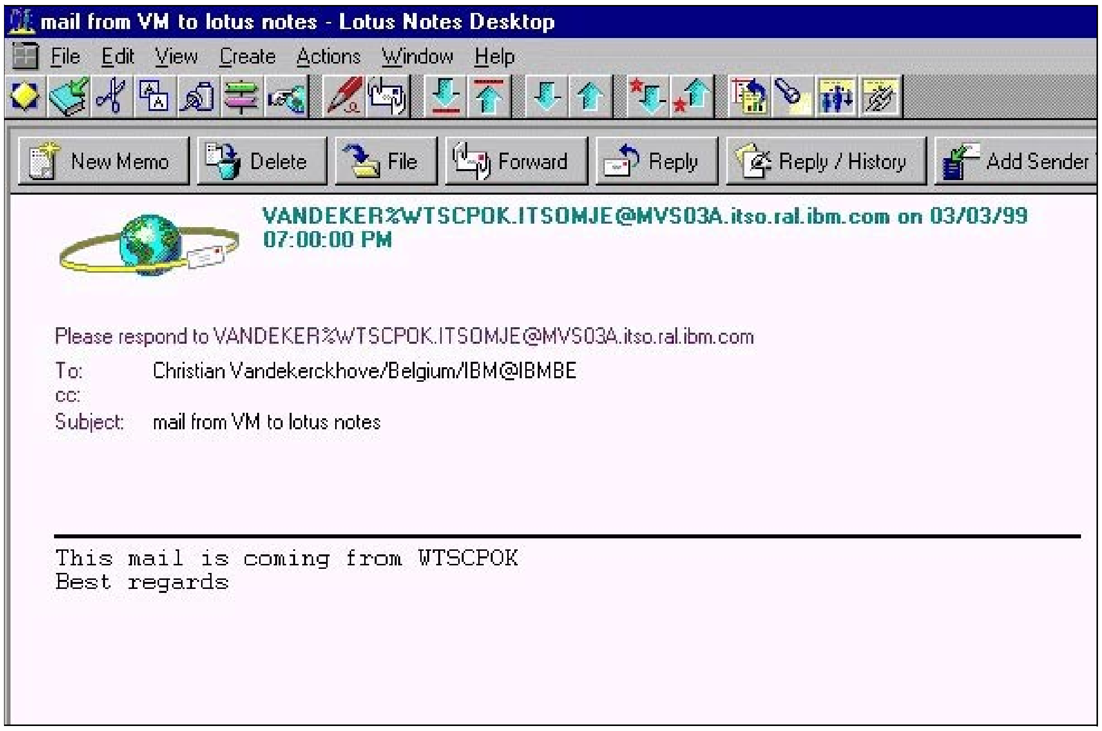
The note will have the following look when it arrives in this sendmail:


Session B - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit

```
 BROWSE -- /tmp/VANDEKE.16:59:12.564004.ishell ------ Line 00000000 Col 001 072
 Command ===> _                                               Scroll ===> PAGE
********************************* Top of Data **********************************
Message 2:
From WTSCPOK@MVS03A.itso.ral.ibm.com Thu Mar   4 16:54:03 1999
Received: from MVS03A.itso.ral.ibm.com (192.168.221.3)
.by MVS39A.itso.ral.ibm.com (8.8.7/8.8.7) with SMTP id QAA1677738
.for <vandeke@mvs39a.itso.ral.ibm.com>; Thr, 4 Mar 1999 15:53:58  -0500
Message-Id; <199903042153.QAA16777238@mvs39a.itso.ral.ibm.com>
Received: from mvs03a by mvs03a.itso.ral.ibm.com  (IBM MVS SMTP CS V2R7)
   with BSMTP id 9328; Thu, 04 Mar 99 13:51:56 EST
Date: 4 March 1999, 16:19:53 ETW
From WTSCPOK.itsomje@mvs03a.itso.ral.ibm.com
To: vandeke@mvs39a.itso.ral.ibm.com
Status: 0
Subject:  mail from VM to sendmail

This mail is coming from WTSCPOK
best regards

Saved 1 message in /u/vandeke/mbox
Held 2 messages in /usr/mail/VANDEKE
  F1=HELP       F2=SPLIT      F3=END        F4=RETURN     F5=RFIND      F6=RCHANGE
  F7=UP         F8=DOWN       F9=SWAP      F10=LEFT      F11=RIGHT     F12=RETRIEVE
MA    b
```

Figure 5-73 Mail received in MVS39A

Figure 5-73 shows a TSO OMVS session with the command **`mail`** executed and the note received.


mail from VM to lotus notes - Lotus Notes Desktop
File Edit View Create Actions Window Help
New Memo
Delete
File
Forward
Reply
Reply / History
Add Sender

VANDEKER%WTSCPOK.ITSOMJE@MVS03A.itso.ral.ibm.com on 03/03/99
07:00:00 PM

Please respond to VANDEKER%WTSCPOK.ITSOMJE@MVS03A.itso.ral.ibm.com

To: Christian Vandekerckhove/Belgium/IBM@IBMBE
cc:
Subject: mail from VM to lotus notes

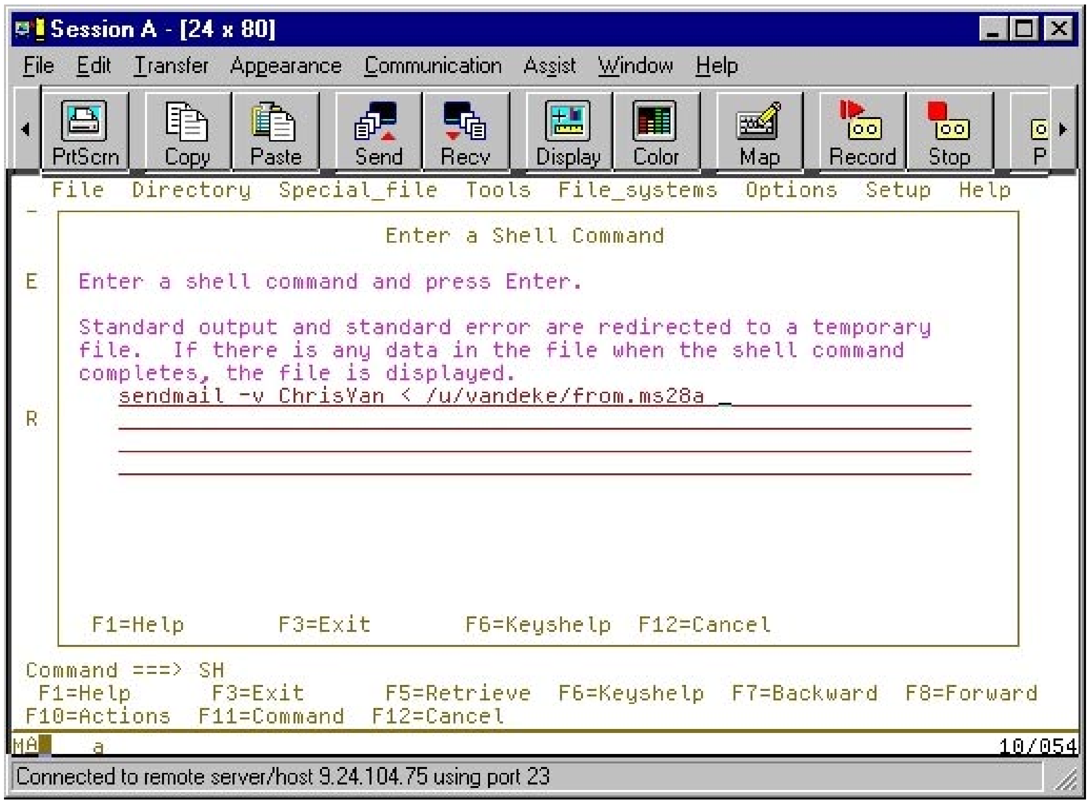
```
This mail is coming from WTSCPOK
Best regards
```

Figure 5-74 Mail received in Lotus Notes from an RSCS node

Figure 5-74 shows a Lotus Notes session and the mail received from a VM RSCS node. This mail passed through the SMTP gateway "mvs03a" and several mail servers.

5.7 sendmail and Lotus Notes

We had the possibility to send mail from MVS running sendmail to a Lotus Notes user. In Figure 5-75, a note is sent from MVS RA28, TCP/IP stack MVS28A to a Lotus Notes user.


Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop P

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
 -
                        Enter a Shell Command

 E    Enter a shell command and press Enter.

      Standard output and standard error are redirected to a temporary
      file.  If there is any data in the file when the shell command
      completes, the file is displayed.
         sendmail -v ChrisVan < /u/vandeke/from.ms28a _
 R




       F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel

 Command ===> SH
  F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
 F10=Actions  F11=Command  F12=Cancel
 MA    a                                                                10/054
```

Connected to remote server/host 9.24.104.75 using port 23

Figure 5-75 Sending mail

The DNS server running in MVS03A is the first DNS server contacted. Several domain name servers are contacted before the mail can be sent. Following are the messages issued due to the -v option in the **sendmail** command:

```
ChrisVan... aliased to Chris_Vandekerckhove@be.ibm.com
Chris_Vandekerckhove@be.ibm.com... Connecting to
            d06relay01.portsmouth.uk.ibm.com. via esmtp...  1
220 d06relay01.portsmouth.uk.ibm.com ESMTP Sendmail 8.8.7.NCA v1.8;
                Fri, 5 Mar 1999 22:45:06 GMT
>>> EHLO MVS28A.itso.ral.ibm.com
250- d06relay01.portsmouth.uk.ibm.com Hello mvs28a.itso.ral.ibm.com
               "9.24.104.42", pleased to meet you
250-8BITMIME
250-SIZE 10000000
250-DSN
250-ONEX
250-ETRN
250-XUSR
250-HELP
>>>> MAIL From:vandeke@mvs28a.itso.ral.ibm.com SIZE-48
250 <vandeke@mvs28a.itso.ral.ibm.com>... Sender ok
>>>RCPT To:<Chis_Vandekerckhove@be.ibm.com>
250 <Chis_Vandekerckhove@be.ibm.com>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself  2
>>> .
250-WAA22243 Message accepted for delivery
Chris_Vandekerckhove@be.ibm.com. Sent (WAA22234 Message accepted)
Closing connection to d06relay01.portsmouth.uk.ibm.com
>>> QUIT
221 d06relay01.portsmouth.uk.ibm.com closing connection
```

Figure 5-76 Verbose output of mail sent to Lotus Notes

1 Another domain name server is contacted to find this destination.

2 The mail is delivered to the destination.

5.7.1 Submit a job to send a mail

In the same way, a job can be submitted to send a mail to a mail server. The following job was used to transfer a small sequential data set to a Lotus Notes user.

```
//VANDEKE1 JOB (SWCE),'VANDEKE',
//              NOTIFY=VANDEKE
//              CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*****************************************************
//S1          EXEC PGM=IKJEFT01,DYNAMNBR=30
//OUTPUT   DD  PATH='/tmp/vandeke.mail.file',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//INPUT    DD  DISP=SHR,DSN=VANDEKE.NOTE
//SYSTSPRT DD   SYSOUT=A
//SYSTSIN  DD   DATA,DLM='*/'
  OCOPY INDD(INPUT) OUTDD(OUTPUT) TEXT
*/
//S1           EXEC PGM=BPXBATCH,REGION=4096k,TIME=NOLIMIT,
          PARM='PGM /usr/lpp/tcpip/sbin/sendmail -v ChrisVan' 3
//STDOUT   DD PATH='/u/vandeke/send.report',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDIN    DD PATH='/u/vandeke.mail.file',
//            PATHOPTS=(ORDONLY),
//            PATHDISP=(DELETE,DELETE)
```

Figure 5-77 Job to send a mail to a Lotus Notes user

3 The -v is the verbose option and is not necessary if you do not want all the messages. ChrisVan is the alias defined in the /etc/aliases. Using the complete name and no alias works as well.

The file that is specified in the //INPUT DD in the JCL contains the note to be sent. The first step in the JCL copies the sequential file to an HFS file with **ocopy**. The second step uses sendmail to send the contents of this hfs file to the Lotus Notes user.

Part 3

File-related applications

In this part, we describe file-related protocols such as the File Transfer Protocol (FTP) and the Network File System (NFS).

6

File Transfer Protocol (FTP)

This chapter introduces and guides you through the implementation and use of one of the most frequently used applications in the TCP/IP protocol suite. It highlights the new features introduced by CS for z/OS V1R2 IP and details their implementation.

6.1 Introduction to FTP

File Transfer Protocol (FTP) is the name of the Unix application and the protocol which allows you to transfer files in a TCP/IP network. There are two machines involved in a FTP session, namely the local host which is the client machine and a remote host which is the server. Using the FTP command and subcommands, you can sequentially access multiple hosts without leaving the FTP environment. The local host or FTP client is the TCP/IP host that initiates the FTP session. The FTP server is the TCP/IP host to which the client's session is established. This host provides the responses to the client's commands and subcommands. CS for z/OS V1R2 IP FTP, includes translation facilities for ASCII/EBCDIC translation to support host file transfer to and from a variety of host platforms and operating systems.

An FTP client connects to an FTP server using the TCP protocol. The FTP protocol requires the FTP server to use two TCP ports. One TCP port is called the control connection over which all control information, such as user ID and password, is transmitted. All FTP commands, subcommands, and responses are exchanged over this connection. Well-known port 21 is used as the default for the control connection port on the FTP server. The other TCP port is called the data connection, which is used for transferring the contents of files based on the FTP client's requests. The output of the ls or dir FTP subcommands are also sent over this data connection. Well-known port 20 is used as the default for the data connection port on the FTP server. Refer to *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780 for details about FTP usage, commands, and subcommands.

During an FTP session, it is extremely important to keep track of which machine is the client and which is the server, because this will determine whether you use a get command or a put command to move files. The **get** command is always used to copy files from the server to the client and the **put** command is used to copy files from the client to the server.

6.2 CS for z/OS IP: FTP overview

This section outlines the FTP server and client included with CS for z/OS IP. Additional information on the subject can be found in *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

6.2.1 Server and client overview

CS for z/OS V1R2 IP continues with the trend of providing only a single version of the FTP server based on the TCP/IP for MVS OpenEdition Applications Feature. Note that in CS for OS/390 V2R5 IP and later the FTP server and client exploit UNIX System Services and provides access to both traditional MVS data sets and UNIX System Services hierarchical file system(HFS) files. It supports all functions found in the C-FTP server in TCP/IP Version 3 Release 2 and replaces it. The fact that the server works with both standard MVS data sets and HFS files also means that functions from the C-FTP server, such as JES file type and SQL file type functions, are supported by the CS for z/OS IP FTP server for both standard MVS data sets and hierarchical file system files. You can run SQL queries based on SQL statements in hierarchical file system files, or you can submit MVS batch jobs written in JCL from a file in the hierarchical file system. CS for OS/390 V2R5 IP and later provides a new FTP client written in C language. The FTP client has the same functions of the Pascal FTP client that are provided in TCP/IP V3R2 for MVS, with the addition of the support of UNIX System Services hierarchical file system. The new FTP client requires the LE runtime library. The FTP client in an IP network may use CS for z/OS IP FTP server for the following major services:

- Transferring files between the client TCP/IP host and the z/OS system.
- Submitting batch jobs to the MVS job queues, displaying the status of batch jobs and retrieving the output for completed jobs.
- Submitting SQL queries to DB2 and retrieving the results of the queries.

Communications Server for z/OS IP is also delivered with translation tables supporting many different languages, including both single-byte character sets (SBCS) and double-byte character sets (DBCS). In CS for OS/390 V2R5 IP and later the FTP implementation supports the transfer of Unicode data encoded in UCS-2, which is a character encoding defined by ISO 10646. UCS-2 is a 16-bit character code capable of encoding all of the characters of the major scripts used throughout the world.

The FTP server does not use InetD as a listener process. The FTP server in CS for z/OS IP uses its own daemon (the listener) and server processes. Refer to Figure 6-1 on page 125 for an overview. The FTP server daemon is called ftpd, and is executed from the SEZALINK library of z/OS. The hierarchical file system (HFS) contains a non-executable dummy module /usr/lpp/tcpip/sbin/ftpd with the sticky bit file attribute turned on. (When the sticky bit is set on, UNIX System Services searches for the program in the user's STEPLIB, the link pack area, or the link list concatenation.) The daemon performs initialization, listens for new connections and starts a separate server process for each connection. The new server process is created for each client connection to process the client commands in the background. The server address space is started with the ftpdns program, which is also loaded from the SEZALINK library of z/OS.

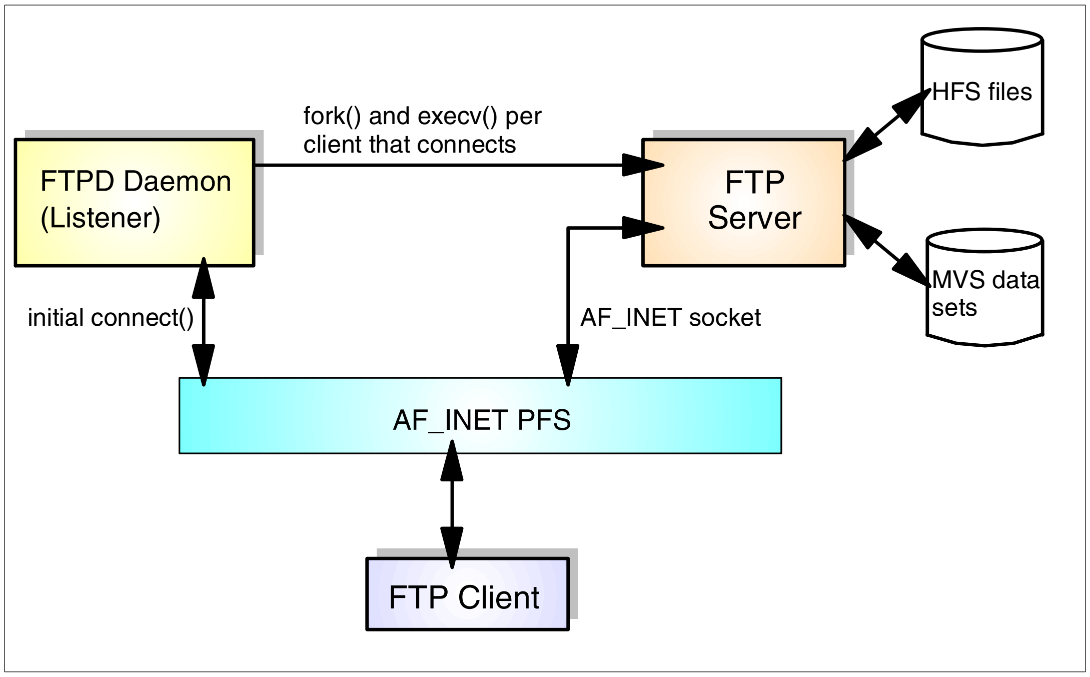


Figure 6-1 FTP server implementation overview

6.2.2 Process flow of UNIX System Services FTP

The FTP server will not complete initialization unless the CS for z/OS IP stack is *enabled* in the IFAPRDxx parmlib member. The FTP client also requires the CS for z/OS IP stack to be *enabled* in IFAPRDxx. Otherwise, client invocation ends with error message EZA2800E.

The FTP.DATA file is read when ftpd is started and the translate tables are set up in the initial address space which has access to DD statement allocation. A process is then forked to run in the background but an execv() is not done at this stage. Therefore, it is an exact copy of the original address space and has all the configuration data. When a client connects, the daemon forks a new address space to handle the client session. In that new address space an execv() is done to load the ftpdns module, and the configuration parameters are passed in a parameter list over the execv(). Refer to Figure 6-2 on page 126 for an overview.

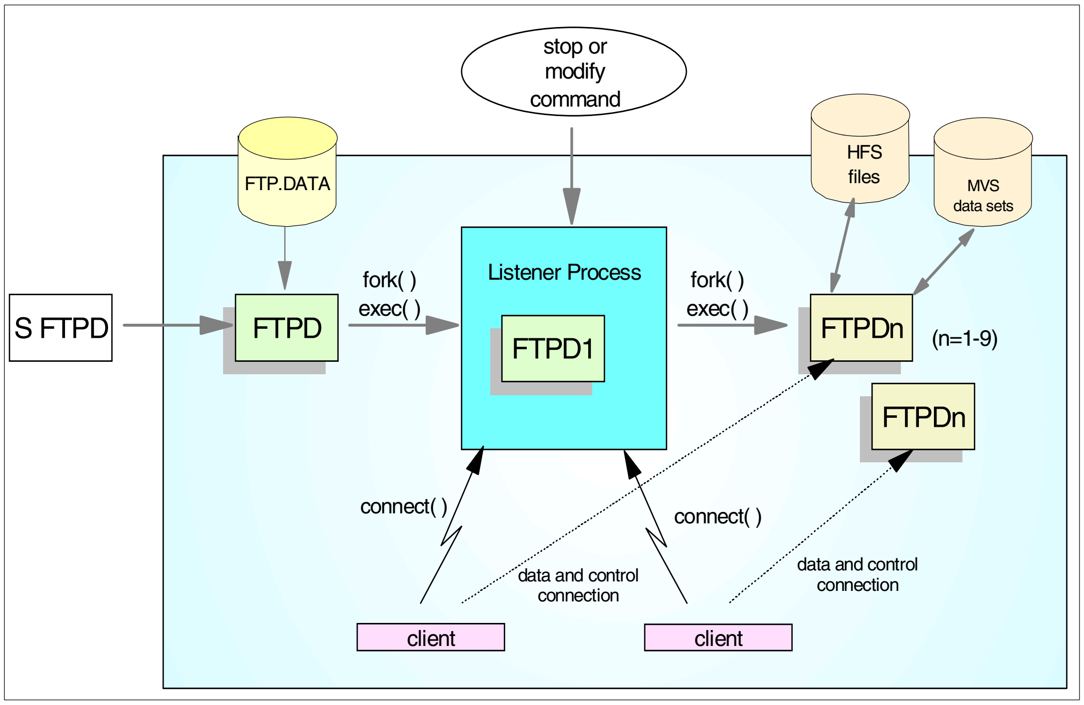


Figure 6-2 Process flow of OS/390 UNIX FTP server

When a new FTP client connects to the FTPD daemon process, the FTP daemon forks another address space that uses the execv() services to start the connection-specific server program, the ftpdns program. When the FTP daemon process forks an FTP server process, a new job name is generated by UNIX System Services. If the original job name is less than 8 characters, UNIX System Services adds a digit between 1 and 9. At TCP/IP startup this will always be a 1. So in most cases the FTP daemon address space would be named FTPD1. If your original job name is 8 characters, UNIX System Services uses the same job name for the background job. Similarly, every instance of the FTP server address space will have a name generated by UNIX System Services. Refer to *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800 for details about how job names are generated in UNIX System Services.

The FTP daemon may be registered in a DNS/WLM sysplex connection balancing group. This is necessary if FTP is to participate in a group of FTP servers in the same sysplex, which will allow clients to attach to them in a balanced manner. You may use the WLMCLUSTERNAME statement in the FTP.DATA configuration data set to specify the group name registered to a DNS/WLM sysplex group. Up to 16 different groups are supported for the FTP server.

6.2.3 SITE and LOCSITE parameters

Each operating system has unique requirements for allocating files or data sets in its file system. These requirements differ so widely between operating systems that it has been impossible to develop a single protocol that embraces all requirements from all operating systems.

In order to cover all requirements, the FTP protocol implements a **SITE** command, which enables an FTP client to send imbedded parameters to the FTP server over the control connection.

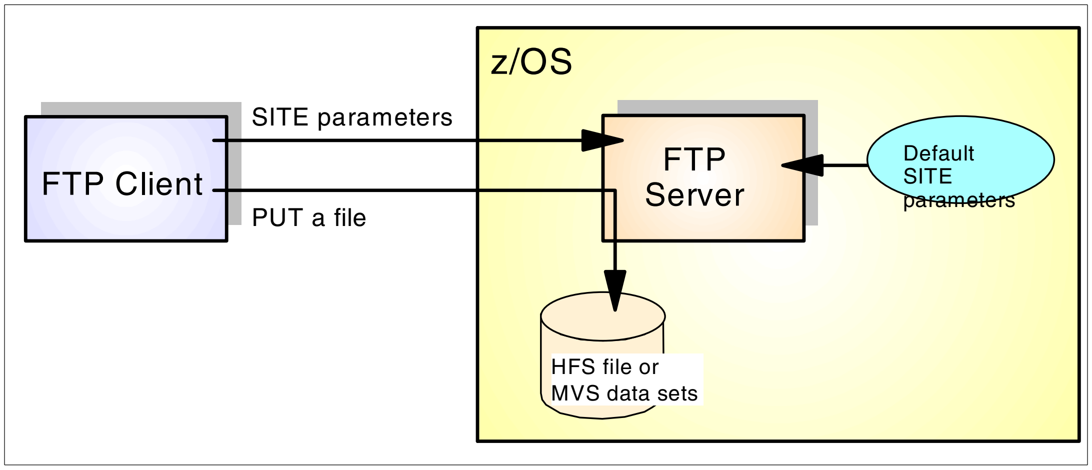


Figure 6-3 When are SITE parameters used? - z/OS FTP server

When an FTP client issues a `put` to transfer a file to the OS/390 FTP server, the FTP server needs specific parameters in order to allocate a data set. These parameters include record format (RECFM), record length (LRECL), unit-type (UNIT), and blocksize (BLKSIZE), plus many others, depending on the specific operation requested. The FTP server has a set of default values for all the parameters it may need. The client can change many of these values for the current FTP session via the **SITE** command.

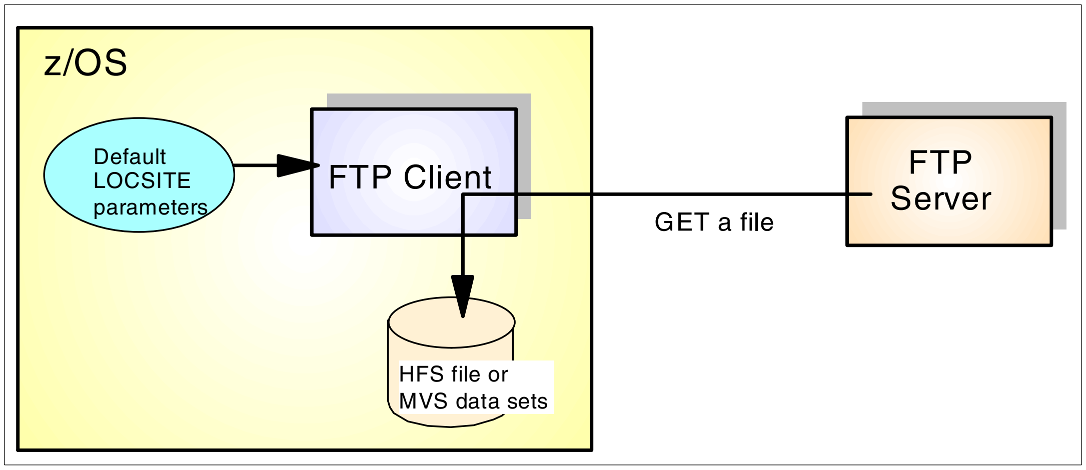


Figure 6-4 When are LOCSITE parameters used? - z/OS FTP client

If you use the z/OS FTP client function and you retrieve a file from an FTP server somewhere in your IP network, the FTP client function also needs a set of parameters similar to those of the z/OS FTP server, in order to allocate a data set in MVS. Again a set of default values exists for the z/OS FTP client, but you as a user may change these via a **`LOCSITE`** command.

You do not necessarily need to specify all the allocation attributes of an MVS data set; you may instead use the Storage Management System (SMS) of IBM Data Facility Systems Managed Storage. You have in both the **`SITE`** and the **`LOCSITE`** commands an option to specify values for the three main SMS constructs:

1. Data class (`site/locsite dataclass=`) is a collection of data set allocation attributes, for example, space requirements, record format, data set type, or retention period.
2. Management class (`site mgmtclass=`) is a collection of management attributes, for example, migration rules, backup frequency, or rules for release of unused space.
3. Storage class (`site storclass=`) is a collection of service attributes, for example, availability requirements and requested storage subsystem response time.

Consult your storage administrator for a list of available SMS constructs in your installation.

In addition to passing data set allocation parameters, the FTP implementation also uses the **`SITE`** and **`LOCSITE`** commands to enable or disable special services, which are outside the scope of the original FTP protocols, such as submitting jobs to JES or initiating DB2 queries.

6.2.4 Specification of FTP default values

Default values for data set and disk parameters can be specified using the FTP configuration data sets.

For the FTP client, the default LOCSITE parameters are specified in an installation-wide default configuration data set, or in a user-specific configuration data set. For the FTP server, the default SITE parameters are specified in an FTP server configuration data set.

Default FTP Client LOCSITE parameters

When a user on your z/OS system starts the FTP client function, a set of default local SITE parameters are in effect. The user can change these parameters during the FTP session by using the **`LOCSITE`** command.

The FTP client function searches for a configuration data set with these default parameters using the following search hierarchy.

Table 6-1 FTP.DATA search order - FTP client

TSO Environment	UNIX Environment
1. SYSFTP DD statement	1. $HOME/ftp.data
2. tso_prefix.FTP.DATA	2. userid.FTP.DATA
3. userid.FTP.DATA	3. /etc/ftp.data
4. /etc/ftp.data	4. SYS1.TCPPARMS(FTPDATA)
5. SYS1.TCPPARMS(FTPDATA)	5. tcpip_hlq.FTP.DATA
6. tcpip_hlq.FTP.DATA	

If none of these data sets is found, the FTP client will use a set of hard-coded default values for the local SITE parameters.

The FTP.DATA configuration data set may be shared between the server and the client function in z/OS. If you need to use different parameters for the server and your clients, you can make an explicit allocation in the server JCL of the server SITE parameters, and let your clients find their default local SITE parameters using dynamic allocation of either SYS1.TCPPARMS(FTPDATA) or datasetprefix.FTP.DATA.

If a user needs to use a different setup, that user may make a temporary modification by means of the `LOCSITE` command, or a permanent modification by creating a userid.FTP.DATA configuration data set. Review the hlq.SEZAINST(FTCDATA) sample member for client SITE parameters.

Note: Not all SITE parameters can be specified in the FTP.DATA file and not all parameters specified in the FTP.DATA file can be changed with SITE or LOCSITE commands.

For details about both the SITE, LOCSITE, and FTP.DATA client statement parameters, consult *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780.

Default FTP Server SITE parameters (FTPDATA)

The FTP server uses a set of default SITE parameters.

When a client connects to the z/OS FTP server, the default SITE parameters will become the actual SITE parameters for that FTP session, unless the client during the session changes them by means of an FTP client `SITE` command.

Note: Some systems do not support the `SITE` command. In order to pass the SITE parameters to MVS, the user has to enter the `QUOTE` command in front of SITE. To pass, for example, the RECFM=U parameter to MVS, the user might have to type in:

```
quote site recfm=u
```

During startup, the server will look for your default SITE parameters in an FTP configuration data set. The search order for FTP.DATA for the server follows:

The FTP.DATA search order is as follows:

1. A SYSFTPD DD-name allocation
2. jobname.FTP.DATA or userid.FTP.DATA
3. /etc/ftpd.data
4. SYS1.TCPPARMS(FTPDATA)
5. *datasetprefix*.FTP.DATA

The search step for FTP.DATA that uses a job name as high-level qualifier will use the original job name and not the UNIX System Services generated FTP daemon job name. This original job name will be used to search for both FTP.DATA and translation tables, for example FTPD.FTP.DATA and FTPD.STANDARD.TCPXLBIN. If the ftpd daemon program were started from the UNIX System Services shell, these search steps would use the user ID of the process that started the listener program or the value of the _BPX_JOBNAME environment variable. The *datasetprefix* used in the last search step comes from the UNIX System Services resolver configuration file, in our setup the SYSTCPD DD statement. If none of these data sets is found, the server will use a set of hard-coded default values. Review the hlq.SEZAINST(FTPSDATA) sample member for server site parameters. Also review the *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for details about the FTP.DATA configuration statements.

6.2.5 FTP translate tables

When either the FTP client or the FTP server is in an FTP session with z/OS is an ASCII host and you use a data type of ASCII, the data will be translated between EBCDIC and ASCII. This translation takes place on the MVS system.

In addition to single-byte ASCII/EBCDIC conversion, V2R5 FTP and later also supports DBCS and Unicode UCS-2 conversion to/from EBCDIC.

In a z/OS FTP client session, the user can select which DBCS translation to use via the DBCS subcommands of the FTP client function. Similarly, the user can issue FTP subcommands to select Unicode data transfer and conversion to/from EBCDIC.

The z/OS FTP server accepts `TYPE FTP` commands with subparameters that can specify DBCS or Unicode translation for the data transfer.

6.2.6 Supported translations

See the following chart for an overview of the DBCS translate tables that are supported:

Table 6-2 DBCS translation support

FTP Client Command	TYPE Command	Translation
BIG5	TYPE B 8	Big-5
EUCKANJI	TYPE B 2	Extended UNIX code Kanji
HANGEUL	TYPE B 5	Hangeul transfer type
IBMKANJI	TYPE F 1	IBM (EBCDIC) Kanji
JIS78JK	TYPE B 4 A	JIS 1978 Kanji using ASCII shift-in
JIS78JK (ASCII	TYPE B 4 A	ASCII shift-in esc sequence
JIS78JK (JISROMAN	TYPE B 4 R	JISROMAN shift-in esc seq.
JIS78JK (JISROMAN NOSO	TYPE B 4 R N	Pure DBCS
JIS83KJ	TYPE B 3 A	JIS 1983 Kanji using ASCII shift-in
JIS83KJ (ASCII	TYPE B 3 A	ASCII shift-in escape seq.
JIS83KJ (JISROMAN	TYPE B 3 R	JISROMAN shift-in esc seq.
JIS83KJ (JISROMAN NOSO	TYPE B 3 R N	Pure DBCS
KSC5601	TYPE B 6	Korean Standard Code
SCHINESE	TYPE B 9	Simplified Chinese
SJISKANJI	TYPE B 1	Shift JIS Kanji
SJISKANJI (SOSI	TYPE B 1 S A	Shift-out / shift-in characters X’1E’ / X’1F’

FTP Client Command	TYPE Command	Translation
SJISKANJI SOSI ASCII	TYPE B 1 S A	Shift-out / shift-in characters X’1E’ / X’1F’
SJISKANJI (SOSI EBCDIC	TYPE B 1 S E	Shift-out / shift-in characters X’0E’ / X’0F’
SJISKANJI (SOSI SPACE	TYPE B 1 S S	Shift-out / shift-in characters X’20’ / X’20’
SJISKANJI (NOSO	TYPE B 1 N	Pure DBCS
TCHINESE	TYPE 7	Traditional Chinese (5550)
UCS2	TYPE U 2	Unicode (UCS-2)

For SBCS translation, the z/OS FTP client user selects a translate table via the FTP command line keyword TRANSLATE, and the z/OS FTP server accepts a `SITE` command with a translate table name from a remote FTP client.

See the following chart for an overview of the SBCS translate tables that are provided:

Table 6-3 SBCS translation support

Table Name	Country or Region	ASCII-EBCDIC Code Page Number
AUSGER	Austrian-German	850 <-> 273
BELGIAN	Belgian	850 <-> 500
CANADIAN	Canadian	850 <-> 037
CUSTOM	Customer	819 <-> 1047
DANNOR	Danish-Norwegian	850 <-> 277
DUTCH	Dutch	850 <-> 037
FINSWED	Finnish-Swedish	850 <-> 278
FRENCH	French	850 <-> 297
ITALIAN	Italian	850 <-> 280
JAPANESE	Japanese	850 <-> 281
JPNALPHA	Japanese code	1041 <-> 1027
JPNKANA	Japanese code	1041 <-> 0290
KOR0891	Korean code	0891 <-> 0833
KOR1088	Korean code	1088 <-> 0833
PORTUGUE	Portuguese	850 <-> 037
PRC1115	P. R. China	1115 <-> 0836
SPANISH	Spanish	850 <-> 284
SWISFREN	Swiss-French	850 <-> 500
SWISGERM	Swiss-German	850 <-> 500
TAI0904	Taiwan	0904 <-> 0037

Table Name	Country or Region	ASCII-EBCDIC Code Page Number
TAI1114	Taiwan	1114 <-> 0037
UK	United Kingdom	850 <-> 285
US	United States	850 <-> 037

6.2.7 Translation tables search order

In OS/390 V2R5 IP and later, a user selects DBCS tables independently of SBCS tables. If a remote FTP client wants to transfer a mixed-mode data stream (a data stream that consists of both DBCS and SBCS data sections intermixed with shift-in and shift-out control characters), the user has to select a DBCS translation table using an `FTP TYPE` command. The user may change an SBCS translation table using a `SITE SBDATACONN` command. The selected DBCS table will be used to translate the DBCS sections, and the SBCS table will be used to translate the SBCS sections of the mixed-mode data stream.

In OS/390 V2R5 IP and later, both the FTP server and the FTP client allow the use of separate SBCS translation tables for the data connection and the control connection.

See the following table for the SBCS translation table search order for the FTP server.

Table 6-4 z/OS FTP server SBCS translation table search order

The Control Connection	The Data Connection (SBCS)
1. CTRLCONN (or CCXLATE) keyword in FTP.DATA	1. SYSFTSX DD Statement in the start procedure
2. Search order to locate a TCPXLBIN data set a. jobname.SRVRFTP.TCPXLBIN b. hlq.SRVRFTP.TCPXLBIN c. jobname.STANDARD.TCPXLBIN d.hlq.STANDARD.TCPXLBIN	2. SBDATACONN (or XLATE) keyword in FTP.DATA
3. 7-bit ASCII	3. Search order to locate a TCPXLBIN data set a. jobname.SRVRFTP.TCPXLBIN b. hlq.SRVRFTP.TCPXLBIN c. jobname.STANDARD.TCPXLBIN d.hlq.STANDARD.TCPXLBIN
4. Internal (hard-coded) 7-bit tables	4. The same conversions established for the control connection

Notes:

1. The *jobname* is the job name of the FTP cataloged procedure, not that of the FTP daemon address space.
2. The CCXLATE and XLATE statements in the FTP.DATA data set have been modified to use an environment variable.

   The *name* parameter specified at the CCXLATE statement used to be the DD name of the data set, but now it is a part of the environment variable called _FTPXLATE_name. If the environment variable exists, its value is used as the data set name.

The environment variable name must be all uppercase, although *name* parameter can be in mixed case.

If the environment variable does not exist, the FTP server looks for *hlq.name*.TCPXLBIN.

3. CCXLATE/XLATE and CTRLCONN/SBDATACONN are mutually exclusive.
4. The SITE XLATE option also has been changed, and the *name* parameter specified at the SITE XLATE option works the same way as the XLATE statement.

The FTP client resolves the SBCS translation table in the following order:

Table 6-5 z/OS FTP client SBCS translation table search order

The Control Connection	**The Data Connection (SBCS)**
with TRANSLATE dsname option	*with TRANSLATE dsname option*
1. $HOME/dsname.TCPXLBIN (UNIX System Services environment only) 2. userid.dsname.TCPXLBIN 3. hlq.dsname.TCPXLBIN	1. $HOME/dsname.TCPXLBIN (UNIX System Services environment only) 2. userid.dsname.TCPXLBIN 3. hlq.dsname.TCPXLBIN
without TRANSLATE sdname option	*without TRANSLATE sdname option*
1. CRTLCONN keyword in FTP.DATA 2. CCTRANS keyword in FTP.DATA 3. default search order a. userid.FTP.TCPXLBIN b. hlq.FTP.TCPXLBIN c. userid.STANDARD.TCPXLBIN d. hlq.STANDARD.TCPXLBIN 4. 7-bit ASCII 5. Internal (hard coded) 7-bit tables	1. SBDATACONN keyword in FTP.DATA 2. SBTRANS keyword in FTP.DATA 3. default search order a. userid.FTP.TCPXLBIN b. hlq.FTP.TCPXLBIN c. userid.STANDARD.TCPXLBIN d. hlq.STANDARD.TCPXLBIN 4. The same conversion established for the control connection 5. The same conversion established for the control connection

Notes:

1. The TRANSLATE option will result in the same SBCS translation tables both for the control and data connection.
2. When the TRANSLATE option is specified, the *hlq*.STANDARD.TCPXLBIN data set is never used. If userid.dsname.TCPXLBIN and hlq.dsname.TCPXLBIN do not exist, or if they were incorrectly created, FTP ends with an error message.
3. The CCTRANS and SBTRANS keyword in the FTP.DATA data set is still supported, but the CTRLCONN and SBDATACONN are preferred respectively, if both are present.

For DBCS translation tables used for the data connection refer to the following corresponding tables for the server and client to review the search order.

Note: As an example, we show the search order for the Japanese KANJI DBCS translation tables here.

Table 6-6 z/OS FTP server DBCS translation search order (Kanji)

1. original_jobname.SRVRFTP.TCPKJBIN
2. hlq.SRVRFTP.TCPKJBIN
3. original_jobname.STANDARD.TCPKJBIN
4. hlq.STANDARD.TCPKJBIN

Table 6-7 z/OS FTP client DBCS translation search order (Kanji)

with TRANSLATE dsname option	**without TRANSLATE dsname option**
1. userid.dsname.TCPKJBIN	1. userid.FTP.TCPKJBIN
2. hlq.dsname.TCPKJBIN	2. hlq.FTP.TCPKJBIN
3. userid.STANDARD.TCPKJBIN	3. userid.STANDARD.TCPKJBIN
4. hlq.STANDARD.TCPKJBIN	4. hlq.STANDARD.TCPKJBIN

Default SBCS tables for the z/OS FTP server may be established via the server FTP.DATA parameters CTRLCONN/CCXLATE for the control connection and SBDATACONN/XLATE for the data connection. The remote FTP client end user can change the SBCS translation tables for both the control and the data connection via a SITE CTRLCONN and SITE SBDATACONN/XLATE respectively.

Default tables for the z/OS FTP client may be established via the client FTP.DATA parameters CTRLCONN for the control connection and SBDATACONN for the data connection. The FTP client end user can override these tables when the FTP client command is entered by passing a TRANSLATE parameter on the command invocation. You also can change the SBCS translate tables used by the OS/390 FTP client for both the control and data connection via a LOCSITE CTRLCONN and LOCSITE SBDATACONN respectively.

The following is a list of SITE/LOCSITE options related to code page translation. XLATE can only be used with a **SITE** subcommand.

CTRLCONN:

Sets the ASCII/EBCDIC SBCS translation option for the FTP control connection. The default translation for the control connection is established via the FTP.DATA keyword CTRLCONN.

You have to specify the ASCII code set using the code set names that can be recognized by the iconv() functions. The FTP server and client use the EBCDIC code set of their respective locales for the control connection.

The code sets that are supported by the FTP server and client are those that are supported by the UNIX System Services iconv() functions. Please refer to *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765 for details on the supported code set conversions. If, for example, you want the FTP server to use ASCII code set IBM-850 for the control connection for your FTP session, you have to specify the following:

```
site ctrlconn=IBM-850
```

SBDATACONN:

Sets the ASCII/EBCDIC SBCS translation option for the FTP data connection.

You specify both an ASCII code page and an EBCDIC code page. If, for example, you want the FTP server to use EBCDIC code page IBM-037 and ASCII code page IBM-850, you specify the following:

```
site sbdataconn=(IBM-037,IBM-850)
```

The code pages that are supported by the FTP server and client are those that are supported by the UNIX System Services iconv() functions. Please refer to *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765 for details on the supported code set conversions.

You may also use the translation table data sets generated by the CONVXLAT utility. You do so by using the following syntax on your **SITE** command:

```
SITE sbdataconn=tcpip.ftpkana.tcpxlbin
```

Please note that you specify the fully qualified MVS data set name and not, as with the obsolete **XLATE SITE** command, a DD-name. You may also place your translation tables in the hierarchical file system and refer to a hierarchical file system file name in a **SITE SBDATACONN** command. Note that the HFS name is case-sensitive.

```
SITE sbdataconn=/u/userx/ftpkana.tcpxlbin
```

Notes:

a. The name must not be enclosed in quotes. If quotes appear, they will be treated as part of the name.

b. The `SBDATACONN` keyword must be the only keyword or last keyword on a **SITE** subcommand.

c. **SITE XLATE** and **SITE SBDATACONN** are mutually exclusive.

UCSHOSTCS:

Specifies the EBCDIC code set to be used when converting to or from Unicode.

The code sets that are supported by the FTP server and client are those that are supported by the UNIX System Services iconv() functions. You must specify the EBCDIC code set using the code set names which can be recognized by the iconv() functions. You can use this technique to transfer DBCS data encoded in Unicode to or from the DBCS EBCDIC code sets supported by the iconv() functions. If, for example, you want to use EBCDIC code page IBM-939 for the FTP server, you specify the following:

```
site ucshostcs=IBM-939
```

If you want to use the same code converter for the FTP client, you have to specify the following FTP subcommand:

```
locsite ucshostcs=IBM-939
```

If you do not use `ucshostcs` to specify a code set, the current code set of the server (or client) process will be used.

The code converter supported by the iconv() functions are listed in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227. For detailed information on the iconv() functions and supported code page, please refer to *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765.

XLATE:

Indicates the desired translation table to be used for the data connection.

This option has been modified to use an environment variable. The usage of this option is:

```
SITE XLATE=name
```

The *name* parameter specified at the option used to be the DD name of the data set, but now it is a part of the environment variable called _FTPXLATE_*name*.

If the environment variable exists, its value is used as the data set name. If it does not exist, FTP server looks for hlq.name.TCPXLBIN. The *name* of * indicates that the translation table set up at initialization for the data connection will be used.

Notes:

a. The environment variable name must be all uppercase, although the *name* parameter can be in mixed case.
b. SITE XLATE and SITE SBDATACONN are mutually exclusive.

If you use the FTP client and want to change the code pages of the client, you have to use the **LOCSITE FTP** subcommand instead of the **SITE** subcommand.

6.2.8 Selecting translation tables

The following are examples of how to select the translation tables. In all of the examples, we have configured the following statement in the TCPIP.DATA used by the FTP server and client.

```
DATASETPREFIX TCPIP
LOADDBCSTABLES SJISKANJI EUCKANJI
```

The FTP server has been started by the cataloged procedure named FTPDB and the FTP daemon address space had a UNIX System Services generated name FTPDB1.

```
ftp> quote site ctrlconn=7bit   1
200 Site command was accepted
ftp> quote site sbdataconn=garthm.ftpkana.tcpxlbin  2
200 Site command was accepted
ftp> quote type b 1           3
200-Representation type is KANJI Shift-JIS
200 Standard DBCS control used
```

Figure 6-5 Selecting translation tables

1 The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII. Since some system may not recognize the **site ctrlconn** subcommand, you would have to use the **quote** FTP subcommand.

If you specify an invalid parameter or the FTP server cannot support the specified translation table, you will see the following error message:

```
ftp> quote site ctrlconn=8bit
200-ctrlconn=8bit ignored.  Requested conversion is not supported.
200 Site command was accepted
```

Figure 6-6 site ctrlconn command on client

2 The FTP client tries to change the translation table for the data connection used by the FTP server to a customized SBCS table.

If you specify an invalid table name, the FTP server will return the following error message to the client:

```
ftp> quote site sbdataconn=tcpip.notable
200-Translate file 'tcpip.notable' not found.  SBDATACONN ignored.
200 Site command was accepted
```

Figure 6-7 site sbdataconn command on client

3 The FTP client asks the FTP server to change the current transfer type to Shift JIS Kanji. The message with the number 200 will be returned to the client, if the FTP server server loads the corresponding DBCS translation table correctly.

If the LOADDBCSTABLES statement is not specified in TCPIP.DATA allocated for the FTP server, or the proper code pages are not specified to the LOADDBCSTABLES statement, your attempt will fail and the client will receive the following message:

```
504-Type not Supported. Translation table not Loaded.
504 Type remains Ascii NonPrint
```

Figure 6-8 Type not supported error

If you set the TRACE option for the z/OS FTP server, the following trace entries will be written to the syslog file (if daemon.debug has been defined in /etc/syslog.conf) or displayed on the MVS console if syslogd is not active:

```
get_command: select rc is 1
get_command: received 20 bytes
Parse_cmd: command line: site ctrlconn=7bit.  1
site_cmd entered with 'ctrlconn=7bit'
site(): processing ctrlconn=7bit
site: freeing site arg: 'CTRLCONN'
get_command: select rc is 1
get_command: received 40 bytes
Parse_cmd: command line: site sbdataconn=kakky.ftpkana.tcpxlbin.  2
site_cmd entered with 'sbdataconn=kakky.ftpkana.tcpxlbin'
site(): processing sbdataconn=kakky.ftpkana.tcpxlbin
site: freeing site arg: 'SBDATACONN'
get_command: select rc is 1
get_command: received 10 bytes
Parse_cmd: command line: type b 1.
xtype: xtype routine entered with type 'B' dataformat '1' opt1 ' ' opt2
trying to open DBCS file //'FTPDB.SRVRFTP.TCPKJBIN'      3
fopen failed for //'FTPDB.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'TCP.SRVRFTP.TCPKJBIN'        3
fopen failed for //'TCP.SRVRFTP.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
trying to open DBCS file //'FTPDB.STANDARD.TCPKJBIN'     3
fopen failed for //'FTPDB.STANDARD.TCPKJBIN'.
                   EDC5049I The specified file name could not be located.
NC1444 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN' 3
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
NC1816 read_db_table() ... checking table header
NC1817 byte1 is 00  and  byte2 is 00
EZY2720I Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 4
```

Figure 6-9 syslogd output

1 The FTP client changes the FTP server translation table for the control connection to 7-bit ASCII.

2 The FTP client changes the FTP server translation table for the single byte data connection to a private translate table.

3 You can see the search order for the DBCS translation table. In this case, the name of the FTPD cataloged procedure is FTPDB, so FTPDB will be used as the original job name.

4 The DBCS translation table loaded by the FTP server.

Here are examples when we started FTP clients with the debug option. You can use either -d or (TRACE for this option.

```
EZA1736I FTP -d (EXIT
EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages. 1
EZY2640I Using dd:SYSFTPD for local site configuration parameters.
EZA2794I Both CCTRANS and CTRLCONN were specified. Using CTRLCONN.       2
                                      CCTRANS will be ignored.
EZA2795I Both SBTRANS and SBDATACONN were specified. Using SBDATACONN.   3
                                      SBTRANS will be ignored.
EZYFT27I Using conversion between 'IBM-850' and 'IBM-1047'
                                      for the control connection.       4
EZYFT31I Using TCP.FTPKANA.TCPXLBIN for FTP translation tables
                                      for the data connection.          5
EP4873 set_dbcs: __ipdbcs() returned 2 parms from LOADDBCSTABLES statement(s)
EP4879 main: dbcstables->__ip_dbcs_listÿ0Ã is: SJISKANJ.  Len is: 8      6
EP4879 main: dbcstables->__ip_dbcs_listÿ1Ã is: EUCKANJI.  Len is: 8      6
EZA1450I IBM FTP CS/390 296 00:10 UTC
EZA1466I FTP: using TCPIPA instead of INET
EZA1772I FTP: EXIT has been set.
EZA1456I Connect to ?
EZA1736I 9.24.104.26
```

Figure 6-10 FTP client with debug option

1 The messages used by the FTP clients are stored in message catalogs to allow message translation.

2 If both the CCTRANS and CTRLCONN keywords are specified in FTP.DATA, CTRLCONN is preferred.

3 If both the SBTRANS and SBDATACONN keywords are specified in FTP.DATA, SBDATACONN is preferred.

4 The translation used by the client for control connection.

5 The translation table used by the client for data connection.

6 The DBCS code pages specified in the TCPIP.DATA configuration data set have been allocated for this client. Each code page name is recognized by the preceding 8 characters.

If you specify the **TRANSLATE FTP** command option, the *hlq*.STANDARD.TCPXLBIN data set is never used. If the translation table search procedure using a data set name specified with this option fails, FTP ends with an error message.

```
KAKKY@/u/kakky$ftp -d 9.24.104.47 \(translate notable  1
Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
Using FTP configuration defaults.
NX0570 read_xlate_files: Unable to open /u/kakky/notable.tcpxlbin :  2
                 EDC5129I No such file or directory.
NX0613 read_xlate_files: Unable to open //'KAKKY.NOTABLE.TCPXLBIN' :
                 EDC5049I The specified file name could not be located.
NX0625 read_xlate_files: Unable to open //'TCP.NOTABLE.TCPXLBIN' :
                 EDC5049I The specified file name could not be located.
Cannot load translate table specified by TRANSLATE parameter notable 3
```

Figure 6-11 Translate table not found

1 If you issue the FTP command in the UNIX System Services shell with MVS style options, you have to precede the left parenthesis with an escape character such as the backslash(\).

2 In the UNIX System Services environment, the translation table in the hierarchical file system is the top of the table search hierarchy.

3 The FTP command stops. No more table search attempts are executed.

When you use the DBCS data transfer mode in the FTP client, the debug option of the FTP client will show you the following messages.

```
EZA1460I Command:
sjiskanji (notype  1
 PC0291 Input to parCmd is :
 PC0293 parseCmd: Number of parameters is 2
 PC0296 parseCmd: parameter 0 is sjiskanji
 PC0296 parseCmd: parameter 1 is (notype
 PC0486 fndCmd: entering with sjiskanji.
 PC0568 fndCmd: Command found is sjiskanji
 PC0305 parseCmd: fndCmd returned the cmdrecord for sjiskanji
 PC0386 parseCmd: Using primary session
 CT2205 sjiskj: routine entered with parmcount=2
 NC1017 get_client_dbcs_table() entered for lang type 1
 NC1053 trying to open DBCS file //'KAKKY.FTP.TCPKJBIN'          2
 NC1079 fopen failed for //'KAKKY.FTP.TCPKJBIN'.
               EDC5049I The specified name could not be located.
 NC1053 trying to open DBCS file //'TCP.FTP.TCPKJBIN'            2
 NC1079 fopen failed for //'TCP.FTP.TCPKJBIN'.
               EDC5049I The specified file could not be located.
 NC1053 trying to open DBCS file //'KAKKY.STANDARD.TCPKJBIN'     2
 NC1079 fopen failed for //'KAKKY.STANDARD.TCPKJBIN'.
               EDC5049I The specified file name could not be located.
 NC1053 trying to open DBCS file //'TCP.STANDARD.TCPKJBIN'       2
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1816 read_db_table() ... checking table header
 NC1817 byte1 is 00  and  byte2 is 00
 NC1061 Using Japanese translation tables in 'TCP.STANDARD.TCPKJBIN' 3
 CT0334 entering cliDBOptopns() for newftpoptformat=0, parmcount=2
 CT0711 cliDBOpt: no cmd sent to server
 PC0452 parseCmd: Using primary session.
 EZA1460I Command:
```

Figure 6-12 Debug log when using DBCS transfer mode

1 Change the data transfer mode to Japanese Shift JIS Kanji. In order to suppress the sending of the corresponding TYPE message to the FTP server which may not recognize it, the end user issues the SJISKANJI FTP subcommand with the NOTYPE parameter.

2 The debug message shows you the DBCS translation search order used by the FTP client.

3 The DBCS translation table loaded by the FTP client.

The following examples show you how to enable Unicode data transfer mode. In this example, we used the OS/2 FTP client and CS for z/OS IP FTP server.

```
ftp> quote type u 2   1
200 Representation type is UCS-2
ftp> quote site noucstrunc noucssub ucshostcs=ibm-939  2
200 Site command was accepted
ftp> put jpdata.ucs 'kakky.test.jpdata'
200 Port request OK.
125 Storing data set KAKKY.TEST.JPDATA
250 Transfer completed successfully.
local: jpdata.ucs remote: 'kakky.test.jpdata'
96 bytes sent in 0.00 seconds (0 Kbytes/s)
```

Figure 6-13   Unicode data transfer mode

1 Issue **TYPE U 2**. Receiving this **TYPE** command, the FTP server set the transfer type to UCS-2 TYPE. Since the OS/2 client cannot recognize this TYPE, you have to precede the FTP subcommand with **QUOTE**.

2 Specify the EBCDIC code set to be used for converting to/from Unicode. In this sample, we use the IBM-939 code character set. You have to specify the EBCDIC code set using the code character set name supported by the iconv() API. At the same time, we disable the substitution and truncation. If you transfer the EBCDIC data that contains DBCS characters, you should disable the truncation, or the EBCDIC data might collapse.

Verify the server status using **STAT FTP** subcommand. The following is part of the output from the **STAT** subcommand.

```
211-using Mode Stream, Structure File, type UCS-2, byte-size 8
211-TYPE U data will be converted to/from IBM-939      1
211-UCS Substitution: OFF, UCS Truncation: OFF         2
211-Byte Order: big-endian                             3
```

Figure 6-14   Partial STAT command output

1 The data encoded in Unicode will be converted IBM-939 character set.

2 Neither the UCS substitution nor UCS truncation are enabled. If you transfer SBCS/DBCS mixed data, you should disable the UCS truncation.

3 The byte order for the Unicode encoding is set to big-endian by default. You can change it to the little-endian byte order by issuing the **TYPE U 2 L** FTP subcommand.

If you want to use the z/OS FTP client in Unicode transfer type, you have to issue the **UCS2** FTP subcommand and optionally the **LOCSITE** subcommand with UCS* parameters before starting the file transfer. You can also verify the status of the FTP client with the **LOCSTAT FTP** subcommand.

6.2.9 Directory mode or data set mode

The directory mode or data set mode specifies how the output from a directory command submitted to the z/OS FTP server should look.

When you customize the z/OS FTP server, you specify the default mode in the server configuration data set (see , “Default FTP Server SITE parameters (FTPDATA)” on page 129).

The client can switch between the two modes by using the **SITE** command specifying either DIRECTORYMODE or DATASETMODE.

An MVS data set name consists of a number of qualifiers separated by periods, each qualifier having a maximum length of eight bytes.

Our normal MVS way of looking at MVS data set names is:

```
ALFREDC.DB2.CNTL
ALFREDC.DB2.OUTPUT
ALFREDC.ESA4.ISPPROF
ALFREDC.ISPF.CLIST
ALFREDC.ISPF.ISPPLIB
ALFREDC.ISPF.TEST.CLIST
ALFREDC.ISPF.TEST.WORKDSN
ALFREDC.ISPFESA.ISPPROF
ALFREDC.PRINT
ALFREDC.SPFLOG1.LIST
ALFREDC.SPFTEMP1.CNTL
```

Figure 6-15 MVS dataset name layout

However, we might also look upon the data sets in a way that more closely resembles the OS/2 or UNIX style:

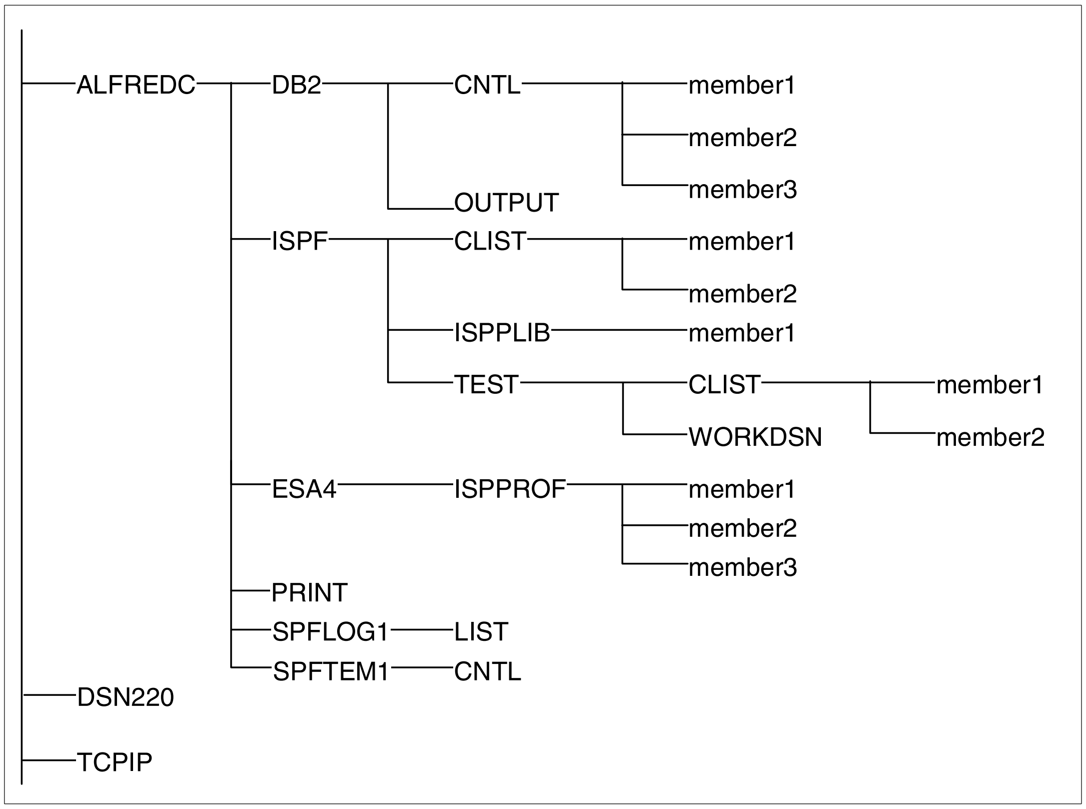


Figure 6-16 MVS data set list in an OS/2 or UNIX style

Working with FTP employs the notion of a directory and a hierarchy of directories. When MVS is the FTP server, the client still uses the directory notion and the server must transform this notion into the traditional MVS file system structure.

When you initiate an FTP session, the z/OS server will try to extract the value of your TSOPREFIX settings in RACF and use that as your default directory. If you do not maintain TSO logon information in RACF, your user ID will be used as your default directory.

Each qualifier level is considered a directory. A directory may contain data sets or subdirectories. A partitioned data set is considered a directory, and the individual members as files in that directory.

You may step down the hierarchy by using **`CD`** commands to name the next low level qualifier you want to view. You may step up to the root by using **`CD ..`** commands.

When at a given directory you issue a **`dir`** command to have the contents of the directory listed, the output you receive from the z/OS server is determined by your settings of datasetmode or directorymode.

When you use the datasetmode, the result of a **`dir`** command will look like your normal data set list output:

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  DB2.CNTL
WTLSTG 3380K  05/13/96  1    6  VB    4092  4096  PS  DB2.OUTPUT
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ESA4.ISPPROF
WTLSTG 3380K  05/13/96  1    9  FB      80  6160  PO  ISPF.CLIST
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  ISPF.ISPPLIB
WTLSTG 3380K  05/13/96  1    1  FB      80  6160  PO  ISPF.TEST.CLIST
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  ISPF.TEST.WORKDSN
WTLSTG 3380K  05/13/96  1    2  FB      80  3120  PO  ISPFESA.ISPPROF
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
WTLSTG 3380K  05/13/96  1    8  VA     125   129  PS  SPFLOG1.LIST
WTLSTG 3380K  05/13/96  1    1  FB      80   800  PS  SPFTEMP1.CNTL
250 List completed successfully.
808 bytes received in 1.3 seconds (0 Kbytes/s)
```

Figure 6-17 The directory command datasetmode

But if you use the directory mode, the results of a **dir** command would be:

```
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit  Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
Pseudo Directory                                  DB2
Pseudo Directory                                  ESA4
Pseudo Directory                                  ISPF
Pseudo Directory                                  ISPFESA
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  PRINT
Pseudo Directory                                  SPFLOG1
Pseudo Directory                                  SPFTEMP1
250 List completed successfully.
493 bytes received in 0.84 seconds (0 Kbytes/s)
```

Figure 6-18 The directory command directory mode

Then you can work down through the directory hierarchy to list the contents of each hierarchy level:

```
ftp> cd ispf
257 "'ALFREDC.ISPF.'" is working directory name prefix.
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit     Date  Ext Used Recfm Lrecl BlkSz Dsorg Dsname
WTLSTG 3380K  05/13/96  1    9  FB      80  6160  PO  CLIST
WTLSTG 3380K  05/13/96  1   10  FB      80  6160  PO  ISPPLIB
Pseudo Directory                                      TEST
250 List completed successfully.
247 bytes received in 0.44 seconds (0 Kbytes/s)
ftp> cd test
257 "'ALFREDC.ISPF.TEST.'" is working directory name prefix.
ftp> dir
200 Port request OK.
125 List started OK.
Volume Unit     Date  Ext Used Recfm Lrecl BlkSz Dsorg Dsname
WTLSTG 3380K  05/13/96  1    1  FB      80  6160  PO  CLIST
WTLSTG 3380K  05/13/96  1    1  VB     136 23476  PS  WORKDSN
250 List completed successfully.
187 bytes received in 0.34 seconds (0 Kbytes/s)
ftp> cd clist
257 "'ALFREDC.ISPF.TEST.CLIST'" partitioned data set is working directory
ftp> dir
200 Port request OK.
125 List started OK.
 Name     VV.MM  Created     Changed     Size  Init   Mod   Id
LOGON18   01.00 96/05/13 96/05/13 12:28   111   111     0 ALFREDC
LOGON20   01.00 96/05/13 96/05/13 12:29   111   111     0 ALFREDC
MYLOGON   01.04 96/05/13 96/05/13 11:29   111   101     0 ALFREDC
250 List completed successfully.
265 bytes received in 0.44 seconds (0 Kbytes/s)
ftp>
```

Figure 6-19 Moving through the directory

You can issue the **SITE DATASETMODE** or **SITE DIRECTORYMODE** FTP subcommand to set the file system mode.

6.2.10 Transfer mode, data type and data structure

At first glance, it may seem to be a trivial matter to transfer files between different computer systems, but when you take a closer look, you soon discover a range of issues originating from the diversity of computer architectures represented in a typical IP network. Some operating systems use 7-bit ASCII to represent character data, others use 8-bit ASCII or EBCDIC, just to mention the most obvious. Some operating systems organize files into records, while others treat files as continuous streams of data, in some situations without any encoded notion of record boundaries (in such a case, it will be up to the program reading or writing the data to impose a structure onto the data stream).

The FTP protocol deals to a great extent with these issues, but you, as a user of this protocol, have to select the proper options in order to let FTP transfer a file in such a way that it is usable on the receiving system.

FTP always transfers data in 8-bit bytes; this is called the *transfer size*. If the sending or receiving system uses another byte length, it is up to the FTP client and the FTP server to implement the proper conversion between local byte sizes and the FTP transfer size. When FTP transfers ASCII data, it always transfers it in 8-bit bytes, where the bits are used to encode the ASCII character according to a specific ASCII standard, which is called NVT-ASCII (Network Virtual Terminal ASCII as defined in the TELNET protocol). This implies that when you transfer ASCII type data between two ASCII hosts, a translation from the local ASCII representation to NVT-ASCII for transmission and back to the receiving hosts local ASCII representation always takes place.

When MVS is involved in an ASCII type transfer, MVS will translate the received NVT-ASCII into EBCDIC, and translate data to be transmitted from EBCDIC into NVT-ASCII.

When you request an FTP file transfer, you can characterize the transfer by means of three attributes:

TYPE The term used in the official sources is *data type*, but you may also meet the term *transfer type* or *representation type*. Data type is used to signal how the bits of the transmitted data should be interpreted by the receiver.

Data type will normally have one of the following three values:

ASCII When you set the data type to ASCII, the receiver knows that the data is character data and that each line of data is terminated via a control sequence of Carriage Control plus Line Feed (CRLF), which in ASCII is X'0D0A'. If MVS is the receiving side, data will be translated from NVT-ASCII to EBCDIC and the CRLF sequences will be removed from the data stream and substituted by traditional MVS record boundaries according to the current settings of the SITE/LOCSITE parameters: RECFM and LRECL. If RECFM is fixed, the data records will be padded with extra spaces in order to fill up a record. If MVS is the sending side, the data will be translated from EBCDIC into NVT-ASCII and, based on existing record boundaries, CRLF sequences will be constructed and inserted into the ASCII data stream.

A data type of ASCII is the default data type in all FTP implementations.

EBCDIC A data type of EBCDIC means that the data transferred is EBCDIC data. In such a case, no translation to NVT-ASCII or from NVT-ASCII takes place in MVS. The 8-bit EBCDIC bytes are transferred as they are. If you transfer text data between two EBCDIC systems, a data type of EBCDIC is the most efficient way to transfer the data. Most ASCII hosts will reject a data transfer where you specify a data type of EBCDIC. Some will treat it as an ASCII transfer, but the point where the translation takes place is at their end of the FTP connection, and not in MVS.

IMAGE A data type of IMAGE means that the data will be transmitted as contiguous bits packed into the 8-bit FTP transfer byte size. No translation takes place, neither at the sending nor at the receiving side. You will normally use this data type for binary data, such as program files. If you transfer text data between two similar ASCII hosts, it will often be more efficient to use an IMAGE data type instead of an ASCII data type. As the two systems use exactly the same ASCII representation, there is no need to impose the overhead of translating to and from NVT-ASCII.

Note: Other data types are all flavors of DBCS and Unicode (UCS-2).

Both the ASCII and EBCDIC data types have a second attribute called *format control*.

Non-print The text data does not include any vertical format control characters. This format control is the only one you will find in the MVS FTP implementation. When you set data type to ASCII, the format control defaults to Non-print.

TELNET The text data includes TELNET control characters. This is not commonly used.

CC The text data includes Carriage Control (ASA) control characters, according to the FORTRAN implementation, which only includes:

- "blank" - move up one line
- "0" - move up two lines
- "1" - move to top of form
- "+" - suppress movement

STRUCT Defines the structure of a file being transferred between systems. This option is especially important when you transfer files between systems with different views of a file system. Some are stream oriented, while others are record oriented. The possible values this option can take are:

File The file has no internal structure and is considered to be a continuous sequence of bytes. File structure can be used with all transfer modes and data types, and is the most widely implemented.

Record The file is made up of sequential records. This is a relatively simple matter to deal with as long as we talk about text files. The ASCII data type with CRLF sequences can be seen as a special case of the record data structure, which in a sense is implied by the data type of ASCII. All data types are generally supported for record structure. In CS for z/OS IP, the explicit use of record structure is only supported with stream mode transfers. When record structure is explicitly used, each record is terminated by an end-of-record (EOR) sequence, which is X'FF01'. End-Of-File (EOF) is indicated by X'FF02'. If the data contains X'FF' bytes, each X'FF' byte will be transmitted as two bytes, X'FFFF', in order to preserve the original value. In CS for z/OS IP both the FTP server and FTP client can support the record structure.

Page A third structure value exists, called page structure. It is not used in conjunction with MVS, and CS for z/OS IP does not support it, either in the FTP client or in the FTP server.

MODE *Transfer mode*, sometimes referred to as *transmission mode*, is used to indicate which kind of services should be associated with the transmission of data.

Stream Data is transmitted as a stream of bytes. The data is passed with very little or no extra processing. With stream mode, the data type is used to determine if any processing at all should be applied to the data: for example, translation of text data or CRLF processing. There is no restriction on data type or data structure. If record structure is used, the end of file will be indicated via the EOF control sequence (X'FF02'). If file structure is used, the end of the file is indicated when the sending host closes the data connection. Stream mode is the default transfer mode, and the most commonly used.

Block In block mode, you transfer data as a series of data blocks, each block preceded by a header. The header contains a count field of the number of bytes in the block (excluding the header itself) and a descriptor code, which defines block attributes (last block in the file, last block in the record or restart marker). The FTP protocols do not impose any restrictions on either data type or structure used with block mode transfers. The actual FTP implementations do, however, impose restrictions of various kinds. In CS for z/OS IP, for example, the block mode transfer is only supported with a data type of EBCDIC. You may use block mode when you transfer files between S/370 hosts. A file transferred between two MVS systems in block mode will preserve its record structure unchanged, including files with variable length records.

Compress Data is transmitted in a compressed format. The compression algorithm is rather simple. It includes the ability to send replicated bytes in a two-byte sequence (max. 128 replications), and to send a filler byte in a one-byte sequence (max. 64 filler bytes). A filler byte is defined as *space* for ASCII or EBCDIC data types and as binary zero for a data type of image. In CS for z/OS IP compressed mode requires a data type of EBCDIC.

See the following chart for an overview of the supported combinations of these three attributes in the FTP server and client implementations. It provides a cross reference between mode, type and structure.

Table 6-8 Cross reference between mode, type, and structure

	Data Type			Data Structure	
Transfer Modes	**ASCII**	**EBCDIC**	**Image**	**File**	**Record**
STREAM	yes	yes	yes	yes	yes
BLOCK	no	yes	no	yes	no
COMPRESSED	no	yes	no	yes	no

When you select among the options listed above, you have to consider the purpose of your file transfer:

1. Are you going to transfer a file to a host, where the file will be processed by programs on that host? In that case, you have to select options that will result in a file that can be used by the target host. If the data is text, the originating host uses EBCDIC and the target host uses ASCII, you have to use an ASCII data type and a stream transfer mode.
2. Are you going to transfer a file to another host for intermediate storage only and later retrieve it again on the original host? In this case it is very important that you can invert the process, so the file you will end up with back on your original host is exactly like the file you started with. If it is text data, you may not need to translate between EBCDIC and ASCII, but you can use a BINARY data type instead.

6.2.11 Stream-oriented or record-oriented

If you want to use a stream-oriented file system as intermediate storage for a record-oriented MVS file, you may have a problem, depending on the record format of the MVS data set you want to store and the data type you use.

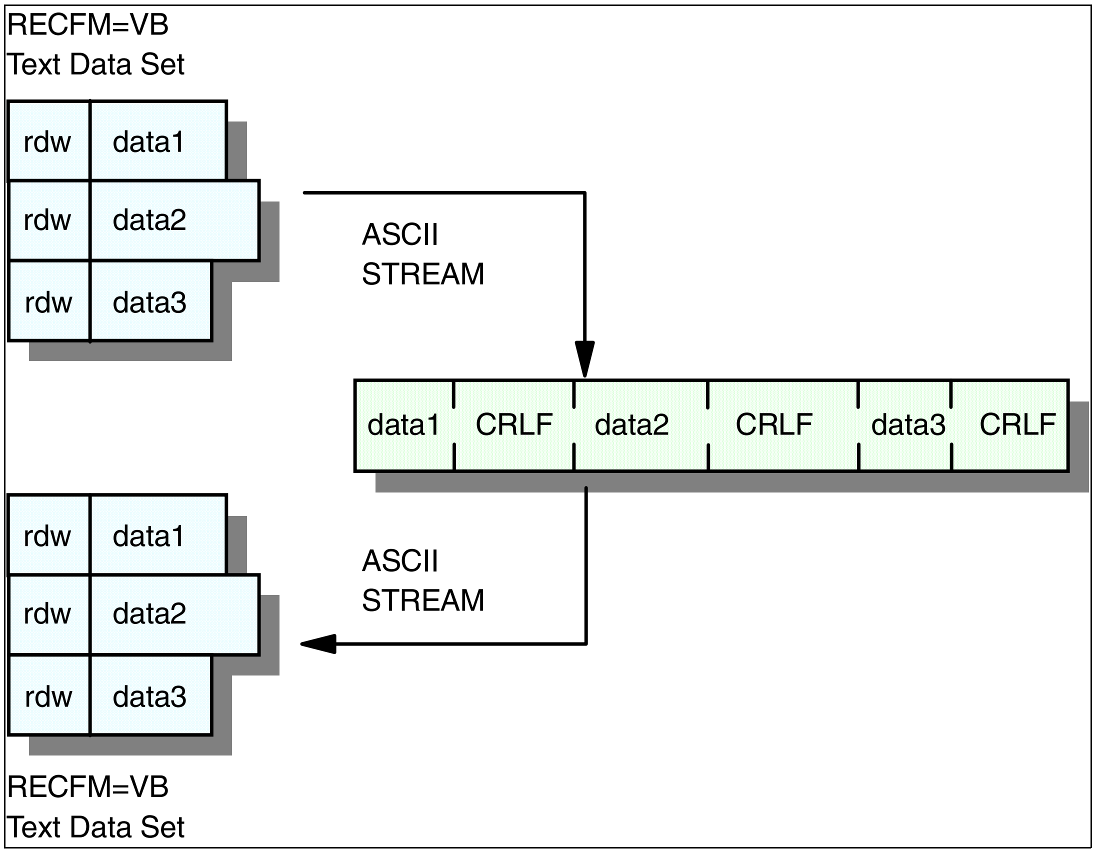


Figure 6-20 ASCII stream transfer of MVS data sets

For ASCII data types, record boundaries do not impose problems. The record boundaries are preserved by means of CRLF (Carriage Return, Line Feed - X'0D0A') for OS/2 and DOS systems or just LF (Line Feed - X'0A') for UNIX systems. If such a data set is transferred from MVS to, for example, UNIX and back to MVS again, the CRLF or LF is used to rebuild the record boundaries and the data set will be identical to the one you originally had on MVS. This is true for both fixed length and variable length record data sets.

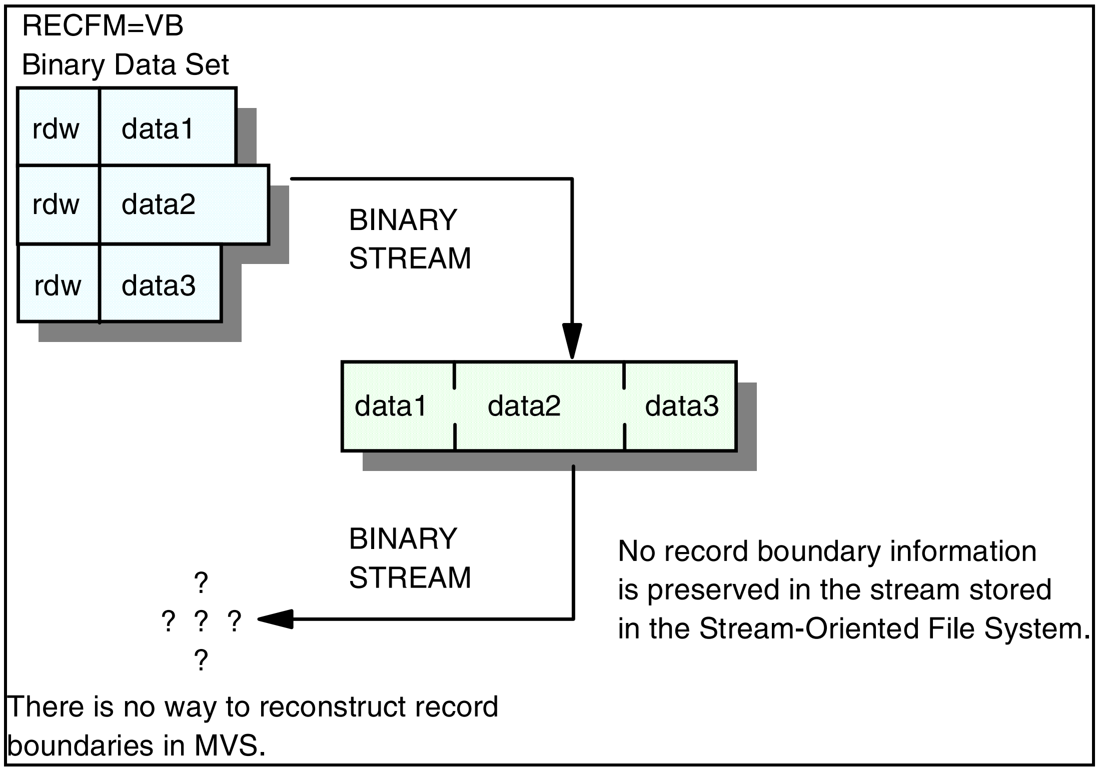

RECFM=VB
Binary Data Set
rdw
data1
rdw
data2
rdw
data3
BINARY
STREAM
data1
data2
data3
BINARY
STREAM
? ? ? ? ?
No record boundary information is preserved in the stream stored in the Stream-Oriented File System.
There is no way to reconstruct record boundaries in MVS.

Figure 6-21 BINARY stream transfer of MVS data sets with RECFM=V

For BINARY or IMAGE transfer from MVS to a stream-oriented file system, the situation is slightly more complicated.

When the records of an MVS data set are stored in a stream-oriented file system, the records are stored one after the other as one long stream of bytes, without any notion of the record boundaries from the MVS system.

If the original data set was a fixed length record data set, you can reconstruct this data set if you transfer the file back to MVS using the same logical record length as the original data set. The long stream of bytes will be chopped into records of exactly the length you specify, thereby reconstructing the same record boundaries as you had in the original data set.

If the original data set was a variable length record data set or a data set with undefined record format, you cannot use the above technique as you have no knowledge of the varying length of each record in the original data set.

There are two ways in which you can move a variable record length file out of MVS and back into MVS again, preserving the record boundaries:

1. Use the RDW option of the z/OS FTP client or z/OS FTP server.
2. Use the record structure option.

Note: We strongly recommend that you use the record structure option. This is one of the standard file structures defined in RFC959. Both the FTP server and FTP client support it.

6.2.12 Using the RDW option

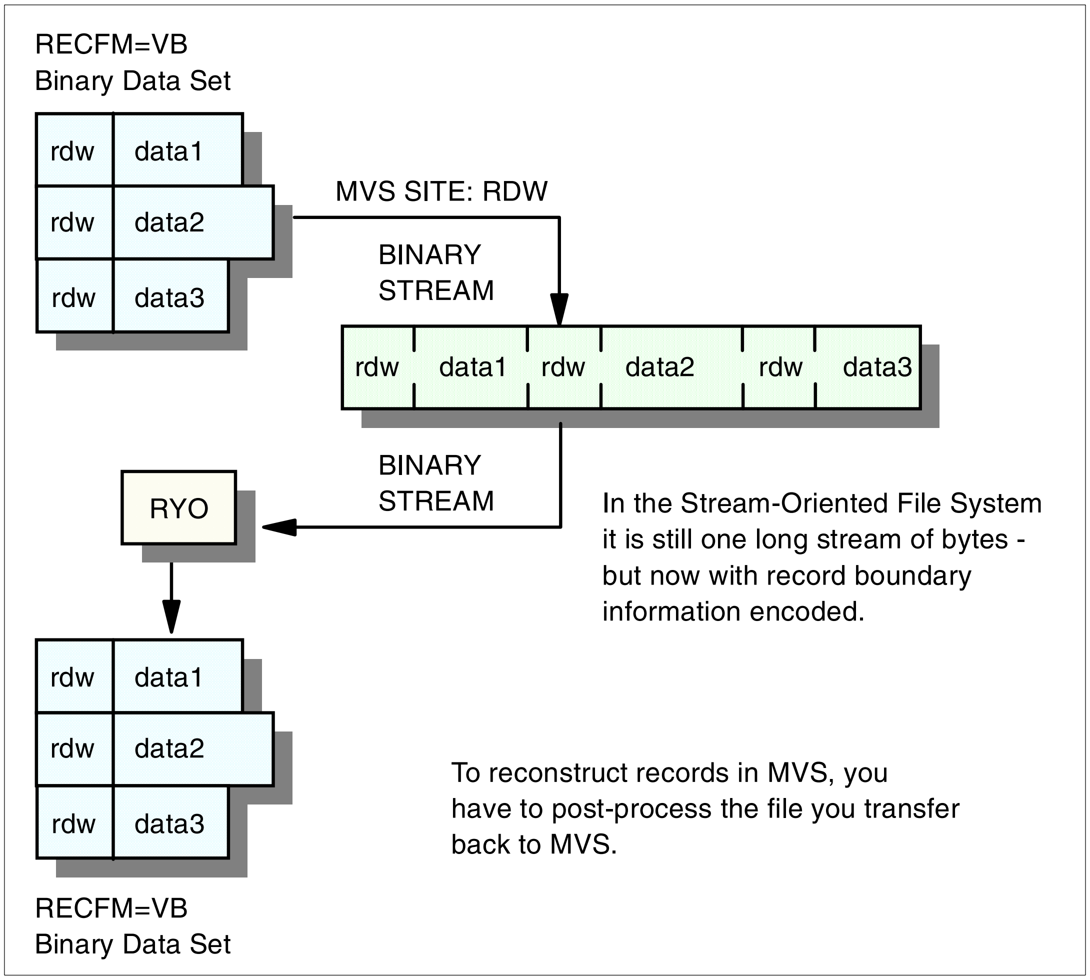

RECFM=VB
Binary Data Set
rdw
data1
rdw
data2
rdw
data3
MVS SITE: RDW
BINARY
STREAM
rdw
data1
rdw
data2
rdw
data3
BINARY
STREAM
RYO
In the Stream-Oriented File System it is still one long stream of bytes - but now with record boundary information encoded.
rdw
data1
rdw
data2
rdw
data3
To reconstruct records in MVS, you have to post-process the file you transfer back to MVS.
RECFM=VB
Binary Data Set

Figure 6-22 BINARY stream transfer of MVS data sets with the RDW SITE option

If you use the MVS FTP client or the MVS C FTP server to transfer a variable length record data set from MVS to a stream-oriented file system and you use the RDW option, the file stored on the stream-oriented file system will include the record descriptor words (RDWs) of each record in the original data set.

If the purpose of including the RDWs was to let an application program on the remote host work with the information in the file including the RDWs, you have accomplished what you wanted, but there may be situations where you might want to get such a file back into MVS again. Unfortunately, the code in CS for z/OS IP that stores the data set on MVS DASD does not use the preserved RDWs to reconstruct record boundaries. Instead, the DCB information given in either the **SITE** or **LOCSITE** command is used. You can implement a solution to this problem in the following way:

1. Use the z/OS FTP client or the z/OS FTP server to transfer a RECFM=V or VB data set to, for example, OS/2 using the BINARY, STREAM, and RDW option. This will give you the file on OS/2 with imbedded RDWs.
2. Transfer the file back to MVS using the BINARY, STREAM, and MVS SITE parameters of RECFM=U and BLKSIZE=some high value.

3. Create a little program that, based on the imbedded RDWs, reconstructs the original record structure. Check E.5, “FTP RDW post process sample program” on page 536 for the sample program written at Raleigh ITSO for testing.

Take care if you use the RDW option with ASCII transfers. Transferring the file out of MVS will work without problems, but if you later want to transfer the file back into MVS, the ASCII-to-EBCDIC translation will also translate the RDWs, which may give some interesting results.

6.2.13 Using the FTP record structure option

When you connect your FTP client to the z/OS Server, you can use the record structure option to accomplish the same results.

The following suggestion might not be the intended use of this option, but it works (at least from an AIX FTP client):

```
230 ALFREDC is logged on.  Working directory is "ALFREDC.".
ftp> binary  1
200 Representation type is Image
ftp> quote stru r  2
250 Data structure is Record
ftp> get vb vb.binary  3
200 Port request OK.
125 Sending data set ALFREDC.VB
250 Transfer completed successfully.
156 bytes received in 0.1165 seconds (1.308 Kbytes/s)
ftp> site recfm=vb lrecl=255 blksize=8192 4
200 Site command was accepted
ftp> put vb.binary vb.back  5
200 Port request OK.
125 Storing data set ALFREDC.VB.BACK
250 Transfer completed successfully.
156 bytes sent in 0.001355 seconds (112.4 Kbytes/s)
```

Figure 6-23 FTP session

1 Connect your FTP client to the z/OS FTP server and set transfer mode to binary.

2 Issue a **quote stru r** command. This command will not be interpreted by the AIX FTP client, but will be sent directly to the z/OS FTP server. The effect of this command is that the OS/390 FTP server will send data to the FTP client with imbedded end-of-record sequences.

3 Issue a **get** command to copy your variable record length file from z/OS to AIX. Because the **structure** command was sent in a quote, the AIX client does not know about it and will receive and store the file as a binary stream of bytes, including the imbedded EOR sequences.

When you want to copy the file back into MVS again, you connect your FTP client to the MVS FTP Server, set transfer mode to binary, and send the **quote** command with a **structure** command telling the MVS FTP Server to expect records with imbedded EORs.

4 Depending on your default MVS SITE parameters, you may have to send another **SITE** command with record format and record length information to MVS, before you put the file.

5 You issue a **put** command, where you put the file you received earlier. Because AIX does not know about the record structure mode it will transfer the file as a stream of bytes. The file still has the imbedded EOR sequences that will be interpreted by the MVS FTP Server, rebuilding the original record boundaries.

While the technique above can be used to transfer the VB data set in binary mode, it is still difficult to use the contents of a file at the remote system, because the file received contains imbedded EOR sequences.

The FTP client provided in OS/390 V2R5 IP and later can support record structure option. Therefore you can transfer any VB files via structure mode among MVS systems.

Here is an example of the usage of the record structure option:

```
 EZA1460I Command:
bin
 EZA1701I >>> TYPE I
 200 Representation type is Image
 EZA1460I Command:
stru r
 EZA1701I >>> STRU R    1
 250 Data structure is Record
 EZA1460I Command:

get test.vb 'kakky.test.vb1'  2
 EZA1701I >>> PORT 9,24,104,231,4,14
 200 Port request OK.
 EZA1701I >>> RETR test.vb
 125 Sending data set KAKKY.TEST.VB
 250 Transfer completed successfully.
 EZA1617I 4197 bytes transferred in 0.490 seconds. Transfer rate 8.57 Kbytes/sec.
 EZA1460I Command:
 EZA1460I Command:
```

Figure 6-24 Usage record structure option

1 Since both FTP server and client can recognize the **stru r** command, you do not need to add the **quote** command any more. Both FTP server and client recognize the file structure that is supposed to be transferred is record.

2 Issue a **get** command. The sending side (in this case the FTP server) will add EOR sequences at the end of each record and EOF to indicate the end of file. Since the receiver (in this case the FTP client) is also able to process these control sequences, the file transferred will be stored without imbedded ERO sequences.

6.2.14 New features introduced with CS for z/OS V1R2 IP

The following is a list of features introduced as part of CS for z/OS V1R2 IP.

- Restricting DIRECTORY output

  Directory output is handled in the same way as the ISPF data set list. Catalog security is provided where users cannot list data set names for which they do not have read access.

- RFC 2389 & 2640 updates

  CS for z/OS V1R2 IP implements additional standards which includes an FTP feature negotiation mechanism and FTP language Internationalization.

- Stream mode file transfer restarting

  If you are transferring a large file over an unstable network, you will keep losing the session. Restartability of a file transfer becomes a key factor in this environment. A large percentage of transfers are done between non EBCDIC nodes. These FTP users were thus left out in the cold, as restarts were only supported for users using block or compressed mode of type EBCDIC. CS for z/OS V1R2 IP has now implemented support for stream mode as per the draft RFC - Extensions to FTP. Stream mode restarts are supported for types I, A, and E.

- Socksifying of the FTP client

  Only a socksified FTP client can get past a fire wall SOCKS server. Clients in other words have to implement SOCKS protocols to enable them to connect to a SOCKS server. Now with CS for z/OS V1R2 IP you have just that, with support for both SOCKS V4 and SOCKS V5.

- Trace enhancements

  FTP server and client traces used to be to be all encompassing. This process has changed to allowing selected trace points and levels of detail.

- Native ASCII support

  Support has been added for tagging of HFS files. A file tag is an attribute of a file that is used by the ASCII Runtime Library to automatically convert between character sets. HFS files tags can thus be set, interpreted or recognized as either tagged, untagged or tagged with a binary or ecdic codepage.

- TLS enablement for FTP

  Support for Transport Layer Security (TLS) has been added to both the FTP server and client environments.

- ISPF stats enhancements

  After editing a dataset using ISPF, it updates the dataset statistics display accordingly. Changes to the member's current size, initial size, creation date, last modified date settings would reflect some form of change. In the previous versions of FTP these statistics remained untouched and thus incorrect. FTP will now create or update the ISPF stats of partitioned datasets after a stream mode data transfer.

- User level FTP server options

  The FTP.DATA configuration file provides settings and characteristics for all FTP clients that log into the server. With CS for z/OS V1R2 IP users are allowed to have user-level configuration files that are unique to each login userid. These config files set the session parameters for each client session overriding those set in FTP.DATA.

- Fending off bounce attacks using PORT commands

  This feature was included to provide the FTP server a way to protect itself from being used to attack well-known network servers. A client can attack a data-receiving server by sending a PORT command to a data-sending server with well-known port number, which allows the client to avoid having direct connection with the data-receiving server. Instructing a third party to connect to the service, rather than connecting directly, makes tracking down the attacker very difficult. In CS for z/OS V1R2 IP FTP can now restrict PORT commands.

- Surrogate RACF support

  Anonymous support can be enabled by specifying a userid and password in FTP.DATA. This really defeats the purpose of being secure. RACF surrogate support eliminates the need for passwords. The user would however include SURROGATE as a password. CS for z/OS V1R2 IP FTP would recognize this and issue a RACF call to check if the user is allowed access without a password.
- Kerberos Support for FTP client and server

  Kerberos support is implemented using the GSSAPI and Kerberos APIs provided by the z/OS Security Server (RACF).
- FTP SMF record changes

  CS V1R2 has implemented new SMF records, type 119.Both type 118 and type 119 records can be collected, although that is not recommended for performance reasons. SMF Type 119 records utilize subtypes that cannot be modified by the customer.

6.3 z/OS FTP server configuration and implementation

In this section, we outline the necessary steps to configure the FTP server. Where appropriate, we include configuration file samples.

6.3.1 Configuring the FTP server

To configure the FTP server functions, you have to perform one or more of the following tasks:

- Set up the RACF security environment.
- Update the PROFILE.TCPIP data set.
- Set up the default MVS FTP server configuration values (FTP.DATA).
- Set up configuration values, which FTP clients on your MVS system will use when they start an FTP session.
- Set up a set of available translation tables to be used for translation between ASCII and EBCDIC.
- Implement FTP server security.
- Set up the syslogd daemon for the FTP server logging.
- Prepare DB2 for the SQL query function.

In TCP/IP V3R2 for MVS you had the choice of using either a C-based MVS FTP server (called EZAFTSRV) or an z/OS UNIX System Services FTP server (called FTPD), which was introduced in TCP/IP for MVS OpenEdition Applications Feature. In CS for OS/390 V2R5 IP and later you have only one FTP server, which is based on OS/390 TCP/IP OpenEdition server and has been extended with functions earlier found in both FTP servers.

The FTP server functions consist of the following modules:

ftpd - The FTP daemon load module. It performs initialization and listens for incoming client connections. For each client connection, the daemon establishes a separate address space that runs the FTP server load module.

ftpdns - The FTP server load module. It runs in a separate address space (one for each client connection) and processes the FTP subcommands until the quit command is received.

The FTP listener program is called ftpd and the program that processes each FTP client connection is called the ftpdns program. During installation, both of these programs are installed into the /usr/sbin directory. The installation process also places copies of these two programs into your SEZALINK library.

The modules (ftpd and ftpdns) installed in the /usr/sbin directory are not executable modules, but symbolic links to dummy modules that are installed into the /usr/lpp/tcpip/sbin directory with the sticky bit set. The files in the hierarchical file system with the sticky bit set are only used to force a search through the CS for z/OS V1R2 IP SEZALINK library for the executable module. Therefore if the sticky bit is turned off, on /usr/lpp/tcpip/sbin/ftpd or /usr/lpp/tcpip/sbin/ftpdns, the process will not run. Note that there are options to change the mount point for the TCP/IP file system in a hierarchical file system environment, so the dummy modules might be installed in another directory.

6.3.2 Setting up the syslog daemon

If the syslogd daemon has been started before the FTP server, all the messages from the FTPD will be sent to syslogd and written in HFS files. The FTP server issues info, warning, and error messages. All trace entries are written with debug priority. If you configure the following statement in /etc/syslog.conf, all trace entries (and all messages) will be recorded in /tmp/daemon.logs:

```
 daemon.debug   /tmp/daemon.logs
```

An alternative is to configure the following statement in /etc/syslog.conf. Then all trace entries (and all messages) will be recorded in /tmp/ftpd.logs, thereby isolating ftpd messages that use the daemon facility name. For more information on the syslogd daemon, refer to Appendix 14, “syslogd” on page 435

```
*.FTPD*.daemon.debug   /tmp/ftpd.logs
```

If syslogd does not start before the FTP, all messages and trace entries will appear on the MVS system console. Figure 6-25 shows a started procedure for the syslog daemon which points to the output file as described above.

```
//SYSLOGD PROC
//* -----------------------------------------------------------------
//* SYSLOG Daemon
//* -----------------------------------------------------------------
//SYSLOGD EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM /usr/sbin/syslogd -f /etc/syslog.conf'
//STDIN    DD PATH='/dev/null'
//STDOUT   DD PATH='/dev/null'
//*STDENV   DD PATH='/etc/std.environment',PATHOPTS=(ORDONLY)
//* -----------------------------------------------------------------
```

Figure 6-25   STC Procedure for SYSLOGD

6.3.3 Security environment for FTP servers

In order to use the FTP server, you have to configure several security elements required by RACF or equivalent security products.

The following is an overview of the security configuration for the FTP server:

If you want to start the FTP server as an MVS started task, define the STARTED class profile with the associated user ID that has an OMVS segment with UID=0.

If you have the BPX.DAEMON facility class defined in the security database, give the user associated with the FTP started task permission to READ this facility class. Then make sure that all load modules that are loaded for the FTP server come from controlled libraries.

If you want to allow ANONYMOUS users to access your FTP server, add a user definition to the security database for user ID ANONYMO (or whatever user ID you define in FTP.DATA on the ANONYMOUS statement). see , “Anonymous user definitions” on page 239.

A user ID must have an OMVS segment in order to log on the FTP server. Therefore each login user ID must either:

- Have an OMVS segment defined in the security database.
- The OMVS default user must be established.

The terminal ID passed from FTP to RACF is an 8-byte hexadecimal character string containing an IP address. RACF interprets this as a terminal logon address and rejects it if it is not previously defined. The terminal access control is optional.

Optionally you may use four FTP server security user exits to control accesses to the FTP server. For more information on the FTP user exit, refer to 6.6.4, “FTPD server security user exit routines” on page 246.

The security considerations for the FTP server is discussed further in 6.6, “Security in the FTP environment” on page 234

6.3.4 The catalogued procedure for FTP servers

The FTP listener can be started in several ways: from a shell script, via a BPXBATCH job, or as a POSIX(ON) program. We decided to start it as a POSIX(ON) program because we wanted to control, by means of the SYSFTPD DD-statement, which FTP.DATA configuration data set the FTP server used. If you need to start only one FTP server in UNIX System Services, you may use the default FTP.DATA location in the hierarchical file system, the /etc/ftp.data file. However, if you want to start more than one instance of the FTP server and need to control the function of each of these servers via different FTP.DATA configuration options, you have to use the SYSFTPD DD-statement approach or use jobname.FTP.DATA.

The JCL we used to start our FTP server looks like the following:

```
//FTPDB  PROC MODULE='FTPD',PARMS=''  1
//*************************************************************
//*  Resulting address space name will be FTPDB1            *
//*************************************************************
//FTPDB  EXEC PGM=&MODULE,REGION=OM,TIME=NOLIMIT,  2
//        PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',
//             '"_BPX_JOBNAME=FTPDB1"',
//             '"TZ=EST")/&PARMS')
//*
//*FTPD is in TCPIP.SEZALINK (on the LINKLST)
//*FTCHKxxx routines are in the STEPLIB load library
//*STEPLIB DD DISP=SHR,DSN=WOZA.GIMLIB   3
//*
//CEEDUMP  DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(FTSD&SYSCLONE.B)   4
//SYSTCPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(TCPD&SYSCLONE.B)   5
```

Figure 6-26   FTP server sample Start Procedure

1 When we use the catalogued procedure to start the FTP listener, the resulting listener process will get a job name of this procedure name suffixed with 1 - FTPDB1. The FTP address space, which you start with an MVS operator START command, starts the FTP listener program, which does some initialization (including reading SYSFTPD) before it forks another address space, the FTPDB1 address space. The FTPDB1 address space is the address space in which the FTP listener program will execute until closed down by an MVS operator STOP command (P FTPDB1) or a UNIX System Services kill signal.

2 The FTP server daemon program is called ftpd, and will be executed from an MVS library on the system link list, the SEZALINK library. The server address space is started with the ftpdns program, which is also loaded from an MVS library on the system link list.

3 If you have developed your own FTP server security exit routines (FTCHKIP, FTCHKJES, FTCHKPWD, FTCHKCMD) or SMF exit routine (FTPSMFEX), they must be either in a library on your system link list, or you must add a STEPLIB library to this JCL procedure. As mentioned earlier, a STEPLIB allocation is the only DD-name allocation that will be passed to forked child processes. The FTP server security user exits are discussed further in 7.6.4, “FTPD server security user exit routines” on page 252.

4 In our sample setup, we use the SYSFTPD DD-statement to direct the FTP server to its FTP.DATA configuration options. The FTP.DATA configuration data sets are discussed in more detail in , “Default FTP Server SITE parameters (FTPDATA)” on page 129.

5 The FTP server gets certain operating parameters from the statement in the TCPIP.DATA data set. In our sample setup, we use the SYSTCPD DD-statement to teach the FTP server the location of this configuration data set.

The search order for TCPIP.DATA is as follows:

- The environment variable RESOLVER_CONFIG
- /etc/resolv.conf
- A SYSTCPD DD-name allocation
- jobname.TCPIP.DATA or userid.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)
- datasetprefix.TCPIP.DATA

You can see our sample configuration for the TCPIP.DATA data set as follows

```
HOSTNAME  MVS28B                   ; OS/390 IP host name
DOMAINORIGIN  itso.ral.ibm.com     ; this is appended to the host name to
                                   ; form the fully qualified host name
DATASETPREFIX TCPIP                ; default hlq for configuration data sets
MESSAGECASE MIXED                  ; case translation for the FTP server
LOADDBCSTABLES SJISKANJI EUCKANJI ; DBCS translation tables
TCPIPJOBNAME TCPIPB                ; name of the TCPIP procedure
```

Figure 6-27 Sample TCPIP.DATA data set

Notes:

1. Please remember that the library must be RACF program controlled and APF authorized. The FTPD server, in addition to using UNIX System Services functions to change the security environment, also uses standard MVS functions to check a user's authorization to standard MVS resources before a user is allowed to transfer MVS data sets in or out of your MVS system.
2. If you want to transfer files that contain DBCS data, you have to configure the LOADDBCSTABLES statement with the proper DBCS code names.
3. If you enable SQL query support, DB2 load modules are going to be loaded from your DB2 DSNLOAD library and most likely also from your DB2 DSNEXIT library. These libraries must be under RACF program control too, in order for the FTPD server address space not to become corrupted when the DB2 load modules are loaded.

```
RALTER  PROGRAM * ADDMEM('db2.version.DSNEXIT'/volser/NOPADCHK) UACC(READ)
RALTER  PROGRAM * ADDMEM('db2.version.DSNLOAD'/volser/NOPADCHK) UACC(READ)
```

Figure 6-28 RACF definitions for DB2

Using FTP servers in multiple-stack environments

In CS for z/OS IP, you can configure multiple TCP/IP stacks in a single z/OS image using the UNIX System Services C-INET feature. In C-INET configuration an application using the UNIX System Services Socket interface can get transparent access to all the TCP/IP protocol stacks configured under C-INET. When an application issues a socket/bind/listen call in a C-INET environment, the request is propagated by C-INET to all TCP/IP stacks. This application can then service clients that arrive into any of the configured TCP/IP stacks without having any awareness of this fact. This type of application is often referred to as a *generic server/daemon*.

Since FTP server can act as a generic server and communicate with C-INET PFS, the TCPIPJOBNAME statement is ignored by the FTP server. In a multiple-stack environment, the FTP server can connect to all TCP/IP stacks that are active at the time the FTP server is initialized.

The following is the output from a **`D TCPIP,,NETSTAT,CONN`** command in our test system at ITSO-Raleigh. In this sample, you will see the FTP server is listening on two TCP/IP stacks that are named TCPIPA and TCPIPB.

```
D TCPIP,TCPIPB,NETSTAT,CONN
RESPONSE=RA28
 EZZ2500I NETSTAT CS V1R2  TCPIPB 533
 USER ID  CONN     LOCAL SOCKET           FOREIGN SOCKET       STATE
 BPXOINIT 00002AD0 0.0.0.0..10007         0.0.0.0..0           LISTEN
 FTPDB1  00002B49 0.0.0.0..21  1      0.0.0.0..0           LISTEN
 TCPIPB   00000013 0.0.0.0..23            0.0.0.0..0           LISTEN
 TCPIPB   0000000C 0.0.0.0..1025          0.0.0.0..0           LISTEN
 TCPIPB   00000012 127.0.0.1..1025        127.0.0.1..1026      ESTBLSH
 TCPIPB   00000011 127.0.0.1..1026        127.0.0.1..1025      ESTBLSH
 SYSLOGD1 0000028C 0.0.0.0..514           *..*                 UDP
 7 OF 7 RECORDS DISPLAYED

D TCPIP,TCPIPA,NETSTAT,CONN
RESPONSE=RA28
 EZZ2500I NETSTAT CS V1R2  TCPIPA 537
 USER ID  CONN     LOCAL SOCKET           FOREIGN SOCKET       STATE
 BPXOINIT 0000000B 0.0.0.0..10007         0.0.0.0..0           LISTEN
 FTPDA1  00001B5A 0.0.0.0..21   1     0.0.0.0..0           LISTEN
 INETD1   0000002A 0.0.0.0..110      2    0.0.0.0..0           LISTEN
 INETD1   00000029 0.0.0.0..109      2    0.0.0.0..0           LISTEN
 INETD1   0000002B 0.0.0.0..2023     2    0.0.0.0..0           LISTEN
 OMPROUTA 00000016 127.0.0.1..1027        127.0.0.1..1028      ESTBLSH
 TCPIPA   00000011 127.0.0.1..1025        127.0.0.1..1026      ESTBLSH
 TCPIPA   0000000D 0.0.0.0..1025          0.0.0.0..0           LISTEN
 TCPIPA   00000010 127.0.0.1..1026        127.0.0.1..1025      ESTBLSH
 TCPIPA   00000014 127.0.0.1..1028        127.0.0.1..1027      ESTBLSH
 TCPIPA   00000015 0.0.0.0..23            0.0.0.0..0           LISTEN
 11 OF 11 RECORDS DISPLAYED
```

Figure 6-29   Netstat connections command

1 The listening local IP address 0.0.0.0 means that the FTP server issued the `bind()` Sockets API with local IP address INADDR_ANY which is predefined and has the value 0. Then the FTP server can receive all incoming requests to the particular server port number over all available network interfaces. This way of listening is mandatory for the socket server programs that work on a multihomed host.

2 As you can see in the sample output above, some other servers, such as the InetD and syslogd daemon, can also work as generic servers.

Although generic servers can help reduce complexity in some C-INET environments, there are also cases where the FTP server is required to connect to a single TCP/IP stack only, because of a security reason and/or to restrict the usage of servers and so on.

This can be accomplished by use of the _BPXK_SETIBMOPT_TRANSPORT environment variable. This variable can be set in the JCL for a started procedure to indicate which TCP/IP stack instance the FTP server should bind to.

The following is a sample configuration in our test environment at ITSO-Raleigh. We configured TCPIPB and FTPDB on images SC63 and SC64 both running CS for z/OS V1R2 IP. The following common started procedure was used for both FTP servers.

```
//FTPDB   PROC MODULE='FTPD',PARMS=''
//FTPDB   EXEC PGM=&MODULE,REGION=OK,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB"',   3
//            '"_BPX_JOBNAME=FTPDB1"',
//            '"TX=EST")/&PARMS')
//CEEDUMP  DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIPB.&SYSNAME..TCPPARMS(FTPDATB)
//SYSTCPD DD DISP=SHR,DSN=TCPIPB.&SYSNAME..TCPPARMS(TCPDATB)
//SYSFTSX DD DISP=SHR,DSN=TCPIPMVS.STANDARD.TCPXLBIN
```

Figure 6-30 Sample procedure for FTPDB FTP server

3 You can indicate which TCP/IP stack instance the FTP server should connect to using the environment variable.

You can verify the association between an FTP server and a TCP/IP instance using the **DISPLAY TCPIP** MVS operator command or **NETSTAT CONN** command as follows:

```
RESPONSE=SC64
 EZZ2500I NETSTAT CS V1R2 TCPIPB 421
 USER ID  CONN     LOCAL SOCKET          FOREIGN SOCKET         STATE
 DB7KDIST 00000019 0.0.0.0..33741        0.0.0.0..0             LISTEN
 DB7KDIST 00000017 0.0.0.0..33740        0.0.0.0..0             LISTEN
 FTPDB1   000002C5 0.0.0.0..21           0.0.0.0..0             LISTEN
```

Figure 6-31 SC64 NETSTAT connections display

The display shows the TCPIPB stack is associated with FTPDB1, the FTP listener. You can also issue the **D,OMVS,CINET** command. The multiple-stack environment is discussed in further detail in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

6.3.5 Starting FTP servers from the z/OS UNIX shell

You may start the FTP server as a UNIX System Services shell command or automatically when OMVS is started, rather than as an MVS cataloged procedure.

Starting the FTP daemon manually, you can issue the following command:

```
_BPX_JOBNAME='FTPD' /usr/sbin/ftpd port 20021 &
```

If you want to start FTP server when the OMVS address space is started, you have to include the following statement in the /etc/rc file:

```
export _BPX_JOBNAME='FTPD'
/usr/sbin/ftpd port 20021 &
```

The _BPX_JOBNAME environment variable specifies the job name for MVS commands. If the name is 8 characters, the UNIX System Services generated name will be the same. If it is less than 8 characters, UNIX System Services will generate a different name by adding a digit that may or may not be 1. The initialization complete message and the FTP server job name will be displayed at an MVS operator console (if syslogd is not active) or sent to the syslogd and written by it in an HFS file.

```
EZY2702I Server-FTP: Initialization completed at 09:38:39 on 02/24/98.
EZYFT41I Server-FTP: process id 33554445, server jobname FTPD8
```

Figure 6-32 FTP Server initialization

In this case, the FTP daemon was named FTPD8.

If FTP is started from the shell without _BPX_JOBNAME, the UNIX System Services generated name comes from the underlying user ID. For example, if you are using USER1, you will get a job name USER1x.

Under prior versions of CS for z/OS IP, if TCP/IP initialization is not completed before FTP is started, the FTP server will be unable to create a socket. The FTP server will continue to try every minute forever until TCP/IP initialization completes, at which time FTP initialization can complete. FTP will not recognize the **STOP** command at this phase of its initialization, so you have to use the **CANCEL** command to stop it.

However, we recommend using the AUTOLOG statement, and starting up the FTP server when the TCP/IP address space is started.

If you start the FTP server from an UNIX System Services environment, you might face several security violation errors. For security considerations, refer to “Security in the FTP environment” on page 234 and *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

6.3.6 PROFILE.TCPIP for FTP servers

If you want the FTP server to be started automatically when the TCP/IP address space is started, then include the name of the member containing the FTP server cataloged procedure in the AUTOLOG statement. To reserve ports 21 and 20 for the FTP server you have to configure the PORT statement too.

The following is our sample configuration for PROFILE.TCPIP.

```
AUTOLOG 1
 ....
 FTPDB   JOBNAME FTPDB1     ; FTP Server Listener  1
 ....
ENDAUTOLOG

PORT
 ....
 20 TCP OMVS      NOAUTOLOG ; FTP Server data port
 21 TCP FTPDB1              ; FTP Server control port  2
```

Figure 6-33 Sample TCPIP.PROFILE configuration for an FTP server

1 The automatic server activation is done by a separate subtask, called the AUTOLOG subtask. This subtask is started at TCP/IP initialization and runs in the TCP/IP address space until the address space is terminated. The AUTOLOG subtask also monitors the procedure. The monitoring function performed is:

- Check if the procedure's address space is still active. If not, restart the procedure.

- For each PORT entry for this procedure that has an AUTOLOG statement keyword, check to make sure each PORT is still active. If there is one or more PORT entry that is not active, the procedure will be restarted.

To avoid needless attempts for the AUTOLOG subtask to restart the FTP server, you need to let it know the correct FTP listener name. Therefore in the AUTOLOG statement, the JOBNAME parameter was added.

If the procedure name for FTP startup is less than 8 characters, you need to provide the UNIX System Services generated job name on the AUTOLOG statement and the PORT statement must specify that job name. If the procedure name is 8 characters, the UNIX System Services generated job name will be the same 8 characters and the JOBNAME parameter does not need to be specified on the AUTOLOG statement.

2 On the PORT statement you should not reserve port 20, which is used for the data connection port, for a particular job name. Instead, you should reserve port 20 for the general job name OMVS because the data connection is used by the forked FTP server processes that have UNIX System Services generated job names, such as FTPDB2 or FTPDB3. If you reserve port 20 for the FTP listener program (in this case FTPDB1), the forked FTP server process (for example, FTPDB2) cannot bind a socket to port 20 and the attempt at opening a socket will fail.

In this case the following error message will be displayed at an MVS operator console or sent to the syslogd and written in an HFS file if daemon.debug is active.

```
SR0388 data_connect: bind() error (111/744C7246) - EDC5111I Permission denied.
```

The client will see the following message:

```
425 Unable to open data connection
```

However, there remain a few ways to reserve both the control and data connection for a particular job name of the FTP server.

One way to do this is by using the 8-character original job name for the FTP server. In this case, all FTP server-related address spaces, including the FTP daemon address spaces, have the same job name. You would see several address spaces running concurrently with the same job name. In this case, to stop the FTP server you just need to issue the **STOP** command with the 8-character job name. Since existing FTP connections will not be terminated, the currently active FTP sessions can continue to work with the z/OS FTP server until a **QUIT** subcommand is entered.

The other way is to reserve the data connection port for all possible FTP server job names. As we mentioned before, the UNIX System Services generated names consist of the original job name plus a one-digit character that is selected from 1 to 9. Therefore by adding eight more PORT statements with all possible job names, we can reserve the data connection port for the particular FTP server.

You will see the sample configuration in Figure 6-34. In this sample, we configured two FTP servers. One is the FTP server with an 8-character name and work on non-standard port numbers. The other one is the same server as shown before, but in the PORT statement the data connection port is reserved for all possible names of the FTP server address spaces

```
AUTOLOG
    ........
    FTPDB  JOBNAME  FTPDB1
    T28FTPSV                3
    ........
ENDAUTOLOG

PORT
    .. ... ........
    20   TCP FTPDB1  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB2  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB3  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB4  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB5  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB6  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB7  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB8  NOAUTOLOG  ; FTP Server1 data port 4
    20   TCP FTPDB9  NOAUTOLOG  ; FTP Server1 data port 4
    21   TCP FTPDB1             ; FTP Server1 control port
    .. ... ........
 20020   TCP T28FTPSV  NOAUTOLOG ; FTP Server2 data port    5
 20021   TCP T28FTPSV            ; FTP Server2 control port 5
    .. ... ........
```

Figure 6-34 Sample TCPIP.PROFILE configuration for an FTP server

3 This cataloged procedure for the FTP server has an 8-character job name, so you do not need to specify a JOBNAME statement with the name of the FTP daemon address space.

4 Reserve data connection port number for all possible job names of the FTP server address spaces. You have to specify the NOAUTOLOG parameter as well.

5 The PORT statement for the FTP server with the 8-character original job name. Since all of the FTP daemon and server processes have the same job name, you can specify the same job name as that in the AUTOLOG statement.

6.3.7 SMF records

With CS for z/OS V1R2 IP two smf record types are available, namely the type 118 record and a type 119 record. All the existing events are recorded via both the Type 118 and the Type 119 records. It is possible to collect both types if desired. This however isn't recommended, due to the performance overhead of generating two records containing largely the same data. Subsequent releases of CS for z/OS IP will see the Type 118 record being retired permanently. The FTP server uses SMF type 118 (X’76’) and type 119 (X’77’) records to record transactions made by the FTP server.

The type 118 record

The **SMFCONFIG** statement is used in TCPIP.PROFILE to provide SMF logging for type 118 records, with standard subtypes. At least one of the SMF subtype statements (SMF, SMFAPPE, SMFDEL, SMFLOGN, SMFREN, SMFRETR, or SMFSTOR) must be coded in FTP.DATA to enable logging.If no subtypes have been specified, the server will not write any SMF records. Using the SMFPARM statement provides similar capability but requires the installation to select subtype numbers to enable SMF logging.

By entering options into the server configuration data set, you control in what situations you want the server to write SMF records.

The following FTP.DATA options are used to enable SMF recording for the listed events. `nn` is used to associate individual SMF record 118 subtypes to each event. You can use any value between 1 and 255. You should ensure that the values you select for your FTP SMF record subtypes are unique within SMF record type 118. Other components of TCP/IP may have been customized to write SMF records too, for example, TELNET.

SMFAPPE nn	Append
SMFDEL nn	Delete or Mdelete
SMFRETR nn	Get or Mget
SMFSTOR nn	Put or Mput
SMFREN nn	Rename

These records can be selected for all file types, or for one or more of the supported file types:

SEQ	Sequential files: enabled by default, when the above options are specified.
JES	JES spool files: FTP.DATA option: *SMFJES*.
SQL	SQL query files: FTP.DATA option: *SMFSQL*.

In addition to these SMF records, you may request that the MVS FTP server write an SMF record whenever an invalid logon attempt takes place (FTP.DATA option: *SMFLOGN*). All SMF records have predefined standard values. You may use the SMF STD statement to set the default SMF record subtype for all SMF records. SMF STD will do two things: Activate all FTP's SMF events and set the subtypes to the standard subtype numbers. Alternatively you can specify STD on the individual SMF* statements to activate recording for selective events using standard subtypes.

You are able to implement a user exit, which will be given control every time the FTP server is about to write an SMF record. The exit receives the SMF record that is about to be written and can either modify the record or suppress writing of the SMF record.

You enable FTP client SMF records by specifying SMFCONFIG FTPCLIENT in the PROFILE.TCPIP configuration data set. SMF records of type 118 and subtype 3 are created when a user invokes the FTP client commands. While the SMFPARMS statement in PROFILE.TCPIP is still supported, we strongly recommend migrating to SMFCONFIG in order to standardize subtypes. If SMFPARMS is encountered after an SMFCONFIG statement, an error message is displayed and the SMFPARMS parameters are ignored. If SMFCONFIG is encountered after an SMFPARMS statement, a warning message is displayed and the SMFCONFIG parameters are accepted.

The Type 119 record

As part of the type 119 record, the FTP Server Transfer Complete records have been reformatted to utilize one subtype value, and to include a "self defining section" and TCP Stack Identification section.

- The FTP Server record has additional optional sections:

  Hostname

  Associated MVS or HFS filename(s)

  - Some of the other information included is:

    Operation type

    Transmission start and end time and date

    Local and remote IP addresses and port numbers, for data and control connections

    Inbound/outbound byte data counts

    File transmission characteristics

- The FTP Login Failure record has been reformatted to include a "self defining section" and the TCP Stack Identification section.
  - Some of the other information included is:

    User ID attempting the login

    Local and remote IP addresses and port numbers, for data and control connections

    Login failure reason code

- The FTP Client Transfer Complete record has been reformatted to include a "self defining section" and the new TCP Stack Identification section.
  - FTP Client record has additional optional sections:

    Associated MVS or HFS data set name

    SOCKS Server information

  - Some of the other information included is:

    Operation type

    Transmission start and end time and date

    Local and remote IP addresses and port numbers, for data and control connections

    Inbound/outbound byte data counts

    File transmission characteristics

Type 119 SMF records include all those available under type 118, but additionally include IFSTATISTICS, PORTSTATISTICS, TCPSTACK, and UDPTERM.

Type 119 records may be configured in FTP.DATA using the TYPE119 keyword following the command as follows.

- SMF TYPE119 requests that type 119 SMF FTP records be created for all file transfer operations.
- SMFAPPE TYPE119 requests that type 119 records be created for append operations.
- SMFDEL TYPE119 requests that type 119 records be created for delete operations.
- SMFLOGN TYPE119 requests that type 119 records be created when a logon failure occurs.
- SMFREN TYPE119 requests that type 119 records be created for rename operations.
- SMFRETR TYPE119 requests that type 119 records be created for retrieve operations.
- SMFSTOR TYPE119 requests that type 119 records be created for store and store unique operations.

The three FTP.DATA statements SMFEXIT, SMFJES, and SMFSQL control FTP's SMF behavior as follows:

- SMFEXIT controls whether the FTP user exit, FTPSMFEX, is called when writing a SMF FTP record. The SMFEXIT statement however affects only type 118 records.
- SMFJES causes type 118 transfer completion records to be issued for transfers of JES files. SMFJES TYPE119 does the same for type 119 records.
- SMFSQL causes type 118 transfer completion records to be issued for transfers of SQL files. SMFSQL TYPE119 does the same for type 119 records.

Layout of the SMF records is documented in *z/OS V1R2.0 CS: IP Configuration Reference,* SC31-8776.

6.4 Server customization and usage

We continue our exploration of the FTP server by illustrating basic customization procedures.

6.4.1 Users of the FTP server

The FTP server sets the initial working directory at the server to one of the following:

- The login user's home directory in hierarchical file system.
- The PREFIX defined in the PROFILE of the login TSO user ID. If no PREFIX has been defined, the login user ID will be the initial working directory.

You can choose which directory to be used by FTP server using the STARTDIRECTORY statement in FTP.DATA. The default is MVS data set.

Note: If you run the FTP client in the TSO environment, the PREFIX (or your TSO user ID if no prefix is defined) is used as the initial local working directory. If you run the FTP client from the OE shell, your $HOME directory is used as the initial local working directory.

If your FTP clients want to transfer both MVS data sets and hierarchical file system files, in or out of the FTP server, they can change the current working directory after they have logged in as shown in Figure 6-35:

```
C:\>FTP 9.24.104.43
Connected to 9.24.104.43.
220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe, 17:34:35 on 2002-05-26.
220-*******************************************************
220-*                                                     *
220-*          Welcome to z/OS ITSO Raleigh SC63          *
220-*     You accessed this system via CS for z/OS V1R2   *
220-*                                                     *
220-*    This is the TCPIP.TCPPARMS(FTPBANR) file for FTP *
220-*                                                     *
220-*******************************************************
220-This is SC63.itso.ral.ibm.com on Sun May 26 17:34:35 2002
220-For administrative assistance contact support@helpdesk.com
220-
220 Connection will not timeout.
User (9.24.104.43:(none)): garthm
331 Send password please.
Password:
230 KAKKY is logged on.  Working directory is "/u/garthm".            1
ftp> cd 'TCPIP.TCPPARMS'
250 "TCPIP.TCPPARMS" partitioned data set is working directory       2
ftp> cd /tmp
250 HFS directory /tmp is the current working directory              3
ftp>
```

Figure 6-35 Changing to a HFS directory

1 The default working directory just after having logged in is the user's HOME directory (we have STARTDIRECTORY HFS defined in the Server).

2 Change working directory to an MVS data set high-level qualifier.

3 Change working directory back to a directory in hierarchical file system.

If you can change the initial working directory in the MVS environment, you can use the **TSO PROFILE** command as follows:

1. Log on to TSO on the MVS system of the FTP server.
2. Set your new prefix using the TSO PROFILE command:

   ```
   TSO PROFILE PREFIX(prefix)
   ```

   where `prefix` is any TSO prefix you choose.
3. Log off to save the new default working directory name.

Using a **PWD FTP** subcommand after logging into an FTP server, you can see where your initial working directory is in the remote system.

A user of the UNIX System Services FTPD server must have a valid OMVS identity in terms of an OMVS segment in the user's RACF profile in order just to log on to the FTP server. This is the case even if the user does not intend to transfer files in or out of the hierarchical file system, but just wants to use the server for transferring standard MVS data sets.

If FTP clients receive an error message like the following when they try to log on to your FTP server, they most likely do not have a valid OMVS segment in their RACF user profile and a default OMVS segment has not been established.

```
[C:\]ftp mvs18o
IBM TCP/IP for OS/2 - FTP Client ver 08:36:08 on Jul 22 1996
Connected to mvs18o.itso.ral.ibm.com.
220-FTPD1 IBM MVS V3R3 at mvs18oe.itso.ral.ibm.com, 13:45:23 on 1997-09-28
220 Connection will close if idle for more than 5 minutes.
Name (mvs18o): alfred
331 Send password please.
Password: ......
550 PASS command failed - getpwnam() error : EDC5163I SAF/RACF extract error.
Login failed.
ftp>
```

Figure 6-36 User signon without valid OMVS segment

6.4.2 MVS datasets and HFS files

The FTP server determines the source and target file (MVS data set or a hierarchical file system file) in the following way:

It does so based on an analysis of the resource name. You do not have to specify explicitly if you are working with MVS data sets or hierarchical file system files.

- Is the name a fully qualified resource name (enclosed in single quotes)?

  Use the name as it is. If the name begins with a slash (/), then treat it as an HFS file name. Otherwise treat it as an MVS data set name.

  – 'mydata.set' - MVS data set MYDATA.SET

  – '/u/user/myfile' - HFS file /u/user/myfile

- Is the name not a fully qualified resource name?

  If the name begins with a slash, use the name as it is and treat it as an HFS file name.

  – /u/user/yourfile - HFS file /u/user/yourfile

  If the name does not begin with a slash, then append the value of the current working directory to the beginning of the name and treat the combined name as a fully qualified name.

  – myfile - if the current working directory is /u/user, the resulting name is an HFS file called /u/user/myfile. If the current working directory is myuserid, the resulting name is an MVS data set called MYUSERID.MYFILE.

MVS data set names are translated to uppercase before the FTPD server accesses the data sets. HFS file names are left as is, because mixed case file names are fully valid in the hierarchical file system.

File names in the hierarchical file system may consist of any characters, including quotes and spaces. This poses a small dilemma if, for example, you want to transfer a file in the HFS that has a name of /u/user/c/'weird.name', which is a fully valid name in the hierarchical file system.

Assume the following sequence of events:

1. Current working directory is /u/user/c
2. User enters `get ‘weird.name'`

According to our rules described earlier, this is a fully qualified MVS data set name, but that is not what the user meant. The FTP server has a new option available to deal with this dilemma: the quotes override option. It can be set to an installation-wide default value in your FTP.DATA configuration and it can be changed by individual users via a **SITE (NO)QUOTESOVERRIDE** command.

QUOTESOVERRIDE refers to overriding the current directory specification in this situation or not.

If QUOTESOVERRIDE=TRUE, the above situation would result in the MVS data set 'WEIRD.NAME' being transferred.

If QUOTESOVERRIDE=FALSE, the above situation would result in the HFS file /u/user/c/'weird.name' being transferred.

Another interesting file name in the hierarchical file system could be the following:

```
&blank.&blank.very&blank.weird'Name&blank.&blank.
```

where the `&blank.` in the name stands for a blank character. This file name starts with two blanks, has a blank and a quote in the middle of the name and ends with two blanks. Although special, it is a valid HFS file name.

The UNIX System Services FTPD server deals with such a name based on the FTP protocol subcommand contents. If a user wants to retrieve the above file, the FTP subcommand would be a **RETR** subcommand. The exact contents of the subcommand would be as follows:

```
RETR   very weird'Name  <eol>
```

The **RETR** command is followed by *one* blank before the file name starts. In this example, there are three blanks following the **RETR** command, which means that the file name starts with two blanks. Everything up until the <eol> control character is considered to be the resource name, including the two trailing blanks in the above example. The resource name is then treated as described earlier. If the current working directory is an MVS data set high-level qualifier, the result would be a invalid MVS data set name, but if the current working directory is a directory path in the hierarchical file system, the resulting name would be a valid file name.

Note: Please note that not all FTP clients will be able to deal with such file names, especially not file names with leading or trailing blanks.

When you work with file names in the hierarchical file system, there are a couple of character sequences that have special meaning:

`~/`	(tilde slash) This stands for your $HOME directory.
`../`	(dot dot slash) This means back up one directory level.

These may be combined in a resource name. Consider the following example: your current working directory is /u/dilbert/c/trojan_horses and your $HOME directory is /u/weirdo.

- You enter `get ~/mysweetie` - the resulting file name becomes: /u/weirdo/mysweetie.
- You enter `get ../../bin/sweeproot` - resulting file name becomes: /u/dilbert/bin/sweeproot.
- You enter `get ~/../boss/salary_list` - the resulting file name becomes: /u/boss/salary_list.

6.4.3 Restartability

Checkpoint restarting can be done for the following modes

- Block mode and type E (ebcdic)
- Compressed mode and type E (ebcdic)
- Stream Mode and type I (binary), E (ebcdic), or A (ascii)

Block transfer mode and the compressed transfer mode supports checkpoint/restart. When checkpointing is used, the sending side inserts checkpoint markers into the data being transmitted (using the descriptor code in block mode and using an escape sequence in compressed mode). The FTP client always keeps track of the checkpoint markers. In the CS for z/OS V1R2 IP FTP client the checkpoints are logged in a data set called userID.FTP.CHKPOINT. The FTP client restarts the failed request using the logged checkpoint marker information.

You have to indicate to the sending side if you want to include checkpoints in your file transfer. If you GET a file, you must use the **`site chkptint=nn`** command. If you PUT a file, you must use the **`locsite chkptint=nn`** command to specify how many blocks to transmit between checkpoints.

```
mode b
 >>>MODE b
 200 Data transfer mode is Block
 Command:
type e
 >>>TYPE e
 200 Representation type is Ebcdic NonPrint
 Command:
site chkptint=2
 >>>SITE chkptint=2
 200 Site command was accepted
 Command:
get large large.mvs1803
 >>>PORT 9,67,41,2,4,8
 200 Port request OK.
 >>>RETR large
 125 Sending data set GARTHM.LARGE FIXrecfm 80

Line error connection terminated
```

Figure 6-37 Setting the checkpoint interval

Note: Set your CHKPTINT based on the formulae below and not on the figure and set in the above example.

Formulae: CHKPTINT = amount of data in interval / record length of the file

CHKPTINT = 200KB / 80 bytes

= 200 * 1024 bytes / 80 bytes

= 2560

Now assume that your connection broke down in the middle of this transfer. The FTP client keeps track of the checkpoints in a data set, which would look like the following if the transfer had broken down:

```
get.large.large.mvs1803.
110 MARK 84 = 0,768,2,320,0,320,0,3200
110 MARK 168 = 0,1280,4,640,0,640,0,3200
```

To restart this transfer, you have to start your FTP client again, reconnect to the server, reestablish your transfer mode and data type settings, and issue a **restart** command:

```
mode b
 >>>MODE b
 200 Data transfer mode is Block
 Command:
type e
 >>>TYPE e
 200 Representation type is Ebcdic NonPrint
 Command:
restart
 >>>REST 0,1280,4,640,0,640,0,3200
 300 Restart command accepted, parameter 0,1280,4,640,0,640,0,3200.
 >>>PORT 9,67,41,2,4,6
 200 Port request OK.
 >>>RETR large
 125 Sending data set ALFREDC.LARGE FIXrecfm 80
 65536 bytes transferred.
 139264 bytes transferred.
 311296 bytes transferred.
 483328 bytes transferred.
 638976 bytes transferred.
 794624 bytes transferred.
 250 Transfer completed successfully.
 921300 bytes transferred in 72.871 seconds. Transfer rate 12.64 Kbytes/
 Command:
```

Figure 6-38 Restarting a block mode file transfer

To restart an interrupted stream mode transfer the SRESTART subcommand has to be used. The **srestart** command arguments indicate the file transfer to restart, and has to match the original command arguments.The environment (FTP.DATA statements, site and locsite commands, subcommands, start options) when you issue srestart, must be identical to the original environment. You the user is responsible for creating this environment. The local file must be HFS and must be a link to a file.

The following SRESTART restrictions apply:

- the mode must be stream
- structure must be file
- the filetype must be SEQ (no JES or SQL restarts)
- sunique must be toggled off
- the type must be I (binary), E (ebcdic), or A (ascii)
- TLS or Kerberos protected data connections cannot be restarted

When type A is set, the file data received from the server on the data connection is encoded in ASCII. Each line is terminated with a <CRLF> (carriage return, line feed) sequence, in accordance with RFC 959. If the data contains a <CR> other than the <CRLF> sequence, the file cannot be restarted. The FTP client can detect this problem and will fail the restart, provided the same ASCII translate table is used for the original file transfer and for the restarted file transfer.

If SRETART fails, there is always the option of running your file transfer again.

To activate this function on the z/OS server, you must code two statements in FTP.DATA as can be seen in Figure 6-41 on page 175:

- extensions SIZE
- extensions REST_STREAM

The SIZE command is part of the sequence needed to restart a file in stream mode. Extensions REST_STREAM configures the server to accept a REST command while in stream mode, and while type is set to I, A, or E.

FTP logs an error message when you code EXTENSIONS REST_STREAM without coding EXTENSIONS SIZE.The server FEAT command can be used as a means of asking the server which extensions to FTP it supports.

We setup a 3CDAEMON FTP server on Windows 2000, and transferred an HFS file to the PC from a z/OS client. We defined a mode of Stream and a type of Image/Binary and issue our PUT command as shown in Figure 6-39.

```
 Command:
mode s
 >>> MODE S
 200 Command OK. Current mode is stream
 Command:
type i
 >>> TYPE I
 200 Type set to I.
 Command:
put /usr/hfstest hfstest
 >>> PORT 9,12,6,30,4,153
 200 PORT command successful.
 >>> STOR hfstest
 150 File status OK ; about to open data connection
 1658880 bytes transferred.

       Line error connection terminated
```

Figure 6-39 Stream FTP failure to PC FTP Server

Once the transfer was well under way we broke the LAN connection to simulate the connection and subsequent FTP termination. After restoring the FTP connection, we recreated the same environment as we had at the time of the failure. We did this by setting Mode to S (Stream) and the type to I (Image or Binary). We then issued the Stream restart command called SRESTART as shown in Figure 6-40 on page 174. The format for this command is as follows:

SRESTART subcommand

SRESTART get foreignfile localfile

SRESTART put localfile localfile

The file transfer continued from the point of termination. As apposed to the block restart scenario, the SITE command CHKPTINT is not required when doing Stream restarts

```
 Command:
mode s
 >>> MODE S
 200 Command OK. Current mode is stream
 Command:
type i
 >>> TYPE I
 200 Type set to I.
 Command:
srestart put /usr/hfstest hfstest
 >>> SIZE hfstest
 213 2114112
 >>> PORT 9,12,6,30,4,155
 200 PORT command successful.
 >>> REST 2114111
 350 Requested file action pending further information
 >>> STOR hfstest
 150 File status OK ; about to open data connection
 1756609 bytes transferred.
 3415489 bytes transferred.
 :
 17608129 bytes transferred.
 226 Closing data connection; File transfer successful.
 18864565 bytes transferred in 114.760 seconds.  Transfer rate 164.38 Kbytes/s
 ec.
 Command:
```

Figure 6-40 Restarted Stream file transfer to PC FTP Server

6.4.4 Using a socks server

A firewall controls access to a companies networks by either allowing or blocking such access. Firewall technology used to isolate the intranet from the internet, normally implements a SOCKS server. The SOCKS server allows client applications inside the fire wall to use servers outside the fire wall, by relaying the application data in the SOCKS server, which is also known as a transparent proxy server. The SOCKS server can be configured to reject, allow, redirect, or log connections requested by clients.

To set up the client to use a socks server the following has to be done.

- Include a SOCKSCONFIGFILE statement in the client's FTP.DATA which points to a configuration file as detailed in Figure 6-41. The SOCKSCONFIGFILE can be HFS, a PDS or a Sequential file.

```
Recfm           FB        ; Fixed blocked record format
Filetype        SEQ       ; File Type = SEQ (default)
RDW             false     ; Do not retain RDWs as data
EXTENSIONS      SIZE      ; Server can respond to SIZE cmd
EXTENSIONS      MDTM      ; Server can respond to MDTM cmd
EXTENSIONS      UTF8      ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
SOCKSCONFIGFILE 'TCPIPB.SC63.TCPPARMS(SOCKSCNF)'  ;SOCKS CONFIG FILE
```

Figure 6-41 FTP.DATA file showing SOCKS config file parm

If the file defined in this statement is absent or invalid it will not default but the statement will be ignored as shown in Figure 6-42 on page 175.

```
BPXF024I (TCPIPMVS) May  8 14:15:20 ftpd 67240208 : EZYFT47I 797
 dd:SYSFTPD file, line 21: Ignoring keyword "SOCKSCONFIGFILE".
```

Figure 6-42 Invalid Socks config file

Socks protocols will therefore not be used in the following circumstances:

- If there is no SOCKSCONFIGFILE statement is in FTP.DATA;
- The SOCKSCONFIGFILE cannot be opened or read;
- The SOCKSCONFIGFILE contains invalid data.

The SOCKSCONFIGFILE contains statements telling the FTP client whether to use SOCKS protocols for a given target server as shown in Figure 6-43. Pertaining to this figure:

- DIRECT statement means don't use SOCKS protocols.
- SOCKD4 statement means use SOCKS V4 protocols to access the FTP server.
- SOCKD5 statement means use SOCKS V5 protocols to access the FTP server.

The statements can be in any order, but the FTP client will use the first statement that applies to the target FTP server.

```
direct 9.0.0.0 255.0.0.0         ; internal net            1
sockd4 @=socks4srv 192.152.1.0 255.255.255.0 ; Test Network1  2
sockd5 @=socks5srv 192.158.3.0 255.255.255.0 ; Test Network2  3
sockd5 @=9.1.2.3 0.0.0.0 0.0.0.0 ; Anything else 4
```

Figure 6-43 Socks configuration file

On the **direct** statement and you must supply an IP address and a subnet mask. Only IP V4 addressing is supported.

1 This **direct** statement indicates to FTP client not to use the SOCKS server for addresses within our network.

On the **sockd4** and **sockd5** statements a DNS name or a dotted decimal IP address can be used for your socks server. A gethostbyname(DNS name) is done at FTP initialization, so the DNS name for the socks server host given in this statement has to be valid.

SOCKD4 @=socks_server_host_name remote_server address mask

As in the direct statement, you use a dotted decimal IP address and mask to denote the FTP server or servers affected by this rule --target FTP server IP address AND'ed with the mask must match the remote server address

2This `sockd4` statement instructs the FTP client to use the SOCKS server socks4srv for any target in the class C 192.152.1.0 network, and to use SOCKS 4 protocols to establish the connection. As can be seen in this example the use of DNS names are allowed.

3This `sockd5` statement instructs the FTP client to use the SOCKS server socks5srv for any target in the class C 192.158.3.0 network, and use SOCKS 5 protocols to establish the connection.The `SOCKD` and `SOCKD5` keywords can be used interchangeably.

4 This `sockd5` statement was defined as a catch all bucket for dealing with all other servers out there.t

Note: If you use `direct 0.0.0.0 0.0.0.0` at end of the SOCKSCONFIGFILE it will bypass SOCKS for all target FTP servers.

You can use the LOCSTAT command to determine which file is used for SOCKCONFIGFILE.The example shown in Figure 6-44 indicates 'garthm.ftp.cnf1' is coded in FTP.DATA. If no SOCKSCONFIGFILE statement is in FTP.DATA or there is something wrong will the file then this entry will not be displayed.

```
locstat
 Trace: FALSE, Send Port: TRUE
 Send Site with Put command: TRUE
 Connected to:9.12.6.63, Port: FTP control (21), logged in
 Local Port: 1208
 Data type:a, Transfer mode:s, Structure:f
 UTF-8 encoding is being used on the control connection
 Automatic recall of migrated data sets.
 Automatic mount of direct access volumes.
 Data set mode. (Do not treat each qualifier as a directory.)
 ISPFSTATS is set to FALSE
 Primary allocation 5 tracks, Secondary allocation 2 tracks.
 Partitioned data sets will be created with 15 directory blocks
 FileType is SEQ (Sequential - the default).
:
:
:
Prompting: ON, Globbing: ON
 ASA control characters transferred as ASA control characters
 New data sets catalogued if a store operation terminates abnormally
 Single quotes will override the current working directory
 UMASK value is 027
 Data connections for the client are not firewall friendly.
 Authentication mechanism: None
 Tape write is not allowed to use BSAM I/O
 Using GARTHM.FTP.CNF1 for SOCKS server configuration
 Using dd:SYSFTPD=TCPIPB.SC63.TCPPARMS(CLFTPD) for local site configuration pa
 rameters.
```

Figure 6-44 STAT command showing SOCKS server file

All messages, tracing, logging, and TCP/IP diagnostics that show peer information will now show the SOCKS server information, not the FTP client information. Exit routines are driven with the SOCKS server IP address and port.

In this example, the STAT reply, which shows FTP server status, displays the SOCKS server IP address, not the client IP address.

This is correct because the connection is truly with the SOCKS server, but it is something to be aware of when debugging and interpreting FTP server replies.

6.4.5 FTP Tracing

Server tracing is determined at the time the client starts its session. Tracing cannot be changed for the session after it starts. If a modify command is issued for the trace, it affects only those clients that start a session after the modify command completes.

The DEBUG command can set trace types as follows:

- ALL sets all of the traces. The following is a list of the available trace types.

  FLO function flow

  CMD command trace

  PAR parser details

  INT program initialization and termination

  ACC access control (logging in)

  UTL utility functions

  FSC file services

  SOC socket services

  JES JES processing (server only)

  SEC Security

  SQL SQL processing
- NONE resets all
- BAS (for Basic) sets the following traces:
  - CMD
  - INT
  - FSC
  - SOC
- ? as a type to show the active traces as shown in Figure 6-45 on page 177:
- Entering ‘DEBUG’ twice toggles tracing on and off respectively

```
debug ?
 PC0315 parseCmd: subcommand:  debug
 PC0318 parseCmd: parameter 1: ?
 Active client traces - CMD INT FSC(1) SOC(1)
 Command:
```

Figure 6-45 Querying active traces

Each trace type entry can be reset by adding the prefix X to the **debug** command. This will then exclude this type entry from the active client trace which is in effect at the time of the command. This is shown in Figure 6-46 on page 178 which demonstrated how to exclude certain trace types.

```
debug ?
 CL0181 debug: entered
 Active client traces - ACC UTL SEC FSC(1) SOC(1) SQL
 Command:
debug xacc xutl
 CL0181 debug: entered
 GU2299 setDebug: entered
 GU2299 setDebug: entered
 Active client traces - SEC FSC(1) SOC(1) SQL
 Command:
```

Figure 6-46 Excluding trace types

File Services(FSC) and Socket Services (SOC) supports different levels of tracing. Setting these different levels is shown in Figure 6-47. The higher levels of tracing produce large amounts of output and should be used only when requested by IBM service.

```
 debug fsc soc
 PC0315 parseCmd: subcommand:  debug
 PC0318 parseCmd: parameter 1: fsc
 PC0318 parseCmd: parameter 2: soc
 Active client traces - CMD INT FSC(1) SOC(1)
 Command:
debug fsc(2) soc(4)
 PC0315 parseCmd: subcommand:  debug
 PC0318 parseCmd: parameter 1: fsc(2)
 PC0318 parseCmd: parameter 2: soc(4)
 Active client traces - CMD INT FSC(2) SOC(4)
 Command:
```

Figure 6-47 Changing trace levels

Server trace can be started as follows:

- F FTPDB1,TRACE
- F FTPDB1,DEBUG=(BAS)
- SITE DEBUG=(INT,ACC)

Note: The server FTP.DATA file must have the statement '**`DEBUGONSITE TRUE`**', for the '**`SITE DEBUG=`**' subcommand to be accepted by the server

Client trace can be started as follows:

- FTP.DATA statements. Figure 6-48 on page 179 show an example of these
- ftp ip_address -d
- ftp ip_address (trace
- debug subcommand

Extended tracing can be done in the form of the DUMP command. The ID parameter of the dump command which activates specific trace points in the FTP code, ranges from 1-99.

Server extended trace activation can be done as follows:

- FTP.DATA statements. Figure 6-48 on page 179 shows an example of these.
- F FTPDB1,DUMP
- SITE DUMP=1

Note: The server FTP.DATA file must have the statement '`DUMPONSITE TRUE`' for the '`SITE DUMP`' subcommand to be accepted by the server

Client EXTENDED trace activation can be done as follows:

- FTP.DATA statements. This is shown with the dump command in Figure 6-48.
- dump subcommand

```
DEBUG BAS ; set basic trace parms
DUMP 21 ; dynamic allocation text units
DUMP 35 ; checkpoint marker error
DUMP 53 ; 1st 5 bytes of data on ctrl connection
DUMP 62 ; job status
DUMP 72 ; SQL FETCH values
DUMP 84 ; SOCKS configuration tables
```

Figure 6-48 Client or Server FTP.DATA dump activation statements

The following controls can be put in place when activating server traces:

- RACF profile use of SERVAUTH resource class

  EZB.FTP.<sysname>.<ftpname>.SITE.DEBUG

  EZB.FTP.<sysname>.<ftpname>.SITE.DUMP

  Example:

  `EZB.FTP.SC63.FTPDB1.SITE.DEBUG`
- MODIFY command filters for userid and for ip address

  MODIFY FTPDB1,DEBUG=(ALL,USERID(USER3*))

  MODIFY FTPDB1,DEBUG=(ALL,IPADDR(9.67.113.57))

Only one filter can be active at a time for DEBUG and DUMP

Details in the DEBUG and DUMP subcommand can be reviewed in the z/OS V1R2.0 CS: IP User's Guide and Commands, SC31-8780 manual.

6.4.6 Using the latest FTP features (RFC2389 and 2640)

CS for z/OS V1R2 IP FTP implements RFCs 959, 1123 and 1579. These RFCs are quite old as they date back to the eighties. New RFCs in the form of RFC 2389 and 2640 has been added in the late nineties which incorporated new features and extensions to FTP. These RFCs represent the feature negotiation mechanism and FTP internationalization respectively.

RFC 2389

The Feature negotiation Mechanism feature enables FTP clients to ask the server which features or options it supports. Commands introduced with this feature are:

- On the z/OS Server:

```
   OPTS
FEAT
```

- On the z/OS Client

```
feature
```

- On the non z/OS Client

```
quote feature
```

Displaying server features depends upon what is included in the EXTENSIONS statement in FTP.DATA. Figure 6-49 on page 180 shows an example of a FTP.DATA showing some extension statements.

```
SpaceType       TRACK      ; Data sets allocated in tracks
Recfm           FB         ; Fixed blocked record format
Filetype        SEQ        ; File Type = SEQ (default)
RDW             false      ; Do not retain RDWs as data
EXTENSIONS      SIZE       ; Server can respond to SIZE cmd
EXTENSIONS      AUTH_GSSAPI  ; GSSAPI authentication is supported
EXTENSIONS      MDTM       ; Server can respond to MDTM cmd
EXTENSIONS      UTF8       ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
EXTENSIONS      AUTH_TLS   ; TLS authentication supported
```

Figure 6-49   FTP.DATA file showing extensions supported

RFC2640 - FTP Internationalization

FTP servers use English alpha numerics for their directory and file names, even though the site serves a nonEnglish speaking community. This is so because FTP constrains pathnames to single byte encodings (ASCII). With Internationalization, 7-bit restrictions on pathnames used in client commands and server responses are removed. UCS transformation format UTF-8 is used, and a new command for language negotiation is defined. To activate the RFC2640 implementation the 'EXTENSIONS UTF8' parameter in TCP is required in FTP.DATA as shown in Figure 6-49 on page 180. This parameter has to be coded in both the z/OS Client and the z/OS Server. A client can thus override statements within FTP.DATA which control code page selection by using the 'LOCSITE CONTRLCONN subcommand.

Requirements to implement these RFCs

- RFC 2389 is available from CS for z/OS V1R2 IP with no additional tasks having to be performed to activate the support.
- RFC 2640 requires you to code an EXTENSIONS UTF8 statement in your server and clients FTP.DATA

The National Language Resources component of z/OS Language Environment must be installed for the UTF-8 function to be available.

Once UTF-8 has been negotiated between client and server, the STAT command will indicate that UTF-8 is active on the control connection. Figure 6-50 on page 181 shows the results of a successful UTF-8 negotiation.

```
stat
 >>> STAT
 211-Server FTP talking to host 9.12.6.30, port 1088
 211-User: GARTHM  Working directory: GARTHM.
 211-UTF-8 encoding in use on the control connection
 211-The control connection has transferred 309 bytes
 211-There is no current data connection.
 211-The next data connection will be actively opened
 211-to host 9.12.6.30, port 1088,
 211-using Mode Stream, Structure File, type ASCII, byte-size 8
 211-Automatic recall of migrated data sets.
 211-Automatic mount of direct access volumes.
 211-Auto tape mount is allowed.
 211-Inactivity timer is set to 300
 211-VCOUNT is 59
 211-ASA control characters in ASA files opened for text processing
 211-will be transferred as ASA control characters.
 211-Trailing blanks are removed from a fixed format
 211-data set when it is retrieved.
```

Figure 6-50 STAT command output

6.4.7 FTP from a Web browser

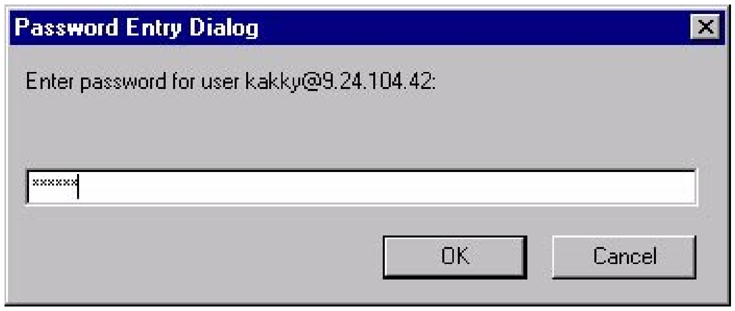
The server now presents a UNIX appearance from a Web browser. For example if you log in to the CS for z/OS V1R2 IP FTP server from a Web browser with the following in the locator field:

```
ftp://userid:password@your.mvshost.com/
```

your browser will present the files and directories with the look of a UNIX environment. Be cautious when you access the FTP server using this URL syntax, because your user ID and password are sent over the IP network in clear text. Also your user ID and password will remain in the locator pull-down list of the Web browser. You may enter a URL without the password by entering the following URL:

```
ftp://userid@your.mvshost.com/
```

Then you will be prompted for the password once the FTP server is contacted.

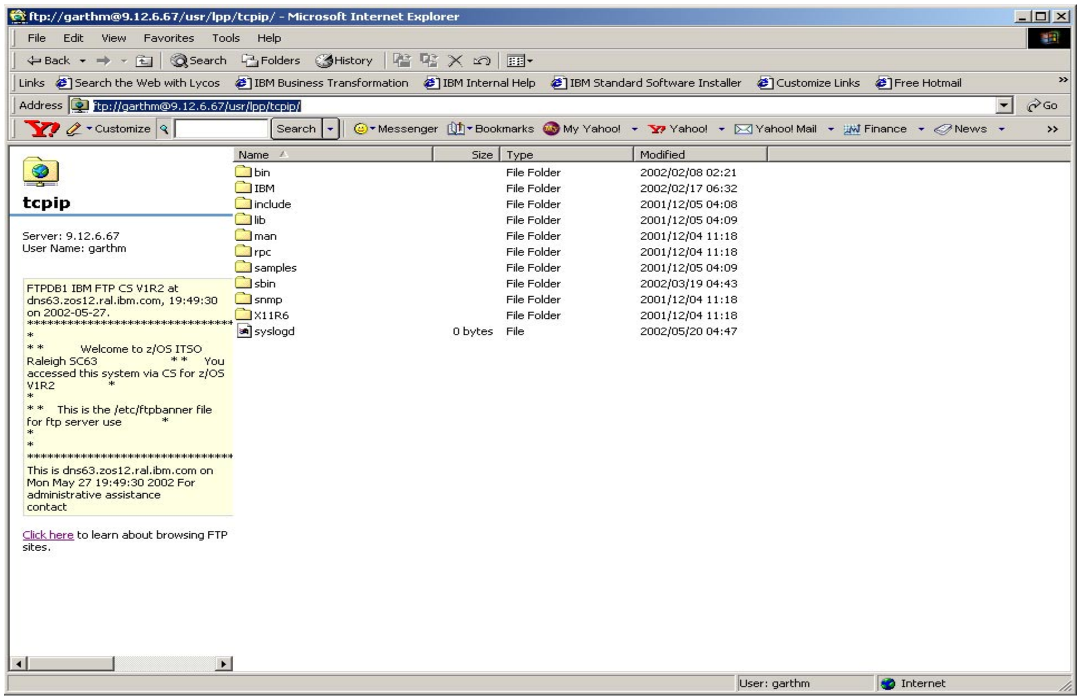

Password Entry Dialog

Enter password for user kakky@9.24.104.42:

Figure 6-51 Password entry dialog for FTP connection using Netscape Navigator

This will keep the password out of the browser locator pull-down list of URLs, but once prompted for the password by the FTP server the password is still sent over the network in clear text. Figure 6-51 is a dialog panel from a browser accessing the CS for z/OS V1R2 IP FTP server.

If your FTP.DATA file has the STARTDIRECTORY set to MVS then the '/' at the end of the URL is crucial if you want to access the UNIX System Services file system. In this case the '/' at the end of the URL assures you are presented the listing of the root directory of UNIX System Services. The information in Figure 6-52 can be displayed by entering `ftp://userid@your.mvshost.com/usr/lpp/tcpip/` in the Web browser locator field. Also make sure that the FTP.DATA keyword 'FILETYPE' is set to SEQ, so that a file listing will be presented. The **SITE** and **LOCSITE** commands cannot be used in a browser interface to make changes to the keywords that affect FTP transfers.


ftp://garthm@9.12.6.67/usr/lpp/tcpip/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Folders History
Links Search the Web with Lycos IBM Business Transformation IBM Internal Help IBM Standard Software Installer Customize Links Free Hotmail
Address ftp://garthm@9.12.6.67/usr/lpp/tcpip/ Go
Customize Search Messenger Bookmarks My Yahoo! Yahoo! Yahoo! Mail Finance News
tcpip

Server: 9.12.6.67
User Name: garthm

```
FTPDB1 IBM FTP CS V1R2 at
dns63.zos12.ral.ibm.com, 19:49:30
on 2002-05-27.
********************************
*
* *         Welcome to z/OS ITSO
Raleigh SC63                 * *     You
accessed this system via CS for z/OS
V1R2             *
*
* *    This is the /etc/ftpbanner file
for ftp server use          *
*
*
********************************
This is dns63.zos12.ral.ibm.com on
Mon May 27 19:49:30 2002 For
administrative assistance
contact
```

Click here to learn about browsing FTP sites.

Name	Size	Type	Modified
bin		File Folder	2002/02/08 02:21
IBM		File Folder	2002/02/17 06:32
include		File Folder	2001/12/05 04:08
lib		File Folder	2001/12/05 04:09
man		File Folder	2001/12/04 11:18
rpc		File Folder	2001/12/04 11:18
samples		File Folder	2001/12/05 04:09
sbin		File Folder	2002/03/19 04:43
snmp		File Folder	2001/12/04 11:18
X11R6		File Folder	2001/12/04 11:18
syslogd	0 bytes	File	2002/05/20 04:47

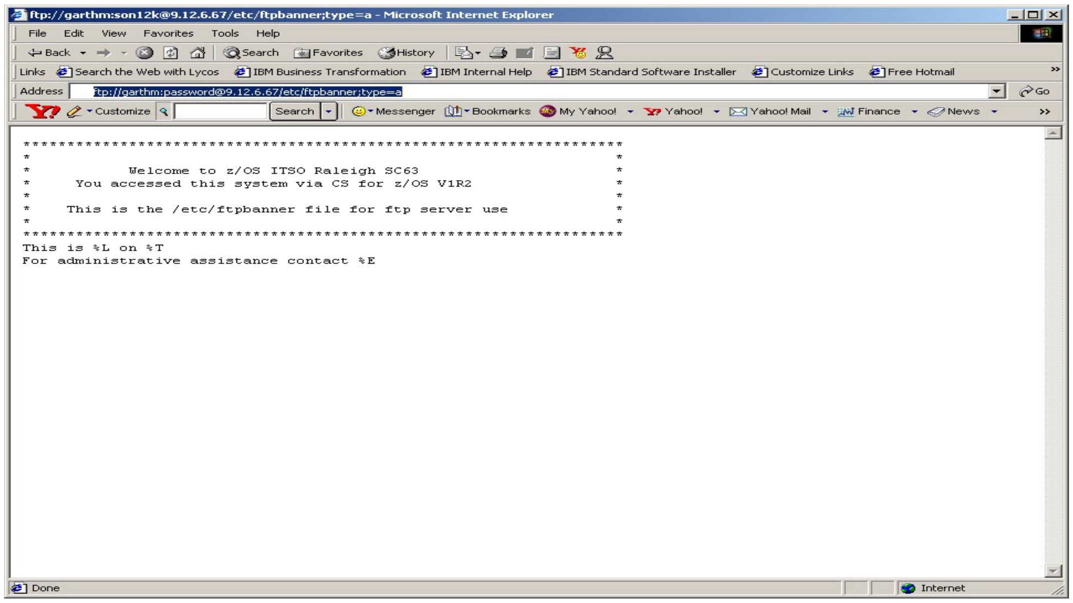
User: garthm   Internet

Figure 6-52 View of CS for z/OS V1R2 IP /usr/lpp/tcpip directory from a browser

Suppose you wanted to view /etc/banner file in your browser. You can view it in the browser by using the following syntax to view the file:

```
ftp://userid:password@your.mvshost.com/etc/banner;type=a
```

The `;type=a` following the file name informs the FTP server that this is an ASCII file. File names with extension that are typically used to describe a text file (for example, txt) can be displayed by a browser without `;type=a`. The FTP server has also extended its system tests to allow it to function with GUI FTP client software.


ftp://garthm:son12k@9.12.6.67/etc/ftpbanner;type=a - Microsoft Internet Explorer
Address ftp://garthm:password@9.12.6.67/etc/ftpbanner;type=a
Done
Internet

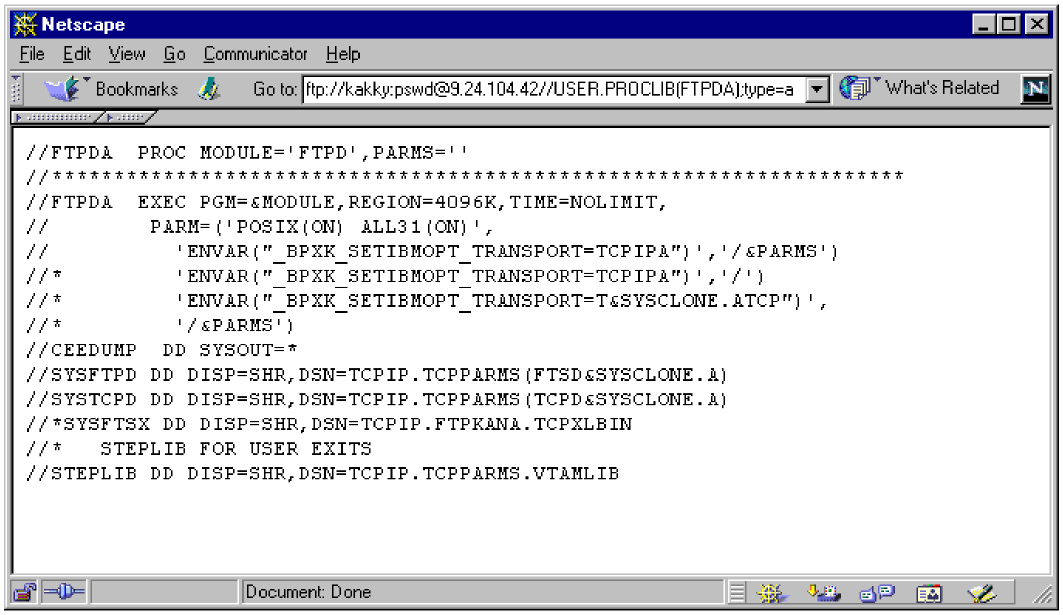
```
*********************************************************************
*                                                                   *
*           Welcome to z/OS ITSO Raleigh SC63                       *
*      You accessed this system via CS for z/OS V1R2                *
*                                                                   *
*     This is the /etc/ftpbanner file for ftp server use            *
*                                                                   *
*********************************************************************
This is %L on %T
For administrative assistance contact %E
```

Figure 6-53   Browsing a text file using an FTP connection

z/OS data set URL support

CS for z/OS V1R2 IP FTP server allows a Web browser to read MVS data sets by entering:

```
ftp://userid@yourhost.com//hlq.jcl(test);type=a
```


Netscape
Go to: ftp://kakky:pswd@9.24.104.42//USER.PROCLIB(FTPDA);type=a
Document: Done

```
//FTPDA   PROC MODULE='FTPD',PARMS=''
//*********************************************************************
//FTPDA   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//         PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")','/&PARMS')
//*          'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")','/')
//*          'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=T&SYSCLONE.ATCP")',
//*          '/&PARMS')
//CEEDUMP  DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(FTSD&SYSCLONE.A)
//SYSTCPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(TCPD&SYSCLONE.A)
//*SYSFTSX DD DISP=SHR,DSN=TCPIP.FTPKANA.TCPXLBIN
//*   STEPLIB FOR USER EXITS
//STEPLIB DD DISP=SHR,DSN=TCPIP.TCPPARMS.VTAMLIB
```

Figure 6-54   Browsing an MVS data set using an FTP connection

The two slashes tell the FTP server that the data set is a traditional data set and not a file in the HFS. CS for z/OS V1R2 IP adds a new configuration statement (MVSURLKEY) to allow the FTP server URL syntax (or an HTML script) to be consistent with WebSphere URL syntax. Refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for more details. The new statement allows you to define a key to indicate an MVS data set name in a URL. I Suppose you code the new statement as follows in an FTP.DATA configuration file:

```
MVSURLKEY         MVSDS
```

Then you will also be able to access MVS data sets with a browser by using the following URL:

```
ftp://userid@yourhost.com/MVSDS/'hlq.jcl(test)';type=a
```

Note that the MVS data set name has apostrophes at the beginning and end of the data set name. Be aware of the following when implementing the new statement:

- When using the MVSURLKEY statement, avoid using metacharacters (for example, ###) in the key that the FTP client browser may interpret instead of passing on to the FTP server. Stick to using the basic alphanumeric characters.
- When selecting a name for a key, WebSphere users should choose a name that matches the WebSphere configuration directive for encoding MVS data set URLs. Typically the key is MVSDS.
- The FTP server won't support the WebSphere URL encoding unless you explicitly encode an MVSURLKEY in FTP.DATA.
- If the MVSURLKEY happens to match the name of an HFS root subdirectory, that directory will be closed to FTP transfers.
- The key you choose is case sensitive.

6.4.8 Transferring load modules

It is possible to transfer load modules between systems provided that they are CS for OS/390 V2R10 IP and above. The load module transfer function uses the IEBCOPY system utility, which must be available on both the origin and destination host. Sufficient temporary DASD to hold the unload file must be available on both the origin and destination systems. On the origin system, it unloads the load module(s) to be transferred into a temporary data set, which is then transferred to the destination system and reloaded. (Refer to *z/OS V1R2.0 CS: IP User's Guide and Commands,* SC31-8780 for further details). If the SITE/LOCSITE parameter AUTOMOUNT is allowed FTP will attempt to mount sufficient temporary DASD to satisfy the request. This can cause a delay if your system does not process mount requests quickly.

The transfer may specify a real or alias name; in either case the real module and all its associated aliases are transferred. The FTP commands **get**, **mget**, **put**, and **mput** are supported in MVS load module transfer. After a successful FTP transfer the load module will be executable on the destination host.

If processing fails, the file transfer may continue but the transferred load modules will not be executable on the target system. See the *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782 for details about load module transfer failure causes. Some reasons for failure of load module transfer but successful base file transfer processing could include:

- Not all hosts are at CS for z/OS V1R2 IP; this includes Proxy situations
- Attempt to rename a file or specify it as anything other than a member name
- Attempt to transfer from a PDSE to a PDS or any other non-PDSE file structure

Some load module transfer failures that are fatal and will terminate without attempting base processing:

- Failure to allocate the temporary file
- IEBCOPY not available on either host
- Failure to unload or reload load modules out of or into the temporary file
- Attempt to transfer from a non-PDSE to a PDSE

An example of a failure may be if AUTOMOUNT is not allowed and there is not sufficient temporary DASD. The transfer will fail with a message or reply indicating an error allocating the temporary data set.

Because of special requirements of the load module transfer, there are some restrictions:

- The current working directory on both the client and the server must be the source and destination load library. A load library is a PDS or PDSE with RECFM=U.
- Only member names may be specified with **get** or **put** commands. No fully qualified names may be specified.
- File rename is not supported on load module transfer.
- Load modules may only be transferred between the same types of libraries. For example, PDS to PDSE transfer is not allowed.
- If load modules are being sent to or from the z/OS FTP client, the client must be started from one of the following environments: TSO terminal session, TSO REXX, TSO batch, TSO background, or UNIX System Services terminal session.
- Hosts must be running CS for OS/390 V2R10 IP or greater.
- A load module loading from a temporary data set will always be a *REPLACE* operation; existing members are overwritten.
- There is no prompting on **mput** and **mget** commands. All files that match the mask provided will be transferred.
- DDNAME token "//DD:" is not supported on load module transfers.

Some additional FTP messages will be seen when doing a **cd** and **lcd** into a potential load library (PDS or PDSE with RECFM=U). The following example shows a successful FTP transfer of the load module TCPIP.SEZALINK(FTP) from SC63, which is running CS for z/OS V1R2 IP FTP. The FTP client was SC64 running CS for z/OS V1R2 IP. The load module was retrieved with the **get** command into BTHOMPS.SEZALINK(FTP).

Note that both client (set with **lcd** command) and server (set with **cd** command) are set to the proper working directory.

```
 220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe.itso.ibm.com, 18:11:09 on 2002-05-16.
 220 Connection will close if idle for more than 5 minutes.
 NAME (9.12.6.67:GARTHM):
garthm
 >>> USER garthm
 331 Send password please.
 PASSWORD:


 >>> PASS
 230-Processing FTPS.RC configuration file - GARTHM.FTPS.RC
 230-SITE command was accepted
 230-No users are allowed to use SITE DEBUG
 230-SITE command was accepted
 230-"GARTHM.GARTHM.FTP.CNF1." is the working directory name prefix.
 230 GARTHM is logged on.  Working directory is "GARTHM.GARTHM.FTP.CNF1.".
 Command:
cd 'tcpip.sezalink'
 >>> CWD 'tcpip.sezalink'
 250-The working directory may be a load library
 250 The working directory "TCPIP.SEZALINK" is a partitioned data set
 Command:
Command:
lcd 'garthm.sezalink'
 Local directory might be a load library
 Local directory name set to partitioned data set GARTHM.SEZALINK
 Command:
get ftp
 >>> XLMT PDS 0  ftp
 250 PDS 1279216 - send next command for load module transfer
 >>> PORT 9,12,6,63,4,9
 200 Port request OK.
 >>> RETR ftp
 125-Transferring load module
 125 DCB 32760 32760
 250 Transfer completed successfully.
 1328284 bytes transferred in 0.260 seconds.  Transfer rate 5108.78 Kbytes/sec
 .
 Command:
```

Figure 6-55 Load module transfer

The following accomplishes the same function using an FTP batch job. This job was submitted from an SC64 system:

```
//FTPBAT  JOB 1,BTHOMPS,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//STEP01 EXEC PGM=FTP,REGION=2048K
//*
//SYSFTPD    DD  DSN=TCPIP.TCPPARMS(FTSD28B),DISP=SHR
//SYSTCPD    DD  DSN=TCPIP.TCPPARMS(TCPD28B),DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//OUTPUT    DD  SYSOUT=*
//INPUT     DD  *
; Comments can be inserted in this area.
; connect to SC63 FTP Server
9.12.6.67
;enter userid password below
userid  password       ;be sure to secure this file
lcd 'garthm.sezalink' ;set client's (local)  directory
cd  'tcpip.sezalink'   ;set server's (remote) directory
get ftp                ;get ftp load module
QUIT
/*
```

Figure 6-56 Batch load module transfer

As a result of the **get** FTP subcommand two load modules were transferred: the alias load module FTP and EZAFTPLC. The transfer preserves the load module structure and directory information. It also preserves alias information.

```
EDIT            GARTHM.SEZALINK
Command ===>
           Name     Prompt        Alias-of     Size
_________ EZAFTPLC                           000EA1F0
_________ FTP                     EZAFTPLC   000EA1F0
```

Figure 6-57 Load module present in SEZALINK

6.4.9 Setting up a welcome page

The following FTP configuration file statements are available to customize your FTP environment for welcome pages. Review the *z/OS V1R2.0 CS: IP Migration*, GC31-8773 for details.

BANNER

This statement is used to tell the FTP server which file contains the information to be displayed when a client connects to the FTP server. The file can contain any alphanumeric characters and the supported *magic cookies*. The magic cookies are special characters that are replaced with the appropriate text definition when the banner is displayed. The file containing the banner can be located in the UNIX System Services HFS or a traditional MVS data set. The slash ' / ' as the first character indicates that it's an HFS file, and no slash indicates that it's an MVS data set.

CS for z/OS V1R2 IP support the following magic cookies:

- %T - Local time (displayed as `Thu May 16 18:15:39 2002`)
- %C - Current working directory

- %E - The FTP server administrator's e-mail address
- %R - Remote host name
- %L - Local host name
- %U - Username (logged-in user)

In order to support the FTP server administrator e-mail address cookie (%E), you must specify the e-mail address in the FTP.DATA file. You must also define the banner page support statement. Refer to the statement definitions below for meaning and use. Following are the definitions used on MVS28B FTP.DATA to test banner page support:

```
ADMINEMAILADDRESS    support@helpdesk.com
BANNER               /etc/ftpbanner
```

```
********************************************************************
*                                                                  *
*           Welcome to z/OS ITSO Raleigh SC63                      *
*     You accessed this system via CS for z/OS V1R2                *
*                                                                  *
*    This is the /etc/ftpbanner file for ftp server use            *
*                                                                  *
********************************************************************
This is %L on %T
For administrative assistance contact %E

```

Figure 6-58 Sample banner file with the magic cookies

Figure 6-58 show a sample banner that makes use of the magic cookies, which produces the messages shown in Figure 6-59.

```
220-*********************************************************
220-*                                                       *
220-*           Welcome to z/OS ITSO Raleigh SC63           *
220-*     You accessed this system via CS for z/OS V1R2     *
220-*                                                       *
220-*    This is the /etc/ftpbanner file for ftp server use *
220-*                                                       *
220-*********************************************************
220-This is wtsc63oe.itso.ral.ibm.com on Thu May  16 18:15:39 2002
220-For administrative assistance contact support@helpdesk.com
220-
```

Figure 6-59 Sample banner messages

The banner file /etc/ftpbanner was copied to an MVS data set for testing. TCPIP.TCPPARMS(FTPBANR) and FTP.DATA were updated as follows:

```
BANNER          TCPIP.TCPPARMS(FTPBANR)
```

The message produced was exactly the same as the one above. There will be an error message if the MVS data set name is coded incorrectly in FTP.DATA.

LOGINMSG / ANONYMOUSLOGINMSG

This statement is used to tell the FTP server which file contains the information to be displayed when a client logs into the FTP server. The file can contain any alphanumeric characters. When a defined user logs in, the information in the file pointed to by LOGINMSG will be displayed. When an anonymous user logs in, the information in the file pointed to by ANONYMOUSLOGINMSG will be displayed. The file can be located in the UNIX System Services HFS or a traditional MVS data set. The slash ' / ' as the first character indicates that it's an HFS file, and the absence of a slash indicates that it's an MVS data set.

```
LOGINMSG              /etc/ftploginmsg
ANONYMOUSLOGINMSG     /etc/ftpanonymousloginmsg
```

```
001
002 ******  This system is for authorized purposes only  ******
003
```

Figure 6-60 Content of /etc/ftploginmsg

```
001
002 **** You will be restricted to the /u/ftp directory  ****
003
```

Figure 6-61 Content of /etc/ftpanonymousloginmsg

MVSINFO / ANONYMOUSMVSINFO

This statement is used to tell the FTP server which file name contains the information to be displayed when a client uses the change directory command and initially enters a specific MVS directory. The MVS directory must contain a *low level qualifier file name* that matches the one specified as the parameter of the statement. A wild card character " * " cannot be used for pattern matching in the parameter. The file name that matches the parameter in the MVS directory will have its content presented to the FTP client. If no match is found nothing will be presented. The information file can contain any alphanumeric characters.

When a defined user initially changes to an MVS directory, the information from the file name in the MVS directory that matches the parameter on the MVSINFO statement will be displayed. When an anonymous user initially changes to an MVS directory, the information from the file name in the MVS directory that matches the parameter on the ANONYMOUSMVSINFO statement will be displayed. Only the initial entry (per FTP session) into a directory will cause the information file to be presented to the FTP client. Subsequent **cd** commands to the same directory during the same FTP session will not cause the information file to be presented again.

For testing purposes, the following MVS-LLQ (*low level qualifier*) was used:

```
MVSINFO             README
ANONYMOUSMVSINFO    README
```

To show the use of this statement, below is a listing of HLQ = "USER":

```
DSLIST - Data Sets Matching USER
Command ===>

Command - Enter "/" to select action
--------------------------------------
         USER
         USER.PARMLIB
         USER.PROCLIB
         USER.PROCLIB.OLD
         USER.README
         USER.SA03.APPCTP
         USER.SA03.APPCTP.DATA
         USER.SA03.APPCTP.INDEX
         USER.TRACE
         USER.VTAMLST
         USER.VTAMLST.RECOVER
```

Figure 6-62   DSLIST showing ‘user.’ files

We edited the USER.README with the content shown below:

```
01
02 This file was created to test FTP MVSINFO and can be used as
03 a reference for information that will be stored in hlq 'USER"
04
05 See GC31-8773 z/OS V1r2.0 CS: IP Migration for details:
06
```

Figure 6-63   Edited USER.README file

When an FTP client changes its working directory to ’USER’ 1, you will see the directory information configured 2 as FTP client messages (see Figure 6-64 on page 190).

```
230 BTHOMPS is logged on.  Working directory is "/u/bthomps".
Command:
cd 'user'               1
>>> CWD 'user'
250-
250-
250-This file was created to test FTP MVSINFO and can be used as          2
250-a reference for information that will be stored in hlq 'USER"
250-
250-See GC31-8773 z/OS V1r2.0 CS: IP Migration for details:
250-
250 "USER." is the working directory name prefix.
Command:
```

Figure 6-64   Display MVS directory information

HFSINFO / ANONYMOUSHFSINFO

This statement is used to tell the FTP server which file name contains the information to be displayed when a client uses the change directory command and initially enters a specific HFS directory. The HFS directory must contain a file name that matches the one specified as the parameter of the statement. A wild card character " * " can be used for pattern matching in the parameter. The first file that matches in the HFS directory will have its content presented to the FTP client. If no match is found nothing will be presented. The information file can contain any alphanumeric characters.

When a defined user initially changes into an HFS directory, the information from the file name in the HFS directory that matches the parameter on the HFSINFO statement will be displayed. When an anonymous user initially changes into an HFS directory, the information from the file name in the HFS directory that matches the parameter on the ANONYMOUSHFSINFO statement will be displayed. Only the initial entry (per FTP session) into the directory will cause the information file to be presented to the FTP client. Subsequent **cd** commands to the same directory during the same FTP session will not cause the information file to be presented again.

```
HFSINFO              README*
ANONYMOUSHFSINFO     README*
```

This function basically performs like MVSINFO and ANONYMOUSMVSINFO. The first file found whose name starts with "README" content will be presented to the FTP client; the use of the wild-card makes that possible. If the content of the file is longer than 100 lines, the messages after the 100th line will not be displayed, and a message indicating truncation will be displayed on the 101st line. See the sample below to show usage of this function.

We had files shown below in the HFS directory of /usr/lpp/dce/examples/xdsxom/cds_xmpl:

```
BTHOMPS:/usr/lpp/dce/examples/xdsxom/cds_xmpl: >ls -l
total 112
-rw-r--r--   2 FTPD2    OMVSGRP      483 Dec  6  1999 Makefile
-rw-r--r--   2 FTPD2    OMVSGRP     7257 Dec  6  1999 README
-rw-r--r--   2 FTPD2    OMVSGRP    42969 Dec  6  1999 cds_xmpl.c
```

Figure 6-65   HFS directory display

```
230 BTHOMPS is logged on.  Working directory is "/u/bthomps".
Command:
cd /usr/lpp/dce/examples/xdsxom/cds_xmpl

250-Note: You must have proper authority to
250-     perform the above.  If you do not,
250-     contact your DCE Administrator.
250-
250-Step #3:    After setting up your account, etc., log in
250-            to DCE.  Enter:
250-
250-            dce_login <principal> <password>
250-
250-Step #4:    Start the CDS_XMPL application with either
250-            the add (a), read (r), modify (m), list (l),
250-            or delete (d) parameter.
250-
250-The message was truncated.
250 HFS directory /usr/lpp/dce/examples/xdsxom/cds_xmpl is
the current working directory
Command:
```

Figure 6-66   Display HFS directory information

This output is produced during the FTP session, after changing into the directory that has a matching filename (README). The first 100 lines of the file is displayed, ending with "`250-The message was truncated.`".

6.4.10 Using the SIZE and MDTM commands

Commands **`SIZE`** and **`MDTM`** are client commands that are honored by the FTP server only if the extensions are properly configured in the FTP configuration file (FTP.DATA). The extensions to FTP are implemented at the server only. The CS for z/OS V1R2 IP client does not support the **`SIZE`** or **`MDTM`** commands. A client such as a Web browser sends these commands to the FTP server. Code the following in FTP.DATA to enable this support:

```
EXTENSIONS    SIZE
EXTENSIONS    MDTM
```

`SIZE` returns the number of bytes that would be transferred to the client if the client were to get the file from the server. This command is only valid for :

- FILETYPE of SEQ
- UNIX System Services HFS files
- Type image
- Structure of file
- Mode of stream

The number returned by the **`SIZE`** command varies according to current settings for type, structure, and file. When type is image, structure is file, and mode is stream the actual file size is equal to the byte transfer size.

Note: This command does not work for traditional MVS datasets.

```
site filetype=seq
 EZA1701I >>> SITE filetype=seq
 200 SITE command was accepted
 EZA1460I Command:
type i
 EZA1701I >>> TYPE I
 200 Representation type is Image
 EZA1460I Command:
structure f
 EZA1701I >>> STRU F
 250 Data structure is File
 EZA1460I Command:
mode s
 EZA1701I >>> MODE S
 200 Data transfer mode is Stream
 EZA1460I Command:
quote size /usr/lpp/internet/bin/httpd
 EZA1701I >>> size /usr/lpp/internet/bin/httpd
 213 57344
 EZA1460I Command:
```

Figure 6-67 Sample use of the SIZE command

As illustrated in Figure 6-67, you can see that in order to send the **SIZE** command to the FTP server it must be preceded by the **quote** command. The value returned is 57,344. Therefore if the file /usr/lpp/internet/bin/httpd is retrieved with the **ftp get** command the amount of data bytes will be 57,344 bytes.

MDTM (Modification Time) returns the date and time the file was last modified. This command is supported only for:

- FILETYPE of SEQ
- UNIX System Services HFS files

```
site filetype=seq
 EZA1701I >>> SITE filetype=seq
 200 SITE command was accepted
 EZA1460I Command:
quote mdtm /usr/lpp/internet/bin/httpd
 EZA1701I >>> mdtm /usr/lpp/internet/bin/httpd
 213 20020518150533
 EZA1460I Command:
```

Figure 6-68 Sample use of the MDTM command

As illustrated in Figure 6-68, you can see that in order to send the **MDTM** command to the FTP server it must be preceded by the **quote** command. The value returned is 20020518150533. Therefore the time /usr/lpp/internet/bin/httpd was last modified was May 18, 2002 at 15:05:33. Other responses from the **MDTM** command are `file does not exist, the modification time is unavailable`, or some other standard ftp 550- message.

6.4.11 Using the STAT and SITE commands

The **STAT** command (short for status) can be used to view configuration statement settings of a remote host (FTP server) from the local host (FTP client). This command allows the client to validate the settings prior to an FTP transfer, and is also key in problem determination of failed transfers. When the **STAT** command is issued current settings of the configuration statements are shown. The initial values of the statements can be initialized from the FTP.DATA configuration file. Refer to "Default FTP Server SITE parameters (FTPDATA)" on page 129 for information on selecting a configuration file for the server. Figure 6-69 is output from the **STAT** command:

```
STAT
211-Server FTP talking to host 9.24.104.113, port 1060
211-User: BTHOMPS  Working directory: /u/bthomps
211-The control connection has transferred 5126 bytes
211-There is no current data connection.
211-The next data connection will be actively opened
211-to host 9.24.104.113, port 1060,
211-using Mode Stream, Structure File, type ASCII, byte-size 8
211-Automatic recall of migrated data sets.
211-Automatic mount of direct access volumes.
211-Auto tape mount is allowed.
211-Inactivity timer is disabled
211-UCOUNT is 5
211-VCOUNT is 50
211-ASA control characters in ASA files opened for text processing
211-will be transferred as ASA control characters.
211-Trailing blanks are removed from a fixed format
211-data set when it is retrieved.
211-Data set mode.  (Do not treat each qualifier as a directory.)
211-Primary allocation 1 track.  Secondary allocation 1 track.
211-Partitioned data set will be created with 27 directory blocks
211-FileType JES (MVS Job Spool). JES Name is JES2
211-Number of access method buffers is 5
211-RDWs from variable format data sets are discarded.
211-SITE DB2 subsystem name is DB2
211-Data not wrapped into next record.
211-JESLRECL is 80
211-JESRECFM is Fixed
211-JESINTERFACELEVEL is 2
211-JESENTRYLIMIT is 20
211-JESOWNER is BTHOMPS
211-JESJOBNAME is BTHOMPS*
211-JESSTATUS is ALL
211-SMS is active.
211-Data sets will be allocated on SPLEX1.
211-New data sets will be catalogued if a store operation abends
211-Single quotes will override the current working directory
211-UMASK value is 027
211-Process id is 16777497
211-Checkpoint interval is 0
211-Record format VB, Lrecl: 256, Blocksize: 6233
```

Figure 6-69 Sample output STAT command

You may have to issue **QUOTE STAT** to produce the desired output, if the FTP client you are using does not send the **STAT** command to the CS for z/OS IP FTP server. To display

JES statements with the **STAT** command the FILETYPE must be set to JES. The FILETYPE statement specifies the mode of operation of the server. For normal file transfers FILETYPE should be set to SEQ, which is the default value.

STAT is used to show configuration settings, and the **SITE** command allows you to set or change the values of these settings. The initial settings can be defined in FTP.DATA. If a statement is not coded in the configuration file the system hardcoded default is used. Review the following to see how the new keywords values were set:

```
site ucount=5
 EZA1701I >>> SITE ucount=5
 200 SITE command was accepted
 EZA1460I Command:
site vcount=50
 EZA1701I >>> SITE vcount=50
 200 SITE command was accepted
 EZA1460I Command:
site volume=splex1
 EZA1701I >>> SITE volume=splex1
 200 SITE command was accepted
 EZA1460I Command:
```

Figure 6-70 The SITE command

As for most **SITE** commands, you can revert to the FTP default value by simply entering the keyword without a value. For example entering `SITE UCOUNT` will revert to the FTP default.

Refer to the *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 and the *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780 for further details on these statements.

Refer to 6.4.12, "JES interface" on page 195 and 6.5.7, "FTP client in a batch job" on page 212 for examples and usage of the JES statements.

6.4.12 JES interface

With this interface enabled you can submit jobs and can view these jobs allowable by the system JESSPOOL RACF class. You are also able to display held/dup jobs on the internal reader, CPU time of running jobs, and number of sysout data sets for a complete job.

Use the **SITE** command to set `filetype=jes`, before issuing any of the JES interface commands. The FILETYPE statement specifies the mode of operation of the server.

To enable this function set JESINTERFACELEVEL to 2 in FTP.DATA. You can now use the DIR command to display a list of started tasks.

```
site jesjobname=*
 EZA1701I >>> SITE jesjobname=*
 200 SITE command was accepted
 EZA1460I Command:
site jesowner=ibmuser
 EZA1701I >>> SITE jesowner=ibmuser
 200 SITE command was accepted
 EZA1460I Command:
dir
 EZA1701I >>> PORT 9,39,64,151,8,91
 200 Port request OK.
 EZA1701I >>> LIST
 125 List started JESJOBNAME=*, JESSTATUS=ALL and JESOWNER=IBMUSER
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I ASCHINT  STC00968 IBMUSER  ACTIVE STC
 EZA2284I ASCHINT  STC00966 IBMUSER  ACTIVE STC
 EZA2284I ASCHINT  STC00965 IBMUSER  ACTIVE STC
 EZA2284I ASCHINT  STC00964 IBMUSER  ACTIVE STC
 250 List completed successfully.
EZA1460I Command:
dir STC00964
 EZA1701I >>> PORT 9,39,64,151,8,92
 200 Port request OK.
 EZA1701I >>> LIST STC00964
 125 List started JESJOBNAME=*, JESSTATUS=ALL and JESOWNER=IBMUSER
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I ASCHINT  STC00964 IBMUSER  ACTIVE STC
 EZA2284I --------
 EZA2284I          STEPNAME ++++++++ PROCNAME     ASCHINT
 EZA2284I          CPUTIME  1347769.376 ELAPSED TIME 1995358.962
 250 List completed successfully.
```

Figure 6-71 DIR command displaying STCs

See the following for a sample of using the **`list Jxx`** command with a JOB:

```
list j2080
 EZA1701I >>> PORT 9,39,64,151,8,72
 200 Port request OK.
 EZA1701I >>> LIST j2080
 125 List started JESJOBNAME=*, JESSTATUS=ALL and JESOWNER=BTHOMPS
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I DDNTEST  JOB02080 BTHOMPS  OUTPUT A        RC=000
 EZA2284I --------
 EZA2284I          ID  STEPNAME PROCSTEP C DDNAME   BYTE-COUNT
 EZA2284I          001 JES2              X JESMSGLG      1351
 EZA2284I          002 JES2              X JESJCL         548
 EZA2284I          003 JES2              X JESYSMSG      1596
 EZA2284I          004 STEP01            X OUTPUT        1389
 EZA2284I 4 spool files
 250 List completed successfully.
```

Figure 6-72 Using the list Jxx command

Refer to 6.5.11, “FTP server interface to JES” on page 221 for a sample of retrieving sysout. And also see *z/OS V1R2.0 CS: IP User’s Guide and Commands*, SC31-8780 for a sample of how to retrieve sysout automatically.

The **LIST** and **DIR** commands are essentially equivalent. They require more system resources because they provide spool information. If you only need a list of job names use the NLST or its equivalent LS. The following will illustrate setting the JESENRYLIMIT to 5, to only list the first five jobs matching the filtering criteria. Then output from the **LS** and **LIST** commands.

```
site jesentrylimit=5
EZA1701I >>> SITE jesentrylimit=5
200 SITE command was accepted
EZA1460I Command:
site jesowner=*
EZA1701I >>> SITE jesowner=*
200 SITE command was accepted
EZA1460I Command:
site jesjobname=bpx*
EZA1701I >>> SITE jesjobname=bpx*
200 SITE command was accepted
EZA1460I Command:
ls
EZA1701I >>> PORT 9,39,64,151,8,141
200 Port request OK.
EZA1701I >>> NLST      1
125 Nlst started JESJOBNAME=BPX*, JESSTATUS=ALL and JESOWNER=*
EZA2284I STC02133
EZA2284I STC00916
EZA2284I STC02134
EZA2284I STC02132
EZA2284I STC02131
250-JESENTRYLIMIT of 5 reached Additional entries not displayed
250 Nlst completed successfully
EZA1460I Command:
list
EZA1701I >>> PORT 9,39,64,151,8,140
200 Port request OK.
EZA1701I >>> LIST
125 List started JESJOBNAME=BPX*, JESSTATUS=ALL and JESOWNER=*
EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
EZA2284I BPXAS    STC02133 ++++++++ OUTPUT STC      RC=000 2 spool
EZA2284I BPXAS    STC00916 ++++++++ OUTPUT STC      RC=000 2 spool
EZA2284I BPXAS    STC02134 ++++++++ ACTIVE STC
EZA2284I BPXAS    STC02132 ++++++++ ACTIVE STC
EZA2284I BPXAS    STC02131 ++++++++ ACTIVE STC
250-JESENTRYLIMIT of 5 reached Additional entries not displayed
 250 List completed successfully.
 EZA1460I Command:
```

Figure 6-73 Showing the jesentrylimit command

1 The **LS** command was entered, but the generic information message EZA1701I displays **NLST**; the **LS** equivalent.

Since JESENTRYLEVEL = 2 allows client access, security is needed to control this access. SAF resources mirror SDSF for the JES2 environment. JES3/non-SDSF installations require SAF setup. To tighten up on security implement the changes listed below:

- Use SDSF SAF Resources
  - JESJOBNAME ISFCMD.FILTER.PREFIX
  - JESJOBOWNER ISFCMD.FILTER.OWNER

- JESSTATUS
  - ISFCMD.DSP.INPUT.<jesx>
  - ISFCMD.DSP.ACTIVE<jesx>
  - ISFCMD.DSP.OUTPUT.<jesx>
- Defaults
  - JESJOBNAME=<userid>
  - JESOWNER=<userid>*
  - JESSTATUS=ALL,OUTPUT,INPUT

6.4.13 User exits

The following is a list of exits available for FTP:

- FTPSMFEX
- FTCHKIP
- FTCHKPWD
- FTCHKCMD
- FTCHKJES
- FTPOSTPR

Refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for detailed information regarding these exits.

FTPSMFEX

This user exit is called before an SMF type 118 record containing session stats is written. You can then modify the record and control if the record should be written to SMF.

FTCHKIP

This user exit is called when a user attempts to log in to the FTP server or when a new connection is established. The IP address and PORT number of both the local and remote host is passed to the user exit. It can be used validate whether port numbers and IP addresses are allowed to access an FTP server.

FTPCHKPWD

This user exit is called after the users password or e-mail address is entered. The following information is passed to the exit:

- The user ID
- The user password or an asterisk (*) if an e-mail address is entered instead of a password
- A userdata buffer
- If an e-mail address is entered to log in, the userdata buffer contains the e-mail address.

This exit can be used to restrict access to a site based on user ID and password.

FTPCHKJES

This exit is called if the server is in FILETYPE=JES mode and the client tries to submit a job. The user ID and the job being submitted are passed to the exit. The exit can allow or refuse the job to be submitted to the JES internal reader. If the job is refused message 550 User Exit refuses this job to be submitted by userid is sent to the user.

FTCHKCMD

The FTCHKCMD user exit is called whenever the user enters an FTP command.The exit is passed the user ID, current directory type (MVS or HFS), working directory name, filetype setting (JES, SQL or SEQ), the FTP command to be executed, command arguments, and a buffer. The following chart shows the offset of each item in the parameter list:

Table 6-9 FTCHKCMD parameter list

Offset	Description
+0	Return 0 to accept the command or to pass new arguments to the command. Return non-zero value to reject the command.
+4	Pointer to a word containing the number of following parameters.
+8	Pointer to the 8-byte user ID the is logged on.
+12	Pointer to the 8-byte command being entered.
+16	Pointer to a string containing arguments after the command. The first halfword of the string contains the number of characters that follow.
+20	4-byte character string with current directory type: MVS or HFS.
+24	4-byte character string with current FILETYPE: SEQ, JES, or SQL.
+28	Buffer with current directory value, first bytes hold length of remaining buffer. 1102-byte output buffer in which to return modified argument strings. The first two bytes must be initialized to the length of the returned command string.
+32	1102-byte output buffer in which to return modified argument strings. You can modify the arguments passed to the command by placing the modified arguments in this buffer. The first two bytes must be initialized to the length of the returned command string.

For example, if a user issues a **`DIR *`** command, the exit can prevent the command from being executed and reply with a message `500- User Exit denies Userid xxxxx from using Command yyy` or the exit can change the " * " parameter into "userid* ". This can prevent the **`DIR *`** command from hanging the catalog. Note also that only the parameter can be changed, not the command. There are three sample FTCHKCMD exits included with CS for z/OS V1R2 IP. All the members are in *hlq*.SEZAINST. FTCHKCMD and FTCHKCM1 deal with restricting erroneous **`DIR`** command with the '*' wildcard. FTCHKCM2 implements customized SITE options.

FTPOSTPR

The exit (FTPOSTPR) gets control whenever a file transfer attempt completes successfully or unsuccessfully. It will give a return code indicating a completion status of the transfer. This exit allows a customer to perform some post processing after FTP processing in user-written batch jobs. This user exit is passed the user ID, client's IP address, client's port number, current directory type, length of the parameter string, current working directory, current file type, FTP reply code, a buffer containing the FTP reply string, FTP command code, current CONDDISP setting, and the close reason code. The following chart show the offset of each item in the parameter list:

Table 6-10 FTPOSTPR parameter list

Offset	Description
+0	Pointer to the word with the user exit return code
+4	Pointer to the number of parameters passed in
+8	Pointer to the 8-byte buffer containing the user ID

Offset	Description
+12	Pointer to the 4-byte client IP address
+16	Pointer to the 2-byte client port number
+20	Pointer to the 4 byte character string with current directory type: MVS or HFS (left justified)
+24	Pointer to a buffer that contains the current directory value, the first two bytes hold the length of the remaining buffer
+28	Pointer to the 4 character byte field that contains the current filetype: SEQ, JES, SQL (left justified)
+32	Pointer to the 3 character byte field that contains the current reply code
+36	Pointer to buffer that contains FTP reply string; first two bytes contain the length of the remaining buffer
+40	Pointer to the 4 byte field that contains the current FTP command code
+44	Pointer to the 1-character byte field that contains the current CONDDISP setting- C for catalog, D for delete
+48	Pointer to the 4 byte binary field that contains the close reason code: 0 -- transfer completed normally 4 -- transfer aborted before data connection was established 8 -- transfer aborted with socket communication errors 12 -- transfer aborted after data connection was established 16 -- transfer aborted with SLQ file errors after data connection was established

User exit FTPOSTPR implementation

The sample exit source code, EZAECCM3, is also located in *hlq*.SEZAINST. You will see the sample user exit and the JCL used to compile and link-edit it in Appendix E, "FTP user exits and sample code" on page 525.

The user exit load modules must be placed in an APF-authorized library to which the FTP server has access via STEPLIB, LNKLST, or LPA. After the compile and link we updated the FTP started task procedure as follows:

```
//FTPDA  PROC MODULE='FTPD',PARMS=''
 //***************************************************************
 //FTPDA  EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
 //        PARM=('POSIX(ON) ALL31(ON)',
 //          'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")','/&PARMS')
 //*         'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")','/')
 //*         'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=T&SYSCLONE.ATCP")',
 //*         '/&PARMS')
 //CEEDUMP  DD SYSOUT=*
 //SYSFTPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(FTSD&SYSCLONE.A)
 //SYSTCPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(TCPD&SYSCLONE.A)
 //*SYSFTSX DD DISP=SHR,DSN=TCPIP.FTPKANA.TCPXLBIN
 //*   STEPLIB FOR USER EXITS
 //STEPLIB DD DISP=SHR,DSN=TCPIP.TCPPARMS.VTAMLIB              1
```

Figure 6-74   FTPD started procedure with steplib

1 is the load module library that contains the FTPOSTPR load module. This library has to be APF-authorized and program controlled, if you have activated the program control.

To verify that the exit was functioning properly we initiated an FTP session and executed a **put** command to store a file on the CS for z/OS V1R2 IP FTP server. Then we reviewed /tmp/syslogd.log file to see the posted messages:

```
FTPOSTPR: FTP process completed with rc of 0, userid 'KAKKY ', client IP 09186A1F,
client port 1569, reply code '250', and reply string 'Transfer completed successfully'
```

This was a successful transfer with a rc=0. A failure will return a non-zero value: for example, a get on an non-existing file will have rc=4. You can see that the user name, hex IP address (translates to 9.24.106.31), port number, FTP standard 250- message, and the actual message `Transfer completed successfully` are recorded by this exit.

6.4.14 Using the directory command

The DIR command considers the users access to the catalog before returning the requested entries. Users can thus not see any datasets in a protected catalog without having the relevant read access to that catalog. As can be seen in Figure 6-75, the user does not have access to 'test.' datasets, and receives the 'no datasets found' response. The same responses are received when using the LS, MGET, MPUT and MDEL commands.

```
dir 'test.*'
 >>> PORT 9,12,6,30,4,12
 200 Port request OK.
 >>> LIST 'test.*'
 550 No data sets found
 Command:
```

Figure 6-75   Directory listing with '' at the end of the qualifier*

The use of ’**’ in the DIR command represents one or more qualifiers in a dataset. Figure 6-76 shows the result of the wild card, where all datasets having its first qualifier as ‘tcpipb’ followed by one or more qualifiers and having its final qualifier as’ files’ being displayed.

```
Command:
dir 'tcpipb.**.files'
 >>> PORT 9,12,6,30,4,25
 200 Port request OK.
 >>> LIST 'tcpipb.**.files'
 125 List started OK.
 Volume Unit    Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
 SBOX02 3390   2002/04/29  1    1  FB      80 27920  PO  'TCPIPB.EXAMPLE.TEMP.F
 ILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.EXAMPLE.TEMP2.
 FILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP.FILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP2.FILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP3.FILES'
 250 List completed successfully.
 Command:
```

Figure 6-76 Double asterisk

The single asterisk wild card ‘*’ can be embedded in between the text of a qualifier as shown in Figure 6-77. This can be done, as opposed to putting the ‘*’ at the beginning or the end of the text of a qualifier. In this case only datasets with a single qualifier between the ‘tcpipb’ and ‘files’ are displayed.

```
Command:
dir 'tcpipb.*.files'
 >>> PORT 9,12,6,30,4,31
 200 Port request OK.
 >>> LIST 'tcpipb.*.files'
 125 List started OK.
 Volume Unit    Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP.FILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP2.FILES'
 SBOX02 3390   **NONE**    1    1  FB      80 27920  PO  'TCPIPB.TEMP3.FILES'
 250 List completed successfully.
 Command:
```

Figure 6-77 Directory Display with ‘’ between qualifiers*

6.5 Client customization and usage

We now provide details into the customization of the FTP client.

6.5.1 Using FTP client in z/OS

CS for z/OS IP ships a C-based FTP client. Since this client is based on the FTP server, the following functions that are part of the server functions are now available for FTP clients:

- The UNIX System Services shell environment support
- The hierarchical file system files support

- Client can read and write to tape
- Support record structure
- Client can transfer data encoded in Unicode (UCS-2)

The messages used by FTP client are stored in message catalogs to allow message translation. Therefore, the FTP client is NLS enabled.

You can run FTP client in several environments such as:

- From TSO command line
- Batch job
- TSO REXX
- UNIX System Services shell command
- UNIX System ServicesREXX

Note that if you enter an FTP command from the UNIX System Services shell, the FTP flags must be entered in lowercase. From TSO, you can use both lowercase and uppercase. Case sensitivity includes names of files in the UNIX System Services environment.

Each FTP client needs an OMVS segment defined in your security database. However, you can use a default OMVS segment instead.

6.5.2 FTP client NETRC data set

In many UNIX environments, you have the option of creating a file that holds the user ID and password you use when you connect to other UNIX systems. The FTP client supports a similar implementation, which allows you to use a userid.NETRC data set ($HOME/.netrc, in the UNIX System Services shell) as an alternative to specifying your user ID and password every time you want to connect to an FTP server at a remote host.

```
machine  9.24.104.47   login jedeye  password secret
machine 9.24.104.26 login jedeye password secret
```

Figure 6-78 userid.NETRC

To use the z/OS FTP client to connect to host 9.24.104.26, for example, type in:

```
ftp 9.24.104.26
```

You would see the following sequence of events on your TSO terminal:

```
IBM FTP CS V1R2
FTP: using TCPIPB
EZA1554I Connecting to:  9.24.104.26 port: 21.
220 rs60001 FTP server (Version 4.1 Sun Jul 28 12:35:09 CDT 1996) ready.
EZA1701I >>> USER jedeye  1
331 Password required for jedeye.
EZA1701I >>> PASS       2
230 User jedeye logged in.
EZA1460I Command:     3
```

Figure 6-79 FTP with userid.NETRC

Your user ID will be sent automatically **1**, as well as your password **2**. Your first prompt will be at **3**.

Whether this approach is a valid implementation in your security environment is something you have to consider. The user ID and passwords are kept in hlq.NETRC data set in clear text. You should as a minimum, ensure that the data set is RACF protected, so only you can read it. Refer to “FTP client in a batch job” on page 212 to review sample JCL using a NETRC DD card.

6.5.3 Setting USER level FTP Server options

This is done using a configuration file that defined per user. The filename ends with the name "FTPS.RC", and it is searched for as follows:

1. TSO_prefix for the userid.
2. userid as a high-level qualifier
3. $HOME directory, filename ftps.rc.

Remember, the HFS environment is case-sensitive, you need to make sure that ftps.rc is in lowercase.

The following can be entered into this file as is shown in Figure 6-80 on page 204

- SITE commands.
- CWD (Change Working Directory) command.
- CD (Change Directory) command
- Comments ‘;’ lines.

```
; This is a sample configuration file for FTPS.RC
SITE automount
SITE debug=all
; Setting directory to a dataset.
CD garthm.ftp.cnf1
```

Figure 6-80 GARTHM.FTPS.RC dataset

The login process as shown in Figure 6-81 on page 205 shows the FTPS.RC configuration file which is selected. It also prefixes messages with a 230 message number. This can also be used to debug any possible errors in your configuration file.

```
garthm
 >>> USER garthm
 331 Send password please.
 PASSWORD:


 >>> PASS
 230-Processing FTPS.RC configuration file - GARTHM.FTPS.RC
 230-SITE command was accepted
 230-Active server traces - FLO CMD PAR INT ACC UTL SEC FSC(1) SOC(1) JES SQL
230-SITE command was accepted
 230-CWD cmd failed : EDC5129I No such file or directory.
 230-"GARTHM.GARTHM.FTP.CNF1." is the working directory name prefix.
 230 GARTHM is logged on.  Working directory is "GARTHM.GARTHM.FTP.CNF1.".
 Command:
```

Figure 6-81   Session initiation with ftps.rc file

6.5.4 LOCSTAT and LOCSITE commands

The **LOCSTAT** command (short for local status) can be used to view configuration statement settings of the local host (FTP client). This command allows the client to validate the settings prior to an FTP transfer, and is also key in problem determination of failed transfers. When the **LOCSTAT** command is issued current settings of the configuration statements are shown. The initial values of the statements can be initialized from the FTP.DATA configuration file. Refer to “Default FTP Client LOCSITE parameters” on page 128 for information on selecting a configuration file for the client. Following is output from the **LOCSTAT** command:

```
locstat
EZA1600I Trace: FALSE, Send Port: TRUE 1
EZA1601I Send Site with Put command: TRUE
EZA2677I Connected to:9.24.104.43, Port: FTP control (21)2
EZA1605I Local Port: 1295
EZA1606I Data type:a, Transfer mode:s, Structure:f 3
EZA2098I Automatic recall of migrated data sets.
EZA2100I Automatic mount of direct access volumes.
EZA2101I Data set mode. (Do not treat qualifier as a directory)
EZA2137I Primary allocation 1 track, Secondary allocation 1 track
EZA2138I Partitioned data sets will be created with 27 directory
EZA2103I FileType is SEQ (Sequential - the default)
EZA2141I Number of access method buffers is 5.
EZA2145I RDW's from VB/VBS files are discarded.
EZA2148I DB2 subsystem name is DB2
EZA2152I Volid of Migrated Data Sets is MIGRAT
EZA2154I Trailing blanks in records read from RECFM F datasets...
EZA2535I Record format: VB, Lrecl: 256, Blocksize: 6233.   4
EZA2080I Data sets will be allocated on PUBTST,PUBPRD
EZA2801I Data not wrapped into next record.
EZA2494I Checkpoint interval is 0
EZA2425I RESTGet    Checkpoint data set will be opened for get.
EZA2817I No automatic mount of tape volumes. 5
EZA2809I CCONNTIME is 30   6
EZA2810I DATACTTIME is 120 7
EZA2811I DCONNTIME is 120  8
EZA2812I INACTTIME is 300  9
EZA2813I MYOPENTIME is 60  10
EZA2814I UCOUNT is 3   11
EZA2815I VCOUNT is 20  12
EZA2689I Prompting: ON, Globbing: ON
EZA2719I ASA control characters transferred as ASA control char
EZA2720I New data sets catalogued if a store operation abends
EZA2722I Single quotes will override the current working directory
EZA2724I UMASK value is 027
EZA2819I Data connections for the client are not firewall friendly
EZY2640I Using 'BTHOMPS.FTP.DATA' for local site configuration...
EZA1460I Command:
```

Figure 6-82   The LOCSTAT command

1 Trace : Trace setting which can be true or false depending on whether a debug command was issued or the FTP was started with the trace option. The current sendport setting is also shown, which can be changed by using the sendport command.

2 Connected to : IP address of the FTP server you are connected to

3 Data type : a (ascii), e (edcdic), i (image), b(dbcs) , or u (unicode))

Transfer mode : s (stream), b (block), or (compressed)

Structure. f (file), or r (record)

4 Record Format : RECFM (record format) for new datasets

LRECL (logical record length)

BLKSIZE (blocksize)

5 AUTOMOUNT : True: permits automatic mounting of tape volumes for data sets on volumes that are not mounted. False: prevents automatic mounting of tape volumes for data sets on volumes that are not mounted. False is the hard-coded default for the client. The value can be set in FTP.DATA and cannot be changed by a client with the LOCSITE command.

6 CCONNTIME : Defines the amount of time to wait after attempting to close a control connection before terminating the connection and reporting an error (default = 30).

7 DATACTTIME : Defines the amount of time to wait after attempting to send or receive data before terminating the connection and reporting an error (default=120).

8 DCONNTIME : Defines the amount of time to wait after attempting to close a data transfer before terminating the connection and reporting an error (default=120).

9 INACTTIME : This statement is used to set the inactivity timer to a specified number of seconds. Any client control connection which is inactive for the amount of time specified on the statement is closed by the server. A value of zero disables the timer. A value greater than zero enables the timer. The valid range is (0 - 86400). The default is 300.The value can be set in FTP.DATA and cannot be changed by a client with the LOCSITE command.

10 MYOPENTIME : Defines the amount of time to wait for a session to open before terminating the attempt and reporting an error (default=60)

11 UCCODE : Specifies the number of devices to allocate. Valid range is (1-59) or P. The P signifies a parallel mount request. When specified without a value, the FTP server does not specify a unit count when allocating data sets. If this statement is not coded in FTP.DATA or set with the LOCSITE command, it will not appear in the output of STAT.

12 VCOUNT : Specifies number of volumes an allocated data set may span. Valid ranges (1-255). When specified without a value, the FTP server uses a volume count of 50 when allocating data sets. LOCSITE can be used to set this value.

The timer values which are described in points 6 through 10 above can be set when the FTP command is issued to initiate a session. The command ‘`FTP ip_address (timeout nn`’ will set the values of the aforementioned timers and that of FTPKeepAlive to ’nn’. These timers can also be individually set in FTP.DATA.

The `LOCSITE` subcommand to specify information that is used by the local host to provide services specific to that host system.The initial FTP session settings are defined in FTP.DATA. If a statement is not coded in this configuration file, the system hardcoded default is used. As for most `LOCSITE` commands, you can revert to the FTP default value by simply entering the keyword without a value. For example entering `LOCSITE UCOUNT` will revert to the FTP default.

Refer to the *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 and the *z/OS V1R2.0 CS: IP User’s Guide and Commands*, SC31-8780 for further details on these statements.

6.5.5 FTP SUNIQUE command

This command can be used to assure that existing files on the remote host are not overwritten on an FTP `put` or `mput` command. It is basically used to change the method of storing files on the remote host. Following is a syntax diagram for the command:

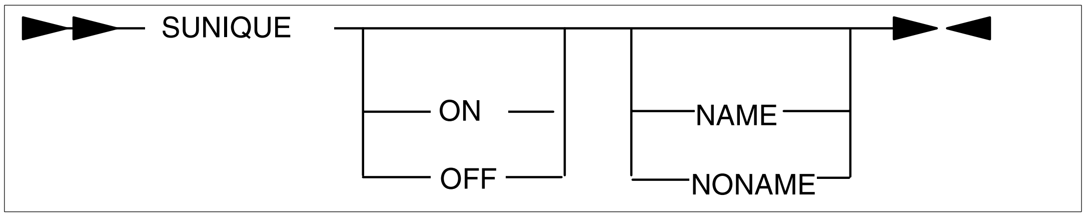


Figure 6-83 FTP SUNIQUE command syntax

The default setting is OFF, NAME and the FTP server uses a store command (**STOR**) with **put** and **mput**. If the file name on the remote host exists, it is overwritten.

Table 6-11 FTP SUNIQUE command syntax description

Parameter	Description
blank	Toggles on/off (default is off)
ON	Turns on store unique
OFF	Turns off store unique
NAME	When specified with ON or OFF, instructs the client to include a name when sending a store-unique command to the server (default)
NONAME	When specified with ON or OFF, instructs the client to omit a name when sending the store-unique command to the server.

If SUNIQUE is set to ON, FTP uses a store-unique command (**STOU**) with the **put** and **mput** command. This prevents you for overwriting an existing file. If the default setting of NAME is in effect, a string will be sent to the server with the store-unique command. The file will be created with a unique name, then FTP sends the unique name to the local host where it is displayed. The following is the screen output that shows usage of the **SUNIQUE** command.

Note: This command (**SUNIQUE**) has no effect on load module FTP transfers. The load module name on the remote host must be the same as the name on the local host. Refer to 6.4.8, "Transferring load modules" on page 184 for further detail.

```
EZA1460I Command:
sunique        1
 EZA1626I Store unique is ON
 EZA1460I Command:
put readme     2
 EZA1701I >>> SITE VARrecfm LRECL=80 RECFM=VB BLKSIZE=27920
 200 SITE command was accepted
 EZA1701I >>> PORT 9,24,104,43,4,6
 200 Port request OK.
 EZA1701I >>> STOU readme
 125 Storing data set /u/bthomps/readme1 (unique name)  3
 250 Transfer completed successfully.
 EZA1617I 38 bytes transferred in 0.070 seconds.
 Transfer rate 0.54 Kbytes/sec.
```

Figure 6-84 SUNIQUE command

1 The **sunique** command is issued to turn on the store-unique option. It is issued without any parameters and since it is off by default, it turns on. NAME is also the default. The initial **SUNIQUE** command is equivalent to issuing **sunique on name**.

2 The file readme already exists on the remote host.

3 The **STOU readme** command stores the file with a unique name /u/bthomps/readme1 instead of overwriting it, then informs you in message #125.

6.5.6 Using FTP client in the z/OS UNIX shell environment

The FTP command can be issued from the UNIX System Services shell. In the UNIX System Services shell, the FTP command and FTP flags must be entered in lowercase. You can also use the traditional MVS command options in the UNIX System Services shell. However, you have to precede the left parenthesis with an escape character such as backslash(\):

```
ftp 9.24.104.74 \(trace
```

In the UNIX System Services shell, the directory where the FTP command was issued is the local working directory. On the other hand, if the FTP client is started in a TSO environment, the user's TSO prefix or logon user ID will be the initial local working directory.

When you use the FTP client in the UNIX System Services shell environment, you can use the **! FTP** subcommand to invoke the UNIX System Services shell with a UNIX System Services command.

Here are examples of the FTP command in the UNIX System Services shell.

```
BTHOMPS:/u/bthomps: > ftp 9.24.104.43     1
FTP: using TCPIPB
Connecting to:  9.24.104.43 port: 21.
220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe, 17:34:35 on 2002-05-26.
220-*********************************************************
220-*                                                       *
220-*          Welcome to z/OS ITSO Raleigh SC63            *
220-*     You accessed this system via CS for z/OS V1R2     *
220-*                                                       *
220-*    This is the /etc/ftpbanner file for ftp server use *
220-*                                                       *
220-*********************************************************
220-This is wtscoe.itso.ral.ibm.com on Sun May 26 17:34:35 2002
220-For administrative assistance contact support@helpdesk.com
220-
220 Connection will not timeout.
>>> USER bthomps
331 Send password please.
>>> PASS
230 BTHOMPS is logged on.  Working directory is "BTHOMPS." 2
Command:
bin
>>> TYPE I
200 Representation type is Image
Command:
get a.out a.out.bin
>>> PORT 9,24,104,43,5,20
200 Port request OK.
>>> RETR a.out
125 Sending data set BTHOMPS.A.OUT
250 Transfer completed successfully.
86016 bytes transferred 0.350 seconds. Transfer rate 245.76KB/sec
Command:
quit
>>> QUIT
221 Quit command received. Goodbye.
BTHOMPS:/u/bthomps: >
```

Figure 6-85   FTP in Unix System Services

1 Issue the FTP command.

2 The current working directory in the remote host is the TSO prefix defined in the PROFILE of the login user ID.

In Figure 6-86 on page 211, the FTP client changes the FTP transfer modes, and gets an MVS VB data set as a record structured file in binary mode. Both the FTP server and FTP client support the transfer mode and file structure shown below.

```
Command:
mode b        1
>>> MODE B
200 Data transfer mode is Block
Command:
mode c        2
>>> MODE C
200 Data transfer mode is Compressed
Command:
mode s        3
>>> MODE S
200 Data transfer mode is Stream
Command:
type i        4
>>> TYPE I
200 Representation type is Image
Command:
struct r      5
>>> STRU R
250 Data structure is Record
Command:
get test.vb test.vb.bin
>>> PORT 9,24,104,231,4,11
200 Port request OK.
>>> RETR test.vb
125 Sending data set GARTHM.TEST.VB
250 Transfer completed successfully.
4197 bytes transferred in 0.350 seconds.  Transfer rate 11.99 Kbytes/sec.
Command:
! ls -laF /tmp/test* 6
-rw-rw-rw-   1 GARTHM    DCEGRP      4080 Feb 25 18:21 /tmp/test.vb.bin
Command:
quit
>>> QUIT
221 Quit command received. Goodbye.
```

Figure 6-86   Changing modes

1 Set the data transfer mode to BLOCK mode. You may also use the **BLock FTP** subcommand.

2 Set the data transfer mode to COMPRESS mode. You may also use the **COMpress FTP** subcommand.

3 Set the data transfer mode to STREAM mode. You may also use the **STREam FTP** subcommand.

4 Set the data transfer type to IMAGE mode. You may also use the **Binary FTP** subcommand.

5 Set the file structure to RECORD. You may also use the **RECord FTP** subcommand.

6 Invoke the UNIX System Services shell with the **LS** command, which displays the list of files in the current working directory. For more information on FTP transfer mode, data type, and data structure, see “Transfer mode, data type and data structure” on page 145.

Figure 6-87 on page 212 shows how to work with directories on a local host, and how to issue the UNIX System Services commands from an FTP client session.

```
Command:
lcd /tmp     1
HFS directory /tmp is the current working directory.
Command:
lpwd         2
Local directory name set to hierarchical file /tmp
Command:
! pwd        3
/u/garthm
Command:     3
! cd /tmp
Command:
! pwd        3
/u/garthm
```

Figure 6-87 Issuing Unix system services commands from FTP

1 Changes the current directory on the local host for the FTP session.

2 Displays the name of the current working directory on the local host. The current directory has been changed from the default directory to /tmp directory.

3 Invokes the UNIX System Services shell with the **PWD** command, which displays the current working directory name. Since the **!** command invokes the new UNIX System Services shell every time it is issued, the passed UNIX System Services command is always started in the user's $HOME directory (in this case /u/garthm). Therefore if you issue the UNIX System Services **cd** command to change the working directory, the current working directory information cannot be inherited to the succeeding UNIX System Services commands.

6.5.7 FTP client in a batch job

The FTP server has an interface so that it can be used to submit jobs to the z/OS internal reader from a remote client system using the TCP/IP network. This allows job submission to be automated and eliminates the need to log on to MVS. The CS for z/OS IP FTP JES interface supports JES2 and JES3 and allows the following functions:

- Submit jobs (JCL and data) from any TCP/IP client to MVS.
- Display current execution status of submitted jobs.
- Receive output of completed jobs at the client.

Note: Receiving of output does not purge output on the JES output queue. Output spool files can be explicitly purged via FTP commands. Only output in HELD status can be retrieved.

- Selectively deletes jobs on the MVS server JES queue.

Figure 6-88 on page 213 contains sample JCL that can be used to submit an FTP command as a batch job:

```
//jobname  JOB MSGCLASS=O,MSGLEVEL=(1,1),CLASS=A,NOTIFY=&SYSUID
//FTP     EXEC PGM=FTP,REGION=3M,PARM='/-d (EXIT'            3
//STEPLIB  DD  DSN=TCPIP.SEZALINK,DISP=SHR
//SYSTCPD  DD  DSN=TCP.TCPPARMS(TDATA03A),DISP=SHR
//SYSFTPD  DD  DSN=TCP.TCPPARMS(FDATACLN),DISP=SHR
//SYSPRINT DD  SYSOUT=*                                      1
//OUTPUT   DD  SYSOUT=*,DCB=(LRECL=160,RECFM=FB,BLKSIZE=16000) 1
//INPUT    DD  *                                             1
9.24.104.26
user1 passwd1    ; note - userid and password is case sensitive
sendsite  2      ; toggle off to suppress sending site command
cd /tmp/
put 'user1.ftptst1' ftp.test1
get ftp.test1 'user1.ftptst2' (replace
quit
/*
```

Figure 6-88 Batch JCL for FTP client

1 You must configure the following three DD statements:

- SYSPRINT DD
- INPUT DD
- OUTPUT DD

You may specify MVS data sets on these DD statements, but you have to ensure that the data set for the OUTPUT DD statement has an LRECL of 160 with any block size that is a multiple of the LRECL. The data set specified on the INPUT DD statement should have an LRECL of 80 with any block size that is a multiple of the LRECL.

2 By default, the FTP client sends the **SITE** subcommands when sending data to a foreign host. This may cause error messages from a non-mainframe FTP server such as an AIX system. You can use the **SENDSITE FTP** subcommand to prevent clients from sending site information. Comments are only allowed with CS for OS/390 V2R10 IP and later. The " ; " semicolon denotes a comment.

Note that each time you use the **SENDSITE** subcommand, it is turned alternately on and off. You may use the **LOCSTAT** subcommand to make sure the sending of site information is disabled on your OS/390 system.

3 The -d parameter turns on tracing. This is for debugging only. The EXIT parameter will provide message EZA1735I with the FTP return code if something goes wrong and inhibits further processing. EXIT=nn will terminate FTP with an non-zero return code of your choice if an FTP error occurs.

Note: the continuation character for FTP batch jobs is the " + " sign. Figure 6-89 shows the continuation for the get statement as being a “+”

```
get ftp.test1 +
    'user1.ftptst2' +
    (replace
```

Figure 6-89 Showing continuation in batch jobs

It will be interpreted as one line: `get ftp.test1 'user1.ftptst2' (replace`

The user ID and password can be placed in userid.NETRC and defined with a DD card in JCL.Figure 6-90 shows the required JCL.

```
//FTPNETRC  JOB MSGCLASS=O,MSGLEVEL=(1,1),CLASS=A,NOTIFY=&SYSUID
//FTP     EXEC PGM=FTP,REGION=3M,PARM='/-d (EXIT'
//STEPLIB  DD  DSN=TCPIP.SEZALINK,DISP=SHR
//SYSTCPD  DD  DSN=TCP.TCPPARMS(TDATA03A),DISP=SHR
//SYSFTPD  DD  DSN=TCP.TCPPARMS(FDATACLN),DISP=SHR
//NETRC   DD DSN=BTHOMPS.NETRC,DISP=SHR   1
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*,DCB=(LRECL=160,RECFM=FB,BLKSIZE=16000)
//INPUT    DD  *
9.24.104.26
sendsite         ; toggle off to suppress sending site command
cd /tmp/
put 'user1.ftptst1' ftp.test1
get ftp.test1 'user1.ftptst2' (replace
quit
/*
```

Figure 6-90 JCL with NETRC DD statement.

1 Notice that the user ID and password is not required in the JCL INPUT DD card. In this case that information is coded in BTHOMPS.NETRC and defined with the //NETRC DD card. Refer to 6.5.2, "FTP client NETRC data set" on page 203 for NETRC file coding syntax as well as *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780.

6.5.8 FTP server DDNAME and batch job comment support

The CS for z/OS V1R2 IP client can refer to local files by DDNAME with FTP **get** and **put** subcommands (**mget** and **mput** are not supported with DDNAME). This function is enabled when a local filename is specified with the //DD: token followed by a 1 - 8 character DD name; otherwise the client does not invoke this support. Because an HFS file name could be allocated with a DD statement, both MVS data sets and HFS files in the local system can be transferred using this function.

This is a client-only function and the user is responsible for allocating and deallocating DDNAMEs that are transferred. The DDNAME can be allocated in a job or by using the **ALLOC TSO** command. DDNAME support does not require any action unless you want to take advantage of the new function. Figure 6-91 on page 215 will show how to use //DD: with ALLOC and then the FTP **put** subcommand.

```
 READY
ALLOC fi(MYTESTDD)  da('BTHOMPS.JCL(DDNTEST)') shr
 READY
ftp 9.39.64.151
 220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe, 23:39:35 on 2002-05-26.
 EZA1554I Connecting to:  9.39.64.151 port: 21.
 220-FTPD1 IBM FTP CS V2R8 at DEMOMVS.DDS.DFW.IBM.COM, 16:54:12
 220 Connection will close if idle for more than 5 minutes.
 EZA1459I NAME (9.39.64.151:BTHOMPS):
ms168
 EZA1701I >>> USER ms168
 331 Send password please.
 EZA1789I PASSWORD:
 EZA1701I >>> PASS
 230 MS168 is logged on.  Working directory is "MS168."
 EZA1460I Command:
put //DD:MYTESTDD 'MS168.DDNTEST'
EZA1701I >>> PORT 9,24,104,43,4,168
200 Port request OK.
EZA1701I >>> STOR 'MS168.DDNTEST'
125 Storing data set MS168.DDNTEST
250 Transfer completed (data was truncated)
EZA1617I 1230 bytes transferred in 0.030 seconds.
Transfer rate 41.00 Kbytes/sec.
EZA1460I Command:
```

Figure 6-91   ALLOC DD with PUT command

When using the **put** command with the //DD: token you must specify the foreign file or data set name, it cannot be blank. In the above example, the local file name is referenced by //DD:MYTESTDD on the **put** subcommand and the foreign data set name will be'MS168.DDNTEST'. Refer to *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780 for further details. Please note that the //DD: token can only be used with local file names; it cannot reference a foreign file name on a remote host. The FTP client used was MVS28B.itso.ral.ibm.com, where the local file is stored and then subsequently referenced with the //DD: token, after ALLOC was used to define the DD name. And at the conclusion of the FTP put, a file named 'MS168.DDNTEST' will exist on the FTP server host DEMOMVS.DDS.DFW.IBM.COM.

Figure 6-92 on page 216 will show you how to use the //DD: token in JCL. The example JCL will demonstrate its use with the ftp **get** and **put** subcommands. Also notice that comments are allowed in the JCL. A semicolon " ; " is the character that denotes the start of a comment. It can be in column one of a line or at the end of a statement following a space.

```
//DDNTEST JOB 1,BTHOMPS,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//STEP01 EXEC PGM=FTP,REGION=2048K
//*
//SYSFTPD    DD  DSN=TCPIP.TCPPARMS(FTSD28B),DISP=SHR
//SYSTCPD    DD  DSN=TCPIP.TCPPARMS(TCPD28B),DISP=SHR
//SYSPRINT   DD  SYSOUT=*
//MYTESTDD DD  PATH='/u/bthomps/ddexample',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//OUTPUT     DD  SYSOUT=*
//INPUT      DD  *
; ip address of the remote host follows
9.39.64.151  ; host name can be used here instead
userid  password   ; please protect this file or use netrc
;issue get & put ftp subcommands
GET /samples/profile //DD:MYTESTDD (REPLACE
;
PUT //DD:MYTESTDD ’MS168.MYTESTDD’
QUIT
/*
```

Figure 6-92 Making use of DD card in the FTP GET & PUT command

Important: The REPLACE option causes a data set on your local host to be overwritten. Also a get issued for an empty or non-existing file erases the content of the existing file on the local host.

Refer to *z/OS V1R2.0 CS: IP User’s Guide and Commands*, SC31-8780 for a "//DD:" token JCL example used with GDG data sets.

6.5.9 FTP client in REXX

The FTP EXEC interface allows you to execute FTP subcommands from an EXEC rather than interactively from a terminal. The FTP subcommand can be in a file (MVS data set or hierarchical file system file), or you can code it directly in the EXEC. By default, the FTP session dialog is printed on the terminal. If you want the message sent to a data set rather than the terminal do the following:

TSO	Specify an OUTPUT data set as part of the ALLOC statement.
UNIX System Services	Redirect the output to an hierarchical file system file when invoking the FTP command.

See Appendix F, “FTP client sample REXX program” on page 549 for a sample REXX program to submit an FTP command. You can run this REXX EXEC in both a TSO environment and UNIX System Services shell.

From TSO command line, you can run this sample REXX as follows:

```
EXEC 'KAKKY.SAMP.JCL(FTPREX01)' 'KAKKY.FTPIN KAKKY.FTPOUT' EX
FTP client return code is: 0
***
```

Figure 6-93 Executing a Rexx exec from the TSO command line

In the UNIX System Services shell environment, the FTP command may be started by the following command:

```
ftprex01 ftpin ftpout
FTP client return code is: 0
```

Figure 6-94 FTP command

The following is an example of the input file (either the input data set KAKKY.FTPIN or the hierarchical file system file ftpin).

```
9.24.104.26
user1 passwd1
sendsite
cd /tmp
put 'user1.ftptst1' ftp.test1
get ftp.test1 'user1.ftptst2' (replace
quit
```

Figure 6-95 Input file

You may also use the REXX stack to hold the FTP subcommands. When using the REXX

```
QUEUE "9.24.104.26"
QUEUE user1 passwd1
QUEUE "put 'user1.ftptst1' ftp.test1"
QUEUE "quit"

cmdargs = "-v -p TCPIPA"
parse source . . . . . . . . env .
/*************************************/
if env = "OpenMVS" then
  ADDRESS LINKMVS "ftp cmdargs"
else
  "ftp" cmdargs
end
```

Figure 6-96 Rexx stack holding FTP subcommands

The output messages from an FTP command are sent to the output file (either the output data set KAKKY.FTPOUT or the hierarchical file system file ftpout).

6.5.10 Proxy FTP

The proxy FTP allows file transfer between two hosts on behalf of another host. There is no need to do a **`get`** for a file on the client system and then do a **`put`** of the file to a different server. The proxy client has two control connections and can initiate data transfer between two servers. The control connection to a secondary (proxy) server, is established using the **`PROXY OPEN`** subcommand.

The FTP client may send most FTP commands using the **`PROXY`** command. These commands and replies are sent via control connections between the FTP client and proxy FTP server.

The LOCSTAT output shows both the primary and secondary connections.

Figure 6-97 on page 218 illustrates how proxy FTP works.

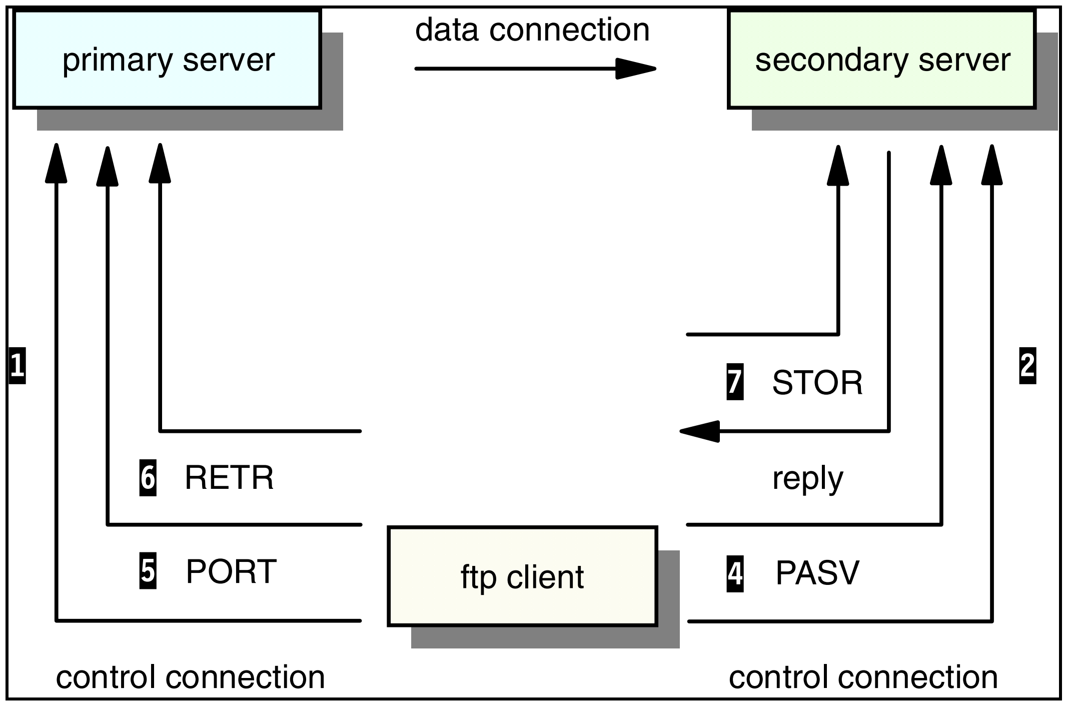


Figure 6-97 Proxy FTP operation

Figure 6-98 on page 219 shows an example of the proxy FTP operation. Note that the numbers in Figure 6-98 match with the numbers in Figure 6-97, so that you will easily see the communications occurred during the proxy FTP operation.

```
ftp 9.24.104.47  1

 EZA1450I IBM FTP CS/390 V2R5  1997 296 00:10 UTC
 EZA1466I FTP: using TCPIPA instead of INET
 EZA1554I Connecting to:  9.24.104.47 port: 21.
 220 wtr05118 IBM TCP/IP for OS/2 - FTP Server ver 02:36:43 on Aug 29 1996 ready.
 EZA1459I NAME (9.24.104.47:kakky):
kakky
 EZA1701I >>> USER kakky
 331 Password required for kakky.
 EZA1789I PASSWORD:
******

 EZA1701I >>> PASS
 230 User kakky logged in.
 EZA1460I Command:
proxy open 9.24.104.74    2
 EZA1554I Connecting to:  9.24.104.74 port: 21.
 220-T18NFTP1 IBM MVS V3R2 at MVS18.ITSO.RAL.IBM.COM, 12:00:18 on 1998/02/24
 220 Connection will close if idle for more than 5 minutes.
 EZA1459I NAME (9.24.104.74:kakky):
kakky
 EZA1701I >>> USER kakky
 331 Send password please.
 EZA1789I PASSWORD:
******

 EZA1701I >>> PASS
 230 KAKKY is logged on.  Working directory is "KAKKY.".
```

Figure 6-98 PROXY FTP operation

1 The FTP client connects to the primary FTP server. In this case, the primary server is running in an OS/2 system.

2 After entering the primary FTP server the client tries to open another control connection to the secondary FTP server. We used the TCP/IP V3R2 for MVS FTP server as a secondary FTP server.

```
 EZA1460I Command:
proxy get c:\autoexec.bat 'kakky.test.proxy01'   3
 EZA1701I >>> PASV  4
 227 Entering Passive Mode (9,24,104,74,4,15)
 EZA1701I >>> PORT 9,24,104,74,4,15 5
 200 PORT command successful.
 EZA1701I >>> RETR c::autoexec.bat 6
 150 Opening ASCII mode data connection for c:\autoexec.bat (389 bytes).
 EZA1701I >>> STOR 'kakky.test.proxy01' 7
 125 Storing data set KAKKY.TEST.PROXY01
 250 Transfer completed successfully.
 226 Transfer complete.
 EZA1460I Command:
proxy ls          8
 EZA1701I >>> PORT 9,24,104,231,4,11
 200 Port request OK.
 EZA1701I >>> NLST
 125 List started OK.
 EZA2284I ISPF52.ISPPROF
 EZA2284I MVS39.ISPPROF
 EZA2284I SAMP.JCL
 EZA2284I SPFLOG1.LIST
 EZA2284I TEST.PROXY01
 250 List completed successfully.
 EZA1460I Command:
quit
 EZA1701I >>> QUIT    9
 221 Quit command received. Goodbye.
 EZA1701I >>> QUIT
 221 Goodbye.
 ***
```

Figure 6-99 The proxy get command

3 The client issues a **proxy get** command, which will initiate file transfer from the primary FTP server to the secondary FTP server. From the client point of view, the secondary server looks like a local host, when you issue FTP subcommands with the **proxy** subcommand. In this case, the client specifies an OS/2 format file name as a remote host name, and uses an MVS data set name for the local host name.

In the PROXY FTP the data connection is initiated by the primary FTP server, so the secondary FTP server must support the **PASV FTP** subcommand.

4 The **PASV** command is sent to the secondary server. The proxy server sends back a reply with an IP address and port number. This will be used for the data connection established by the primary server.

5 A **PORT** command is sent to the primary server. The IP address and port number received on the PASV reply are sent on the **PORT** command.

6 A **RETR** command for the source file is sent to the primary server.

7 A **STOR** command for the target file is sent to the secondary server.

8 To make sure the file was stored successfully, the FTP client issues a **proxy ls** command. This command will display the data set name lists of the current working directory in the secondary FTP server.

9 Terminate both FTP control connections. If you want to terminate only the control connection with the secondary server, you may use the `proxy close` command.

6.5.11 FTP server interface to JES

The FTP server JES interface can be used to submit MVS batch jobs from any FTP client in your IP network, and to retrieve spool output from the JES spool queues to your FTP client. In a sense, this function can be characterized as a form of MVS Remote Job Entry function, based on the FTP protocols. It is important to stress that this function does not require any special RJE software on your TCP/IP workstations.

No special customization of the FTP server is required to implement this function. It is available and ready to use. You can control its use via the FTCHKJES exit routine.

The JES interface, which is used by the FTP server to query information in the spool data set, is based on the same interface that is used by the TSO commands `STATUS` and `OUTPUT`. If the JESINTERFACELEVEL is set to 1 in FTP.DATA the following is true. This interface is not based on the OWNER attribute of a spool file. It will only select jobs based on a match of user ID and the corresponding number of characters in a job name.

The job name in the JCL must be the USERIDx, where x is a 1-character letter or number and USERID must be the user ID you will use to log on to the FTP server to submit the job. Only held output can be retrieved by the FTP client, so you may have to review your JES SYSOUT class specifications. The output class for MSGCLASS and SYSOUT files contained in your JCL must specify a JES HOLD output class.

If the JESINTERFACELEVEL is set to 2, the job name in the JCL can be any name you are authorized to view via the Security Authorization Facility (SAF), such as RACF. Review “JES interface” on page 195 and z/OS V1R2.0 CS: IP User’s Guide and Commands, SC31-8780 for further details.

At a client host, you use the `SITE` command to set `filetype=jes`. The FILETYPE statement specifies the mode of operation of the server.

In filetype JES mode, the `put`, `dir`, `ls`, `delete` and `get` commands work according to the following rules:

- A `put` will submit the file directly to the JES internal reader in the MVS FTP server.
- A `dir` or an `ls` command will list jobs that have as the job name your user ID plus one character.
- A `delete` command will delete a job or a SYSOUT data set from the spool data set.
- A `get` command will either retrieve one or more held SYSOUT files from the spool data set, or it will perform the combined function of submitting a job, waiting for it to run, collecting all output files and returning them to the FTP client.

The following is an example of the output from a `dir` command with a job waiting in the input queue and two jobs in the output queue.

```
ftp> site filetype=jes
200 Site command was accepted
ftp> dir
200 Port request OK.
125 List started OK
ALFREDCA  JOB03192  OUTPUT    3 Spool Files 1
ALFREDCA  JOB03193  OUTPUT    0 Spool Files 2
ALFREDCA  JOB03194  INPUT 3
125 List completed successfully
222 bytes received in 14 seconds (0 Kbytes/s)
```

Figure 6-100 The dir command

1 A job in the output queue with three held spool files that can be retrieved.

2 A job in the output queue, but without any held output.

3 A job that is waiting in the input queue, but not held. If it were held, the line would have shown a status of `-HELD-`.

If you want to retrieve a spool file, you will have to use the following syntax:

```
ftp> get j3192.1 1
200 Port request OK.
125 Sending data set ALFREDC.ALFREDCA.JOB03192.D0000002.JESMSGLG
250 Transfer completed successfully
local: j3192.1 remote: j3192.1
1958 bytes received in 2.2 seconds (0 Kbytes/s)
```

Figure 6-101 Retrieving a spool file

1 You can specify the full job ID (job03192) or abbreviate to J3192. In this example, you retrieve spool file number 1 of the three available.

```
ftp> dir
200 Port request OK.
125 List started OK
ALFREDCA  JOB03192  OUTPUT    3 Spool Files
ALFREDCA  JOB03193  OUTPUT    3 Spool Files
ALFREDCA  JOB03194  INPUT
125 List completed successfully
222 bytes received in 14 seconds (0 Kbytes/s)
ftp> get j3192.x job3192.out 1
200 Port request OK.
125 Sending all SPOOL files for requested JOBID
250 Transfer completed successfully
local: job3192.out remote: j3192.x
2758 bytes received in 19 seconds (0 Kbytes/s)
```

Figure 6-102 Retrieving spool files

You can request to retrieve all spool files for a given job in a single operation, by using an `x` 1 as spool file number:

In the file you receive on your FTP client system, the individual spool files are separated by a line with the following contents:

```
 !! END OF JES SPOOL FILE !!
```

You are able to submit a job and retrieve the output from the job in a single operation. The sequence of events are illustrated in Figure 6-103.

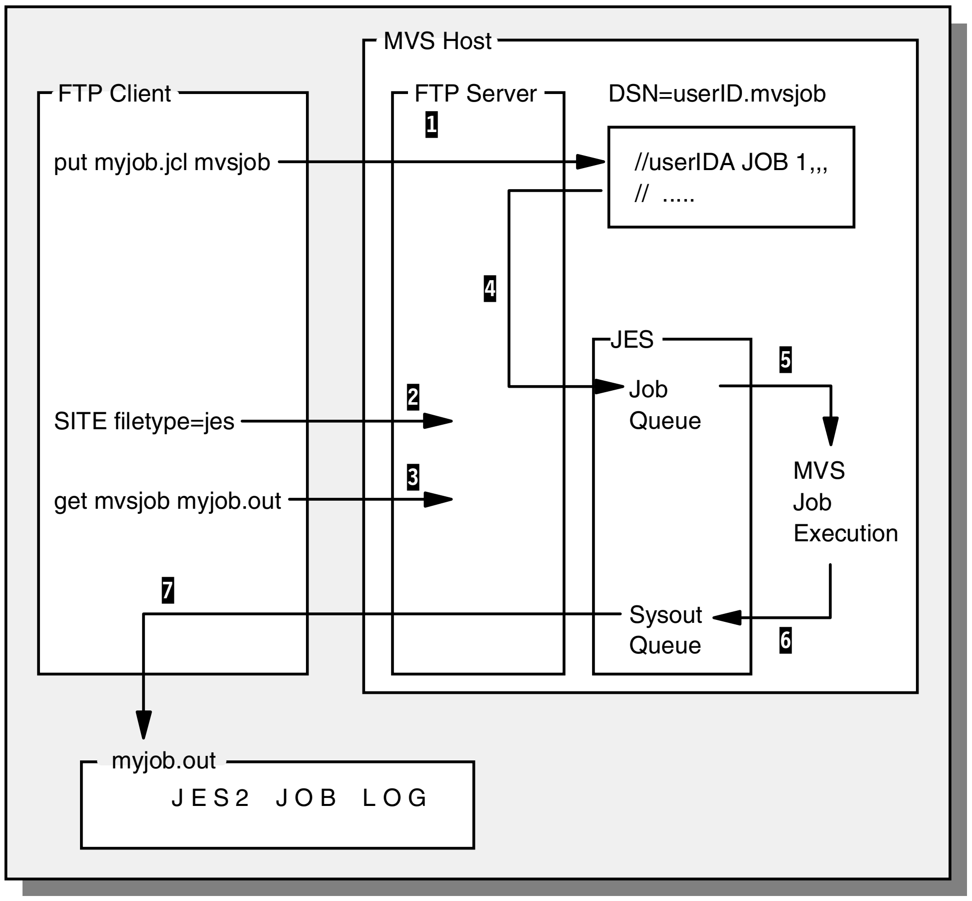


Figure 6-103   FTP server job submission and SYSOUT retrieval

1 First you have to create your JCL data set and transfer it to MVS. In this example we transfer the local file called myjob.jcl to MVS as userID.mvsjob in a traditional ASCII transfer.

2 Then you must set the filetype to JES.

3 To submit the job, wait for it to execute and retrieve the output. Enter a `get` command, where you name the data set on MVS, which you just created, and name your local file (in this example myjob.out). Your local FTP client is blocked until the job has been executed and SYSOUT files transmitted to you.

4 The MVS FTP server will submit the job to JES, where it will be placed in the job queue.

5 MVS will schedule the job when an initiator becomes available.

6 Output from the job will be placed in the JES SYSOUT spool queue.

7 The FTP server will pick up the individual JES SYSOUT files and assemble them into a single file, which will be transmitted to your FTP client and placed in the file, you specified on the **get** command: myjob.out.

The command you issue from your client and the responses you get back at your client workstation look like the following:

```
ftp> put myjob.jcl mvsjob  1
200 Port request OK.
125 Storing data set ALFREDC.MVSJOB
250 Transfer completed successfully.
local: myjob.jcl remote: 'alfredc.mvsjob'
96 bytes sent in 0.032 seconds (2 Kbytes/s)
ftp> site filetype=jes  2
200 Site command was accepted
ftp> get mvsjob myjob.out  3
200 Port request OK.
125-Submitting job 'ALFREDC.MVSJOB' FIXrecfm 80
125 When JOB03193 is done, will retrieve its output   4
250 Transfer completed successfully
local: myjob.out remote: 'alfredc.mvsjob'
2758 bytes received in 26 seconds (0 Kbytes/s)
ftp>
```

Figure 6-104 Client commands and responses

The **JESPUTGETTO** parameter in FTP.DATA specifies how long a client should wait for JES output after a job has been submitted. The default for value is 600 (10 minutes).

If the **JESPUTGETTO** timeout value is exceeded before the job is executed and has returned output, the transfer operation is aborted, and you will have to retrieve the JES output files manually.

6.5.12 Use NJE network to forward files

From your FTP client, you can use the OS/390 FTP server to forward files using an existing NJE network.

When you PUT a file to the FTP server, you can specify that the file is not to be stored on the MVS system, but is to be forwarded to a user on an NJE node. You do this by issuing the following command before you put your file:

```
site dest=nodename.userid
```

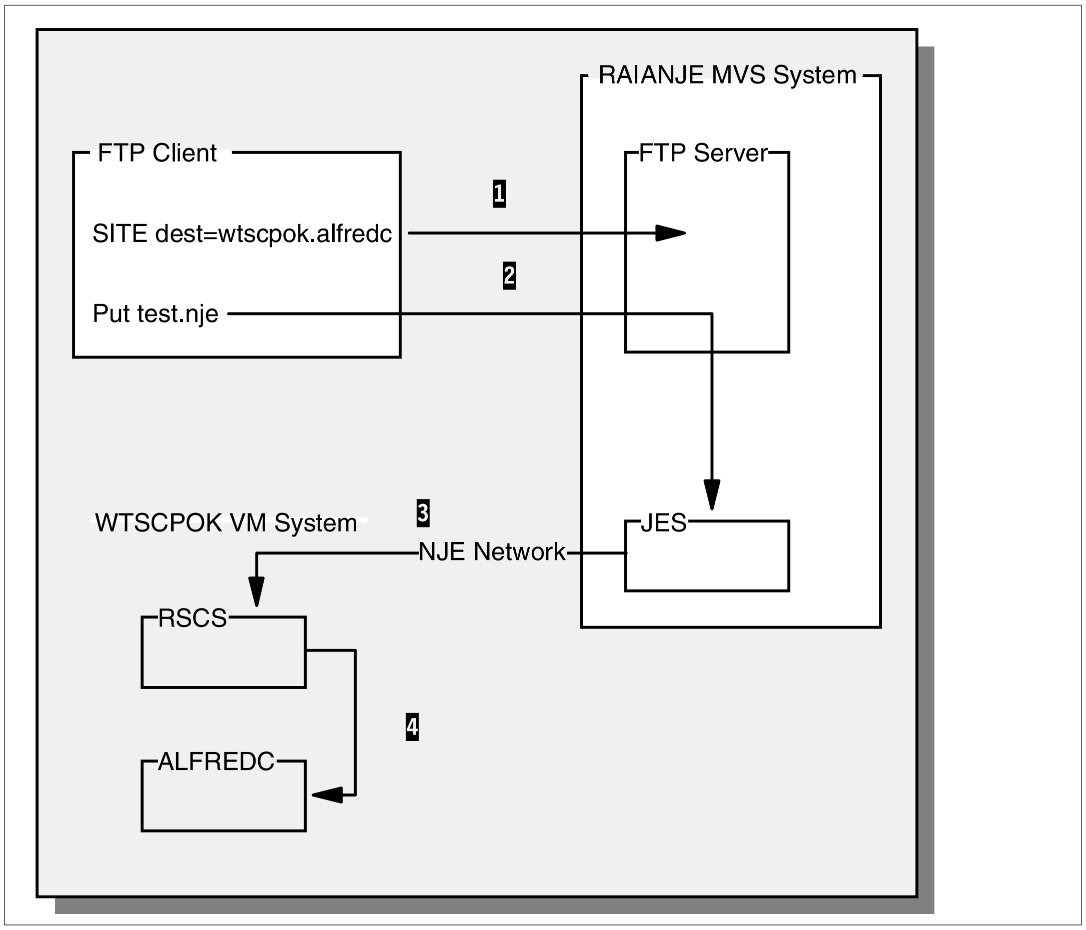


Figure 6-105 Forwarding files to NJE destinations from FTP

1 You set the NJE destination via a **SITE** command.

2 The file is forwarded to the FTP server as an ASCII file.

3 The FTP server forwards the file to JES with the NJE node ID and user ID from the preceding **SITE** command.

4 The file is forwarded through the NJE network to the destination NJE node ID, which in this example is an RSCS virtual machine on a VM operating system.

The file is finally placed into the virtual reader of the user.

```
ftp> site dest=wtscpok.alfredc 1
200 Site command was accepted
ftp> put test.nje 2
200 Port request OK.
125 Sending file via NJE to requested destination. 3
250 Transfer completed successfully.
local: test.nje remote: test.nje
96 bytes sent in 0 seconds (0 Kbytes/s)
```

Figure 6-106 FTP NJE session

1 You specify the NJE receiver of the file via the SITE DEST option.

2 You put the file to the FTP server.

3 The server acknowledges that the file is transmitted via the NJE network.

6.5.13 FTP and use of tape data sets

You are able to read from and write data to tape devices, which are allocated by the FTP server or FTP client.

You request tape data set allocation via the **`UNIT SITE`** parameter for the FTP server and the **`UNIT LOCSITE`** parameter for the FTP client. You can use the following **`SITE`** and/or **`LOCSITE`** command:

- For the FTP server

```
site unit=tape
```

- For the FTP client

```
locsite unit=tape
```

You can disable automatic volume mounting via the **`AUTOTAPEMOUNT`** parameter in the FTP.DATA data set for your server and/or client.

If the number of users who have to be able to access tape data sets is limited, you can set up a second FTP server address space on a separate set of port numbers. You can then set **`AUTOTAPEMOUNT`** to **`false`** in the primary FTP server's FTP.DATA data set, and **`AUTOTAPEMOUNT`** to **`true`** in the secondary FTP server's FTP.DATA data set. You can control access to this secondary FTP server in more ways:

1. By IP address via the FTCHKIP user exit class
2. By user ID via the FTCHKPWD user exit or RACF application class

6.5.14 DB2 SQL queries with FTP

A user at an FTP client may use this support to extract data from a DB2 database. The user at the client will have to do the following:

1. Edit a file with an SQL SELECT statement.
2. Transfer the file to the FTP server with a filetype of SEQ.
3. Issue the **`SITE`** command to set the name of the DB2 subsystem to use and to set the filetype to SQL.
4. Issue a **`GET`** command referring to the file just transferred, thereby signalling to the FTP server that the SQL query should be executed and the result returned to the client.

Similarly, you can use the z/OS FTP client to extract data from a local DB2 database and send the result to any server.

DB2 SQL query function

This function can be used to execute queries (SQL SELECTs) against DB2 objects.

You must BIND the DBRM called EZAFTPMQ to the plan used by FTP, and GRANT execute privilege to PUBLIC. You may specify the name of the plan by the DB2PLAN statement in FTP.DATA. The default name is EZAFTPMQ. This FTP facility performs the SELECT operation only on the DB2 tables. The UPDATE, INSERT, and DELETE operations are not supported.

The FTOEBIND member of the SEZAINST data set may be used as a sample job to enable the FTP server and client to do SQL queries.

You can start a second FTP server that was configured so it would, by default, process all **get** requests as SQL queries.

The JCL procedure to start this server looks like the following:

```
//FTPDSQL PROC
//*
//*  OpenEdition MVS FTP SQL Query Server main process
//*
//FTPD  EXEC PGM=FTPD,REGION=40M,
//           PARM='POSIX(ON),ALL31(ON)/PORT 2021' 1
//*
//* FTPD is in TCPIP.SEZALINK (on the LINKLST)
//* FTCHKxxx routies are in the following load library
//* DB2 modules are needed for SQL support
//*
//* NB: All steplib libraries MUST be RACF PROGRAM controlled!
//*
//STEPLIB DD DSN=WOZA.GIMLIB,DISP=SHR
//        DD DSN=SYS1.DSN230.DSNEXIT,DISP=SHR     2
//        DD DSN=SYS1.DSN230.DSNLOAD,DISP=SHR
//SYSFTPD DD DSN=TCP.TCPPARMS(OEFTPSQL),DISP=SHR
```

Figure 6-107 FTP SQL server started procedure

1 This server starts on an alternate control port, port number 2021, and it will use 2020 as the data port. These port numbers should be reserved to the FTP server job name (for control connection) and OMVS (for data connection) in your TCP/IP PROFILE data set:

```
PORT
2020 TCP  OMVS       ;FTP Alternate Server data port
2021 TCP  FTPDSQL1   ;FTP Alternate Server control port
```

Figure 6-108 Port statements in TCPIP profile for SQL

2 As this server is used for SQL queries, we add the DB2 load libraries to our STEPLIB. If your DB2 load libraries are on your system link list, you do not need to add them to STEPLIB. Please remember to mark your DB2 load libraries as RACF program control.

The FTP.DATA options used for this server look like the following:

```
;*****************************************************************
;*                                                              *       *
;*   Name of File:      TCP.TCPPARMS(OEFTPSQL)                  *       *
;*                                                              *       *
;*   FTP.DATA for SQL FTP server on ports 2020/2021             *       *
;*                                                              *       *
;*****************************************************************
Primary         1        ;Primary allocation is 1 track
Secondary       20       ;Secondary allocation is 20 tracks
Directory       15       ;PDS allocated with 15 directory blocks
Lrecl           255      ;Logical Record Length of 255
BlockSize       6144     ;Block Size of 6144
AutoRecall      true     ;Migrated HSM files recalled automatically
AutoMount       false    ; Non-mounted volumes not mounted auto.
DirectoryMode   false    ; Use all qualifiers (Datasetmode)
Volume          WTLTCP   ; Volume serial number for new data sets
SpaceType       TRACK    ; Data sets allocated in tracks
Recfm           VB       ; Variable Blocked record format
Smf             170      ; The SMF record sub type to be used
Mgmtclass       TCPMGMT  ; SMS management class for new data sets
Ctrlconn        IBM-850  ; ASCII code page for control connection
Sbdataconn      (IBM-1047,IBM-850) ; EBCDIC, ASCII code page data
Quotesoverride  TRUE     ; Single quotes means qualified name
Umask           027      ; Make new HFS files rw- r-- ---
Filetype        SQL   1  ; File Type = SQL
DB2             DSNI  2  ; DB2 subsystem DSNI on mvs18
Spread          False      ; Do not use spread-sheet format
SQLCol          Any        ; Use labels or names
Anonymous FTPANOM/NOSECRET ; The anonymous userid and password 3
STARTDIRECTORY HFS         ; Start in OE
```

Figure 6-109 FTP.DATA statements for SQL

1 This server processes all requests as SQL queries by default.

2 The default DB2 subsystem to connect to is called DSNI.

Each FTP client runs in its own MVS address space, two clients can concurrently be connected to two different DB2 subsystems.

3 This server instance allows anonymous login. An anonymous user will be associated with the FTPANOM user ID. The anonymous user will not be prompted for a password.

Assume that the file /u/ftpanom/sqldept.txt has the following contents:

```
SELECT * FROM ALFREDC.ABCDEPT
```

An anonymous user that retrieves this file via this FTPD server can use the following FTP dialog:

```
[C:\]ftp mvs18o 2021
IBM TCP/IP for OS/2 - FTP Client ver 15:51:28 on Nov 19 1994
Connected to mvs18o.itso.ral.ibm.com.
220-FTPDSQL1 IBM MVS V3R3 at mvs18oe.itso.ral.ibm.com, 21:06:07 on 1997-
220 Connection will close if idle for more than 5 minutes.
Name (mvs18o): anonymous
230 'ANONYMOUS' logged on.  Working directory is "/u/ftpanom".
ftp> get sqldept.txt sqldept.out
200 Port request OK.
125 Sending data set /u/ftpanom/sqldept.txt
250 Transfer completed successfully.
local: sqldept.out remote: sqldept.txt
1490 bytes received in 0.44 seconds (3 Kbytes/s)
ftp>
```

Figure 6-110 Anonymous FTP session

The output file, sqldept.out, will hold the result of the SQL query:

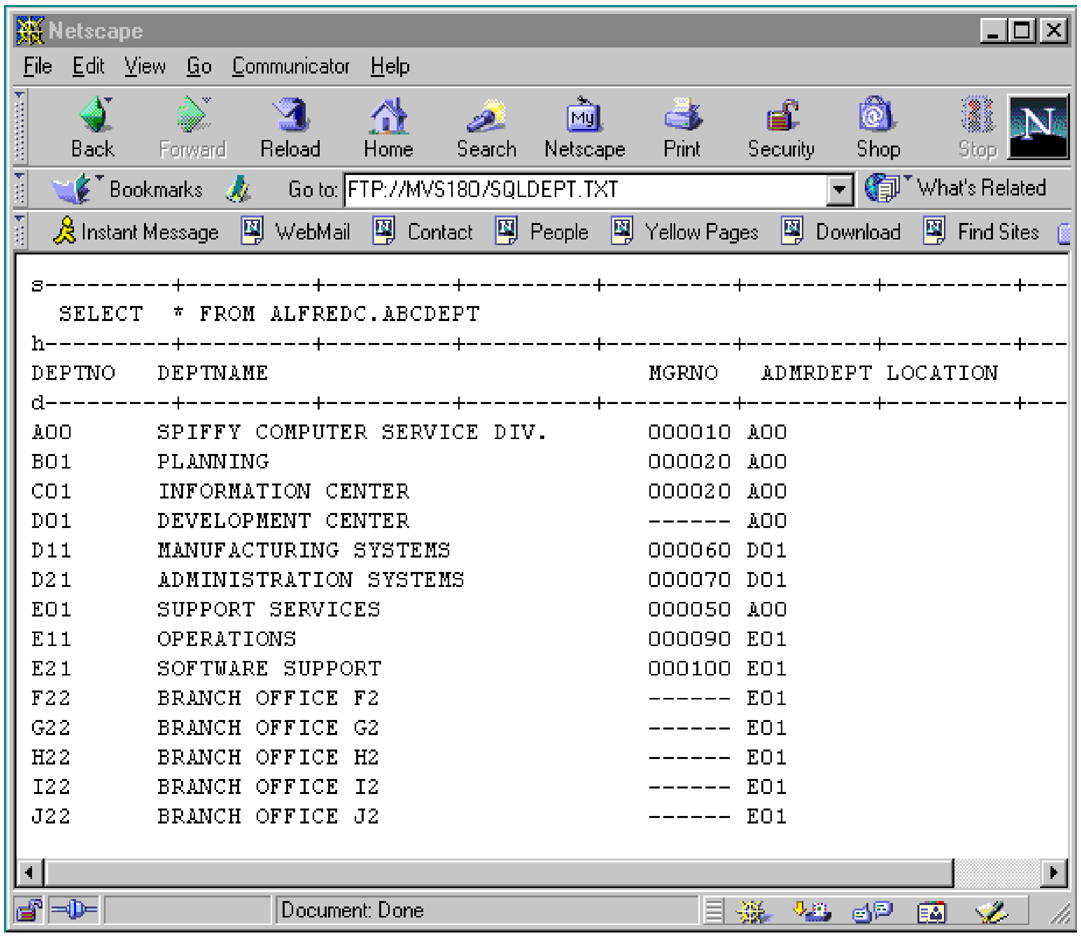
```
s--------+---------+---------+---------+---------+---------+---------+--
 SELECT * FROM ALFREDC.ABCDEPT
h--------+---------+---------+---------+---------+---------+---------+--
DEPTNO DEPTNAME                              MGRNO  ADMRDEPT LOCATION
d--------+---------+---------+---------+---------+---------+---------+--
A00    SPIFFY COMPUTER SERVICE DIV.          000010 A00
B01    PLANNING                              000020 A00
C01    INFORMATION CENTER                    000030 A00
D01    DEVELOPMENT CENTER                    ------ A00
D11    MANUFACTURING SYSTEMS                 000060 D01
D21    ADMINISTRATION SYSTEMS                000070 D01
E01    SUPPORT SERVICES                      000050 A00
E11    OPERATIONS                            000090 E01
E21    SOFTWARE SUPPORT                      000100 E01
F22    BRANCH OFFICE F2                      ------ E01
G22    BRANCH OFFICE G2                      ------ E01
H22    BRANCH OFFICE H2                      ------ E01
I22    BRANCH OFFICE I2                      ------ E01
J22    BRANCH OFFICE J2                      ------ E01
```

Figure 6-111 SQL query output

The FTP request could also be imbedded in an HTML document, as the following example shows:

```
<A HREF="ftp://mvs18o/sqldept.txt">
Sample DB2 SQL query - <EM>Is now working - check it out</EM></A>
```

By clicking on this link, the Web browser will log on as the anonymous user to the FTP server, send a **RETR FTP** command for the specified file, sqldept.txt in the default home directory (/u/ftpanom), and display the output from the SQL query. The reason we use the txt suffix is that the Web browser then will treat the output as a text file and show the result with its default text file browser (see Figure 6-112 on page 230).


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: FTP://MVS180/SQLDEPT.TXT What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites
Document: Done

```
s---------+---------+---------+---------+---------+---------+---
  SELECT  * FROM ALFREDC.ABCDEPT
h---------+---------+---------+---------+---------+---------+---
DEPTNO   DEPTNAME                            MGRNO   ADMRDEPT LOCATION
d---------+---------+---------+---------+---------+---------+---
A00      SPIFFY COMPUTER SERVICE DIV.        000010 A00
B01      PLANNING                            000020 A00
C01      INFORMATION CENTER                  000020 A00
D01      DEVELOPMENT CENTER                  ------ A00
D11      MANUFACTURING SYSTEMS               000060 D01
D21      ADMINISTRATION SYSTEMS              000070 D01
E01      SUPPORT SERVICES                    000050 A00
E11      OPERATIONS                          000090 E01
E21      SOFTWARE SUPPORT                    000100 E01
F22      BRANCH OFFICE F2                    ------ E01
G22      BRANCH OFFICE G2                    ------ E01
H22      BRANCH OFFICE H2                    ------ E01
I22      BRANCH OFFICE I2                    ------ E01
J22      BRANCH OFFICE J2                    ------ E01
```

Figure 6-112 FTP SQL query from a Web browser

Note: Please note that this technique will work only if the FTPD server uses the default FTP server port number. Even though the HTTP protocol allows you to specify an alternate port number in an FTP URL, our testing shows that this does not work from the majority of currently available Web browsers.

6.5.15 Using the FTP SQL query function from a remote FTP client

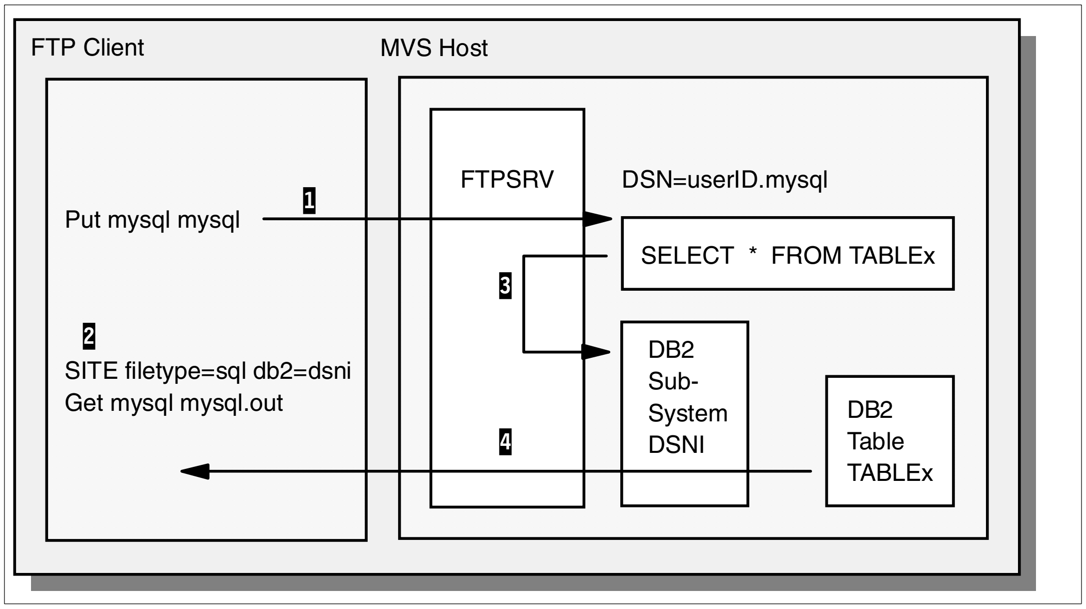


Figure 6-113 Using the FTP SQL query function from a remote FTP client

1 This step is optional. The remote user may edit a local file with an SQL SELECT statement, and transfer this file to the MVS system. Or the user may use a file that already has been prepared with an SQL statement and stored on the MVS system.

If a file is prepared and transferred to MVS, it is done via a traditional ASCII mode transfer.

To execute the SQL query, issue the following two commands 2:

1. First, a **SITE** command, which instructs the MVS FTP server to switch to filetype SQL mode, and names the DB2 subsystem to connect to.
2. Then a **get** command, where the remote file name points to the MVS data set holding the SQL statement to be executed, and the local file name points to the local file, where the output from the SQL query must be placed.

The MVS FTP server will 3 read the MVS data set, but instead of sending it to the client, it will be handed to the DB2 subsystem for execution. The output from the query will be picked up, and transferred 4 to the client.

If you have an SQL query that you want to execute frequently, you should probably store it permanently on MVS. Periodically you enter your FTP client function, where you go into filetype SQL mode, execute the query, and retrieve the current results of the query.

6.5.16 Using the FTP SQL query function from a local TSO FTP client

Note: The FTP client does not have to be invoked under TSO to use the SQL support. You can do this also when the client is invoked from the UNIX System Services shell.

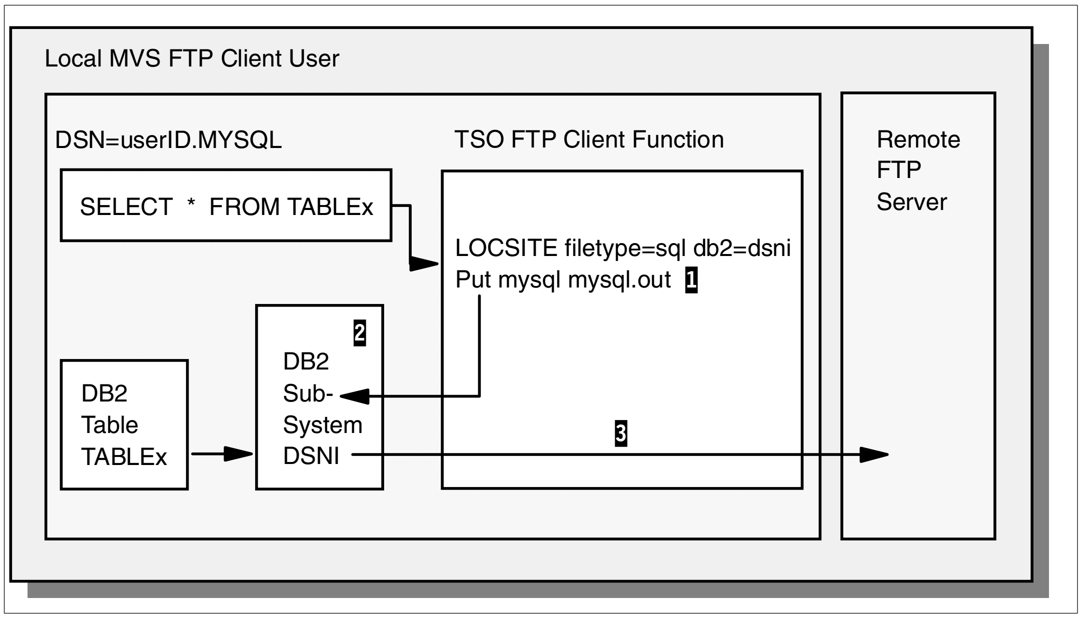


Figure 6-114 Using the FTP SQL query function from a local TSO FTP client

When you are in an FTP session with the remote FTP server, enter the following two commands in order to use the SQL query function:

1. A **LOCSITE** command, where you set the FTP client local filetype=sql, and name the DB2 subsystem to use for the query function.
2. A **PUT** command 1, where your local file name points to a data set holding the SQL query to execute, and the remote file name points to the remote file, where the output of the query should be placed.

The FTP client will hand the data set with the SQL query to the DB2 subsystem 2, pick up the response from the DB2 subsystem 3 and transfer the response to the remote host.

You may use this facility in combination with the batch FTP facility to automate distribution of DB2 data to remote FTP hosts. You could let your batch production scheduling system schedule jobs periodically, which enters a batch FTP session with remote FTP servers, executes predefined SQL queries, and transfers the results to the remote system

```
//jobname  JOB 1,MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=tsouser
//SQL    EXEC PGM=FTP,PARM='(EXIT'
//SYSPRINT DD SYSOUT=*
//INPUT    DD *
<remote host name>
<remote userid>
<remote password>
locsite filetype=sql db2=dsni
put v3r4.db2cntl(select) query1.out 1
close
quit
//OUTPUT   DD SYSOUT=*,DCB=(RECFM=FB,LRECL=160,BLKSIZE=3200)
```

Figure 6-115 Batch FTP with SQL query

1 The SQL query is in the data set called userID.v3r4.db2cntl(select). The result of the query should be placed into the remote file called query1.out.

In a batch invocation of FTP you have to specify `PARM='(EXIT'` in order to terminate FTP in case an error is encountered. If you do not specify `(EXIT`, FTP will try to execute all FTP subcommands in the INPUT data set even if one of the first commands fails. Another advantage of the `(EXIT` keyword is that in case of an error the FTP step will terminate with a non-zero return code, which you may use to control the execution of succeeding steps in your JCL stream.

If you get a non-zero return code, you can check the OUTPUT file to see what went wrong.

6.5.17 FTP SQL query output

You can control the format of the output from DB2 via two SITE/LOCSITE parameters:

SPread/NOSPread	If you specify `SPread`, output will be in such a format that the file can be directly imported into a spreadsheet program of your choice.
SQLCol=	With the `SQLCol` keyword, you control how you want column headings to look. You have three options:
Names	Use the column names of the DB2 tables.
Labels	Use the labels associated with columns in the DB2 tables. If a column does not have a label, that column will be given a system-generated heading of the form `COLnnn`.
Any	Use the labels if they are defined. If a column does not have a label, then use the column name.

Default output format (SQLC`ol=names NOSPread`) looks very much like the output you would get from SPUFI under TSO:

```
s--------+---------+---------+---------+---------+---------+---
 select deptno, deptname, mgrno, admrdept from abcdept
h--------+---------+---------+---------+---------+---------+---
DEPTNO DEPTNAME                             MGRNO  ADMRDEPT
d--------+---------+---------+---------+---------+---------+---
A00    SPIFFY COMPUTER SERVICE DIV.         000010 A00
B01    PLANNING                             000020 A00
C01    INFORMATION CENTER                   000030 A00
D01    DEVELOPMENT CENTER                   ------ A00
D11    MANUFACTURING SYSTEMS                000060 D01
D21    ADMINISTRATION SYSTEMS               000070 D01
E01    SUPPORT SERVICES                     000050 A00
```

Figure 6-116 SQL query output

The only difference from SPUFI output is that the separator lines have a one-letter prefix, which indicates the type of information following that separator line:

s	The SQL query statement follows.
h	A header line is following with column headings.
d	The rest of the data set is rows selected as a result of the query.
e	An error message follows this separator line.

A program processing this output can use these codes to determine the contents of the lines following.

Output with `sqlcol=any nospread` would appear as follows:

```
s--------+---------+---------+---------+---------+---------+---------+--
 select deptno, deptname, mgrno, admrdept from abcdept
h--------+---------+---------+---------+---------+---------+---------+--
Number Name                                MGRNO  Admin
d--------+---------+---------+---------+---------+---------+---------+--
A00    SPIFFY COMPUTER SERVICE DIV.        000010 A00
B01    PLANNING                            000020 A00
C01    INFORMATION CENTER                  000030 A00
D01    DEVELOPMENT CENTER                  ------ A00
D11    MANUFACTURING SYSTEMS               000060 D01
D21    ADMINISTRATION SYSTEMS              000070 D01
E01   SUPPORT SERVICES               000050 A00
```

Figure 6-117 sqlcol=any output

The labels were created with the following SQL statement:

```
label on abcdept
   (deptno is 'Number',
    deptname is 'Name',
    admrdept is 'Admin')
```

Please note that the MGRNO column did not get a label. When you use the `sqlcol=any` option, the MGRNO column heading will be the name of the column.

If you intend to load the output from DB2 into a spreadsheet program, you have to use the `spread` option. Here we used it in combination with `sqlcol=labels`:

```
Number Name                                COL003 Admin
A00    SPIFFY COMPUTER SERVICE DIV.        000010 A00
B01    PLANNING                            000020 A00
C01    INFORMATION CENTER                  000030 A00
D01    DEVELOPMENT CENTER                  ------ A00
D11    MANUFACTURING SYSTEMS               000060 D01
D21    ADMINISTRATION SYSTEMS              000070 D01
E01   SUPPORT SERVICES               000050 A00
```

Figure 6-118 sqlcol=labels output

Each value in the spreadsheet output is separated by a tab character (X'05' in EBCDIC). These characters are not visible in the above print.

Because the MGRNO column does not have a label, it will get a system-generated label when we specify `sqlcol=labels`.

6.6 Security in the FTP environment

Each FTP server relies on the use of an external security manager such as RACF. Several servers provide some built-in security functions for additional security. Quite a few security loopholes exist in the FTP environment so it is imperative to plug as many of these with whatever tools are available to you. The arsenal to plug these loopholes is ever increasing, just as the threats against security are. We will discuss a few of the common exposures that exist in the environment and how to guard against them.

6.6.1 FTP server RACF definition

The FTP server requires, as do other servers in UNIX System Services, the ability to change the user security environment to that of the client. The FTP server listener process (ftpd) must therefore be started under a user ID with UID=0. If you want to start the FTP server as an MVS started procedure or activate the FTP server using the AUTOLOG statement in PROFILE.TCPIP, you need to configure the STARTED class profile for the FTP server in the RACF database. In the STARTED class profile, you also have to specify a user ID that has an OMVS segment with UID=0.

If the job name of the FTP server is *FTPDB* and the associated user ID is *TCPIP3*, the following RACF definitions would be needed:

```
SETROPTS GENERIC(STARTED)  1
SETROPTS CLASSACT(STARTED) RACLIST(STARTED)  1
RDEFINE STARTED FTPDB.* STDATA(USER(TCPIP3) GROUP(OMVSGRP) TRUSTED(NO))
SETROPTS RACLIST(STARTED) REFRESH
```

1 The first two commands are needed if you have not configured the RACF STARTED resource class on your z/OS system.

If you have the BPX.DAEMON facility class configured in the RACF database, the user associated with the FTP server must have READ access to this facility class resource. Activating the BPX.DAEMON level of security is not required. However, it is recommended, because it provides additional security in the UNIX System Services environment.

The following RACF definitions will give the user TCPIP3 the READ permission to the BPX.DAEMON facility class.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(TCPIP3) ACCESS(READ)
```

If you define the BPX.DAEMON facility class, then you must enable program control for certain z/OS load libraries. The following commands define three load libraries as controlled libraries. These definition are required for the FTP server, when you configure the BPX.DAEMON facility class.

```
SETROPTS WHEN(PROGRAM) 1
RDEFINE PROGRAM *  ADDMEM ('TCPIP.SEZALINK'/O35RZ1/NOPADCHK) UACC(READ)
RDEFINE PROGRAM *  ADDMEM ('SYS1.LINKLIB'/O35RZ1/NOPADCHK) UACC(READ)
RDEFINE PROGRAM *  ADDMEM ('CEE.SCEERUN'/O35RZ1/NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

1 The first command is needed only if you have not defined program control in your RACF database.

If the server is configured for DB2 query support, you also need to define the DB2 load libraries as controlled libraries.

If you do not give the user associated with the FTP server the READ authority to the BPX.DAEMON facility class, your FTP client will fail with the following message:

```
Name (9.24.105.126): garthm
331 Send password please.
Password: ......
550 error processing PASS command : EDC5113I Bad file descriptor.
Login failed.
```

Figure 6-119   Read authorization to BPX.DAEMON failure

Then you will see the following security violation message at the MVS console:

```
ICH408I USER(TCPIP3  ) GROUP(OMVSGRP ) NAME(TCPIP TASKS         )
  BPX.DAEMON CL(FACILITY)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Figure 6-120   Security violation on system console

In this case, you would have to check your RACF definition, and make sure the user associated with the FTP server started task has the READ permission to the BPX.DAEMON facility class.

If you do not define the load libraries as program controlled correctly, your FTP client would receive the following error message when you try to log on to the FTP server:

```
Name (9.24.105.126): garthm
331 Send password please.
Password: ......
550 error processing PASS command : EDC5157I An internal error has occurred.
Login failed.
```

Figure 6-121   Load libraries not defined as pgm controlled error

With the TRACE option, you will see the following error message from the FTP server, which may be displayed at the MVS console or sent to the syslog daemon log file:

```
RA0638 pass2: seteuid failed (157/0B7F02AF) :
                       EDC5157I An internal error has occurred.
```

Figure 6-122   Trace output of load library error

To see the list of the program controlled libraries, use the RACF Display General Resource Services panel or issue the following RACF command:

```
RLIST PROGRAM *
```

If you decide not to define the BPX.DAEMON Facility Class, then assigning `UID(0)` for UIDs associated with the FTP server is sufficient to process.

For more information on the RACF definition, such as the BPX.DAEMON facility class or program control, refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

6.6.2 Signing on to the z/OS FTP server

During a logon to the z/OS FTP server, the user is prompted for an MVS user ID and password, which are verified via the SAF interface in MVS. The user is able to change the MVS password during a logon to the FTP server.

The z/OS FTP server will, on the RACROUTE VERIFY call, pass both an application ID and a terminal ID:

APPLID	The first seven characters of the FTP server address space name.
TERMID	A character representation of the hexadecimal byte value of the IP address of the originating host.

```
If the IP address is 9.24.104.79
      Then the  Terminal ID becomes 0918684F
```

All requests for access to a data set in MVS are checked via the SAF interface based on the user ID entered when you log on to the FTP server. A user must have a valid user ID on the MVS system to be able to access the FTP server.

The z/OS FTP server does not require the OPERATIONS RACF attribute on its started task user ID. It has a start option, which allows it to accept **`anonymous`** connections. An anonymous connection is a connection where the user types in the word **`ANONYMOUS`** for a user ID.

When the FTP server processes a logon request from a client, it will issue a RACROUTE REQUEST=VERIFY, where the following parameters will be passed:

1. User ID
2. Password and, optionally, a new password
3. The first seven characters of the FTP server job name as application name
4. A terminal ID based on the IP address of the client

The terminal ID passed will be constructed as an 8-byte character string consisting of characters representing hexadecimal characters:

```
IP address  = 9.24.104.79  (X'09.18.68.4F')
Terminal ID = C'0918684F'
```

If your RACF SETROPTS options are TERMINAL(READ), all terminals are allowed access to your system, and you do not need to add extra resource definitions to your RACF database.

If your RACF SETROPTS options are TERMINAL(NONE), you will have to add TERMINAL resource definitions to the RACF database for your IP addresses:

```
RDEFINE TERMINAL 091868* UACC(NONE) 1
PERMIT 091868* ID(USER3) CLASS(TERMINAL) ACCESS(READ) 2
```

With the RDEFINE in line **1** above, you define a generic resource in the TERMINAL class. This resource covers all IP addresses in the 9.24.104 subnet. In this example the universal access code is NONE, but if you had specified UACC(READ), all IP addresses in the 9.24.104 subnet would be available to all users for logging on to the FTP server.

With the PERMIT in line **2**, you allow the user with user ID USER3 to log on to the FTP server from any IP host in the 9.24.104 subnet.

You can use this support to control from which IP hosts your users log on to your FTP server.

When FTP opens a data set on behalf of the network user, then MVS OPEN uses the credentials of the network user.

6.6.3 Implementing an anonymous user

The use of an FTPD server brings up a special security aspect. It is quite common to provide FTP services for undefined users that logs in with a user ID of ANONYMOUS. The UNIX System Services FTPD server supports both defined and anonymous users. Anonymous user support is enabled in FTP.DATA with a configuration option. The decision to enable this option is left up to your discretion and requirements.

Identifying the user

The FTP daemon is a never-ending server process that waits for connection requests on the well-known port number 21. After an FTP client connects, the FTPD listener process starts another process that is going to be used for this individual FTP session. Refer to “CS for z/OS IP: FTP overview” on page 124 for details about the processes. In our setup, the FTP daemon process was started with a user ID of FTPD2 as shown in Figure 6-123.

```
D OMVS,A=ALL
BPXO040I 15.29.24 DISPLAY OMVS 650
OMVS     000E ACTIVE         OMVS=(28)
USER     JOBNAME  ASID       PID       PPID STATE   START     CT_SECS
FTPD2 1   FTPDB1   006B   33554458         1 1FI   16.38.17       .118
  LATCHWAITPID=        0 CMD=FTPD
FTPD2 2   FTPDB1   0062        107   33554458 1FI   15.45.08       .099
  LATCHWAITPID=        0 CMD=/usr/sbin/ftpdns 0 0 27 1 80 128 256 5
```

Figure 6-123 FTPD server before user enters login information

1 This is the FTPD listener process that executes the ftpd program.

2 This is the new process that handles the new client connection. The program in this process is ftpdns. At this time, the FTP client has not yet entered a user ID and password, which is the reason why this process still executes under the same user ID as the FTPD process.

```
D OMVS,A=ALL
BPXO040I 15.29.24 DISPLAY OMVS 650
OMVS     000E ACTIVE         OMVS=(28)
USER     JOBNAME  ASID    PID       PPID STATE   START     CT_SECS
FTPD2    FTPDB1   006B   33554458  3      1 1FI   16.38.17       .110
  LATCHWAITPID=        0 CMD=FTPD
BTHOMPS 1 FTPDB1   0083   16777284   33554458 2  1FI   15.29.11       .185
  LATCHWAITPID=        0 CMD=/usr/sbin/ftpdns 2138244376
```

Figure 6-124 FTPD server after user has entered login information

1 After the user enters a user ID and password, the process changes its user ID. You can verify the relationship between parent and child processes by comparing the PPID of the user process 2, and the PID of the server process 3.

Please note that *all* users of the FTPD server must have a valid OMVS segment in their RACF user profiles (see Figure 6-128 on page 241). This is true even when the user only wants to access MVS data sets via the FTPD server. If the FTP client maneuvers in a way not allowed in the file system, RACF immediately notices it, the request fails, and a corresponding RACF message is issued on the MVS console (see Figure 6-125).

```
ICH408I USER(GARTHM  ) GROUP(WTCRES  ) NAME(GARTHM MADELLA  )
  /etc/orouted.03a.env
  CL(FSOBJ   ) FID(01E2D7D3C5E7F34E2B0D0000106A0000)
  INSUFFICIENT AUTHORITY TO OPEN
  ACCESS INTENT(R--)  ACCESS ALLOWED(OTHER ---)
```

Figure 6-125 RACF error message when a user request violates the access permission

Notes:

1. CL(FSOBJ) identifies an internal RACF class. It refers to a file system object. You do not need to define it anywhere; it is used for auditing purposes. You may see the following classes in similar messages: DIRACC, DIRSRCH, FSSEC, IPCOBJ, PROCACT, PROCESS.
2. FID identifies the file system object in question.

Anonymous user definitions

A special situation with respect to user identification occurs with FTP. This service may be accessed by an ANONYMOUS user. The UNIX System Services FTPD server must be specifically configured to support anonymous access. The default option is to refuse anonymous logons.

Anonymous support must first be enabled by specifying the ANONYMOUS statement in FTP.DATA or specifying ANONYMOUS as a parm in the FTP started procedure. As follows:

```
//FTPDB  PROC MODULE='FTPD',PARMS='ANONYMOUS'
//FTPDB  EXEC PGM=&MODULE,REGION=OM,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//          'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB")','/&PARMS')
//*         'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=T&SYSCLONE.BTCP")',
//*          '/&PARMS')
//CEEDUMP  DD SYSOUT=*
//SYSFTPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(FTSD&SYSCLONE.B)
//SYSTCPD DD DISP=SHR,DSN=TCPIP.TCPPARMS(TCPD&SYSCLONE.B)
```

Figure 6-126 FTP started proc with ANONYMOUS parm

Note: Specifying ANONYMOUS as a parm in the started procedure will not enable the enhanced support that allows the e-mail address to be used as the password.

Following is a syntax diagram of the ANONYMOUS statement for the FTP.DATA configuration file that is used to define what credentials are required for the user that accesses the FTP server using anonymous as the user ID. Refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 for further assistance configuring the anonymous user.

The following statements are used in FTP.DATA to manage the anonymous user:

Table 6-12 FTP.DATA parameters for an anonymous user

FTP.DATA Statement	Description
ANONYMOUS	Used to allow FTP clients to log in as an anonymous user. **no options**: When an FTP client enters a user ID of anonymous, the FTPD server uses an MVS user ID of ANONYMO. The user will be prompted for a valid password If the user ID ANONYMO is not defined in RACF or if ANONYMO is defined but no OMVS segment created for it, the logon request fails. **userid:** When the FTP client enters a user ID of anonymous, the FTPD server will act as though the user entered the user ID specified. Then the user must enter the valid password for that user ID. All access to resources will be made based on the specified user ID. If this user ID has not been defined in RACF or if the user ID has been defined but no OMVS segment created, the logon request fails. **userid/password:** When the FTP client enters a user ID of anonymous, the FTPD server will act as though the user entered the user ID specified. The FTPD server will also use the specified password in its request to RACF to create a user security environment. (see note **1**)
ANONYMOUSLEVEL	Used to set the type of access permitted to the user who logs in as an anonymous user. **1**: No enhanced support, all other statements listed below in this chart are ignored. FTP server functions using the ANONYMO user ID for anonymous access. The anonymous user has unrestricted access to host resources. **2**: Same as 1, but calls chroot() to restrict the anonymous user to the anonymous user's home directory. (see note **2**) **3**: Enhanced support. All other statements listed below in the chart are enforced. Login credentials are based on parameters of the ANONYMOUS statement. (see note **3**)
ANONYMOUSHFSFILEMODE	Used to set the permission bits for files created by the anonymous user. The default is 000 (---------).
ANONYMOUSHFSDIRMODE	Used to set the permission bits for directories created by the anonymous user. The default is 333 (-wx-wx-wx).
ANONYMOUSFILEACCESS	Sets the type of files that the anonymous user can access. If this access type contradicts the STARTDIRECTORY option, the anonymous user will not be allowed to log in. **MVS**: Only MVS data set access **HFS**: Only HFS file access **BOTH**: The user can access MVS and HFS files
ANONYMOUSFILETYPESEQ	Used to restrict the anonymous user's ability to issue the command `SITE FILETYPE = SEQ` **TRUE**: allowed **FALSE**: not allowed

FTP.DATA Statement	Description
ANONYMOUSFILETYPESQL	Used to restrict the anonymous user's ability to issue the command `SITE FILETYPE = SQL` **TRUE**: allowed **FALSE**: not allowed
ANONYMOUSFILETYPEJES	Used to restrict the anonymous user's ability to issue the command `SITE FILETYPE = JES` **TRUE**: allowed **FALSE**: not allowed
EMAILADDRCHECK	Used to control the extent to which the FTP server validates the e-mail address entered by the FTP client. **NO**: No validation performed **FAIL**: The FTP server reject the login if the e-mail address is not valid (crude validation: only checks for @ sign). **WARNING**: The FTP server returns a warning reply to an invalid e-mail address, then closes the control connection.

Notes:

1 Use this syntax with ANONYMOUSLEVEL 3 to allow the anonymous user to enter an e-mail address in place of a password.

2 For ANONYMOUSLEVEL 2 and greater, when STARTDIRECTORY is HFS, you must create a specific directory structure and contents within the anonymous user's home directory. This directory structure is needed so the FTP client maintains addressability to needed executables once the chroot() is executed.

3 When STARTDIRECTORY HFS, chroot() is called to limit the anonymous user to the anonymous user's home directory. The **`FTP USER`** subcommand is not allowed by the anonymous user and the **`USER ANONYMOUS`** command is not allowed by known FTP users.

The support for anonymous users consistent with the UNIX implementation is only possible with ANONYMOUS user ID/password and ANONYMOUSLEVEL 3. In our sample setup, we defined a user ID called FTPANOM in FTP.DATA (see Figure 6-127) and defined that user with a valid password in RACF (see Figure 6-128).

```
ANONYMOUS ftpanom/password
```

Figure 6-127 Anonymous user support in FTPD

```
ADDUSER  FTPANOM  OMVS(UID(999) HOME(/u/ftp) PGM(/bin/sh)) PASSWORD(password)
```

Figure 6-128 Defining anonymous RACF user ID

The following is a sample of anonymous user statements in FTP.DATA from mvs28b.itso.ral.ibm.com:

```
ANONYMOUS                ftpanom/password
  ANONYMOUSLEVEL           3
  ANONYMOUSHFSFILEMODE     000       1
  ANONYMOUSHFSDIRMODE      333       2
  ANONYMOUSFILEACCESS      BOTH      3
  ANONYMOUSFILETYPESEQ     TRUE      4
  ANONYMOUSFILETYPESQL     FALSE     5
  ANONYMOUSFILETYPEJES     FALSE     6
  EMAILADDRCHECK           FAIL      7
```

Figure 6-129 Anonymous user FTP.DATA keywords

The ANONYMOUS statement includes ftpanom/password which is the actual user ID and password that is used when the anonymous users log in. The ANONYMOUSLEVEL is set to 3, which enables the enhanced anonymous user support. The home directory is /u/ftp for the anonymous user, refer to Figure 6-128 on page 241 where it is defined.

1 A file stored by an anonymous user with the **put** command will have permission bits set to 000 (---------).

2 A directory created by an anonymous user with the **mkdir** command will have permission bits set to 333 (d-wx-wx-wx).

3 The anonymous user can issue the **cd** command to access MVS data sets and HFS files (for example, MVS data sets `cd 'BTHOMPS.JCL'` and HFS files cd /u/ftp/pub). To restrict access of MVS resources, use RACF. The anonymous user will have access to files that the FTPANOM user ID has access to within the MVS file space. If the anonymous user does not have a need to access MVS data sets this statement should be set to HFS as follows:

```
ANONYMOUSFILEACCESS    HFS
```

4 The anonymous user can put or get files based on the imposed permissions.

5 The anonymous user cannot issue FILETYPE=SQL and therefore cannot submit SQL queries or perform any other tasks that require FILETYPE=SQL.

6 The anonymous user cannot issue FILETYPE=JES and therefore cannot submit JCL to have jobs executed or perform any other tasks that require FILETYPE=JES.

7 If the anonymous user enters an invalid e-mail address it will be rejected.

Creating the anonymous user HFS directory structure

If you choose an ANONYMOUSLEVEL greater than one and the STARTDIRECTORY HFS, you must create an anonymous directory structure in the HFS. You can manually create the directory structure or run the sample script /usr/lpp/tcpip/samples/ftpandir.scp and it will do it for you. The script will create the required directories and content as well as the suggested directories (pub, incoming, and extract). If you choose to manually create the directory/file structure for the anonymous user, please consult the *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775. Superuser authority is required to execute the sample script.

The following shows the usage syntax and description of the parameters that can be used with the sample script, /usr/lpp/tcpip/samples/ftpandir.scp.

```
usage: ftpandir <root> [<owner> [<pub> [<incoming> [<extract>]]]]
        <root> Anonymous root directory: default is /u/ftp
        <owner> owner of directory structure and files
           <owner> :=  <user>[:<group>]
           <user> := user name, from the user data base, of
                     the owner of the directory structure
           <group> := group name, from the group data base, of
                      the owner of the directory structure.
        Group - owning group of directory structure and files
                the default is the user's group
        Public subdirectory name: default is pub
        Incoming subdirectory name: default is incoming
        Extract subdirectory name: default is extract
        The owning user and group should not be the anonymous user
```

Figure 6-130   Usage syntax used with script

Running the sample script without parameters creates the following structure.

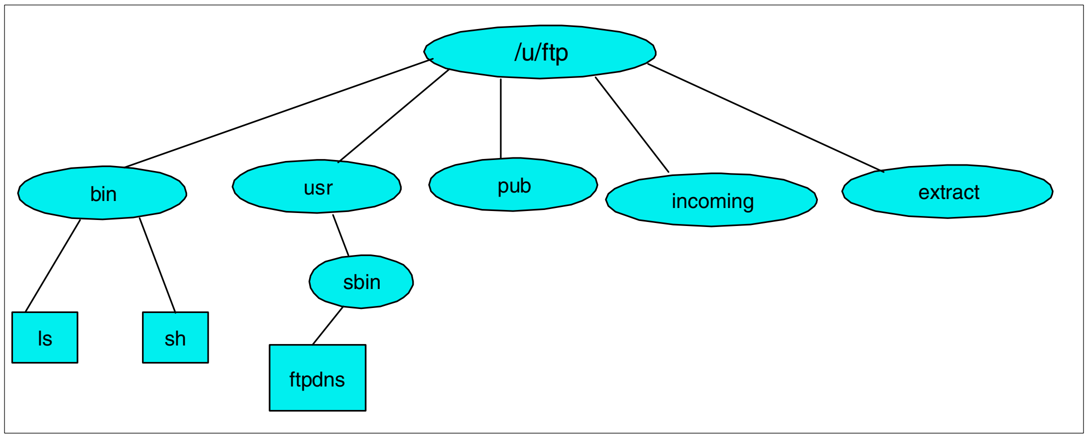


Figure 6-131   ftpandir.scp directory layout

Running the sample script without parameters creates the following permissions:

```
ls -al /u
drwx--x--x   7 FTPD2    SYSPROG     8192 Jul 12 17:55 ftp

ls -al /u/ftp
drwx--x--x   7 FTPD2    SYSPROG     8192 Jul 12 17:55 .
drwxr-xr-x  17 FTPD2    SYSPROG     8192 Jul 12 17:19 ..
drwx--x--x   2 FTPD2    SYSPROG     8192 Jul 12 17:19 bin      1
drwx--x--x   2 FTPD2    SYSPROG     8192 Jul 12 17:52 extract  2
drwx-wx-wx   2 FTPD2    SYSPROG     8192 Jul 12 17:53 incoming 3
drwxr-xr-x   2 FTPD2    SYSPROG     8192 Jul 12 17:55 pub      4
drwx--x--x   3 FTPD2    SYSPROG     8192 Jul 12 17:19 usr      5
```

Figure 6-132   Permissions created by script

If any of the files or directories exist when ftpandir.scp is called, ftpandir.scp will reset the permissions of the files and directories to the ones shown above. When migrating to ANONYMOUSLEVEL 2 or greater, due to the **chroot()** function, the ANONYMOUSLOGINMSG banner may need to be relocated. For example, if you were using /etc/anonymousloginmsg, you may have to move the file to /u/ftp/etc/anonymousloginmsg. At ITSO we were able to use the login message stored in file /etc/anonymousloginmsg without having to relocate it to /u/ftp/etc/anonymousloginmsg.

1 The required directory /u/ftp/bin has the **ls** and **sh** command files stored there so that the anonymous user shell is created and the **ls** command can be performed. The permission bits (drwx--x--x) of the directory do not allow the anonymous user to view a file listing or store (put) files in this directory.

2 The suggested directory /u/ftp/extract can be used for special purposes. Support personnel can place files here and give explicit instructions to the anonymous users to retrieve the file. The permission bits (drwx--x--x) of the directory do not allow the anonymous user to view a file listing or store (put) files in this directory.

3 The suggested directory /u/ftp/incoming can be used by the anonymous user to store files with the **put** command in this directory. The stored file will have permissions bit set to (---------). The permission bit value is set by ANONYMOUSHFTFILEMODE 000. The permission bits (drwx-wx-wx) of the directory do not allow the anonymous user to view a file listing but does allow them to store (put) files in this directory.

4 The suggested directory /u/ftp/pub permission bits (drwxr-xr-x) allow the anonymous user to view file listing and retrieve files. Storing files with the **put** command is prohibited. This is the public directory where various files are available to the general public.

5 Required directory /u/ftp/usr/sbin contains the ftpdns server process.

A sample FTP logon sequence

The anonymous user can also have a predefined banner (welcome page), login message, and informational message presented during the FTP session. Refer to 6.4.9, “Setting up a welcome page” on page 187 for configuration assistance.

```
BTHOMPS:/u/bthomps: >ftp 9.24.104.43
220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe, 23:39:35 on 2002-05-26.
FTP: using TCPIPB
Connecting to:  9.24.104.43 port: 21.
220-FTPDB1 IBM FTP CS V1R2 at wtsc63oe, 17:34:35 on 2002-05-26.
:
:
:
220-This is wtsc63oe.itso.ral.ibm.com on Sun May 26 23:39:35 2002
220-For administrative assistance contact support@helpdesk.com
220-
220 Connection will not timeout.
NAME (9.24.104.43:BTHOMPS):
anonymous   1
>>> USER anonymous
331 Send email address as password please. 2
PASSWORD:

>>> PASS
230 'ANONYMOUS' logged on.  Working directory is "/".
Command:
cd /incoming    3
>>> CWD /incoming
250 HFS directory /incoming is the current working directory.
Command:
put readme.ftp  3
>>> PORT 9,24,104,43,4,29
200 Port request OK.
>>> STOR readme.ftp
125 Storing data set /incoming/readme.ftp
250 Transfer completed successfully.
16 bytes transferred in 0.010 seconds Transfer rate 1.60 Kbytes/sec
Command:
cd /pub  4
>>> CWD /pub
250 HFS directory /pub is the current working directory.
Command:
ls -l     4
>>> PORT 9,24,104,43,4,21
200 Port request OK.
>>> NLST -l
125 List started OK
total 80
-rwxr-xr-x   1 FTPANOM  SYSPROG     5846 Jul 12 21:46 ftp.test.0
-rwxr-xr-x   1 OMVSKERN SYSPROG     5846 Jul 20 19:44 ftp.test.1
-rwxr-xr-x   1 OMVSKERN SYSPROG     5846 Jul 20 19:44 ftp.test.2
-rwxr-xr-x   1 OMVSKERN SYSPROG     5846 Jul 20 19:44 ftp.test.3
-rwx--xr-x   1 OMVSKERN SYSPROG     5846 Jul 20 19:44 ftp.test.4
250 List completed successfully.
Command:
get ftp.test.1  4
>>> PORT 9,24,104,43,4,23
200 Port request OK.
>>> RETR ftp.test.1
125 Sending data set /pub/ftp.test.1
250 Transfer completed successfully.
5939 bytes transferred in 0.020 seconds.  Transfer rate 296.95 Kbytes/sec.
```

1 anonymous entered at the user ID prompt

2 guess@ibm.com entered at the email address prompt

3 Change to the incoming directory and put a file named readme.ftp

4 Change to the pub directory, listing of the file content and get a file named ftp.test.1

Securing ANONYMOUSLEVEL 1

The following can be done with ANONYMOUSLEVEL 1 to enforce some restrictions on the anonymous user.

To limit access to resources in the hierarchical file system, we connected this user to the same RACF group we used for public access to our MVS Web server, the EXTERNAL group.

The user FTPANOM TSO segment is undefined, so that even if someone were able to read FTP.DATA where the password has to be stored in clear text, no TSO access is possible. If the password is compromised, another possible entry into MVS could be through rlogin or telnet. To manage that situation, we defined a .profile in the /u/ftp directory that just contained an **exit** command (see Figure 6-133).

```
echo Hello $LOGNAME.
echo Sorry, you do not have access to the system.
exit
```

Figure 6-133 The /u/ftp/.profile file for the anonymous FTP user

Further we removed ownership of /u/ftp/.profile, assigned OMVSKERN as the owner, and modified the attribute to 045, which restricts FTPANOM to just reading and executing this file. To write a shell history log, we gave .sh_history attributes of 060 so FTPANOM can write to it.

```
pwd=/u/ftp: >ls -nal
total 24
drwx------    2  999      999          0  May  7 01:18  .
drwxr-xr-x   13    0        0          0  May  7 00:58  ..
----r--r-x    1    0      999         69 May  7 01:08  .profile
----rw----    1  999      999         14 May  7 01:48  .sh_history
-rw-r------   1  999      999         57 May  7 01:17  test.scr
```

Figure 6-134 Ownership removed from .profile

6.6.4 FTPD server security user exit routines

In CS for z/OS V1R2 IP you have the option to enable four FTP server security user exits:

FTCHKIP You may use this exit to control which TCP/IP hosts are allowed to connect to the z/OS FTP server. The exit is given control when a new connection is being opened. The exit receives the IP address and the port number of the client host and may return a return code indicating whether the connection should be allowed or denied. At ITSO-Raleigh we used this exit to prevent TCP/IP hosts outside of our test subnets from connecting to the FTP server.

If the connection is denied, the following message will be sent to the user:

```
421 User Exit rejects open for connection.
```

This function can also be performed within the SAF environment, which in a way makes this exit superfluous.

FTCHKPWD You may use this exit to control which user IDs log on to your z/OS FTP server. The exit is given control just after the user has entered a user ID and password. The exit receives both the user ID and the password of the user and may return a return code indicating whether the logon should be allowed or rejected.

If logon is denied, the following message will be sent to the user:

```
530 Logon attempt by 'userid' rejected by user exit.
```

FTCHKCMD In this exit you may control which FTP subcommands a user is allowed to use. For example, you may want to use this exit to disable the **`delete`** command for all users, or to disable the **`put`** command for certain users. The exit receives the user ID, the command, and the argument string entered with the command. The exit may accept or reject the command by returning a return code. At the International Technical Support Organization we used this exit to disable all **`delete`** commands and to prevent any anonymous user from changing the password for our default anonymous user ID (FTPANOM). Refer to 6.4.13, "User exits" on page 198 to learn about user exits FTCHKCMD, and FTPOSTPR.

If the **`DELE (delete)`** command is denied, the following message will be sent to the user:

```
500 User Exit denies Userid 'userid' from using Command 'DELE'
```

FTCHKJES Using this exit you may control which users are allowed to submit MVS batch jobs via the FTP server. You may also implement a check to see if a user is trying to submit a job with a user ID that is not the same as the user ID that was used for logging on to the MVS FTP server. The exit receives user ID and the buffer containing the JCL statements that are about to be submitted via an internal reader to JES. The exit is allowed to modify the passed buffer. The exit may reject the job submission or accept it.

If the remote job submission is denied, either of the following messages will be sent to the user depending on the command issued by the user:

```
125 User Exit refuses this Job to be submitted by userid
550 JesPutGet aborted
```

or

```
550 User Exit refuses this Job to be submitted by userid
```

Security user exit implementation

The FTP server allows you to implement several security-related exit routines. See *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775, for details on these exits.

The user exit load modules must be placed in an APF-authorized library to which the FTP server has access via STEPLIB, LNKLST, or LPA.

Because the FTPCHKIP user exit is loaded at FTP daemon initialization time, if you want the server to use a new version of your exit routine, you have to recycle the FTP server. You may use a test version of a server that runs over non-default port numbers in order to debug a user exit.

Note: You cannot use the System Programming C Facilities for the user exits.

For examples of the exits used at ITSO-Raleigh, see Appendix E, "FTP user exits and sample code" on page 525.

Both the FTCHKIP and the FTCHKPWD routines were very straightforward and we did not experience any problems. An IP address of 9.32.6.2 would be passed to FTCHKIP as X'09200602'.

In the FTCHKCMD routine, you have to be aware that the functions passed to the exit are the standard function names as listed in RFC 959. They are all in EBCDIC. Refer to Appendix E, “FTP user exits and sample code” on page 525 for details.

Some client functions result in multiple FTP commands going to the FTP server, which means that your FTCHKCMD exit may be driven more times per client function. For example, when the user at the client issues a **rename** command, two commands will actually flow: first a rename from (RNFR) followed by a rename to (RETO).

The FTCHKJES exit is called for every logical record that the FTP server receives while it is in filetype=JES mode. The exit sees the records after they have been translated to EBCDIC. The format of the buffer that is passed to the exit depends on:

Jesrecfm	The jesrecfm value set by a **SITE** command; default is F for fixed length
Jeslrecl	The jeslrecl value set by a **SITE** command; default is 80 bytes.

If you plan to use this exit, please see Appendix E, “FTP user exits and sample code” on page 525 for details on the interface.

While you debug your FTCHKJES exit, you may find it of value to issue two **Modify** commands:

1. **F ftpd1,JTRACE**
2. **F ftpd1,JDUMP**

Detailed trace information about the interface to your FTCHKJES exit is then sent to syslogd.

If you have configured the BPX.DAEMON facility class in your security database, make sure that these modules are located in a program controlled library too. The FTPD server modules load the security exits when they are needed. If these exit routines are located in a library that is not program controlled, MVS sets the dirty bit and the FTP server modules will not be able to change the user security environment as required. If you inadvertently place the FTP server security exits in a wrong library, the FTP client end user will receive an error message, and the following messages will appear in your syslogd destination for the *daemon* logging facility name:

```
parse_cmd: entering parse_cmd.
get_command: select rc is 1
get_command: received 10 bytes
parse_cmd: calling user exit FTCHKCMD with: rc 0, numparms 3, userid '',
                                            cmd 'USER' args '3/hdm'.
parse_cmd: return from FTCHKCMD with rc: 0.
user: user routine entered with username 'hdm'.
get_command: select rc is 1
get_command: received 15 bytes
parse_cmd: calling user exit FTCHKCMD for PASS cmd.
parse_cmd: return from FTCHKCMD with rc: 0.
pass: pass routine entered.
pass: calling user exit FTCHKPWD with: rc 0, numparms 3, userid 'HDM'.
pass: return from chkpwd with rc: 0.
pass: termid is '091868D0'.
pass: return from racf with: saf: 0, racf: 0,racf reason: 0 acee: 006F10D8.
pass: calling delacee to delete useracee 006F10D8.
pass2: seteuid failed (157/0B7F02AF) : EDC5157I An internal error has occurred.
```

Figure 6-135 Daemon error log for FTPD

Note: We have discarded the time stamp, process ID, etc., from the above lines.

The errno 157 means that an MVS environmental or internal error occurred. The setuid() API returns the 4-byte long UNIX System Services reason code, which consists of a halfword reason code qualifier (2-bytes long) and a halfword reason code (2 bytes). The reason code 02AF means that the address space has become dirty and the specified function is not supported in a dirty address space.

6.6.5 Using a SURROGATE user

Another way to enable anonymous support in the z/OS FTP server is to specify a RACF userid in the servers FTP.DATA configuration file.RACF SURROGATE support bypasses the need for defining a password in FTP.DATA. As can be seen in Figure 6-136 a userid as defined in FTP.DATA is allowed to logon with a password of SURROGATE. When this password is used, the FTP server calls RACF and checks if userid ‘NUTALL” is allowed to login without a password

```
DEBUGONSITE TRUE
DUMPONSITE TRUE
PORTCOMMAND ACCEPT
PORTCOMMANDPORT NOLOWPORTS
PORTCOMMANDIPADDR NOREDIRECT
ANONYMOUS guest/SURROGATE
```

Figure 6-136 FP.DATA showing SURROGATE userid

1. Activate the SURROGATE class support in RACF

- SETROPTS CLASSACT(SURROGAT)

  This has to be done only once on the system. It is possible that the SURROGAT class may already have been set up on your system.

2. Create the Surrogate class profile for user ‘guest’

- RDEFINE SURROGATE BPX.SRV.guest UACC(NONE)
- SETROPTS RACLIST(SURROGAT) REFRESH

3. Permit the FTP daemon FTPDB to create a security environment for user ‘guest’, by issuing the PERMIT command

- PERMIT BPX.SRV.guest CLASS(SURROGATE) ID(FTPDB) ACCESS(READ)
- SETROPTS RACFLIST(SURROGATE) REFRESH

```
StartDir        MVS     1
RDW             false      ; Do not retain RDWs as data
EXTENSIONS      SIZE       ; Server can respond to SIZE cmd
EXTENSIONS      MDTM       ; Server can respond to MDTM cmd
EXTENSIONS      UTF8       ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
ANONYMOUS GUEST/SURROGATE
ANONYMOUSLEVEL 3
ANONYMOUSFILEACCESS MVS     2
```

Figure 6-137 FTP.DATA with anonymous parms

Note: Is is important to keep in mind that **ANONYMOUSFILEACCESS** defaults to HFS. Your FTP.DATA **STARTDIRECTORY** entry and **ANONYMOUSFILEACCESS** entry has to match, otherwise message ‘530-StartDirectory MVS/HFS is disabled for anonymous’ will be issued.

As is shown in Figure 6-138 we logged in with a userid of ‘ANONYMOUS” and a password of ‘SURROGATE”.

```
 Connecting to:  9.12.6.63 port: 21.
 220-FTPDB1 IBM FTP CS V1R2 at wtsc64oe.itso.ibm.com, 18:39:12 on 2002-05-11.
 220 Connection will close if idle for more than 5 minutes.
 NAME (9.12.6.63:GARTHM):
ANONYMOUS
 >>> USER ANONYMOUS
 331 Send email address as password please.
 PASSWORD:
SURROGATE

 >>> PASS
 230 'ANONYMOUS' logged on.  Working directory is "GUEST.".
 Command:
```

Figure 6-138 Surrogate login session

6.6.6 Guarding against bounce attacks

One of the well known ways of poaching data from a remote server without having the required access is by using the “FTP server bounce attack” mechanism. A client can send a PORT command with IP address of SMTP server and number of 25. The FTP server will connect to the SMTP server, assuming that it is an FTP client. A client can attack the SMTP server by issuing a RETR command to send SMTP files over to the FTP server, even though the client was never authorized for a such transaction. Instructing a third party to connect to the service, rather than connecting directly, makes tracking down the attacker very difficult.

In Figure 6-139 on page 251, we see FTP client A connecting to the FTP server that is listening to PORT 21. FTP client A then uses the PORT command to tell the FTP server to initiate a dataconnection to PORT 25, where the SMTP server is listening. FTP client A follows the PORT command with RETR command that prompts the FTP server to send file A that contains SMTP commands to the SMTP server. Because SMTP is receiving recognizable SMTP commands, it returns OK replies and receives whatever data the FTP server is sending.This demonstrates that it is quite trivial for an outside FTP client to bypass security restrictions you impose within your environment. Using a third party to connect to well known services can be used for many destructive purposes. Similar methods can be used to post virtually untraceable mail, cause disruption on servers, fill up disks, bypass firewalls, and generally be annoying and difficult to track down.

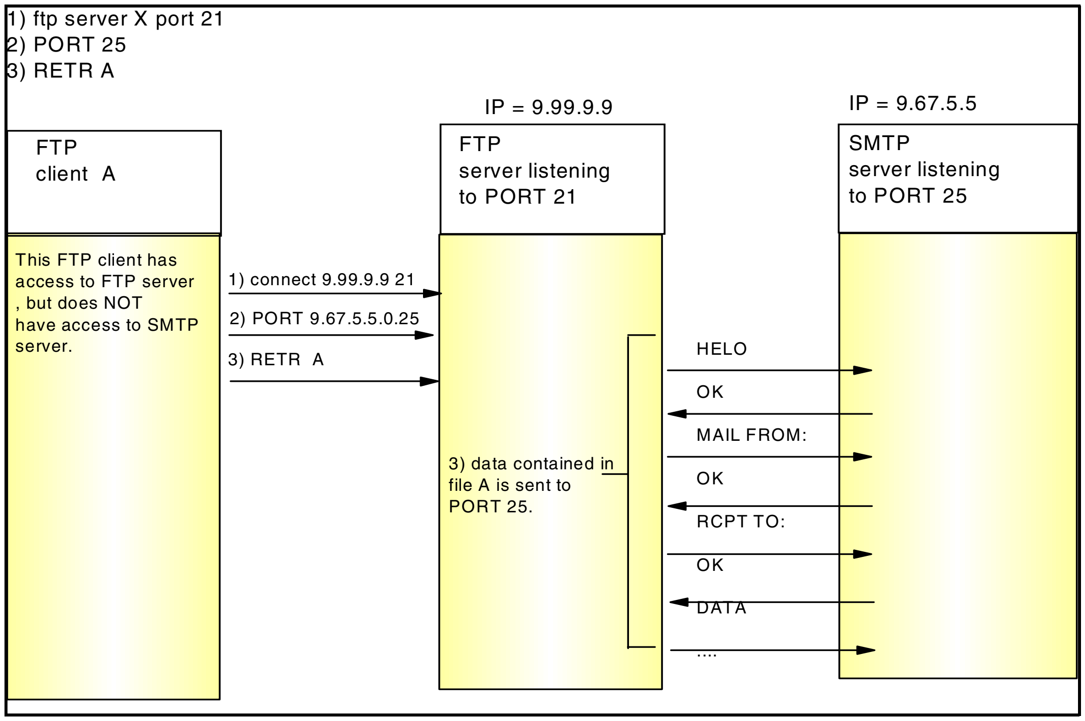

1) ftp server X port 21
2) PORT 25
3) RETR A
IP = 9.99.9.9
IP = 9.67.5.5
FTP client A
FTP server listening to PORT 21
SMTP server listening to PORT 25
This FTP client has access to FTP server , but does NOT have access to SMTP server.
1) connect 9.99.9.9 21
2) PORT 9.67.5.5.0.25
3) RETR A
3) data contained in file A is sent to PORT 25.
HELO
OK
MAIL FROM:
OK
RCPT TO:
OK
DATA
....

Figure 6-139 FTP bounce attack

What can be done to prevent this?

Preventing the use of the PORT command will go a long way to preventing this exposure. These command are defined as part of FTP.DATA and can be coded as follows:

PORTCOMMAND, which has default value of ACCEPT, tells FTP server whether it should accept PORTCOMMAND or not.

PORTCOMMANDIPADDR, which has default value of UNRESTRICTED, tells FTP server whether it should accept PORT commands with IP address that's different from the IP address of the FTP client.

PORTCOMMANDPORT, which has default value of UNRESTRICTED, tells FTP server whether they should accept PORT commands with PORT number that is less than 1024.

Figure 6-140 on page 252 shows how we restricted the use of the PORT command in FTP.DATA.

```
EXTENSIONS      UTF8       ; Server can respond to LANG & UTF-8 enc.
EXTENSIONS      REST_STREAM  ; Server can respond to SIZE cmd
DEBUGONSITE TRUE
DUMPONSITE TRUE
PORTCOMMAND ACCEPT
PORTCOMMANDPORT NOLOWPORTS     1
PORTCOMMANDIPADDR NOREDIRECT
```

Figure 6-140 FTP.DATA including PORT restrict commands

1 Restrict the use of the Port command by not accepting a PORT number that less than 1024. In this case getting to our SMTP server using the ‘PORT 25’ command will fail.

6.6.7 Transport Layer Security (TLS) and Kerberos

Details for both TLS and Kerberos can be found in the *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840. This book deals with how to setup your FTP environment for TLS using digital certificates and for Kerberos using the Key Distribution Centre(KDC).

7

Network File System (NFS)

The Network File System (NFS) support is provided by z/OS Network File System (z/OS NFS). With the z/OS NFS server, you can remotely access MVS/ESA conventional data sets or z/OS UNIX files from workstations, personal computers, and other systems that run client software for the Sun NFS Version 2 protocols, the Sun NFS Version 3 protocols, and the WebNFS protocols over a TCP/IP network.

This chapter contains the following sections:

7.1 Introduction to NFS

The Network File System (NFS) was developed by Sun Microsystems in 1985 and has been implemented on various platforms including as Sun Solaris, HP/UX, AIX, Linux, Windows, and z/OS. NFS clients can use the resources of NFS servers transparently as if they were their own local file systems.

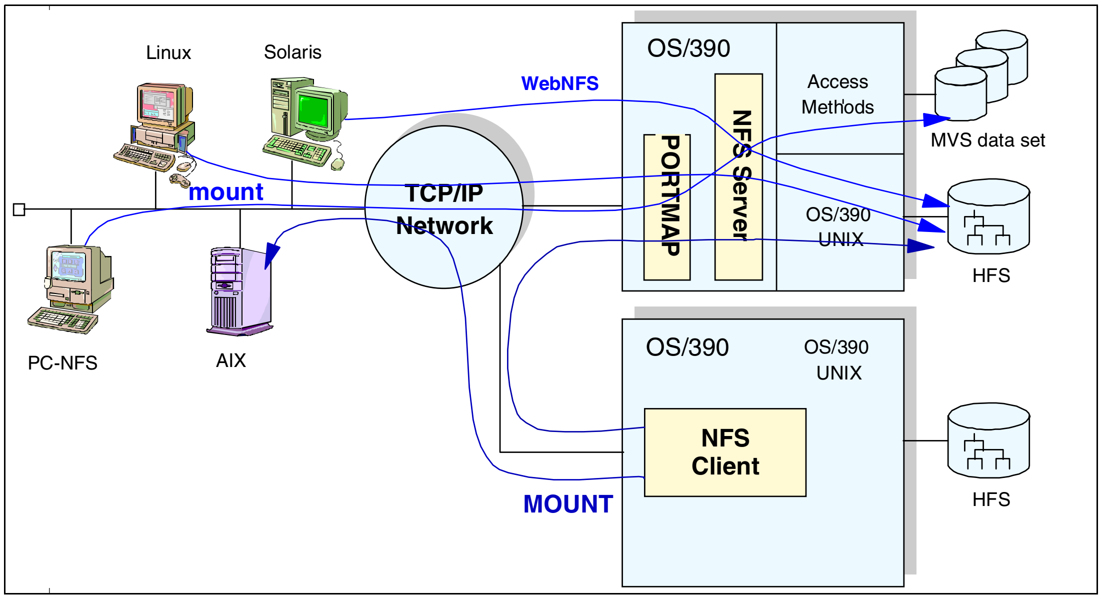


Figure 7-1 z/OS NFS overview

Client systems in a TCP/IP network that support the NFS client protocol can use traditional MVS data sets and UNIX System Services HFS files as part of their file system. Currently, z/OS UNIX System Services HFS files cannot be physically shared in read/write mode among multiple z/OS systems. Therefore, the combination of the z/OS NFS server with the z/OS NFS client is one of the alternative solutions that you can use to share z/OS UNIX System Services HFS files through a TCP/IP network.

NFS can use TCP or UDP to transport its data, and has been enhanced in recent releases of z/OS. Since the purpose of this section is to discuss z/OS NFS security, if you want to know more about the z/OS NFS Server, see *z/OS Network File System Customization and Operation,* SC26-7417.

The z/OS Network File System server allows TCP/IP client systems to access the following data set types on your z/OS:

1. MVS data sets
   - Direct access (DA)
   - Partitioned data set (PDS)
   - Partitioned data set extended (PDSE)
   - Physical sequential (PS)
   - Virtual storage access method (VSAM)
     - Entry-sequenced data set (ESDS)
     - Key-sequenced data set (KSDS)
     - Relative record data set (RRDS)

These data sets can use the following record formats:

- Fixed length
- Variable length
- Undefined
- Blocked and spanned

Note: Only ICF catalogued data sets are supported. There is no support for tape data sets or generation data groups.

2. UNIX System Services (Hierarchical File System) files

   In OS/390 V2R6 IP, OS/390 NFS was enhanced with the following functions:

   - WebNFS server and Internet

     WebNFS eliminates the overhead of PORTMAP and MOUNT protocols, making the protocol easier to use through corporate firewalls. It also reduces the number of LOOKUP requests that are required to identify a particular file on the server.

   - File locking over the z/OS NFS server

     The z/OS Network File System Network Lock Manager (NLM) and the z/OS Network File System Network Status Monitor (NSM) are Remote Procedure Call (RPC) based servers that execute as autonomous "daemon" servers on NFS server systems. NSM and NLM work together to provide file locking and access control capability with the z/OS NFS server.

   - File name extension mapping support

     The z/OS Network File System server is enhanced to provide file extension mapping for members of a Partition Data Set (PDS) and Partition Data Set Extended (PDSE) that are mounted via the Network File System from a client machine. The file name extension mapping capability is provided by the use of "side files" on the MVS host specified by the system administrator and the user during the MOUNT operation.

   - Sun NFS Version 3 protocol support for both client and server functions

     The NFS Version 3 protocol is a revision of the NFS Version 2 protocol. It supports larger files and file systems by allowing 64-bit sizes and offsets. The NFS Version 3 protocol enhances performance by returning attributes for every procedure, thus reducing the number of calls requesting modified file attributes. The NFS Version 3 also enhances performance by adding support to allow the NFS server to do asynchronous writes and by relaxing the limitations on transfer sizes.

   - TCP support

     In addition to UDP, the z/OS NFS server supports communications over the TCP protocol. Any client platform with TCP support can choose to access the z/OS NFS server using TCP.

7.1.1 Accessing data sets

The z/OS NFS server acts as an intermediary to read, write, create or delete z/OS UNIX files and MVS data sets that are maintained on an MVS host system. The remote MVS data sets or z/OS UNIX files are mounted from the host processor to appear as local directories and files on the client system. This server makes the strengths of an MVS host processor -- storage management, high-performance disk storage, security, and centralized data -- available to the client platforms.

We recommend that you refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840, which addresses z/OS NFS security issues.

Reading and writing data sets

When the Network File System server accesses a data set, it first completes the client's request before continuing with the next request for that data set. The Network File System server treats each request as an individual operation, which allows different NFS clients to simultaneously have read access to a file.

The Network File System server does not maintain control information about the client's activities. This means that if a request fails, a client may retry the request without re-establishing the session with the Network File System server.

The Network File System server also uses the remote procedure call (RPC) protocol. This allows users on TCP/IP NFS client systems to create applications that ensure that any attempts to write to DASD are synchronized.

Creating data sets

Users on TCP/IP NFS client systems can also create MVS data sets with the Network File System server. When creating data sets, the user on the client system needs to know the data set attributes that describe how data sets are structured, located, and stored. The Network File System server gives experienced users the flexibility to pass these creating and processing attributes directly to MVS.

7.2 Configuring the Network File System (NFS)

The network file system (NFS) server is shipped as a component of DFSMS/MVS.

Please refer to the following manuals for details on NFS:

- For installation and setup: *z/OS Network File System Customization and Operation*, SC26-7417.
- For usage guidelines: *z/OS Network File System User's Guide*, SC26-7419.

The NFS protocol was developed by SUN Microsystems in 1984 and is a mechanism for sharing data across heterogeneous independent systems.

With file transfer protocol (FTP), you can copy files between TCP/IP host systems, creating a copy of a file in the local file system of each host. If you distribute a file to three hosts, the file will exist in three physical copies.

With NFS you share one physical copy of your file among hosts in your TCP/IP network. The file exists once and can be accessed simultaneously by all authorized hosts in the network.

NFS is based on the SUN remote procedure call (RPC) and the accompanying eXternal Data Representation (XDR).

The version of the NFS protocol that is implemented on MVS is Version 2 as described in RFC 1094 and Version 3 as described in RFC 1813.

One of the fundamental ideas put into the NFS protocol is the idea of being as *stateless* as possible. This means that an NFS server should not need to maintain any protocol state information about any of its clients in order to function correctly. In case of a failure, an NFS client need only retry a request until the server responds; it does not need to be aware of the fact that the server may have crashed and has been restarted, or that the whole network went down and came up again. Each NFS operation should be as *idempotent* as possible. Idempotent means that each request from the client to the server can be repeated an unlimited number of times and will return the same result each time. Not all operations can be implemented in an idempotent fashion; if you delete a file, it is deleted after the first time the server executes the operation. Retries of the operation, caused by network failures or other malfunctions, will not yield the same result as the first time the operation was performed. Operations such as read and write are implemented in an idempotent fashion in NFS.

7.2.1 The NFS file system model

NFS has its roots in the UNIX world, which impacts the assumptions made by NFS. NFS assumes that a *file system* is hierarchical with directories of files. Each entry in a directory represents a file, a directory or even a device as you will find in UNIX implementations, and has a name expressed as a character string. Each operating system may put limitations on the depth of the hierarchy or the allowed syntax for names. An OS/2 or a DOS system uses a "\" character to separate directory entries in a full path name, whereas a UNIX system uses a "/" character and an MVS system uses a ".". UNIX System Services uses the same syntax as other UNIX systems.

```
DOS or OS/2:  d:\level1\level2\filename.xyz
UNIX:         /level1/level2/filename.xyz
MVS:          level1.level2.filename.xyz
       - or level1.level2.filename(xyz)
```

Figure 7-2 Path Name Syntax

With NFS you *mount* a file system or parts of a file system residing on the NFS server into your local file system, combining the two file systems into one virtual file system.

Look at Figure 7-3 for an extract of a UNIX file system.

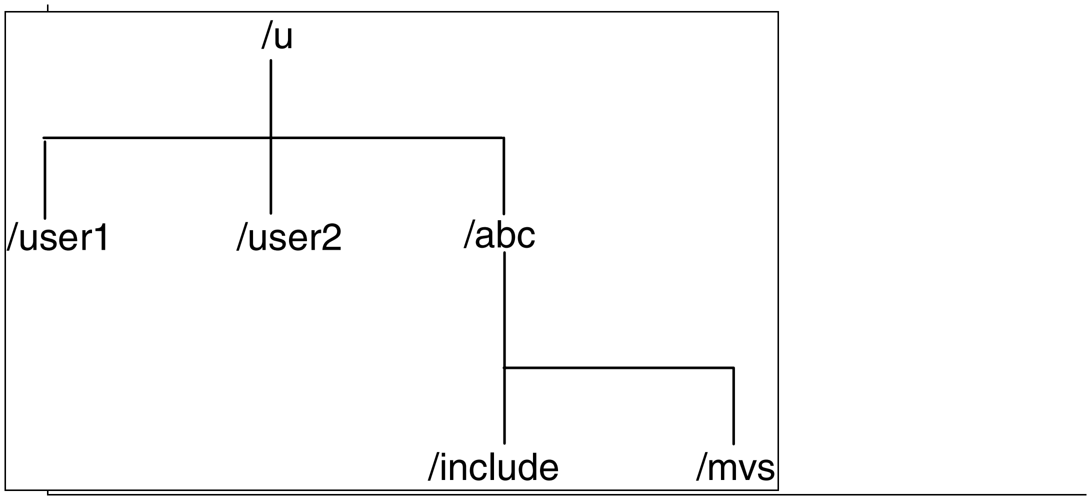


Figure 7-3 Extract of a UNIX file system

The /u/abc/mvs subdirectory is an empty directory we have created as a place holder for the MVS file system we want to mount.

Look at Figure 7-4 for an extract of an MVS file system.

In NFS, a partitioned data set (PDS or PDSE) is considered a directory and the members of the partitioned data set as individual files.

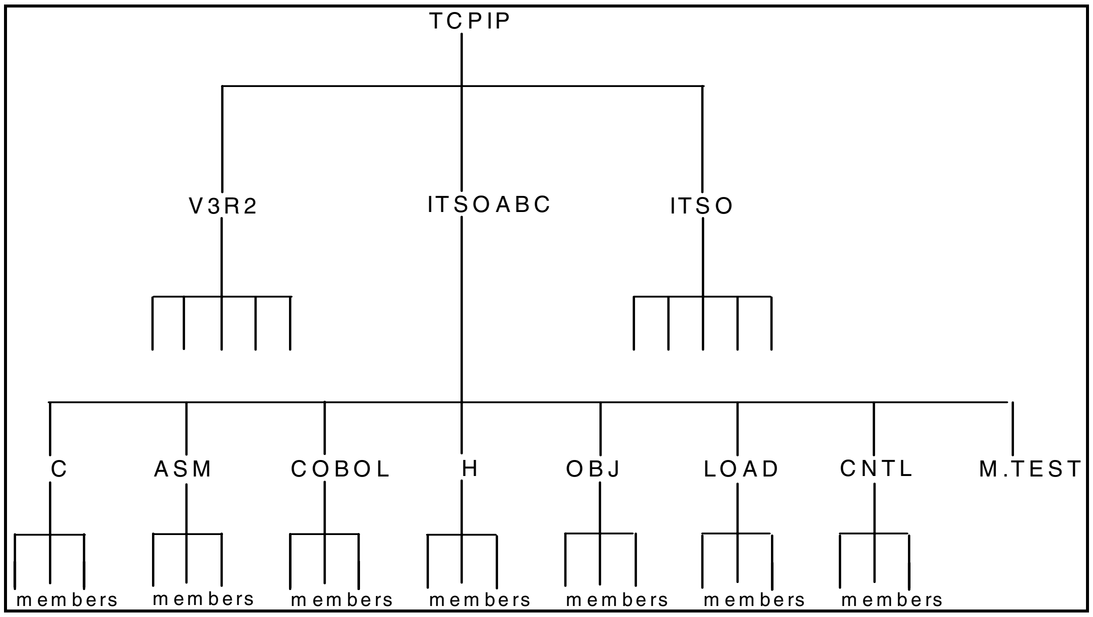


Figure 7-4 Extract of an MVS file system

To combine the TCPIP.ITSOABC part of the MVS file system with the file system on our AIX system, issue a **`mount`** command on AIX:

```
mount mvs18:tcpip.itsoabc /u/abc/mvs
```

Figure 7-5 Mount an MVS file system onto an AIX file system

This command will map our MVS file system starting from TCPIP.ITSOABC into the AIX file system at the /u/abc/mvs point, giving us a view of a file system, as shown in Figure 7-6.

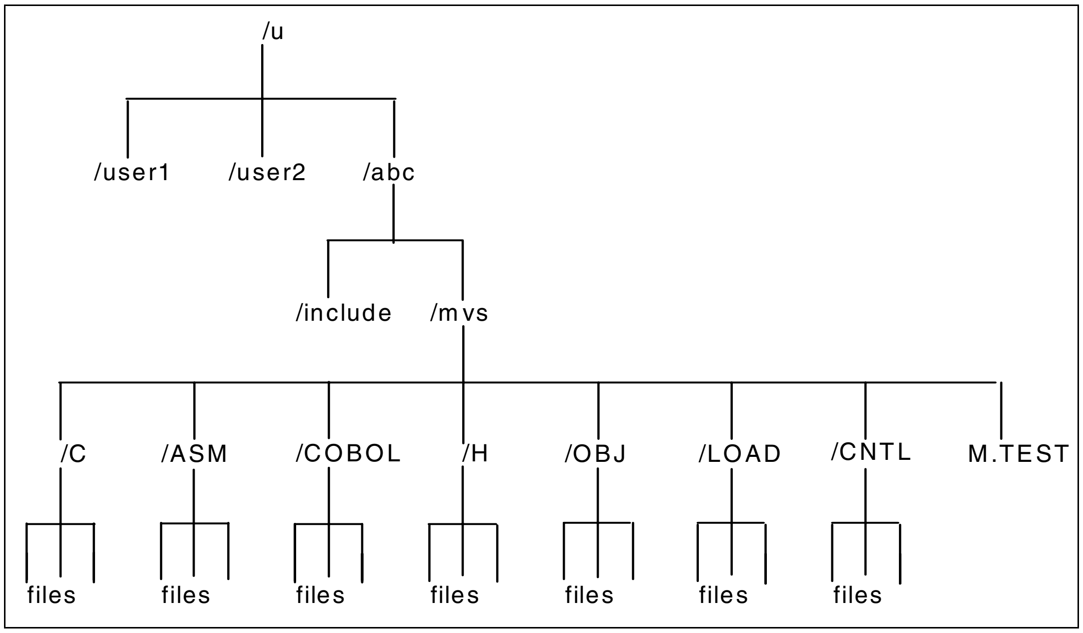


Figure 7-6 Combined file system - AIX view

A list of the /u/abc/mvs directory on AIX will give the following result:

```
$ ls -n
total 4016
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 asm
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 c
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 cntl
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 cobol
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 h
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 load
Frw-rw-rw-   1 0       0       2048838 Aug 21 09:30 m.test
Drwxrwxrwx   2 0       0          1024 Aug 21 09:30 obj
$
```

All the partitioned data sets are considered directories while all other data set organizations on MVS will be considered as files residing in this directory.

You can traverse down the directories to list members of, for example, the tcpip.itsoabc.h partitioned data set:

```
$ cd h
$ ls -n
total 45
Frw-rw-rw-   1 0       0          3344 Aug 13 15:00 bank
Frw-rw-rw-   1 0       0           633 Aug 13 15:10 bankdefs
Frw-rw-rw-   1 0       0           106 Aug 12 09:07 rg
Frw-rw-rw-   1 0       0         10291 Aug 13 10:54 rpc
Frw-rw-rw-   1 0       0          7015 Aug 13 10:53 socket
$
```

The members of a partitioned data set are considered to be files.

The NFS server on MVS will use the statistical information generated by ISPF/PDF to obtain date and time information for files, which are members of MVS partitioned data sets. When you create a file, which ends up as a member in an MVS partitioned data set, the DFP NFS server will generate the same kind of statistical information for the new member, as would have been done by ISPF/PDF.

7.2.2 Byte stream and record mapping considerations

The basic file structure of operating systems such as OS/2, AIX or DOS is byte stream-oriented. The MVS concept of records and blocks is unknown to these kinds of operating systems, and as NFS assumes a byte stream oriented file structure, the NFS MVS-based server must hide the notion of records and blocks from the NFS clients.

NFS client requests to read or write data from/to a file are expressed as: read/write a number of bytes starting at a given offset in bytes from the beginning of the file. Before the MVS NFS server can execute such a request, it must first determine which record in the MVS data set contains the offset specified in the NFS request. To do so, the MVS NFS server reads forward from the last known location in the data set until the record that contains the requested byte offset is found. The MVS NFS server will maintain an in-storage cache over mappings between records and byte offsets. This cache will be kept updated as long as the data set is opened by the MVS NFS server.

Accessing large data sets in a random fashion may result in a significant number of I/O operations, until a complete cache of record-to-byte offset mappings has been constructed for the data set.

7.2.3 Accessing EBCDIC data sets from ASCII hosts

When you access MVS text data sets via NFS, you have to specify what kind of *end-of-line processing* is appropriate for the operating system you are working from. An OS/2 or DOS operating system uses the carriage return/line feed sequence (CRLF) to denote the end of the line, while a UNIX-based system only uses the line feed character (LF) to denote the end of a line. The DFP NFS server supports the following end-of-line processing options:

Option	Description
lf	Line feed only is terminator. This is a standard UNIX notation.
cr	Carriage return only is terminator.
lfcr	The sequence of line feed followed by carriage return is terminator.
crlf	The sequence of carriage return followed by line feed is terminator. This is standard OS/2 and DOS.
noeol	No end-of-line terminator is used.

If you access an MVS mounted file as text and crlf, the MVS NFS server will translate the data from EBCDIC to ACSII and append the CRLF character sequence to every record it reads in an MVS data set. For writes, it will translate from ASCII to EBCDIC, and strip off all CRLF sequences considering each CRLF as a logical record boundary.

If you are working with fixed length records in text mode, you can save transmission time if you use the blankstrip processing attribute. During reads, the MVS NFS server will strip trailing blanks at the end of each record and will append trailing blanks up to the specified record length on writes. The blankstrip attribute is the default. You can switch it off by using the noblankstrip processing attribute.

The MVS NFS server will by default use the STANDARD.TCPXLBIN translate table for ASCII - EBCDIC translations.

7.2.4 Access serialization to data sets

The MVS NFS server will use shared enqueues for read and write, and exclusive enqueues on QNAME=SPFEDIT and RNAME=datasetname for writes.

The exclusive enqueue on write is kept after the write has been completed for as long a period as specified in the processing attribute *writetimeout*, which defaults to 30 seconds. You have also a readtimeout attribute, which defaults to 90 seconds before a data set allocated for read operations will be released.

If you want to share data sets for update between the NFS server and TSO users using ISPF/PDF, you have to consider the following:

- The exclusive write enqueue is kept only for the writetimeout period. If you begin editing an NFS mounted file on your local TCP/IP host, the NFS server will enqueue on QNAME=SPFEDIT protecting your data set or PDS member from TSO users, but the protection will last only for the writetimeout period. If you have not saved the file from your local TCP/IP host before this timer expires, the file is no longer protected and other users may update it. You will then end up overwriting when some time later you exit your local editor. From this point of view, you may want to set a high writetimeout value.
- If you update a data set or a member of a PDS, the enqueue on QNAME=SPFEDIT will remain active for the period specified in writetimeout making it unavailable to other users in that period. From this point of view, you may want to set low writetimeout value.

In a UNIX environment, there is normally no protection when more users are allowed to update the same file. If two users edit the same file at the same time, the one who saves the file first will lose his changes when the second user saves the file. This is the way of life in a UNIX environment. The MVS NFS server does not protect two NFS clients from each other. The serialization scheme imposed by the MVS NFS server does not impose serialization between NFS clients.

7.2.5 Preparing to use the z/OS NFS server

To set up the network file system server, you will have to perform the following tasks:

- Define your default NFS attributes.
- Define the EXPORTS entries.
- Distribute the **`mvslogin`**, **`mvslogout`** and **`showattr`** commands to NFS client hosts.

Please note that the RACF user ID, under which the MVS NFS server task is running, must be defined with the OPERATIONS attribute in RACF.

NFS attributes data set

This data set contains three sets of attributes:

1. The NFS server site attributes:

   The site attributes specify the normal operating parameters for the NFS server which include the size of buffers and cache, and the number of subtasks to start. These attributes cannot be changed by the NFS client.

We will mention two important site attributes:

– Security

  The NFS server supports various levels of security:

  - NONE
  - Exports data set (exp)
  - RACF (saf)
  - RACF and the exports data set (safexp)

– UNIX System Services file access

  The HFS attribute specifies the prefix to use to indicate that the path name points to an HFS file. The default is hfs; nohfs means no access to HFS files.

2. The NFS server processing attributes:

   These attributes affect the way a data set is transmitted (binary or text transfer, fastfilesize or not, write) between the MVS NFS server and the NFS client and readtimeout values. An NFS client can modify the processing attributes for the duration of an NFS session. The values you specify here are the default values used.

3. Data set creation attributes:

   These attributes correspond to the data control block (DCB) parameters you specify when you create a new data set which includes space, blksize, lrecl, etc. The values you specify here are the default values. The NFS client can override these values during the NFS session.

Please refer to *z/OS Network File System Customization and Operation*, SC26-7417 for details about these attributes.

To change the default attributes, you have to close down the NFS server and restart it with the new attribute definitions. The MVS NFS server maintains information about active mounts in a file handle data set. When the server is restarted, the mounts that were active, when the server was closed down, will remain active. The new NFS processing and data set creation attributes will not become active for those mounts until the client unmounts and mounts the file systems again.

The NFS EXPORTS data set

In the EXPORTS data set, you specify what parts of your MVS file system may be mounted from which NFS client hosts.

An entry in the EXPORTS data set consists of a partial or full MVS data set name plus a list of host names that are allowed to mount the file system.

```
/HFS/u/karl              -rw=mvs03a:mvs03c
 /HFS/u/camiluc           -rw=mvs03a:mvs03c
 /HFS/u/vandeke           -rw=mvs03a:mvs03c
 WOZA.PRIV.LIB
```

Figure 7-7 NFS EXPORTS dataset

The above example exports all HFS files listed as read/write to the TCP/IP hosts listed and read-only to all other TCP/IP hosts; the UNIX System Services directory */u/karl*, for example, can be accessed by mvs03a and mvs03c only.

You can use the following options in an EXPORTS entry: An entry for a directory is specified as follows:

```
>>--directory-----ro----------------------------------------------------->
               Ã      <-,----+                        Ã
               +--rw=--client--------------------------Ã
               Ã          <-,----+                    Ã
               +--access=--client----------------------+
                                    Ã      <-,----+  Ã
                                    +-,rw=--client---Ã
                                    +-,ro------------+
```

directory

MVS high-level qualifier, data set name, or alias to a user catalog; the name must conform to MVS data set naming conventions unless UNIX System Services is used (for an HFS directory entry, you need to use the hfs prefix that is used at your site).

-ro

Export the directory as read only. If not specified, the directory is exported as read/write.

rw=client:client...

The directory is exported as read/write to specified clients, and read-only to everyone else. Separate client names by colons.

-access=client:client...,rw=client†:client... Ι,ro

Gives access only to clients listed.

If neither rw nor ro is specified for the -access parameter, then the clients listed have read/write access and the rest of the clients have no access.

If the rw parameter is specified for the -access parameter, the associated clients have read/write access to the directory, and the clients specified in the access list but not in the rw list has read-only access.

If the ro parameter is specified for the -access parameter, the clients in the access list have read-only access to the directory, and the rest of the clients have no access.

If no options are specified, the default value allows any client to mount the given directory with read/write access. Separate client names by colons.

You cannot mix the three options on the same entry.

The actual data set read or write authorization will be done based on the RACF user ID, specified during the NFS logon to the MVS NFS server.

Later updates to the EXPORTS data set may be activated by the NFS modify command:

```
f  procname,exportfs
```

Distributing MVSlogin, MVSlogout and Showattr commands

The NFS protocol assumes authorization is based on the UNIX style uid and gid values, which are passed from the NFS client to the NFS server on each request. MVS has no knowledge of uid and gid but requires the user to identify himself by means of a valid MVS user ID and password.

As the NFS protocol itself does not include logon or logoff functions, these functions have been added by three commands, which are distributed along with the MVS NFS server code. You have to download these commands to all NFS clients that intend to mount and use a file system on MVS.

Operating systems that do not allow command names to be more than eight bytes long will use an mvslogut command name and not the mvslogout command name.

Executable modules are delivered for OS/2 and DOS. For all other environments, C source code is delivered, which you have to download and compile.

See *z/OS Network File System Customization and Operation*, SC26-7417 for details.

You may mount the MVS file system from your TCP/IP host during system startup, but no files will be available until an *mvslogin* is performed.

For multi-user hosts, the MVS file system may be mounted once, but each user will have to issue an mvslogin command in order to gain access to the file system.

The MVS NFS server is not stateless in respect to logon. If the server is stopped and restarted, all users will have to reissue logon. If a user is inactive for more than the period specified in the *logout* attribute, the user will be automatically logged off MVS, and the user will have to issue a new mvslogin command in order to regain access to files in the mounted file system.

The MVS NFS server also supports the PCNFSD protocol.

7.2.6 Using the DFP Network File System server from AIX

To mount a file system, you may either have it done during AIX startup, or you may manually mount a file system any time after startup.

To mount the TCPIP.ITSOABC data sets from MVS into the /u/mvsmnt directory, we used the following input parameters to the *add a file system for mounting* function in SMIT:

```
* PATHNAME of mount point                 1      [/u/mvsmnt]
* PATHNAME of remote directory            2      [tcpip.itsoabc,text,lf]
* HOST where remote directory resides     3      [mvs18]
  Mount type NAME                                  []
* Use SECURE mount option?                          no
* MOUNT now, add entry to /etc/filesystems or both?  now
* /etc/filesystems entry will mount the directory    no
   on system RESTART.
* MODE for this NFS file system                     read-write
* ATTEMPT mount in foreground or background         background
  NUMBER of times to attempt mount                 []
  Buffer SIZE for read                    4      [1024]
  Buffer SIZE for writes                  5      [1024]
  NFS TIMEOUT. In tenths of a second               []
  Internet port NUMBER for server                  []
* Mount file system soft or hard          6       soft
  Allow keyboard INTERRUPTS on hard mounts?         yes
  Minimum TIME, in seconds, for holding            [3]
   attribute cache after file modification
  Maximum TIME, in seconds, for holding            [60]
   attribute cache after file modification
  Minimum TIME, in seconds, for holding            [30]
   attribute cache after directory modification
  Maximum TIME, in seconds, for holding            [60]
   attribute cache after directory modification
  Minimum & Maximum TIME, in seconds, for          []
   holding attribute cache after any modification
  The Maximum NUMBER of biod daemons allowed       [6]
   to work on this file system
* Allow execution of SUID and sgid programs         yes
   in this file system?
* Allow DEVICE access via this mount?               yes
* Server supports long DEVICE NUMBERS?              yes
```

1 The AIX directory onto which we mount the TCPIP.ITSOABC data sets from MVS.

2 The data sets are mounted with a default processing attribute of text and line feed end-of-line processing.

3 The file system is on a TCP/IP host with the name mvs18.

The LAN segment, to which this AIX system was connected, had about 30% errors. The only way to get acceptable performance was to use a low buffer size for both read 4 and write 5.

6 Because of the poor LAN segment quality, we only wanted the default number of retransmissions in case of serious errors. If you specify hard mount, the NFS server and client will retry requests indefinitely. When you specify soft mount, the client and server will only retry failing requests a certain number of times before an error is reported to the user.

After having mounted the file system, the individual users on the AIX system must issue **mvslogin** commands to be allowed access.

```
$ mvslogin mvs18 christe   1
Password required
GFSA973A Enter MVS password:
GFSA955I christe logged in ok.
```

1 The host name is mvs18, and the user ID is christe. The user is prompted for the password, which is not echoed onto the screen.

To edit a member of the TCPIP.ITSOABC.H data set, the following commands are used:

```
$ cd h
$ ls -n
total 6
-rw-rw-rw-   1 0        0           3531 Aug 13 16:00 bank
-rw-rw-rw-   1 0        0            650 Aug 13 16:10 bankdefs
-rw-rw-rw-   1 0        0            112 Aug 23 1992  rg
-rw-rw-rw-   1 0        0          10765 Aug 13 11:54 rpc
-rw-rw-rw-   1 0        0           7310 Aug 13 11:53 socket
$ vi rg
```

After an NFS edit command, the member rg of the TCPIP.ITSOABC.H partitioned data set is locked. An attempt to use the PDF editor to edit the member will give a `MEMBER IN USE` message. The member is locked until the writetimeout period expires.

If you want to override the processing attributes specified during the mount, you have to enclose the file name in quotation marks and append the overriding attributes to the file name.

```
$ cd /u/mvsmnt
$ copy /u/abc/ndbclient/ndbreq "mvsndb,binary"                        1
$ copy "mvsndb,binary,executebiton" /u/abc/ndbclient/ndbreqback  2
$ cd /u/abc/ndbclient
$ ndbreqback     3
usage: ndbreqback hostname "begin"
     or
usage: ndbreqback hostname "select..."
```

1 The file system was mounted with the text processing attribute. To copy binary files we have to override that attribute during the copy process. This command copies the ndbreq file from the /u/abc/ndbclient directory to a sequential data set on MVS, with the name TCPIP.ITSOABC.MVSNDB. The file is copied in binary mode.

2 This command copies the file back from MVS in binary mode. When the file is written into the local AIX file system, the execute bit should be turned on.

3 A test to see if the file came back in executable form.

You could also execute the mvsndb file directly from MVS using the following sequence of commands on AIX:

```
$ cd /u/mvsmnt
$ "mvsndb,binary,executebiton"
usage: mvsndb,binary,executebiton hostname "begin"
     or
usage: mvsndb,binary,executebiton hostname "select..."
$
```

To use MVS as a code server, you would normally mount the file system with the binary and the executebiton attributes. If you had done so, you could have executed the command just by typing in the name of the file mvsndb, without having to append the two overriding processing attributes:

```
$ cd /u/mvsmnt
$ ls -n mvsndb
-rwxrwxrwx   1 0       1         22483 Aug 28 1992  mvsndb  1
$ mvsndb
usage: mvsndb hostname "begin"
     or
usage: mvsndb hostname "select..."
$
```

1 The file system is mounted binary and with the executebit on.

If you are in doubt about your active NFS server attributes, you can use the **showattr** command, which was one of the commands you downloaded from MVS together with the **mvslogin** and the **mvslogout** commands.

```
$ showattr mvs18

MVS/DFP Network File System Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(100,10)           blks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4096)    keys(64,0)              nonspanned
shareoptions(1,3)       model()
mgmtclas()              dsntype(pds)            norlse

MVS/DFP Network File System Server Processing Attributes:

binary                  lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30)

MVS/DFP Network File System Server Site Attributes (not modifiable):

mintimeout(0)           nomaxtimeout    logout(1800)
nfstasks(8)
bufhigh(2097152)        readaheadmax(16384)     cachewindow(16)
percentsteal(20)        maxrdforszleft(32)      logicalcache(1048576)
```

If you have not used the MVS NFS server for a period of time, and you suddenly receive messages such as `permission denied` or `cannot read`, it is presumably because the *logout* timeout value has been reached. When this happens, you have to reissue your **mvslogin** command to regain access to the MVS NFS server.

7.2.7 Using the DFP Network File System server from OS/2

Mount a file system using the following OS/2 command:

```
[C:\]mount x: mvs18:tcpip.itsoabc,text,crlf  1
mount: mvs18:tcpip.itsoabc,text,crlf

NFS Drive 'x:' was attached successfully.

[C:\]
```

1 The file system is mounted as text with CRLF end-of-line processing applied.

You have to log on to MVS using the downloaded **mvslogin** command:

```
[C:\]mvslogin mvs18 christe
Password required
GFSA973A Enter MVS password:
GFSA955I christe logged in ok.

[C:\]
```

You can now use normal OS/2 commands to look at the X drive:

```
[C:\]x:
[X:\]dir

 The volume label in drive X is NFS.
 The Volume Serial Number is ED36:0002
 Directory of X:\

 8-26-92   9:52p     <DIR>          0  .
 8-26-92   9:52p     <DIR>          0  ..
 8-22-92   3:59a     <DIR>          0  asm
 8-26-92   9:52p     <DIR>          0  c
 8-26-92   9:52p     <DIR>          0  cntl
 8-22-92   3:59a     <DIR>          0  cobol
 8-26-92   9:52p     <DIR>          0  h
 8-22-92   3:59a     <DIR>          0  load
 8-26-92   9:52p   2049840          0  mega.testnfs
 8-22-92   3:59a     <DIR>          0  obj
        14 file(s)    2140455 bytes used
                     61440000 bytes free

[X:\]
```

You can edit files using the Enhanced Editor on OS/2. We had problems using the normal `E` editor, but the Enhanced Editor worked without problems on NFS mounted drives:

```
[X:\]cd h

[X:\]epm rg
```

If you, as in this example, have mounted the file system in text mode, you can copy files in and out of MVS in binary mode using the following syntax:

```
[X:\]copy c:\tcpip12\bin\arp.exe "arpmvs.exe,binary"   1
       1 file(s) copied.

[X:\]copy "arpmvs.exe,binary" c:\arpback.exe           2
       1 file(s) copied.

[X:\]c:\arpback -a                                     3
         ARP table contents:

interface     hardware address         IP address   minutes since
                                                      last use
    0          10005a4f58ce             9.67.38.73     0
    0          400000032210             9.67.38.93     7
    0          400050013172             9.67.38.96     0
    0          10005aa87023             9.67.38.72     4
```

1 Copy an executable file to MVS in binary mode. If you want to use the file as an executable file on the NFS mounted file system, you must preserve the exe suffix. OS/2 will not recognize a file without that suffix as an executable command.

2 Copy it back again to OS/2 in binary mode.

3 Test that it is still an executable file.

You can also execute the command **arpmvs** directly from the MVS file system:

```
[X:\]"arpmvs.exe,binary" -a    1
         ARP table contents:

interface     hardware address         IP address   minutes since
                                                      last use
    0          10005a4f58ce             9.67.38.73     3
    0          400000032210             9.67.38.93     12
    0          400050013172             9.67.38.96     0
    0          10005aa87023             9.67.38.72     4

[X:\]
```

1 You have to use the full name of the file with the exe suffix.

If you want to use the MVS NFS server as a code server for OS/2 workstations, you will have to mount the MVS file system in binary mode:

```
[C:\]mount z: mvs18:tcpip.itsoabc,binary    1
mount: mvs18:tcpip.itsoabc,binary

NFS Drive 'z:' was attached successfully.

[C:\]z:

[X:\]dir arpmvs.exe                         2

 The volume label in drive X is NFS.
 The Volume Serial Number is ED36:0002
 Directory of X:\

 8-26-92  10:16p     16665           0  arpmvs.exe
         1 file(s)      16665 bytes used
                     61440000 bytes free

[Z:\]arpmvs -a                              3
          ARP table contents:

interface     hardware address         IP address   minutes since
                                                      last use
    0          10005a4f58ce             9.67.38.73     1
    0          400000032210             9.67.38.93     0
    0          400050013172             9.67.38.96     0
    0          10005aa87023             9.67.38.72     2

[Z:\]
```

1 Mount the file system in binary mode.

2 A **dir** command to see if we have the file on the mounted file system.

3 A normal command invocation in an OS/2 environment.

When you are finished using the MVS NFS server, unmount and log out from MVS:

```
[C:\]umount *
Unmounting drive X:... successful.
Unmounting drive Z:... successful.

[C:\]mvslogut mvs18
GFSA958I uid -2 logged out ok.

[C:\]
```

7.3 Configuring NFS as a client

NFS may also be configured as a client.

7.3.1 Changes to SYS1.PARMLIB and SYS1.PROCLIB

To use the NFS client requires modifications to the SYS1.PARMLIB.

The following statement needs to be added to member BPXPRMxx to define the NFS client to OMVS:

```
FILESYSTYPE
   TYPE(NFS)
   ENTRYPOINT(GFSCINIT)
   PARM('installation parms')
   ASNAME(proc_name)
```

The PARM may be omitted if the defaults are workable. Please refer to *z/OS Network File System Customization and Operation*, SC26-7417 for a detailed description of all the installation parameters, including their defaults.

`proc_name` is the address space name and the name of the procedure in SYS1.PROCLIB that is used to start the NFS client. A sample procedure GFSCPROC may be taken from SYS1.NFSSAMP. The STEPLIB libraries need APF authorization. These will normally be the SYS1.NFSLIB and the Language Environment run-time libraries.

The procedure and all its data sets have to be available when starting OMVS.

Note: The NFS client can only be started and stopped together with OMVS.

Before starting, it should be verified that DFSMS/MVS NFS has been properly enabled in the SYS1.PARMLIB member IFAPRDxx.

7.3.2 Using the NFS client

Execute the next steps:

1. Get superuser authority:

   The user VANDEKE was logged in to TSO and the OMVS shell and had issued the "SU" command to be a superuser.

2. Do an **`mvslogin`**:

   After the **`mvslogin`** command was entered, we received the message there was a mismatch in uid/gid but we could continue.

```
VANDEKE @ RA03:/u/vandeke>/usr/lpp/NFS/mvslogin mvs28a vandeke
Password required
GFSA973A Enter MVS password:
GFSA978I VANDEKE logged in ok.
         Mismatch in uid/gid:  uid is 4027, gid is 1,
         client uid is 0, gid is 0.
VANDEKE @ RA03:/u/vandeke>su vandeke
FSUM5019 Enter the password for vandeke:
```

   Use the **`mvslogin`** command to log in to the remote MVS system. The mvslogin command can be issued multiple times and the last one overrides the previous one.

The **mvslogin** command is only required when accessing data on systems where the DFSMS/MVS NFS server site security attribute is set to saf or safexp.

Following is an example of the **mvslogin** command where mvs28a is the name of the MVS host and vandeke is the user's ID on MVS:

```
VANDEKE @ RA03:/u/vandeke>/usr/lpp/NFS/mvslogin mvs28a vandeke  2
Password required
GFSA973A Enter MVS password:    3
GFSA955I vandeke logged in ok.  4
```

In the example where the user enters mvslogin mvshost1, the current login client user ID is used as the MVS user ID.

In the example, **2** the user enters `mvslogin mvs28a vandeke`, the system then prompts for vandeke's MVS password. **3** If vandeke logs in successfully, this message appears **4**

```
GFSA955I vandeke logged in ok.
```

Otherwise, an appropriate error message appears.

Note: Messages that start with GFSA and GFSC apply to network file system requests. These messages are further explained in *z/OS Network File System User's Guide*, SC26-7419.

3. Mount the network file system

Use the **TSO MOUNT** command to make a connection between a mount point on your local UNIX System Services HFS file system and one or more files on a remote MVS, AIX, UNIX, OS/2, z/OS, or other file system. The **MOUNT** command can only be used by an MVS superuser.

The same mount function can also be performed using the UNIX System Services automount facility or /etc/rc shell scripts support. UNIX System Services does not support NFS mounts in the SYS1.PARMLIB member statement. When the automount facility is used to manage remote NFS mount points, the DFSMS/MVS NFS client user could experience ESTALE/EIO errors if the automounter unmounts the accessed mount point when the time limits specified by the automount DURATION and DELAY parameters have been exceeded. The automount default, DURATION=NOLIMIT, disables the DURATION timeout period. The DELAY for unmounting file systems should be longer than the time DFSMS/MVS NFS clients are likely to keep NFS mounts to the files and directories active.

MOUNT command syntax:

```
MOUNT FILESYSTEM(file_system_name)
          TYPE(NFS)
          MOUNTPOINT(local_mountpoint)
          MODE(RDWR|READ)
          PARM('hostname:"path_name,server_attributes", options')
          SETUID|NOSETUID
          WAIT|NOWAIT
```

Operands:

– FILESYSTEM(file_system_name) specifies the name of the file system to be added to the file system hierarchy. This operand is required. The file system name specified must be unique among previously mounted file systems. It may be an arbitrary name up to 44 characters in length of a file system. You can enclose file_system_name in single quotes, but they are not required.

– TYPE(NFS) specifies the type of file system that performs the logical mount request. The NFS parameter must be used.

– MOUNTPOINT(local_mountpoint) specifies the path name of the mount point directory, the place within the file hierarchy where the file system is to be mounted. The local_mountpoint specifies the mount point path name. The name can be a relative path name or an absolute path name. The relative path name is relative to the current working directory of the TSO session (usually the HOME directory). Therefore, you should usually specify an absolute path name. A path name is case-sensitive, so enter it exactly as it is to appear. Enclose the path name in single quotes.

Note: The mount point must be a directory. Any files in that directory are inaccessible while the file system is mounted. Only one file system can be mounted to a mount point at any time.

– MODE(RDWR|READ) specifies the type of access for which the file system is to be opened.

– RDWR specifies that the file system is to be mounted for read and write access. This is the default option.

– READ specifies that the file system is to be mounted for read-only access.

– PARM('hostname:"path_name,server_attributes", options') specifies the hostname of the remote network file system server, the server attributes, and the options. The double quotes are omitted if no server attributes are specified. Enclose the entire string in single quotes. You can specify lowercase or uppercase characters.

– SETUID|NOSETUID

Refer to *z/OS V1R2.0 UNIX System Services Command Reference*, GA22-7802 for details on SETUID and NOSETUID.

– WAIT|NOWAIT

Refer to *z/OS V1R2.0 UNIX System Services Command Reference*, GA22-7802 for details on WAIT and NOWAIT.

Now the general users may start using the NFS client.

We used a REXX exec to execute the **mount** command to mount the /u/vandeke/ in host mvs28a to /u/vandeke/mnt/ of host mvs03a:

```
/* REXX */
trace r
hfs = 'MVS3NFS1'
hfs = "FILESYSTEM('"||hfs||"')"
mp = '/u/vandeke/mnt'
mp = "MOUNTPOINT('"||mp||"')"
tp = 'TYPE(NFS)'
parms = 'mvs28a:/hfs/u/vandeke'
parms = '9.24.104.42:/hfs/u/vandeke'
parms = "PARM('"parms"')"
MOUNT hfs mp tp parms NOWAIT
exit
```

7.4 Performance

NFS is implemented on top of the SUN RPC programming interface using the remote procedure call protocols. Data is transformed between the extended data representation (XDR) and the individual TCP/IP host's local representation. As a transport protocol, NFS uses UDP. The MVS NFS server has to take care of the translation between the stream-oriented file structure assumed by NFS clients and the record-oriented file structure of MVS. It all adds up to some degree of processing overhead, which impacts the performance you may expect.

Other considerations apply as well. During mount or during NFS client startup, you are often allowed to specify a buffer size to be used during data transfer between the NFS client and the NFS server. The NFS protocol itself imposes an upper limit of 8 KB on this buffer size. UDP datagrams of this size will be used for the transfer of data. If your network's MTU size is lower than this value, the UDP datagrams will be fragmented into IP datagrams fitting into the MTU size. If your network is reliable with very few transmission errors, you will achieve good performance with a high buffer value, but if your network generates many temporary errors, a high buffer value may give you very poor performance.

Please also note that the hard/soft mount option applies not only to the mount operation itself, but to all requests between the NFS client and the NFS server. If you specify hard mount, the NFS client and server will retry failing requests indefinitely. If you specify soft mount, the NFS client and server will retry a specified number of times before the error will be reported to the user. Using *hard* mount with a high buffer size on a poor network may cause NFS operations to perform very poorly.

There are quite a few APARs which address performance issues with MVS NFS. If you get into performance problems with MVS NFS, it might be worthwhile bringing the MVS NFS server to the highest possible PTF level.

8

Trivial File Transfer Protocol (TFTP)

The trivial file transfer protocol (TFTP) that enables the transfer of files to and from a server is a standard protocol with STD number 33. It is defined in RFC 1350 with the status of elective. TFTP uses a start-and-stop protocol on well-known port 69, sending a packet, and waiting for acknowledgment from the receiver. Updates to TFTP can be found in the following RFCs: RFC 1785, 2347, 2348, and 2349.

TFTP is an extremely simple protocol to transfer files and is implemented on top of the User Datagram Protocol (UDP). The TFTP client initially sends a read/write request via port 69; then the server and the client determine the port that they will use for the rest of the connection. TFTP lacks most of the features of FTP; the only thing it can do is read or write a file from or to a server.

This chapter contains the following sections:

8.1 tftpd command syntax

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory as an z/OS UNIX executable module, namely tftpd.

The **tftpd** command syntax is:

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit]     [-s
maxsegsize] [-f file] [-a archive directory [-a ...]]  [directory ...]
```

where:

-l	Logs all the incoming read and write requests.
-p port	Uses the specified port. The default port is 69.
-t timeout	Sets the packet timeout in seconds. The TFTP server usually waits five seconds before presuming that a transmitted packet has been lost.
-r maxretries	Sets the retry limit. The default is 5.
-c concurrency_limit	Sets the concurrency limit. The TFTP server spawns both threads and processes to handle incoming requests. You can specify the limit for the number of threads that may be concurrently processing requests under a single process. When the limit is exceeded, a new process is spawned to handle requests. The default is 200 threads.
-s maxsegsize	Sets the maximum block size that can be negotiated by the TFTP block size option. The default is 8192.
-f file	You can specify files to be pre-loaded and cached for transmission. An entry has the form: `a	b <pathname>`

where: `a` indicates that the specified file is cached in ASCII form. `b` indicates that the specified file is cached in binary form.

The following are examples of cache file entries:

```
a  /usr/local/textfile
b  /usr/local/binaryfile
```

Caching is not dynamic. The cache files are read in when the TFTP server is started and are not updated, even if the file on disk is updated. To update or refresh the cache, the TFTP server must be recycled.

-a archive	Specifies an archive directory. The files in this directory's subdirectories are treated as binary files for downloading. This option is useful on EBCDIC machines that act as file servers for ASCII clients. Multiple -a options can be specified; one directory per -a option. Directories must have absolute path names.

Note: For IBM Network Station Manager, the root of the client code hierarchy (for example, /usr/lpp/nstation/standard) should be specified as an archive directory.

directory	Specifies an absolute path name for a directory. You may specify no more than 20 directories on the tftpd command line.

If the TFTP server is started without a list of directories, all mounted directories are considered active.

Activating a directory activates all of its subdirectories.

For a file to be readable by the TFTP server, the file must be in an active directory and have world ("other") read access enabled.

For a file to be writable by the TFTP server, the file must already exist in an active directory and have world ("other") write access.

The TFTP server pre-forks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist. In case of a flood of concurrent TFTP requests, the TFTP server may fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

8.2 Starting the z/OS TFTP server

You can start the TFTP server in different ways:

1. Using the MVS procedure: Enter `S TFTPD` from the MVS console.

   We used the following procedure:

   ```
   //TFTPD PROC
   //*
   //*  TFTP Server main process
   //*
   //TFTPD  EXEC PGM=TFTPD,REGION=0M,
   //  PARM='POSIX(ON),ALL31(ON)/-l -a /usr/lpp/nstation/standard'
   //*
   ```

2. Issue the **tftpd** command from the command line. Be sure to include:

   ```
   tftpd -a /usr/lpp/nstation/standard /usr/lpp/nstation/standard
   ```

8.3 z/OS TFTP server security

The TFTP server has no user authentication. Any client that can connect to port 69 at the server has access to TFTP. If TFTP is started without a directory, access to the entire HFS is allowed. To restrict access, start the TFTP with a list of directories.

If you want to use the TFTP server on z/OS, you should specify fully qualified directory names that TFTP clients are allowed to access. You may specify no more than 20 directories on the **tftpd** command line.

If a list of directories is specified, only those specified directories are active. That list is used as a search path for incoming requests that specify a relative path name for a file. Activating a directory activates all of its subdirectories. TFTP will not allow access to any other parts of the file system.

On the other hand, if the TFTP server is started without a list of directories, all mounted directories are considered active. In this case, all HFS files are open to every TFTP client because TFTP does not provide user authentication.

TFTP may sound like a very dangerous alternative to FTP, but it is extensively used to download code and initial configurations to routers and simple workstations, because of its simplicity. Due to its lack of security functions, you ought to take these simple steps if you need to run a TFTP server on your z/OS:

- Ensure that the TFTP home directory list is short and harmless.
- Use IPSec whenever a nonsecure network is involved.
- Restrict access to port 69 on packet filters in firewalls.

Part 4

Bootstrapping functions

In this part, we explore some of the bootstrapping functions shipped with CS for z/OS IP. We term these function bootstrapping because they aid in network participation.

9

Dynamic IP with DHCP/PXE, BINL and DDNS

Dynamic IP is a way to bring flexibility and ease of administration into the normally very static world of IP address and name definitions. It enables an end user to get to work without having to wait for an administrator to assign an IP address to him and it eases the work of the administrator by letting the machine do the boring job of assigning an address to a user and typing this into a computer. Dynamic IP gives mobility to the users. They can easily move to a new location and bring their IP identity (name) with them.

In addition to dynamic IP address assignment there is now available a pre-boot execution procedure that gives more flexibility to locate or distribute functions to servers in the network that do IP address assignment, directory services for boot images, and boot services to store and deliver boot files of different system architectures.

This chapter contains the following sections:

9.1 Overview of Dynamic IP

The term *dynamic IP* includes four functions:

1. Dynamic Host Configuration Protocol (DHCP)
2. Dynamic Domain Name System (DDNS)
3. Pre-Boot Execution Services (PXE)
4. Bind Image Negotiation Layer Services (BINL)

CS for z/OS IP provides all of these services. There are also other platforms such as Windows NT and Warp Server which are capable of acting as DHCP and DDNS servers.

DHCP provides a configuration of mobile and stationary workstations that are connected to the network with TCP/IP protocols. The IP configuration can deliver any or all of the following values to a client.

Network stations will use the benefits of PXE and BINL services:

1. IP address
2. IP subnet mask
3. Default router address
4. Local host name
5. Domain name
6. Name server address
7. Time zone indication
8. Pre-boot extension information (OS/390 V2R7+ only)
9. Bind Image Server address (OS/390 V2R7+ only)
10. Boot files
11. Other types of information for configuration

Note: For PXE information provided by the z/OS server, you need a PXE-enabled client.

DDNS permits a Domain Name System (DNS) server to learn dynamically about the host names and IP addresses of clients dynamically configured via DHCP. This alleviates the manual efforts involved in maintaining a name server. DDNS administrators, DDNS clients, and DHCP servers running a DDNS client can request DDNS services with the **`nsupdate`** command.

The benefit of this dynamic approach is the simplification of network administration. Instead of having to hard code IP addresses and IP names in individual workstations and in the name server, symbolic settings of these values can allow the addresses and names to be dynamically generated. This simplification is particularly relevant in today's mobile computing world in which a workstation's move from one site to another can necessitate a change in adapter DLC addresses and in IP (network layer) addresses and names. Managing this process with statically defined files becomes a near impossibility when the workstations involved number in the hundreds or in the thousands, as is now often the case.

DHCP with its PXE extensions will provide an address of the BINL server. The BINL server recognizes the specific client system architecture and selects the address for the appropriate boot server. The client workstation may retrieve the boot image from this boot server.

A prerequisite, however, for using DHCP/PXE extensions is a PXE-enabled client workstation.

9.2 Dynamic Host Configuration Protocol (DHCP)

This section explains the normal DHCP protocol without Pre-Boot Execution Environment (PXE) extensions. These extensions will be described in 9.4, “Pre-Boot eXecution Environment (PXE)” on page 303.

DHCP is a TCP/IP protocol that enables you to centrally locate and dynamically distribute configuration information.

DHCP allows clients to obtain IP network configuration information, including an IP address, from a central DHCP server. The addresses provided to clients by the DHCP server are allocated permanently or are leased for a period of time. When a client receives a leased address, it must periodically request that the server revalidate the address and renew the lease.

Note: BOOTP was the predecessor to DHCP. BOOTP did not support address leasing.

The processes of address allocation, leasing, and lease renewal are all handled dynamically by the DHCP client and server programs.

DHCP defines three IP address allocation policies:

Dynamic	A DHCP server assigns a temporary, leased IP address to a DHCP client.
Static	A DHCP server administrator assigns a static, predefined address reserved for a specific DHCP client.
Permanent	A DHCP server administrator assigns a permanent IP address to a DHCP client. No lease renewal is required.

Note: If your network uses routers or gateways, you need to ensure that they can be enabled as BOOTP relay agents. Enabling the routers or gateways as relay agents allows the DHCP packets to be sent across the network to other LAN segments.

9.2.1 How does DHCP work?

DHCP allows clients to obtain IP network configuration information, including an IP address, from a central DHCP server. DHCP servers control whether the addresses they provide to clients are allocated permanently or are "leased" for a specific time period. When a client receives a leased address, it must periodically request that the server revalidate the address and renew the lease.

The DHCP client and server programs handle the processes of address allocation, leasing, and lease renewal.

Acquiring configuration information

DHCP allows DHCP clients to obtain an IP address and other configuration information through a request process to a DHCP server. DHCP clients use RFC-architected messages to accept and use the options served them by the DHCP server. For example:

1. The client broadcasts a message (containing its client ID) announcing its presence and requesting an IP address (DHCPDISCOVER message) and desired options such as subnet mask, domain name server, domain name, and static route.
2. Optionally, if routers on the network are configured to forward DHCP and BOOTP messages (using BOOTP Relay), the broadcast message is forwarded to DHCP servers on the attached networks.
3. Each DHCP server that receives the client's DHCPDISCOVER message sends a DHCPOFFER message to the client offering an IP address.

   The server checks the configuration file to see if it should assign a static or dynamic address to this client.

   In the case of a dynamic address, the server selects an address from the address pool, choosing the least recently used address. An address pool is a range of IP addresses to be leased to clients. In the case of a static address, the server uses a Client statement from the DHCP server configuration file to assign options to the client. Upon making the offer, the IBM DHCP server reserves the offered address.
4. The client receives the offer message(s) and selects the server it wants to use.
5. The client broadcasts a message indicating which server it selected and requesting use of the IP address offered by that server (DHCPREQUEST message).
6. If a server receives a DHCPREQUEST message indicating that the client has accepted the server's offer, the server marks the address as leased. If the server receives a DHCPREQUEST message indicating that the client has accepted an offer from a different server, the server returns the address to the available pool. If no message is received within a specified time, the server returns the address to the available pool. The selected server sends an acknowledgment which contains additional configuration information to the client (DHCPACK message).
7. The client determines whether the configuration information is valid. Upon receipt of a DHCPACK message, the IBM DHCP client sends an Address Resolution Protocol (ARP) request to the supplied IP address to see if it is already in use. If it receives a response to the ARP request, the client declines (DHCPDECLINE message) the offer and initiates the process again. Otherwise, the client accepts the configuration information.
8. Accepting a valid lease, the client enters a BINDING state with the DHCP server, and proceeds to use the IP address and options.

To DHCP clients that request options, the DHCP server typically provides options that include subnet mask, domain name server, domain name, static route, class-identifier (which indicates a particular vendor), user class, and the name and path of the load image.

However, a DHCP client can request its own, unique set of options. The default set of client-requested DHCP options provided by IBM includes subnet mask, domain name server, domain name, and static route.

Renewing leases

The DHCP client keeps track of how much time is remaining on the lease. At a specified time prior to the expiration of the lease, usually when half of the lease time has passed, the client sends a renewal request, containing its current IP address and configuration information, to the leasing server. If the server responds with a lease offer, the DHCP client's lease is renewed.

If the DHCP server explicitly refuses the request, the DHCP client may continue to use the IP address until the lease time expires and then initiate the address request process, including broadcasting the address request. If the server is unreachable, the client may continue to use the assigned address until the lease expires.

Moving a client out of its subnet?

One benefit of DHCP is the freedom it provides a client host to move from one subnet to another without having to know ahead of time what IP configuration information it needs on the new subnet. As long as the subnets to which a host relocates have access to a DHCP server, a DHCP client will automatically configure itself correctly to access those subnets.

For a DHCP client to reconfigure itself to access a new subnet, the client host must be rebooted. When a host restarts on a new subnet, the DHCP client may try to renew its old lease with the DHCP server which originally allocated the address. The server refuses to renew the request since the address is not valid on the new subnet. Receiving no server response or instructions from the DHCP server, the client initiates the IP address request process to obtain a new IP address and access the network.

Implementing changes in the network

With DHCP, you can make changes at the server, re-initialize the server, and distribute the changes to all the appropriate clients. A DHCP client retains DHCP option values assigned by the DHCP server for the duration of the lease. If you implement configuration changes at the server while a client is already up and running, those changes are not processed by the DHCP client until the client attempts to renew its lease or until it is restarted.

9.2.2 Implementing DHCP

The IBM DHCP server in CS for z/OS IP provides configuration information to clients based on statements in the server's configuration file and based on information provided by the client. The server's configuration file defines the policy for allocating IP addresses and other configuration parameters.

Before you start the DHCP server, create or modify the DHCP server configuration file.

Once the DHCP server is running, you can also make dynamic changes to the configuration by modifying the configuration file and using the DHCP Server Maintenance program, **`dadmin`** to re-initialize the DHCP server. Status information for retrieving old lease information is found in /etc/dhcps.ar and /etc/dhcps.cr.

DHCP configuration file

You configure the DHCP server by manually editing the DHCP server configuration file.

Note: Configuring the server incorrectly causes few, if any, warning messages. The DHCP server normally runs even when it encounters errors in the configuration file and typically ignores incorrect data and may optionally post a message to its log.

The DHCP server will locate the configuration file by default in /etc/dhcpsd.cfg. A sample server configuration file called DHCPSD.CFG is located in the /usr/lpp/tcpip/samples/ directory and in Appendix C, “Sample DHCP configuration file” on page 503.

You can create a hierarchy of configuration parameters by nesting statements within the DHCP server configuration file. This allows you to specify the scope of some configuration values that are served to all clients, while other configuration values are served only to certain clients. The statement used and its position in the file determines what information is supplied to the clients. The options, also known as BOOTP vendor extensions, are defined in *RFC2131* and *RFC2132*. You will find a detailed description in the book *S/390 Network Station Manager*, SC31-8546 or on the Web page:

```
http://www.ietf.org/rfc.html.
```

Figure 9-1 on page 286 shows the dhcpsd.cfg configuration file we used.

```
#   SMP/E Distribution name:  EZATDDSD
#
#   See /usr/lpp/tcpip/samples/dhcpsd file for description
#   of parameters
numLogFiles     4  9
logFileSize     400   OS/390 Firewall Technology Guide and Reference
logFileName     /tmp/dhcpsd.log  8
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE  6
leaseTimeDefault        60 minute
leaseExpireInterval     600 seconds
supportBOOTP            no
supportUnlistedClients  yes

subnet 9.24.104.0 255.255.255.0  9.24.104.185-9.24.104.220  2
{
  Option 1 255.255.255.0                   # mask
  Option 6 9.24.104.108                    # DNS
  Option 15 itso.ral.ibm.com               # domain name
  Option 3 9.24.104.1                      # router
client 6 0000e568afdf ANY                  # MAC Address   1
 {                                         # dynamic IP
  Option 1 255.255.255.0
  Option 6 9.24.104.108
 }
client 6 0000e111afff 9.24.104.221         # MAC Address + 3
 {                                         #  IP Address
  Option 1 255.255.255.0
  Option 51 43000                          # Lease time 12 hours
 }
client 6 0000e345aefe 9.24.104.225         # MAC Address + 4
 {  5                                    #  IP Address
  Option 1 255.255.255.0
  Option 51 0xffffffff                     # Infinite lease
 }  5
}
Class "IBM Network Station"
 {
  Option 6 9.24.104.108
  Option 15 itso.ral.ibm.com
  Option 3 9.24.104.1
 }
# end of dhcpsd.cfg
```

Figure 9-1 DHCP configuration file (dhcpsd.cfg)

1 We defined the MAC-address (0000e568afdf) and ANY to get a dynamic policy for the IBM Network Station.

2 address range (9.24.104.185-9.24.104.220) where the DHCP server decides which IP address is given to the name server (see Appendix C, “Sample DHCP configuration file” on page 503 for information of the Options).

3 is a static allocation, specific MAC address gets a specified IP address with a lease time of 12 hours.

4 is permanent. IP and MAC address defined, no lease renewal required.

Note how the scope of a keyword is limited by a pair of curly brackets ({, }) within which the keyword is located. If a keyword is located outside of any pair of curly brackets, its scope is applicable to all the clients served by this server. In the example depicted in **5**, only Client 6 is affected by the subsequent parameters within the curly brackets ({, }).

6 shows how TRACE has been enabled. This is particularly useful when you are first enabling DHCP or when you have new client types that need to be added. The TRACE option can be turned on and off by using the `dadmin -t` option. (For more about *dadmin*, see “DADMIN utility” on page 291)

7 shows how we have increased the size of the DHCP log data set from the default of 100 KB to 400 KB. Our experience during debugging showed that the logs wrapped too quickly when the file was small, forcing us to review all four generations of the log. With 400 KB, we often captured what we needed in one generation of the log.

8 specifies the location of the DHCP log. We placed it in the /tmp/ directory. Your logging will build four generations of logs **9** before the logs start wrapping. Those logs are named:

```
_ File  dhcpsd.log       (most recent)
_ File  dhcpsd.log.1     (second most recent)
_ File  dhcpsd.log.2     (third most recent)
_ File  dhcpsd.log.3     (fourth most recent)
```

To enable logging by the server, all of the following must be specified in the DHCP configuration file:

- Number of DHCP log files
- Size of DHCP log files
- Name of DHCP log files
- At least one information type to log

This set of parameters specifies the log files maintained by the server. Each parameter is identified by a keyword and followed by its value.

numLogfiles 0 to n
If 0 is specified there will be no log file and no message is displayed. The most recent are kept.

logFileSize in kbytes
If the size is reached it is renamed and a new file is created.

logFilename path and filename
The name of the most recent log file. Renamed files are 1 to n.

logItem Item
An item defined will be logged.

SYSERR	System error, at the interface to the platform.
OBJERR	Object error, in between objects in the process.
PROTERR	Protocol error, between client and server.

WARNING	Warning, worth of attention from the user.
EVENT	Event occurred to the process.
ACTION	Action taken by the process.
INFO	Information that might be useful.
ACNTING	Accounting information on clients served.
TRACE	Code flow, for debugging.

Appendix D, “The DHCP log data set” on page 519 shows the entries in the dhcpsd.log data set after starting the IBM Network Station with the configuration file we used in Figure 9-1.

9.2.3 Configuring DHCP for dynamic IP (DDNS client)

Figure 9-2 on page 289 shows the configuration file used to add Dynamic IP capabilities to the DHCP server. The information between 1 and 2 is used by the DHCP server to run the **nsupdate** command (the DDNS client program) to add, change or delete A records or PTR records from DDNS server files. Records are released (deleted) when a DHCP lease expires or when an administrator issues the nsupdate command to delete them.

A request for an A record update can be submitted only if the DHCP configuration file contains the keyword `proxyARec` 3. This keyword enables the DHCP server to execute **updateDNSA**, which tells the DDNS client to submit an A record update to the DDNS server. The new A record will represent the DHCP client. (If we had omitted `proxyArec`, the DDNS client would have been permitted to submit only the request for the PTR record update: **updateDNSP**.).

Note: Certain DHCP clients, such as the Network Station, do not have DDNS client capability, and therefore require that the DHCP server submits requests for A record updates and deletions. Other DHCP clients, such as OS/2 Warp, include DDNS client function. For such DDNS clients you would probably want to be consistent about whether the DDNS client at the workstation or the client at z/OS executed requests for A record updates; otherwise, the first client requesting the function implants his KEY record in the zone file and locks out processing by any other client.

The placement of the proxyARec statement is important. You can put it at the general defaults, at specific vendor options, at a specific subnet as shown in Figure 9-2 on page 289 or at the specification for a specific class to control the update of the A records.

```
#   SMP/E Distribution name:  EZATDDSD
#
numLogFiles     4
logFileSize     400
logFileName     /tmp/dhcpsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE
#
leaseTimeDefault        7 minute
leaseExpireInterval     20 seconds
# supportBOOTP            yes
supportBOOTP            no
supportUnlistedClients  yes
vendor ibm
{
        option 42 hex"ab dc"
}
vendor sun hex"ef 12 34 56 78"
subnet 192.168.100.0 255.255.255.0  192.168.100.101-192.168.100.120
{
option 1    255.255.255.0
option 3    192.168.100.100
option 4    192.168.100.100
option 5    192.168.100.100
option 6    192.168.100.100
option 15   small.isp.com                # domain name
option 51   10800
}
#   1
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;3110400;q" -q"
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q" -q"
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q" -q"
#   2
proxyARec  3
#
class fruit
{
        option 48 6.5.4.3
        option 48 8.8.8.8
}
class veggie
{
        option 49 1.2.3.4
        option 48 6.6.6.6
}
# end of dhcpsd.cfg
```

Figure 9-2   DHCP configuration file

The DDNS client statements look complicated, but you can just use the samples and they will work fine. The following two lines are always required for pointer record updates:

```
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q" -q"    1
```

The following two lines are only required for A record updates if you specify proxyARec:

```
                                                  2
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;3110400;q" -q"
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q" -q"
```

1 defines the amount of output you get from the **nsupdate** command:

- -q specifies quit mode for very little output.
- -v specifies verbose mode. Use this for debugging only.

2 specifies how long the A record will be kept at the name server after the lease has expired. The value of 3110400 seconds will give a 36-day buffer for inactivity times like vacation.

9.2.4 Generating keys for DDNS updates by the DHCP server

To secure the updates to the DDNS server we have to create keys to be used by the DDNS client. Keys for PTR record updates are always needed; keys for A record updates are needed only if we specified proxyARec.

The **nsupdate** command that will generate the DHCP PTR record key for use when DHCP requests a DDNS update for the dynamic zone 192.168.100 at name server mvs03.itso.ral.ibm.com is:

```
nsupdate -a -f -g -h *.100.168.192.in-addr.arpa -p mvs03.itso.ral.ibm.com
```

The **nsupdate** command that will generate the DHCP A record key for use when DHCP requests a DDNS update for the dynamic zone small.isp.com at name server mvs03.itso.ral.ibm.com is:

```
nsupdate -a -f -g -h *.small.isp.com -p mvs03.itso.ral.ibm.com
```

Each successful key generation results in an additional key in /tmp/ddns.dat and the following shell messages:

```
********************************* Top of Data ***************************
--- NSUPDATE Utility --- ---
Key Gen  ...        ... succeeded  ...
******************************** Bottom of Data *************************
```

After you have generated the keys you should save the ddns.dat file. We backed it up in the /etc/ directory but you may wish to do so in a /backup/ directory that you have created for your installation.

9.2.5 Start the DHCP server

If you have installed Network Station Manager, DHCPSD is installed in the /usr/lpp/tcpip/nsm/sbin directory. When you are using Communications Server for z/OS IP, you will find DHCPSD in the /usr/lpp/tcpip/sbin directory.

DHCP is considered a *generic server/daemon*, meaning that it can gain transparent access to all TCP/IP stacks configured under CINET. Normally you would run only one copy of DHCP in your OS/390 image and it would be associated with only one TCP/IP stack. If, however, you decide to use multiple DHCP servers, each with affinity to a different stack, it is extremely important that you review the multi-stack discussion of the environment variable _BPXK_SETIBMOPT_TRANSPORT in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227.

To start the DHCP server, use the following form of the **dhcpsd** command:

```
dhcpsd [-q or -v] [-f configFile]
```

where:

- `-q` , which starts the server in quiet mode, which means that no banner is displayed when the server starts.
- `-v`, which starts the server in verbose mode. This causes messages dealing with client communication to print to the screen.

  We recommend that you not use verbose mode because it can tie up your OpenEdition shell screen. If you must use verbose mode to obtain additional diagnostic messages, then remember to specify it with the & character as follows: `dhcpsd -v &`. Then, once you have obtained the messages, stop dhcpsd before doing any editing.
- -f configFile, which is the name of the DHCP server configuration file. The default filename is /etc/dhcpsd.cfg.

You also can use an MVS procedure. When starting the DHCP server with a procedure the sample start proc is found in the DHCP member of the SEZAINST library.

If you had run dhcpsd in verbose mode with the following command:

```
/usr/lpp/tcpip/sbin/dhcpsd -v >/u/gdente/dhcpsd.trc &
```

The you would see output like the following:

```
: INFO:   EZZ7277 DHCP Server Initialized at Tue Mar 10 11:54:12 1998
Request from:    6-0x08005a0d2856
.Type:. DISCOVER
...Status:. Offering reserved address to the client - REPLY OFFER.
....IP Addr:    192.168.100.101
....Options:. 1 3 4 5 6 15 51
Request from:    6-0x08005a0d2856
.Type:. REQUEST
...Status:. Requesting a reserved address - REPLY ACK.
....IP Addr:    192.168.100.101
....Options:. 1 3 4 5 6 15 51
Request from:    6-0x08005a0d2856
.Type:. REQUEST
...Status:. Requesting an existing lease - REPLY ACK.
....IP Addr:    192.168.100.101
....Options:. 1 3 4 5 6 15 51
```

9.2.6 DADMIN utility

If you implement DHCPSD, you will want to make use of the DADMIN utility for debugging DHCP or understanding how your DHCP is operating. Figure 9-3 on page 292 shows how you obtain help for dadmin from the shell environment.

```
dadmin -?
dadmin Version 3.20 - IBM DHCP Server

This utility is used to perform administrative functions on the specified
DHCP Server.  If no server is specified, the local DHCP Server is used.

USAGE:  dadmin [-?] | [-v] [[-h]<host>] [-f] -d <ipaddress>

| [-v] [[-h]<host>] -i

| [-v] [[-h]<host>] -s

| [-v] [[-h]<host>] -t <on/off>

| [-v] [[-h]<host>] -n <intervals>

| [-v] [[-h]<host>] -p <ipaddress>

| [-v] [[-h]<host>] -c <client id>
WHERE:

-?             Display help message.
-v             Execute in verbose mode.
-f             Don't prompt when deleting a lease. Force it to yes.
-h <host>      DHCP Server being used (local server if not specified).
-d <ipaddress> DELETE the lease for the specified IP address.
-i             ReINITIALIZE the specified server.
-s             Display address pool STATUS of the specified server.
-t <on/off>     Turn server trace on or off.
-n <intervals>  Display server statistics, summary and any requested inte
                intervals
-q <ipaddress>  Display information for an IP address
-p <ipaddress>  Display information for a Pool of IP addresses.
-c <client id>  Display information for a client ID.
```

Figure 9-3 DADMIN utility: results of dadmin -?

There are many helpful options for dadmin; we show you first the results when you reinitialize your DHCPSD definitions by issuing: **dadmin -i**:

```
 PLEASE WAIT....Gathering Information From the Server....PLEASE WAIT
 Server successfully reinitialized.
```

Next you see the results of your allocated and leased addresses when you issue the command, **dadmin -s**:

```
******************************** Top of Data **************************

PLEASE WAIT....Gathering Information From the Server....PLEASE WAIT


IP Address      Status  Lease Time Start Time  Last Leased Proxy ClientID

192.168.100.101 Leased     3:00:00 03/09 16:53 03/10 07:55 FALSE 6-08005a0d2856
192.168.100.102 Free
192.168.100.103 Free
192.168.100.104 Free
192.168.100.105 Free
192.168.100.106 Free
192.168.100.107 Free
192.168.100.108 Free
192.168.100.109 Free
192.168.100.110 Free
192.168.100.111 Free
192.168.100.112 Free
192.168.100.113 Free
192.168.100.114 Free
192.168.100.115 Free
192.168.100.116 Free
192.168.100.117 Free
192.168.100.118 Free
192.168.100.119 Free
192.168.100.120 Free
******************************** Bottom of Data ************************
```

For testing purposes do not overlook the fact that you can terminate a lease from your side using **dadmin -d <ipaddress>**.

You might prefer for debugging reasons to pipe the results of the dadmin into a file:

```
 /usr/lpp/tcpip/sbin/dadmin -s > /u/gdente/dadmin.leases
```

9.3 Dynamic Domain Name System (DDNS)

DDNS is simple to implement once you already understand the concepts of a name server in general. Its purpose is to allow dynamic updates to the DNS server files: changes to PTR records and changes to A records. The dynamic updates occur when a DDNS administrator or a DDNS client issues an **nsupdate** command to notify the name server that new records are required.

You could have a secondary name server retrieve a DDNS zone from a primary in a zone transfer operation. This would allow the secondary to respond to queries for information about dynamically defined entries. However, a secondary name server cannot respond to an update of the DDNS zone obtained in this fashion. In this sense, a secondary name server is not a backup for the DDNS function itself, but it is a backup for the basic domain name system functionality.

If the secondary name server needed to be a DDNS as well, it would need to generate its own keys and files and become a primary name server for the dynamic domain while remaining a secondary for other domains.

There are two ways to implement a Dynamic Domain Name System:

- Dynamic Secured Updates

  Keyword `dynamic secured` or `dynamic` appears in the boot file.

  Clients are allowed to update their records dynamically using their own encryption keys.

- Dynamic Presecured Updates

  Keyword `dynamic presecured` appears in the boot file.

  Clients are allowed to update their records dynamically only if they have been given encryption keys generated by the DDNS administrator.

  The KEY resource record must have been predefined by the DDNS administrator in the domain file for each client.

Here is a summary of the process you follow to create a DDNS environment:

1. Execute nsupdate command with appropriate keywords and values to generate keys necessary for records. The keywords for the nsupdate command are:

   ```
   optional parameters are:
           -kkeyfile -hhostname -ddomainname -pprimaryname
           -rIPaddress(for in-addr.arpa hostname) -s"command string"
   switches:
           -a administrator mode
           -g key generation mode
           -q quiet (no prompts)
           -v verbose output
           -? display this help
   ```

   Update DHACP configuration file if you plan to have DHCP communicate with the DDNS client to request DDNS server updates.

2. Back up the output of nsupdate in /tmp/ddns.dat and copy to a directory that is safe from deletions (for example, /etc/ddns.dat.bak or a backup directory that can be reached by administrative personnel).
3. Create boot file for DDNS.
4. Cut and paste forward public zone key in forward dynamic file which you have created for DDNS and back it up in case you ever have to restore the original version.
5. Cut and paste reverse public zone key in forward dynamic file which you have created for DDNS and back it up in case you ever have to restore the original version.

Note: For dynamic presecured mode, additional keys must be generated for each user. We do not show an example of dynamic presecured in this book.

As Figure 9-4 on page 295 shows, our dynamic network is 192.168.100.0 and our dynamic domain is small.isp.com. MVS03 reaches this network through a token-ring OSA port with IP address of 192.168.100.100. For the DDNS updates, we will focus on the interaction between workstation #1 and MVS03. The workstation, an OS/2 Warp V4.0 platform, will have been assigned IP address 192.168.100.101 by its DHCP server, MVS03. Its DDNS client function will then request an A record update of the DDNS server at MVS03 and will provide the name myhost to the DDNS server for that update. The DDNS server at MVS03 also goes by the name `ns-updates.small.isp.com`.

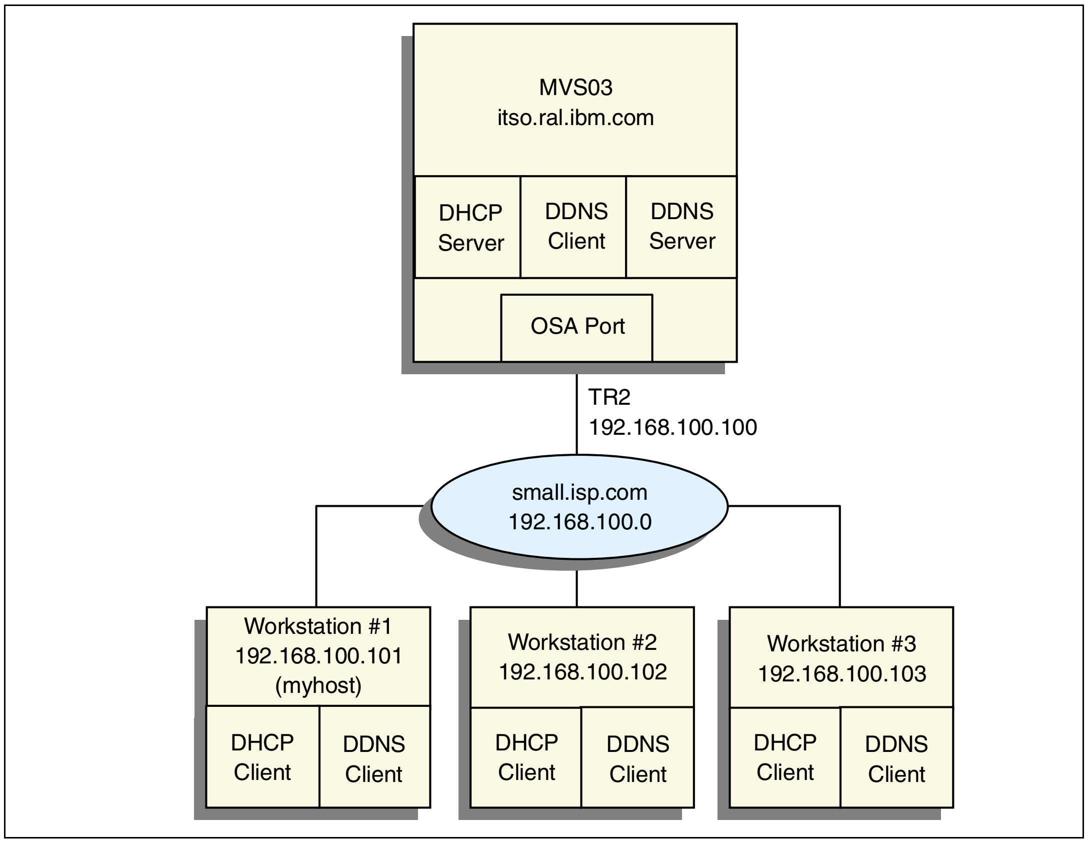


Figure 9-4 Network diagram for DDNS and DHCP

In the documentation that follows, watch for the manner in which these pieces of information have been coded and dynamically updated:

- Addresses 192.168.100.100 and 192.168.100.101
- Domain name `small.isp.com`
- Host name `myhost.small.isp.com`
- DDNS server name `ns-updates.small.isp.com`

In an environment as shown above you should not code `proxyARec` in your DHCP server configuration because all workstations have a DDNS client installed and will update their A records.

9.3.1 Generating zone keys

We have already generated keys to be used by DHCP server initiated DDNS updates in 9.2.4, "Generating keys for DDNS updates by the DHCP server" on page 290. If the administrator of the DDNS server is to perform this function, then you need to generate *zone keys*: one for the dynamic forward file and one for the dynamic reverse file. In fact, the zone keys must always be present if you expect to initiate a server that has been defined as dynamic. Figure 9-5 shows partially what these four keys can look like when they are stored in /tmp/ddns.dat.

```
*.small.isp.com mvs03.itso.ral.ibm.com Hd1CgHs5eMgGtXu05ORn98nOi2dnOz8FE  1
*.100.168.192.in-addr.arpa mvs03.itso.ral.ibm.com 4jDkIDnrU4Ii9WyqvGcx9y  2
100.168.192.in-addr.arpa mvs03.itso.ral.ibm.com Y3BU88qXZQOEOryPqC1n9+vx  3
small.isp.com mvs03.itso.ral.ibm.com mJw7G8UfJAuUaKGk3A1D97VM5757d6c5sY6  4
```

Figure 9-5 DDNS.DAT with four keys

The keys are much longer than they appear above and include both a private key portion and a public key. The private key is required by any zone administrator who wishes to update the zone with **nsupdate**. (He can take the key with him and use it on a different platform from which he executes the **nsupdate** command.) The public key, the second key in the records, is copied into the domain file and the reverse file. In sequence the keys you see above are:

1 The DHCP A record key for use when DHCP requests a DDNS update for the dynamic zone `small.isp.com`.

2 The DHCP PTR record key for use when DHCP requests a DDNS update for the dynamic zone `192.168.100`.

3 The DDNS zone key for use in the reverse (in-addr.arpa) file when DDNS requests an update for the dynamic zone `192.168.100`. The syntax of the **nsupdate** command that will generate this key is:

```
nsupdate -a -g -p mvs03.itso.ral.ibm.com -h 100.168.192.in-addr.arpa
```

4 The DDNS zone key for use in the domain (forward) file when DDNS requests an update for the dynamic zone `small.isp.com`. The syntax of the **nsupdate** command that will generate this key is:

```
nsupdate -a -g -p mvs03.itso.ral.ibm.com -h small -d isp.com
```

The syntax of -h implies that you are giving a host name; in fact you are simply naming the first part of a domain name. The above command to generate the forward zone key would have executed equally as successfully had you keyed it as follows:

```
nsupdate -a -g -p mvs03.itso.ral.ibm.com -h small.isp.com
```

Each successful key generation results in an additional key in /tmp/ddns.dat and the following shell messages:

```
********************************* Top of Data ***************************
--- NSUPDATE Utility --- ---
Key Gen  ...        ... succeeded  ...
******************************** Bottom of Data *************************
```

You will want to perform the key generation task prior to building the new boot file and zone records. The results of our **nsupdate** commands are visible in the DDNS.DAT file in Figure 9-8 on page 299.

Note: Remember to back up the /tmp/ddns.dat file in another data set outside of the /tmp/ directory so that the periodic installation cleanup of the /tmp/ directory does not leave you without a valid copy of the DDNS.DAT file. We backed it up in the /etc/ directory but you may wish to do so in a /backup/ directory that you have created for your installation.

9.3.2 MVS03 DDNS boot file

Please examine the DDNS boot file in Figure 9-6 on page 297. You will notice that the majority of the domain (forward) files, reverse files, hints file, and Loopback file have already been made available in A.1, "BIND 4.9.3-based DNS implementation" on page 468. Therefore, we print here only the files that are different.

```
;    /etc/named.boot.dyn (modeled after named.boot)
;     requires new boot file, new for.dyn file, and new rev.dyn file
;     optional=slave
;  TYPE        DOMAIN                    FILE OR HOST
directory      /etc/dnsdata
;
primary    itso.ral.ibm.com       1  named.for    3
primary    small.isp.com          2  named.for.dyn  dynamic secured 4
primary    100.168.192.in-addr.arpa  named.rev192.168.100.dyn dynamic secured 5
primary    251.168.192.in-addr.arpa  named.rev192.168.251
primary    252.168.192.in-addr.arpa  named.rev192.168.252
primary    236.168.192.in-addr.arpa  named.rev192.168.236
primary    235.168.192.in-addr.arpa  named.rev192.168.235
primary    105.24.9.in-addr.arpa     named.rev9.24.105
primary    104.24.9.in-addr.arpa     named.rev9.24.104
primary    221.168.192.in-addr.arpa  named.rev192.168.221
primary    109.168.192.in-addr.arpa  named.rev192.168.109
primary    0.0.127.in-addr.arpa      named.lbk
cache       .                        named.ca
forwarders   9.24.104.108
slave
options query-log
```

Figure 9-6 Boot file for DDNS

Significant points in the boot file for a DDNS are:

- You may have more than one domain file (forward file), as you see in 1 and 2 of Figure 9-6. One forward file is used when MVS03 manages the `itso.ral.ibm.com domain`. One is used when MVS03 provides services for the `small.isp.com` domain.
- The keyword `dynamic secured` appears on both the domain file for the dynamically managed domain (4) and the reverse file for that domain (5).

Note: You will recall that another keyword, `cluster`, appears on only the domain (forward) file when you implemented DNS/WLM.

9.3.3 MVS03 DDNS domain file

Figure 9-7 shows the DDNS domain file.

```
;  /etc/dnsdata/named.for.dyn for MVS03 as DHCP and DDNS
;
$ORIGIN small.isp.com.
@ 1  IN KEY 80 0 1 AQPYZeXmV/uIJXttTwIlvLcvtDfH5RNE+7GeYix01+JRWqsFluxiSUfJzrRS
@ 2  IN SOA        mvs03.itso.ral.ibm.com. gdente@itso.ral.ibm.com. (
                      6 10800   3600   604800   86400 )
     IN     NS    mvs03.itso.ral.ibm.com.
;
; OWNER    CLASS   TYPE    RECORD DATA
localhost    IN      A       127.0.0.1
murli750     IN      A       192.168.100.254   ;  DNS on small.isp.com.
ns-updates.small.isp.com. 3  IN  A       192.168.100.100   ;
```

Figure 9-7 Domain file for DDNS

1 shows a new record type in the domain file: a KEY record. This key is extracted *by you* from the information in the /tmp/ddns.dat file. The key contents, beginning with the characters *AQ* do not resemble the part of the key that is visible to you in the ddns.dat file above (Figure 9-5). This is because the key in the KEY record is the *public key*, and it is found at an offset of several hundred bytes into the full key that is in the /tmp/ddns.dat file. There are currently no tools on z/OS to create these keys and find the public portion, so it is a labor-intensive process, although easy to implement.

2 identifies mvs03.itso.ral.ibm.com in the SOA record for the domain `small.isp.com`; we could just as well have declared mvs03 to be a member of the small.isp.com domain, but chose not to for the sake of simplicity.

3 shows how we have added an A record for the name the DDNS client uses to address the DDNS server. If you look in Appendix A, "BIND DNS sample configuration" on page 467, you will see that we had also added an address for mvs03.itso.ral.ibm.com in named.for; this address is the address of the OSA port that we used to isolate our DHCP/DDNS testing.

Instead of an A record for ns-updates, we could have added a CNAME record:

```
ns-updates    IN  CNAME   mvs03.itso.ral.ibm.com.
```

This is merely a matter of preference; some installations find it less confusing to have multiple A records for the same address; others find it less confusing to use aliases associated with the actual A records.

Extracting the Public Key from ddns.dat

Browse the /tmp/ddns.dat file. Find the key you need for a particular file you are building (forward or reverse). Now perform a search looking for a blank character in the relevant key record. The blank will be found several hundred characters from the start of the record (we are concerned here with only the zone keys, which are depicted below in **1** and **2** of Figure 9-8).

```
    *.small.isp.com mvs03.  &ellip.  gqpSW36Qk= AQ0eJS8yuRoXzqvJrCN4uOsB

    *.100.168.192.in-addr.  &ellip.  4RaEHMvz0Khbv3k7n4w== AQ06/FfPf3yIo
                                                          ^ 5
1  100.168.192.in-addr.ar  &ellip.  kFwHh8yslFjYFIhcg== AQO/dB7EGk1IPpp
                                                      ^ 3

2  small.isp.com mvs03.it  &ellip.  V+38Y/I= AQPYZeXmV/uIJXttTwIlvLcvtD
                                           ^ 4
```

Figure 9-8 Keys in DDNS.DAT file at MVS03

Once you see the blank character as in ^ 3 and 4 above, you have found the public key. (In our key generation, the public key started with the characters *AQ*, but this is not always the case.) Cut and paste the string of characters behind the blank 4 into the KEY record of the forward file.

You can verify your cut-and-paste operation of both keys by counting the number of characters in each public key. The length of the key in the domain file should match that of the key in the reverse file. (Perhaps you could use the line mode editing command *cols* to count the characters.)

Note: At this point you should make a backup of the forward file. Changes to it through **nsupdate** commands in the future will alter the file's appearance. If at any time you wish to return to what you had coded originally, you may use this backup. This is especially useful during testing.

9.3.4 MVS03 DDNS reverse file

You see in 2 of Figure 9-9 that we have added the name that is used for the DDNS server when the server is sought by the DDNS client. This is the OSA port we used to isolate our DHCP/DDNS testing on.

```
;  /etc/dnsdata/named.rev for MVS03-osa  192.168.100.100
;
;$ORIGIN 100.168.192.in-addr.arpa.
@    IN  1  KEY 80 0 1 AQO/dB7EGk1IPpp19eSSbnazFpvtcxkFoWor7JpIMM/om6CrZrsMUO4
@    IN     SOA   mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
                      6 10800   3600   604800   86400 )
     IN     NS    mvs03.itso.ral.ibm.com.
100  IN     PTR   ns-updates.small.isp.com.  2
254  IN     PTR   murli750.small.isp.com.
```

Figure 9-9 DDNS reverse file (in-addr.arpa)

If this paragraph looks familiar to you, it is because you have just read essentially the same information in the section on the domain file. 1 shows a new record type in the domain file: a KEY record. This key is extracted *by you* from the information in the /tmp/ddns.dat file. The procedure is the same as for the domain file. See Figure 9-8 3. Cut and paste the string of characters behind the blank into the KEY record of the reverse file.

Note: At this point you should make a backup of the reverse file. Changes to it through nsupdate commands in the future will alter the file's appearance. If at any time you wish to return to what you had coded originally, you may use this backup. This is especially useful during testing

9.3.5 Starting the DDNS

The start procedure is the same as what you have seen before; the only difference is the boot file you point to. The log shows that you are loading a configuration that is prepared to perform dynamic updates, as you can see in the notations about `SECURED` below (**1**, **2**).

```
EZZ6699I name server starting. @(#) ddns/ns/ns_main.c, dns_ns, dns_r1.1 1
EZZ6701I named established affinity with 'T03ATCP'
EZZ6540I Static primary zone 'itso.ral.ibm.com' loaded (serial 5)
EZZ6540I SECURED primary zone 'small.isp.com' loaded (serial 6)   1
EZZ6540I SECURED primary zone '100.168.192.in-addr.arpa' loaded (serial 6)  2
EZZ6540I Static primary zone '251.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '252.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '236.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '235.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '105.24.9.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '104.24.9.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '221.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '109.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '0.0.127.in-addr.arpa' loaded (serial 2)
EZZ6540I Static cache zone '' loaded (serial 0)
```

Figure 9-10 DDNS files loading

9.3.6 A DDNS client requesting dynamic update

We started DHCPSD, the server function, at MVS03, ensuring that we were pointing to the correct resolver configuration, since we had multiple TCP/IP stacks running. Then we started the DHCP client at an OS/2 Warp workstation. The DHCP client caused the DDNS client function at the workstation to issue its own nsupdate request to MVS03, the DDNS. You see the results in the `named.run` data set in Figure 9-11 on page 301 (We had enabled DNS debug tracing at level 11, the trace has been heavily edited to show only the interesting data.)

```
 1
datagram from  192.168.100.101 .49400, fd 4, len 42; now Fri Mar  6 19:57:38
;; QUESTIONS:
;; ns-updates.small.isp.com, type = A, class = IN
;; ANSWERS:
ns-updates.small.isp.com. 86400 IN CNAME mvs03.itso.ral.ibm.com.
mvs03.itso.ral.ibm.com. 86400 IN A 192.168.100.100

   2
datagram from  192.168.100.101 .49401, fd 4, len 38; now Fri Mar  6 19:57:35
;; QUESTIONS:
;; myhost.small.isp.com, type = KEY, class = IN
   2
datagram from  192.168.100.101 .49403, fd 4, len 38; now Fri Mar  6 19:57:38
;; QUESTIONS:
;; myhost.small.isp.com, type = A, class = IN

   3
TCP connection from  192.168.100.101 .49168 (fd 7)
streamq = 0x14ce5a48
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id: 9684
;; ADD:
myhost.small.isp.com. 3600 IN KEY  0x0000 0 1 AQPHNSRI2+nPVg2eCf8ekMgGRQdevKuz1Y
 .myhost.small.isp.com. 4294967295 IN A *

Init_Dyn_Update: Mode=Secured, changingKEY=1, addingothers=1
do_dyn_update: dname myhost.small.isp.com type 25 class 1 ttl 3600
do_dyn_update: dname myhost.small.isp.com type 1 class 1 ttl 4660
do_dyn_update: dname myhost.small.isp.com type 24 class 1 ttl 4660
   4
fclose(8) succeeded
ns_req: answer ->  192.168.100.101 .49168 fd=7 id=9684 size=4 Local
streamq = 0x14ce5a48
   5
close(7) succeeded
```

Figure 9-11 named.run after nsupdate request from DDNS client

1 shows that the client wants to get the address of his DDNS. At 2 the client checks if there are already entries in the database. 3 shows the actual update. Notice that this is done using the reliable TCP connection. At 4 the update was successfully written to the database. At 5 the TCP connection was closed.

The DDNS client at the OS/2 workstation requested the update of the A record. You will later see that the DDNS client at MVS03 requests the update of the PTR record. (You will recall that we had imbedded **nsupdate** commands in the DHCP server configuration file in Figure 9-2.) The new view of the updated `named.for.dyn` shows that a record called `myhost.small.isp.com` was added, but it bears not the key of the A record in the ddns.dat of MVS03, but rather the key in the ddns.dat record of the OS/2 Warp client.

After these updates, a view of the named.for.dyn file shows that myhost.small.isp.com has been added:

```
$ORIGIN isp.com.
small..IN.KEY.0x0080   0   1 AQPYZeXmV/uIJXttTwIlvLcvtDfH5RNE+7GeYix01+JR
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=3
..IN.SOA.mvs03.itso.ral.ibm.com. gdente@itso.ral.ibm.com. (
..42 10800 3600 604800 86400 300 ).;Cl=3
$ORIGIN small.isp.com.
murli750.IN.A.192.168.100.254.;Cl=3
ns-updates.IN.A.192.168.100.100.;Cl=3
myhost.3600.IN.KEY.0x0000   0   1 AQO357s/vZCMhgwD+QLttfbsa7vp0LDtRGRpN5z  1
.4660.IN.A.192.168.100.101.;Cr=auth  2
.4660.IN.SIG.A 1 4 4660 892245244 892234444 0x71e6 myhost.small.isp.com Q 3
.4660.IN.SIG.KEY 1 4 4660 895355644 892234444 0x71e6 myhost.small.isp.com 3
localhost.IN.A.127.0.0.1.;Cl=3
```

Figure 9-12 Dynamically updated named.for.dyn after nsupdate request from client

Notice in Figure 9-12 above how new KEY, A, and SIG records have been added (1, 2, 3). Note that the key record for myhost 1 is the one generated by the DDNS client at the workstation and is visible only in the DDNS.DAT file at that OS/2 DDNS client. This means that the OS/2 DDNS client issued the **nsupdate** command that led to the A record update.

Note: If we had used a CNAME record instead of an A record, as described in Figure 9-7, the ns-updates entry in the dynamically altered file would have looked like this

```
$ORIGIN small.isp.com.
ns-updates.IN.CNAME.mvs03.itso.ral.ibm.com.;Cl=3
```

Although the updated domain file has a somewhat different syntax from the one you originally created, you may still manually make changes to the file in its current form. These updates will, however, be picked up only at the next recycle of the DDNS.

OS/2 DDNS clients are not capable of requesting PTR record updates with the **nsupdate** command. However, we see in Figure 9-13 on page 302 that the reverse file (in-addr.arpa) at MVS03 has been updated through someone's nsupdate:

```
$ORIGIN 168.192.in-addr.arpa.
100..IN.KEY.0x0080   0   1 AQO/dB7EGk1IPpp19eSSbnazFpvtcxkFoWor7JpIMM/om6  1
..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..7 10800 3600 604800 86400 300 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
$ORIGIN 100.168.192.in-addr.arpa.
254..IN.PTR.murli750.small.isp.com..;Cl=5
100..IN.PTR.ns-updates.small.isp.com..;Cl=5
101.3600.IN.KEY.0x0000   0   1 AQO6/FfPf3yIotmqUtCUDPRnARwWi/NnOxtd5kxJEz
.4660.IN.PTR.myhost.small.isp.com..;Cr=auth
.4660.IN.SIG.PTR 1 6 4660 892060119 892049319 0xea50 101.100.168.192.in-a
.4660.IN.SIG.KEY 1 6 4660 892060119 892049320 0xea50 101.100.168.192.in-a
```

Figure 9-13 Dynamically updated named.rev192.168.100.dyn from DDNS client

Note that the key record for 101 **1** contains the key that MVS03 generated; compare the key here with the one (**5**) in the DDNS.DAT file in Figure 9-8. They are the same. The nsupdate command was executed based upon the parameters in the DHCP configuration file of MVS03: see **7** in Figure 9-2. Therefore, MVS03 was the DDNS client for the PTR record update.

You will recall that we advised you to make a backup copy of the original static files. If you are making manual updates to the current version of these files, you may also wish to keep the original backup copy synchronized by adding the new static entries to it as well. But remember that the static entries to the dynamic zone file(s) cannot be made known to the running DDNS through a signal to the named process; you must stop and start the DDNS to pick up the new static entries. Alternatively you may add them as dynamic entries with the nsupdate command while the DDNS is running.

9.4 Pre-Boot eXecution Environment (PXE)

The Pre-Boot eXecution Environment (PXE) is an extension to dynamic IP that allows PCs and workstations to extract boot images from a server use them to boot up. This section describes this enhancement.

9.4.1 Current state of Dynamic IP

The current Dynamic Host Configuration Protocol (DHCP)/Dynamic Domain Name Server (DDNS) solution provides dynamic IP address and name administration for client workstations accessing intranets. This means a user doesn't have to wait for an administrator to get assigned an IP address and a host name which has to be typed in during a configuration process at the user's PC/workstation.

The automatic configuration process starts with the DHCPDISCOVER broadcast message issued from the PC/workstation after hardware startup. It requests for an IP address and some options like subnet mask, domain name, IP address for the domain name server and static route information. It finally ends up with a DHCPACK and binding state with a DHCP server which also may provide the name and the path of a load image to boot the PC/workstation. This boot image is based on additional client information.

The client may provide:

- A vendor statement which indicates a specific vendor's hardware equipment
- An operating system
- A class statement to configure a unique configuration to the client's workstation, for example, shared printers or a specific load image including special application software.

9.4.2 New requirements

Many IT managers request to separate the dynamic IP address assignment from the boot process into selectable three phases. In these phases different servers at different locations should be used for these tasks. For example:

- In the first phase, the dynamic IP address maintenance for clients' PC/workstations should be done through a DHCP server.
- In the second phase in a pre-boot process the appropriate boot image directory server should be selected which points to the boot server. The boot server stores the boot module for the appropriate PC/workstation depending on system architecture and vendor specifications and also user information.

- In the third phase the download of the boot image should be done by the PC/workstation from the selected boot server and the boot process should be executed.

The first phase can be skipped when a client already has a static IP address.

Separation into three phases makes it possible to use different boot servers at different locations for the acquired boot procedure relating to configuration parameters for the various clients.

Since IT managers also want to include application software to the boot process, different configurations have to be booted. This requires configuration parameters defined differently depending on the tasks for which the clients are responsible. For example, some clients in an enterprise are located in the sales department, others may be working on production tasks, etc.

Therefore, different boot images are needed for the client groups. This requires selected boot images for the different groups which might be defined for the department's servers while the dynamic IP address assignment process is done by centrally installed DHCP servers.

Of course, all clients should boot consistently also in an interoperable way regardless of vendors' hardware with or without hard disks (for example, using a network station) software, and available servers.

9.4.3 Common solution

This goal may be accomplished only through a uniform and consistent set of pre-boot protocol services within the client which ensures network-based booting through industry standard protocols used to communicate with the servers.

Implementation of clients

The client needs the support of certain vendor option fields in the DHCP protocol. Clients and servers that are aware of these extensions will recognize and use this information, and those that do not recognize the extensions will ignore them. The client also should be able to request the download of an executable image from the boot server and execute the downloaded image so that the system is ready for use by its user.

Implementation of servers

Servers must have the capability to support and recognize clients' DHCP extensions and provide information for a redirection of the client to an appropriate boot server. This redirection service may be organized in two ways:

- *Combined standard DHCP and redirection services.* This means DHCP servers supply IP addresses for clients and redirect PXE-enabled clients to boot servers.
- *Separate standard DHCP and redirection services.* This means PXE redirection servers (Proxy DHCP servers) are added to the existing network environment. They respond only to PXE-enabled clients, and provide only redirection to boot servers.

PXE APIs

For the interoperability of clients and the download process of boot modules, the client PXE code provides a set of services. These are:

- Pre-boot services API, which contain several control information and information functions (like client system environment)
- Trivial file transfer protocol (TFTP) API, used as TCP/IP application protocol

- User datagram protocol (UDP) API, used as a TCP/IP datagram transport protocol over the network, which means not a secure transport mechanism
- Universal network driver interface (UNDI) API, which enables basic control of and I/O through the clients' network interface device. This allows use of universal protocol drivers.

9.4.4 IBM's solution

IBM's solution for CS for z/OS IP is implemented by using separate standard and redirection services. It uses the DHCP server, and for redirection, the Bind Image Negotiation Layer (BINL) server.

Two standard protocols will support this solution:

1. Pre-Boot Execution Environment Protocol (PXE)
2. Bind Image Negotiation Layer (BINL) Protocol

Both protocols are specified in the Intel Wired For Management Baseline document.

- PXE is an enhancement to DHCP. This allows a PXE-enabled client to read DHCP information provided by the BINL server, find where the boot server is located and which boot image has to be loaded.
- BINL is a server that directs a PXE-enabled client to the boot server and the specific boot image file. In order to retrieve the appropriate boot image or boot load module, the client has to provide information about its system architecture and network interface type. BINL uses DHCP messages for its communication with the clients.

Therefore, in addition:

- The z/OS Communications Server for the Dynamic Host Configuration Protocol (DHCP) is enhanced to support the PXE extensions.

9.4.5 The overall DHCP, DDNS, PXE, BINL, boot server environment

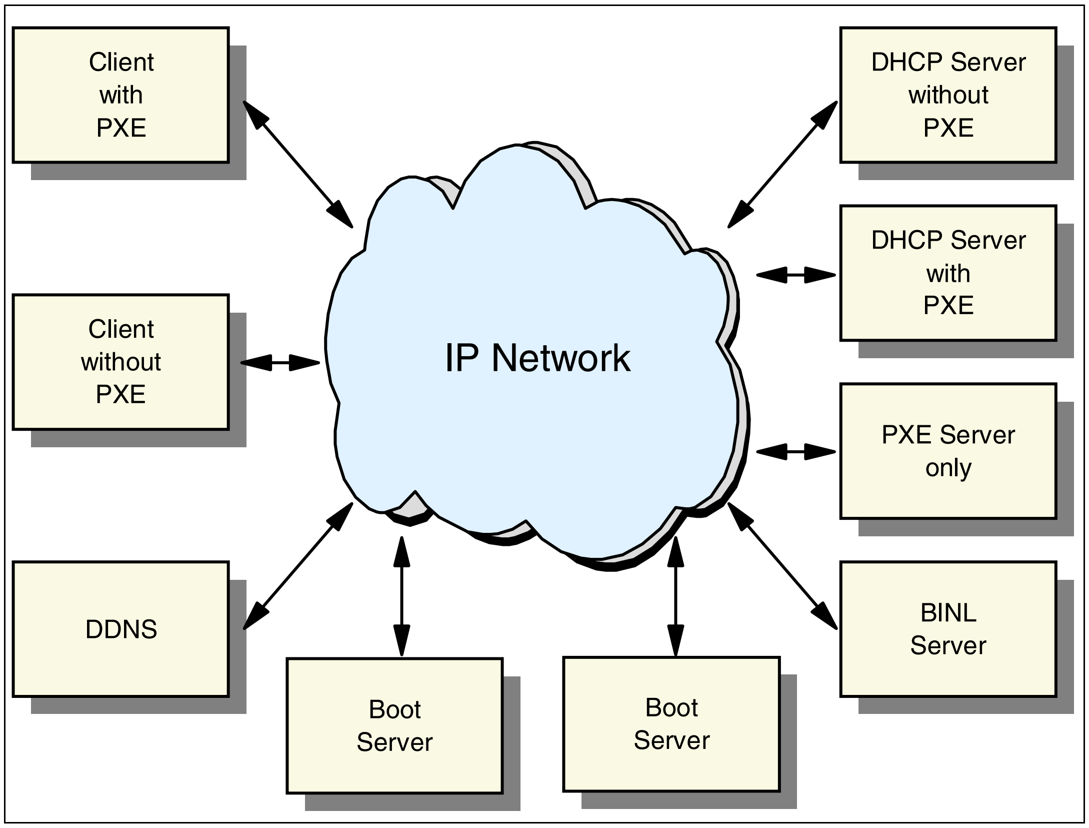


Figure 9-14 DHCP, DDNS, BINL, boot server environment

This figure shows that in a networking environment there might clients with and without PXE extensions.

Clients without PXE extensions will get their IP address assignment from the DHCP server without PXE support while clients with PXE support will take the offerings from the DHCP server with PXE extensions.

The BINL server provides the IP address for the specific boot server.

This kind of technology gives more flexibility in locating servers within a network.

DHCP and PXE extensions

DHCP extensions

The current DHCP header is extended with DHCP/PXE extensions or in some documents also called options. These options contain the following fields:

- Client User/Group ID (UUID), which is a universally unique ID, retrieved from the client system
- Client Network Name (for example 1 = UNDI)
- Client System Architecture

0	IA x86 PC
1	NEC/PC98
2	IA64 PC

3	DEC Alpha
4	ArcX86
5	Intel Lean Client

- Parameter Request List

1	subnet
3	router
43	vendor
60	class
128 - 135	vendor options

- Class Identifier
  a. Used for transactions between client and server; PXEClient:
     Arch:xxxxx:UNDI:yyyzzz
  b. Used for transactions between servers:
     xxxxx = client system architecture
     yyy = UNDI major version
     zzz = UNDI minor version
  c. Vendor Options (multiple DHCP_VENDOR options can be used)
  d. Message Type

1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNAK
7	DHCPRELEASE
8	DHCPINFORM

  e. Server ID
  f. Message Length
  g. End of Options

PXE exensions (commands)

PXE_MTFTP_IP (Multicast IP address)

PXE_MTFTP_CPORT (UDP port number of the client)

PXE_MTFTP_SPORT (UDP port number of the server)

PXE_MTFTP_TMOUT (Timeout, number of seconds a client must listen for activity before trying to start a new transfer)

PXE_MTFTP_DELAY (Reopen timeout)

PXE_DISCOVERY_CONTROL (Disable broadcast/multicast discovery)

DISCOVERY_MCAST_ADDR (Boot server discovery multicast address)

PXE_BOOT_SERVERS (Boot server type)

Type 0	PXE boot strap server
Type 1	Windows Microsoft NT server
Type 2	Intel LCM boot server
Type 3	DOS/UNDI boot server
Type 4	NEC ESMORO boot server
Type 5	IBM WSoD boot server
Type 6	IBM LCCM boot server
Type 7	CA Unicenter TNG boot server
Type 8	HP OpenView boot server
Type 9 - 32767	Reserved
Type 32768 - 65534	For vendor use
Type 65535	PXE API test server

PXE_BOOT_MENUE (Boot server type; type = local boot)

PXE_MENUE_PROMPT (Timeout)

PXE_MCAST_ADDRS_ALLOC (Multicast addresses for boot services)

PXE_CREDENTIAL_TYPES (Credential types retrieved from a security subsystem)

Loader Options
PXE_BOOT_ITEM (Boot server type and layer)

Vendor Options
Vendor specific information

PXE_END

9.4.6 DHCP/PXE protocol flow overview

This description gives you an overview of the protocol steps between the client and DHCP/PXE, BINL and boot server.

The procedure is divided into three subtasks:

- Dynamic IP address assignment
- Pre-boot procedure
- Boot procedure

Dynamic IP address assignment

1. The client initiates broadcasting a DHCPDISCOVER request to port 67 containing client PXE extension tags that identifies the client request coming from a PC/workstation with implemented PXE support.

   The DHCP/PXE header for this DHCPDISCOVER contains the following fields:

   – A tag for a client identifier (UUID)

   – A tag for the client UNDI version
   – A tag for the client system architecture
   – A DHCP option class 60, Class ID set to

     PXEClient:Arch:xxxxx:UNDI:yyyzzz

2. The DHCP server or Proxy DHCP server will first do the IP address assignment according to the rules defined for the client. Assuming the DHCP server has also implemented PXE support, it has recognized in the received DHCPDISCOVER request that the client supports PXE. Based on the server's knowledge about the client's PXE capability the server returns in its DHCPOFFER to the client a list of appropriate IP addresses of BINL servers. The DHCPOFFER is addressed to the client's port 68. The DHCP/PXE header for this DHCPOFFER contains the following fields:
   – Client IP address (and other parameters) offered by standard DHCP
   – BINL server list in the PXE tags
   – Discovery control options (if provided)
   – Multicast discovery IP address (if provided)
3. Because the client broadcasted the DHCPDISCOVER to the network the client may now receive more than one DHCPOFFER from other DHCP servers. Some servers may support PXE too and have included a list of BINL server addresses; others may not support PXE, they don't provide BINL server addresses.
4. The client selects the appropriate BINL server and starts to continue the normal DHCP flow to get its IP address reserved by the selected server. It starts with the DHCPREQUEST to the selected DHCP server requesting the registration of the reserved IP address. The request again is addressed to server port 67.
5. The server responds to the client's DHCPREQUEST after registration with a DHCPACK to the client's port 68.
6. After receiving the DHCPACK from the DHCP server the client checks through issuing an ARP request if the provided IP address is already in use. An ARP response will tell the client that the provided IP address is already implemented by another IP host. Therefore, the client sends a DHCPDECLINE to the server which provided the IP address and the entire process has to be restarted until the client has a valid IP address.
7. When this phase of the protocol flow is finished, the second phase, the pre-boot phase, will be started by the client.

Pre-boot procedure

1. The client selects a BINL server and sends a DHCPREQUEST to the IP address of the BINL server, which it had learned from the DHCP server from which it got its IP address. The client also provides in this request information describing its system architecture and network interface type. The DHCPREQUEST may be sent as broadcast to port 67, as multicast to port 4011`or unicast to port 4011 depending on the discovery control options included in the previous DHCPOFFER of the PXE service extension tags. This packet is the same as the initial DHCPDISCOVER as in step 1 (Dynamic IP Address Assignment), except that it is coded as a DHCPREQUEST and now contains the following information:
   – Client IP address assigned to the client from DHCP services
   – A tag for the client identifier (UUID)
   – A tag for the UNDI version
   – A tag for the system architecture

- A DHCP option 60, class ID set to

  PXEClient:Arch:xxxxx:UNDI:yyyzzz
- The boot server type in a PXE option field

2. The BINL server uses a table driven approach based on system architecture and network interface type values to determine which boot server and which boot file the client should use. These two pieces of information, the IP address of the boot server and the name of the load module (the boot image), are returned in a DHCPACK to the client addressed to port 4011.

   The acknowledgment contains:

   - Boot server's IP address
   - Boot file name
   - MTFTP configuration parameters
   - A tag for the system architecture

Note: The table-driven approach provides the intelligence in the server and boot file selection. For example, boot files for all Intel machines with PCI network interface cards can be installed in certain boot servers while all others with plug-and-play network interface cards can be installed in other boot servers, etc.

Boot procedure

1. The client then uses this information to contact the boot server's IP address, which it got from the BINL server, and starts via TFTP to download the proper boot kernel from the boot server. The name of the boot file on the boot server was also provided by the BINL server.
2. The download will be done via TFTP (port 69) or MTFTP (the port assigned in the BINL server's ACK).
3. If the download of the boot file was successful the client now starts the execution of the loaded boot file. The placement of the downloaded code in the memory is dependent on the client's CPU architecture.

9.4.7 Location of the DHCP/PXE/BINL servers

If we regard how systems support for PCs in large networks are organized then in many cases there will be one group in charge of administering IP addresses while another group might be taking care of servers for various boot images for these PCs or network stations.

When extending the IP address assignment by PXE functions the tasks for DHCP servers might be separated. One group might be responsible for a normal DHCP server while another group will support the network computing environment PXE-only DHCP server and BINL server.

The DHCP and PXE servers can be combined in one machine or installed separately on dedicated machines.

Also the BINL server can run on the same machine as the DHCP and PXE server. It may also be installed on any combination of multiple machines. Whatever combination of server and location is selected, DHCP, PXE and BINL work seamlessly for the end user.

This flexibility meets the requirements of many enterprises.

9.4.8 Definition of the DHCP/PXE/BINL servers

The z/OS DHCP server is controlled using a configuration file that resides in the Hierarchical File System (HFS).

DHCP/PXE configuration file

This configuration file contains new keywords to distinguish between DHCP without and with PXE extensions and a pointer to the BINL server.

You will find a sample of the DHCP configuration file in the samples directory /usr/lpp/tcpip/samples/dhcpsd.cfg. You copy the sample configuration file to the /etc directory as /etc/dhcpsd.cfg, which is the default location of this file.

Then you specify the type of server using the servertype keyword in the DHCP configuration file.

The values of this keyword cause the following behavior:

DHCP

The server behaves exactly as it does today. PXE-related options in the client requests are ignored. This is the default.

PXEDHCP

The server behaves as a combined PXE and DHCP server. It manages IP addresses as it does today, and additionally it provides PXE extensions in its interactions with clients that request them (by setting option 60 to PXEClient). This is how the PXE/DHCP server shown in the protocol description would be configured.

PXEPROXY

The server provides only PXE extensions. It does not manage or provide IP address assignment or any other DHCP information. It simply waits for DHCPDISCOVER messages for clients that contain option 60 with the value of PXEClient and responds to these DHCPDISCOVER messages specifying the address of the BINL server. If this configuration is used, there must be another DHCP server in the network that provides the rest of the DHCP functions.

The PXE client receives a DHCPOFFER from the PXEPROXY server and combines the BINL information provided with an IP address received from another DHCP server using the normal processing, and continues its boot process.

You may also use another file location by using the -f argument when starting the DHCP server, for example:

```
dhcpd -f /etc/dhcpd.cfg.test
```

In the DHCP configuration file there is also a pointer to the location of the BINL server. The keyword imageserver points to the IP address or host name of the BINL server.

For performance reasons, we recommend that you use the IP address to avoid the name server lookup.

If the keyword is not specified, the siaddr (server IP address) value is left null (0.0.0.0) on PXE responses. This means to the client that the BINL server resides on the same host as the PXE server.

BINL server

The BINL server is a new executable file binlsd. It is located in /usr/lpp/tcpip/sbin/binlsd and has a symbolic link to /usr/sbin/binlsd.

The BINL server uses DHCP messages to implement its protocol, and operates similarly to DHCP.

This server receives requests and sends replies on port 4011. Therefore, it can reside on the same machine as a DHCP and/or PXE server.

BINL configuration file

A sample of the configuration file for BINL server is located in the directory /usr/lpp/tcpip/samples/. This sample configuration file should be copied to /etc/binlsd.cfg.

If you want to use another file name you may use the -f parameter when starting the BINL server, for example:

```
binlsd -f /etc/binl/binlsd.cfg1
```

The configuration file gives you choices to define the boot image for the client. Please see an overview of some choices in the following screen.

```
#   The directives are specified in the form of
#   <keyword> <value1> ... <valueN>.
#
#   Here is a partial list of keywords whose value can be specified
#   in this file:
#
#   Keyword            Effect
#   -------------      ------------------------------------------------
#   sa                 Specifies the system architecture.
#   nit                Specifies the network interface type.
#   lsa1nic            Specifies the lsa1 nic type.
#   tftp               Specifies the address of the tftpd server.
#   bpname             Specifies the install filename given to clients.
#   numLogFiles        The number of log files desired.
#   logFileSize        The size of log files in kilobytes.
#   logFileName        The name of the most recent log file.
#   logItem            An item to be logged.
#   call               Server exit definition.
#   servername         The LCM server name for LSA-1 clients. Not used
#   serverdomainname   The LCM domain name for LSA-1 clients. Not used
```

Figure 9-15 Sample BINL configuration file (part 1 of 2)

Definitions are continued in Figure 9-16 on page 313.

```
#   client              Definition of a set of options for a specific cl
#                       or a definition of a client not to be serviced.
#
#   pxevendor           Configuration delimiter to indicate pxe options
#
#   pxeoption           A pxe configuration option value to pass to clie
#
#   The scope of a keyword is limited by a pair of curly brackets ({, })
#   within which the keyword is located.  If a keyword is located outsid
#   of any pair of curly brackets, its scope is applicable to all the
#   clients served by this server.  The curly brackets must appear alone
#   a line.
#
#  Log files.  This set of parameters specifies the log files that will
#  maintained by this server.  Each parameter is identified by a keyword
#  and followed by its value.
#  Keyword      Value          Definition
#  --------     ------------   --------------------------------------
#  numLogFiles  0 to 99        number of log files.  If 0 is specified,
#                              no log file will be maintained and no lo
#                              message is displayed anywhere.  When a l
#                              file reaches maximum size, a new log fil
#                              is created, until the maximum number of
#                              log files have been created.  Only the m
#                              recent n log files are kept/
#
#  logFileSize  in K bytes     maximum size of a log file.  When the si
#                              of the most recent log file reaches this
#                              value, it is renamed and a new log file
#                              created.
#
#  logFileName  file path      name of the most recent log file.  Less
#                              recent log files have the number 1 to
#                              n-1 appended to their names; the larger
#                              the number, the less recent the file.
#                              the number, the less recent the file.
#
#  logItem                     An item that will be logged.
#              SYSERR          System error, at the interface to the pl
#              OBJERR          Object error, in between objects in the
#              PROTERR         Protocol error, between client and serve
#              WARNING         Warning, worth of attention from the use
#              EVENT           Event occurred to the process.
#              ACTION          Action taken by the process.
#              INFO            Information that might be useful.
#              ACNTING         Accounting information on clients served
#              TRACE           Code flow, for debugging.
#
#  call <dll name> <function name> [options]
#                              If present, the server will load the dll
#                              and load the function given in function
#                              it may have.
#  servername <server name>    If present this name is sent to LSA-1 cl
#                              packet.  If this option in not present t
#                              its name to the client as the LCM server

#  serverdomainname  <domain name>  <workgroup | domain>
#                              Specifies the domain name of the LCM ser
#  ............                LSA-1 clients in the offer packet.
#  ............
#  see continuation  in the file or in the manual.
```

Figure 9-16 Sample BINL configuration file (part 2 of 2)

10

BIND Domain Name System (DNS)

This chapter contains information about configuring the name server in a BIND-based Domain Name System (DNS). *Berkeley Internet Name Domain* (BND) was developed at the University of California, Berkeley and is currently maintained by the Internet Software Consortium (ISC) and is used on the vast majority of name server machines on the Internet providing a stable system on top of which an organization's name architecture can be built.

CS for z/OS IP provides a port of the BIND-based version 9 name server to the zSeries platform. It is different from the BIND 4.9.3-based name server that existed in previous CS for OS/390 releases. Both modes of the name server are available through one command interface. The BIND 9 mode of the name server allows for greater security, has IPv6 support, and brings an industry standard Dynamic DNS (DDNS) to the zSeries platform. However when run in the BIND 9 mode, the name server does not have DNS/WLM (connection optimization) capability nor is it compatible with prior CS for OS/390 Dynamic DNS (DDNS) support.

DNS/WLM provides intelligent sysplex distribution of requests through cooperation between the WLM and the DNS server and is implemented only on BIND 4.9.3. For customers who elect to place a DNS in a z/OS sysplex, DNS will invoke WLM sysplex routing services to determine the best system to service a given request. For customers who choose to implement BIND 9, but still need to balance the DNS requests in a sysplex environment, this can be accomplished through Sysplex Distributor, the strategic load distribution technology. Please see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517 for more information. In CS for z/OS V1R2 IP, customers will be able to implement and run their Domain Name System in three different ways:

- Using BIND 9 only
- Using BIND 4.9.3 only
- Using both BIND 9 and BIND 4.9.3 simultaneously

This chapter contains the following sections:

- 10.1, “Domain Name System overview” on page 316
- 10.2, “Setting up a BIND 4.9.3-based Domain Name Server” on page 323
- 10.3, “DNS/WLM - Connection Optimization in a sysplex domain” on page 340
- 10.4, “Setting up a BIND 9-based Domain Name Server” on page 355
- 10.5, “Securing your DNS environment” on page 385
- 10.6, “Running DNS in BIND 9 and BIND 4.9.3 simultaneously” on page 393
- 10.7, “DNS tools” on page 397

For information about DNS, please reference the following RFCs:

- **1033** - *Domain Administrators Operations Guide*
- **1034** - *Domain Names - Concepts and Facilities*
- **1035** - *Domain Names - Implementation and Specification*
- **1183** - *New DNS RR Definitions*
- **1886** - *DNS Extensions to support IP version 6*
- **1995** - *Incremental Zone Transfer in DNS*
- **1996** - *A Mechanism for Prompt Notification of Zone Changes*
- **2065** - *Domain Name System Security Extensions*
- **2136** - *Dynamic Updates in the Domain Name System (Non-Secure)*
- **2137** - *Secure Domain Name System Dynamic Update*
- **2181** - *Clarifications to the DNS Specification*
- **2308** - *Negative Caching of DNS Queries*
- **2845** - *Secret Key Transaction Authentication for DNS (TSIG)*

Valuable information about DNS is also available in:

- *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *z/OS V1R2.0 CS: IP Migration*, GC31-8773

10.1 Domain Name System overview

The following text provides a very brief overview of the major components of the DNS. Note that this subject is a very complex one. Numerous books on DNS have been written. If you are going to be implementing a DNS, it is strongly recommended you obtain a good text on the subject.

10.1.1 Why DNS?

The TCP/IP protocols rely upon a strict system of addressing in order to reach a host on a network. These host addresses are numeric, in the format `nnn.nnn.nnn.nnn` where `nnn` can range from 0 to 255 (with a few exceptions). The major drawback of this system is that, for most people, numbers are difficult to remember. As a result, today's IP-based networks use a mapping of host names to host numbers or addresses. The obvious advantage of this name-to-address mapping is that we can assign easy-to-remember names to hosts on the network. For example what if we map the host `Garth` to the number 9.24.104.200? The immediate advantage is that we no longer need to memorize the address, we can use the name `Garth` in its place. What happens, though, if another `Garth` wants to use this name on the network too? The Domain Name System not only handles the name-to-address (and vice versa) mapping, it also encompasses a system that is capable of ensuring that names are unique throughout all interconnected networks.

10.1.2 What is the Domain Name System?

Domain Name System (DNS) is a protocol developed to handle the use of host names in a network. It provides the host name-to-IP address mapping (and vice versa) through network server hosts called Domain Name Servers. DNS can also provide other information about server hosts and networks such as the TCP/IP services available at a server host and the location of domain name servers in a network.

DNS organizes the hosts in a network into domains. A domain is a group of hosts that share the same namespace and are usually controlled within the same organization. Domains are arranged in a hierarchical form. A special domain known as the root domain exists at the top of the hierarchy. The root domain servers store information about server hosts in the root domain and the name servers in the delegated, top-level domains, such as com (commercial), edu (education), and mil (military). The name servers in the top-level domain, in turn, store the names of name servers for their delegated domains, and so on.

Controlling the names

The complete name of a host, also known as the fully qualified domain name (FQDN), is a series of labels separated by dots or periods. Each label represents an increasingly higher domain level within a network. A domain name server requires the FQDN. The client resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name. The complete name of a host connected to one of the larger networks generally has more than one subdomain, as shown in the following example:

```
small.itso.raleigh.ibm.com
```

Here, itso.raleigh.ibm.com is the lowest level domain name, a subdomain of raleigh.ibm.com, which again is a subdomain of ibm.com, a subdomain of com. We can also represent this naming concept by a hierarchical tree as shown on Figure 10-1 on page 318.

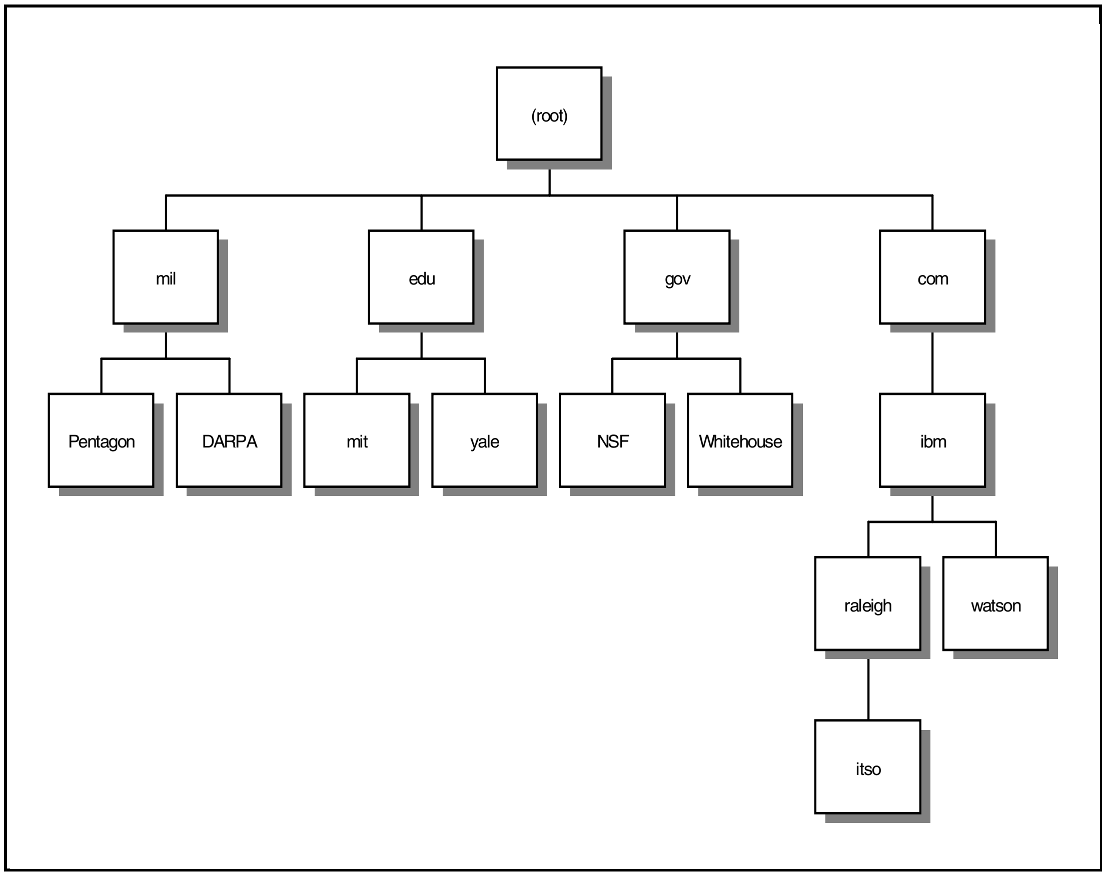


Figure 10-1 Hierarchical distribution of Domains

The uniqueness of host names within a domain is managed in much the same fashion that file names are used within a directory. For example, the files /bin/matt and /sbin/matt are obviously different. The DNS uses the same principle, but the root directory is listed on the far right, and successive subdomains (equivalent to successive subdirectories) are listed from right to left. For example, if we have an address such as `buddha.ral.ibm.com`, our highest (or closest to the root) domain is com. Note that the root domain is represented by a dot, just as on a file system the root directory is a slash. The same address could correctly be written as `buddha.ral.ibm.com.`. Often we leave the dot out of the address, but you will see situations later in this text where it becomes very important. The next subdomain is ibm, and we continue on down to the lowest subdomain of buddha.

At this point, you might be wondering where the hosts are. Any domain name can represent a host while at the same time it can represent the domain of a group of hosts (or more subdomains even). In other words, we know the domain `ibm.com` represents the domain of the IBM corporation, but there could also be a host called `ibm.com` out there as well.

So how does DNS get a hold of these name-to-address mappings? The DNS is essentially a distributed database system. A network administrator chooses a host on the network that will run as a DNS server. This server will usually have a zone for which it is responsible for resolving names to addresses (and addresses to names, called reverse mapping). A zone can describe an entire domain of mappings, more than one domain, or only a subset of a domain. Either way, the server will refer to a configuration file containing simple lists of address and name records, often referred to as a data file. A host to address (and vice versa) mapping is referred to as a resource record. For example:

```
mvs03a     IN    A   168.192.221.3
```

is the resource record that maps host mvs03a to the address 168.192.221.3.

The name server's job is to respond to queries by providing either an address for a name, or else a name for a supplied address. Each server will only know about the hosts within the zones for which it is configured to respond. It should also be noted that the name-to-address mapping can be one to many -- a host can have more than one IP address.

Finding an address

Hosts on a network must be configured to look for a DNS server when a host name is used instead of an address. The request for name resolution is handed to a resolver routine. The resolver routine will have an address or list of addresses that point to hosts running a DNS server. In the MVS TCP/IP environment this is controlled by the NSINTERADDR parameter. The resolver routine will send the query to the host listed in NSINTERADDR, and the resolver routine waits for a response and passes the answer back to the application that requested the resolution. When a query is sent to a name server and the name server is expected to find the answer, this is referred to as a recursive query. Later, we'll discuss a situation where we simply want a name server to give us the best answer it has (that is, the name and address of a more likely name server). This is referred to as an iterative query.

It can happen that a name server doesn't know the mapping that is being requested. When this happens, there are several courses of action the server can take. Usually, there is a name server record pointed to by a data file that maps a domain name to a specific server for resolution. This hard-coded record gives the name server a mapping between a domain it doesn't have data for and the address of a name server that should have the mapping. That name server will respond back to the original name server with its answer, and the original name server will then respond back to the resolver routine on the host that originated the request.

What if we get a request for a domain that is completely separate from any domain we have data files for? This distributed database Domain Name System contains something called a root name server. A root name server's purpose in the DNS world is to provide other servers with information on where to find the top-level domain server for a given domain. In other words, if a name server gets a request for `where.world.ca.`, and the name server knows nothing about the ca domain we can send the request to our root name server. The root name server probably will not resolve the request, but it should return the address of a name server more likely to be able to resolve the request (for example, it could return the address for a name server responsible for a world.ca zone). This process of sending the request to another name server, and receiving iteratively better responses, is called iterative resolution.

Once we have the IP address of the host, the work of the DNS is done.

Reverse mappings

Sometimes, we might already know an address, but we want to find the host name associated with the address. When such a request comes to a name server, it is referred to as a reverse lookup. Reverse mappings are considered to be in the in-addr.arpa domain. The term in-addr.arpa is associated with the actual coding of the resource record for a reverse mapping. For example, the reverse mapping for host mvs03a would look like:

```
3.221.168.192    IN  PTR     mvs03a.buddha.ral.ibm.com.
```

10.1.2.1 DNS implementation

To implement this process, the network uses three components:

- *Domain name space* and *resource records* which are specifications for a tree-structured name space and the data associated with the names. The whole domain name space is partitioned into areas called *zones*.
- *Name Server programs* which hold and administer the information about a domain tree's structure or part of it. With the name server, the network can be broken into a hierarchy of domains.
- *Resolvers programs* which extracts the information from name servers in response to client requests.

Domain Name Space

The structure of the DNS is tree-like where each portion represents a domain or a subdomain.The root of this tree-liked structure is a dot (.). A domain is a full subtree in the name space. Each domain administrator has authority over his portion of the tree and may delegate authority and control for any of his subsections of the tree to other administrator. Resource Records (RRs) are the data associated with the names in a domain. A domain name identifies a node. The authoritative data for a zone is composed of all of the RRs attached to all of the nodes from the top node of the zone down to a single host.

Domain name servers

A name server is said to be authoritative for some part of the domain name space, called zone. A zone consists of the resources within a single domain (for example, commercial or .com) or subdomain (for example, raleigh.ibm.com). Typically, a zone is administered by a single organization or individual. All host systems in a given zone share the same higher level domain name,for example, host1.ral.ibm.com, host2.ral.ibm.com, host3.ral.ibm.com, and so on. As system administrator, you create a zone of authority by listing all the host systems in your zone in the database file of the name server that is authoritative for the zone.

If a domain name server receives a query about a host for which it has information in its database or in its cache, it performs the name resolution and returns all the address records associated with the host to the client. Some hosts (for example, routers or gateways between two or more networks) might have more than one IP address. Alternatively, the name server can query other name servers for information. This process is called iterative resolution. The local name server successively queries other name servers, each of which responds by referring the local name server to a remote name server that is closer to the name server authoritative for the target domain. Finally, the local name server queries the authoritative name server and gets an answer. If the information about a requested host name does not exist or if a name server does not know where to go for the information, it sends a negative response back to the client.

There are multiple name server modes in the DNS:

- *Authoritative*
  - *Master* (primary)
  - *Slave* (secondary)
- *Caching-only*
- *Forwarders*
- *Stealth*

A single server can perform multiple functions. For example, it can be a primary server and a slave server for different zones. The purpose of having these different kinds of servers is to provide redundancy (in case of system failure), to distribute the workload among multiple servers, to speed up the name-resolution process, and to provide flexibility in network design. In addition to being an authoritative or caching-only server, a name server can be defined to only contact a specific set of name servers if queries cannot be resolved locally (through the use of forwarders).

Authoritative servers

An *authoritative server* is the authority for its zone. It queries and is queried by other name servers in the DNS. The data it receives in response from other name servers is cached. Authoritative servers are not authoritative for cached data.

There are two types of authoritative servers: *master* and *slave*. Each zone must have only one master name server, and it should have at least one slave name server for backup. Calling a particular name server a master or slave is misleading. Any given name server can take on either or both roles, as defined by the boot or conf file.

A *master name server* maintains all the data for its zone. Static resources are kept in database files called domain data files. Master name servers can also receive zone updates dynamically.

A *slave name server* acts as an alternate to the master server if the master name server becomes unavailable or overloaded. The slave name server receives zone data directly from the master name server in a process called zone transfer. A slave server, like a master server, is authoritative for a zone.

Caching-only servers

All name servers cache (store) the data they receive in response to a query. A caching-only server, however, is not authoritative for any domain. When a caching-only server receives a query, it checks its cache for the requested information. If it does not have the information, it queries a local name server or a root name server, passes the information to the client, and caches the answer for future queries.

Forwarders

Normally, name servers answer queries either from their data files or their cached data and, if that does not succeed, they attempt to contact other name servers identified in their data files as authoritative for certain domains. However, name servers can also be configured to contact special servers called *forwarders* before contacting the name servers listed in their data files. Forwarders are typically used when you do not wish all the servers at a given site to interact with the rest of the Internet servers. A sample scenario is a network with a number of internal DNS servers and an Internet firewall. Those servers which are not allow to pass packets through the firewall would forward their packets to the server designated as forwarder and this server would query the Internet DNS servers on the internal server's behalf.

Stealth server

A stealth server is a server that answers authoritatively for a zone, but is not listed in that zone's NS records. Stealth servers can be used as a way to centralize distribution of a zone, without having to edit the zone on a remote nameserver. When the master file for a zone resides on a stealth server in this way, it is often referred to as a hidden primary configuration. Stealth servers can also be a way to keep a local copy of a zone for rapid access to the zone's records, even if all official name servers for the zone are inaccessible.

10.1.2.2 What is a BIND

BIND, an acronym for Berkeley Internet Name Domain, implements the Domain Name System (DNS) protocols, and is used on the vast majority of name serving machines on the Internet, providing a robust and stable system on top of which an organization's name architecture can be built. The resolver library included in the BIND distribution provides the standard Application Programs Interface (APIs) for translation between domain names and Internet addresses and is intended to be linked with applications requiring name service.

The BIND components are a nameserver daemon called *named*, a resolver library, and a set of tools for verifying the proper operation of the DNS server. A nameserver is a program that stores information about named resources and responds to queries from programs, called resolvers, which act as client processes. (A client is a program that requests services from a server). The basic function of an Name Server is to provide information about network objects by answering queries.

CS for z/OS V1R2 IP added BIND 9-Based DNS to line up with the industry standards, implementing the new features released in this new version, such as:

- Data authentication security (DNSSEC) and access control
- IPV6 support
- Performance improvements (multi threaded)
- Industry standard Dynamic DNS (DDNS)
- Split DNS

However, since BIND 9 does not have DNS/WLM and is not compatible with prior DDNS support, CS for z/OS V1R2 IP still supports BIND 4.9.3-Based DNS, so you have the option to define which one is the best choice to implement in your network.

10.1.3 Files to support a DNS implementation

The following files are available to configure a DNS server in CS for z/OS V1R2 IP:

1. A startup file or boot file:
   - BIND 4.9.3 called by default named.boot.
   - BIND 9 called by default named.conf
2. The domain data files (used only by the master name server)
   a. A data file or zone file that maps host names to IP addresses for specific domains. We refer to this zone file as the *forward domain* file, because it conventionally uses a suffix of *for*, as in *named.for*. The file name is usually the zone name, but it can be anything.
   b. A data file or zone file that resolves IP addresses to host names for IP networks. We refer to this zone file as the reverse domain file, or in-addr.arpa file; it conventionally uses a suffix of *rev*, as in named.rev. Again, the file name is usually the zone name, but it can be anything.
3. Hints (root server) file. The Hints file, also known as Cache file, contains the names and IP addresses of the authoritative root domain name servers. The root domain name servers contain the names of name servers in the top-level domains such as com, edu, mil, etc..... By convention it uses the suffix ca.
4. A loopback file, which by convention uses the suffix lbk, define the loopback names and addresses that a host uses to route queries to itself. This file must be defined if you decide to start both BIND 4.9.3 and BIND 9 in the same host.

5. Log file. Used only when you start BIND 9 and is configured via the `logging` statement in the named.conf file.

Optionally a DNS server may have additional configuration files:

1. Extra forward files for additional zones that the name server may be servicing
2. Extra reverse files if the name server manages more than one IP network
3. A loopback file, which by convention uses the suffix lbk, to define the loopback names and addresses
4. Forward and reverse zone files that are used to manage a name server that cooperates with WLM. (used only with BIND 4.9.3)
5. Forward and reverse zone files that are used in a network with DHCP and DDNS

All of these files may be present at a secondary name server as well as at the primary name server. A secondary name server obtains most of its data from the primary name server through a process called *zone transfer*. The secondary server uses its forward and reverse files to store the data obtained from the primary name server.

10.2 Setting up a BIND 4.9.3-based Domain Name Server

The following steps will show you what is required to implement and run a basic master (primary) Domain Name Server environment, using BIND 4.9.3-based DNS:

1. 10.2.1, “Define your zone” on page 323
2. 10.2.2, “Create a configuration file for your environment (named.boot)” on page 325
3. 10.2.3, “Specify stack affinity (multiple stack environment)” on page 325
4. 10.2.4, “Specify port ownership” on page 326
5. 10.2.5, “Update the name server start procedure (optional)” on page 327
6. 10.2.6, “Create the domain data files” on page 327
7. 10.2.7, “Create the loopback file” on page 331
8. 10.2.8, “Create the cache file (hints file)” on page 332
9. 10.2.9, “Starting the DNS server” on page 332
10. 10.2.10, “Verifying if the name server has started correctly” on page 335
11. 10.2.11, “Reloading the Domain Name Server V4.9.3” on page 336
12. 10.2.12, “Stopping the DNS server” on page 337
13. 10.2.13, “Implementing a secondary name server DNS64” on page 338

10.2.1 Define your zone

The first thing to do when you implement a Domain Name Server in your network environment is to plan and define what you are to control, the boundaries of your domain, which servers are going to control your domain, how you are going to interact with external domains. By defining these issues, you will be able to draw a good picture of the domain you are going to control, and use it as input to go through all the necessary steps to built your domain name server. A good way to get these definitions is to answer a few questions and draw the scenario you are willing to create.

So we create our scenario by using the following guidelines:

1. What will be the name of your Domain?
   a. We defined in our test environment a Domain named itso.ral.ibm.com
2. Which host will become your master domain name server (primary)?
   a. The host we are going to use as primary Server is 9.12.6.67 (sc63)
3. Which host will be defined as your slave domain name server (secondary)?
   a. The host we are going to use as secondary server is 9.12.6.61 (sc64)
4. To which servers will you forward your unresolved queries?
   a. In our scenario all unresolved queries will be forwarded to 9.24.104.108, the ITSO DNS
5. What kind of environment do you have (single or multiple TCPIP stacks)?
   a. Our environment has multiple TCPIP stacks and we are going to use a stack named tcpipb in our scenario

Based on the answers provided, we are able now to draw our scenario, which, in our case would look like the one showed in Figure 10-2:

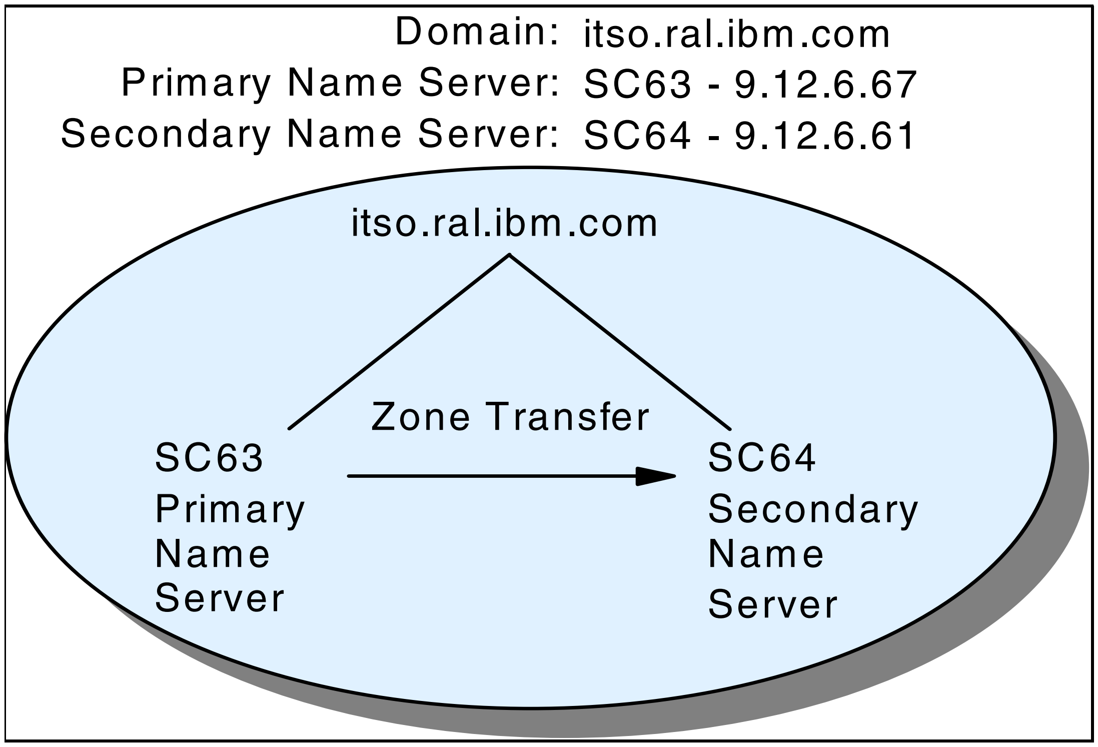

Domain: itso.ral.ibm.com
Primary Name Server: SC63 - 9.12.6.67
Secondary Name Server: SC64 - 9.12.6.61
itso.ral.ibm.com
Zone Transfer
SC63 Primary Name Server
SC64 Secondary Name Server

Figure 10-2 Primary and secondary name servers

The domain is itso.ral.ibm.com. There is one primary name server platform: SC63. The secondary name server SC64 obtains its information from zone transfers with SC63.

10.2.2 Create a configuration file for your environment (named.boot)

The boot file initializes the name server environment and points to the individual name server definition files and to the options that the name server will provide for each zone it supports. The boot file is referenced in the start procedure used from the MVS console or in the UNIX System Services shell command used to start the domain name system server. If no boot file is referenced in the start procedure or command, the default of /etc/named.boot is assumed. Figure 10-3 shows the contents of the named.boot files being used in our scenario.

```
;      boot file for BIND 4.9.3-Based Name Server
;
;type      domain                 source file or host
;
 directory  /etc/octavio/dnsdata                       1
 primary    itso.ibm.com            itso.for.v4        2
 primary    6.12.9.in-addr.arpa     itso.rev.v4        3
 primary    0.0.127.in-addr.arpa    itso.lbk.v4        4
 cache    .                       itso.ca.v4           5
 forwarders 9.24.104.108                               6
 options    query-log                                  7
```

Figure 10-3 named.boot file

The contents of this boot file are:

1 The directory entry (or directive) defines the location of the files that are listed within the boot file. Those HFS files will be found in the /etc/dnsdata/ subdirectory.

2 This primary directive defines the domain name for the zone followed by the file to read for the name-to-IP address mappings.

3 This primary directive defines the domain name for the zone followed by the file to read for the IP address-to-name mappings.

4 This special primary directive is used to map the loopback address.

5 The cache directive defines the root level domain, identified by a dot (.), and the file in which the IP address of the root DNS servers can be found. The cache file is also known as the hints file.

6 The forwarders directive identifies a list of DNS servers to query if the primary or secondary mappings cannot resolve a DNS request. Forwarders act as intermediaries between name servers at one site and name servers at multiple other sites. They query off-site name servers and cache the results, thus minimizing network traffic that could otherwise have been generated by allowing all name servers to query all other name servers without requesting help from a forwarder first.

7 This options directs the V4 name server to log all queries with the syslog daemon.

10.2.3 Specify stack affinity (multiple stack environment)

A BIND DNS name server must be associated with a TCP/IP stack. This process occurs by default if there is only one copy of CS for z/OS IP in the z/OS image. The establishing of affinity to a particular stack is an issue only if you are running a CINET configuration with multiple stacks. In a multiple stack environment, the name server is like any application. It binds to the stack specified by TCPIPJOBNAME, which is defined in the TCPIPDATA file. To determine the location of the file that contains the correct TCPIPjobname parameter, you must use an environmental variable called RESOLVER_CONFIG. If you are going to start the name server from MVS, this variable can be specified in a file defined by STDENV DD card in the startup procedure as shown in Figure 10-4.

```
//STDENV   DD PATH='/etc/octavio/named.env.v4',
//            PATHOPTS=(ORDONLY)
```

Figure 10-4 STDENV DD card example

An example of the STDENV file contents is shown in Figure 10-5

```
 RESOLVER_CONFIG=/etc/octavio/named.resolv
 RESOLVER_CONFIG=//'TCPIPB.SC63.TCPPARMS(TCPDATB)
 the next variable directly defines the TCPIP stack to connect
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
```

Figure 10-5 Contents of STDENV file

If you decide to start the name server named from the z/OS UNIX shell, you can identify the location of the resolver configuration using the **export** command to include the variable before the named command, as shown in Figure 10-6:

```
export RESOLVER_CONFIG="//'TCPIPB.SC63.TCPPARMS(TCPDATB)'" named
```

Figure 10-6 Export command to change the RESOLVER_CONFIG variable

10.2.4 Specify port ownership

By default, the name server uses a single port (53) for TCP and UDP sessions. DNS servers can share port 53 if they connect to clients or other servers on different IP addresses. A BIND 4 server cannot specify IP addresses in its configuration file but will use any IP address not already selected by a BIND 9 server. To specify port ownership when using the named start procedure for BIND 4.9.3, you must include the name server procedure name in the port definition statement of the PROFILE TCPIP dataset adding the suffix 2 to the procedure name you are going to use, as shown in Figure 10-7:

```
PORT
    53 TCP NAMED42 1          ; NAME SERVER V4 TCP PORT
    53 UDP NAMED42            ; NAME SERVER V4 UDP PORT
```

Figure 10-7 Reserving port 53 to NAMED4 startup procedure

1 The procedure name is NAMED4 and the suffix 2 indicates a BIND V4 name server

If you decide to start the Name Server using the z/OS UNIX shell, the port can be generically reserved to UNIX applications, as shown in Figure 10-8:

```
PORT
        53 TCP OMVS          ; reserved to z/OS UNIX
        53 UDP OMVS          ; reserver to z/OS UNIX
```

Figure 10-8 Reserving port 53 to z/OS UNIX Shell

10.2.5 Update the name server start procedure (optional)

When choosing to start the name server from MVS, create a start procedure. This is not necessary if the name server is started from the z/OS UNIX shell. Move the sample start procedure, SEZAINST(NAMED4), to a recognized PROCLIB. Specify name server parameters and change the data set names as required to suit local configuration. The boot file path (for BIND 4.9.3-DNS) can also be changed in the sample start procedures, as shown in Figure 10-9 below:

```
//*****    Proc used to start DNS server bind 4.9.3 *******
//NAMED4   PROC B='/etc/octavio/named.boot'
//NAMED4   EXEC PGM=BPXBATCH,REGION=OK,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/named -b &B '
//STDENV   DD PATH='/etc/octavio/named.env',
//            PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
```

Figure 10-9 NAMED4 startup procedure

If you are migrating your name server from an older OS/390 version, you can continue to use the PGM name provided with those versions (**ezansnmd**), which continue to be supported. In this case, the procedure should be defined as shown in Figure 10-10:

```
 //NAMED PROC B='/etc/octavio/named.boot',P='53'
//***************************************************************
 //NAMED    EXEC PGM=EZANSNMD,REGION=OK,TIME=NOLIMIT,
 //*     PARM='POSIX(ON) ALL31(ON)/ -b &B -p &P -d 11 -q'
 //        PARM=('POSIX(ON) ALL31(ON)',
 //          'ENVAR("RESOLVER_CONFIG=/etc/octavio/named.data")',
 //*         'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPB")',
 //          '/ -b &B -p &P -d 11')
 //*SYSTCPD  DD DISP=SHR,DSN=TCPIPMVS.SC63.TCPPARMS(TCPDATA)
 //SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
 //SYSIN    DD DUMMY
 //SYSERR   DD SYSOUT=*
 //SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
 //CEEDUMP  DD SYSOUT=*
```

Figure 10-10 Using PGM EZANSMD to start the name server

10.2.6 Create the domain data files

The domain data files contain information about a domain, such as the IP addresses and names of the hosts in the domain for which the master name server is authoritative. The forward domain data file contains entries that provide forward mapping (host names-to-IP addresses for each host system in the zone) as well as additional information about system resources. The reverse domain data file contains entries that provide reverse mapping (IP addresses-to-host names).

10.2.6.1 The forward file

The forward files are identified in the /etc/named.boot and contain the name-to-IP address mappings. Every zone must have a separate forward file. If your DNS server services three zones, then you must have three separate forward files. These files are stored in HFS format and their UNIX permission bits must allow name server read/write access. The entries in the forward files are defined using a special format that is defined by RFC1034. These entries are called *resource records*. All resource records (RR) are written in the file with the following format:

```
name ttl address_class record_type record_data
```

The record_type and record_data fields are the only required fields. The name, ttl and address_class have defaults that may be set if they are not specified. The record_type indicates the type of resource record which defines the record data format. Valid resource record types include, but are not limited to, the following:

Table 10-1 DNS valid Resource Records:

DNS Resource Record type	Record Type description
SOA	Start Of Authority for the stated Zone
A	Name -to-IP address translation record
SRV	Service-to-Location definition
PTR	IP address-to-Name translation record
NS	Name of the authoritative DNS server for the stated Zone
CNAME	alias name of a stated host

You also can include in the zone files other records, called Master File Directives, to define some specific values, as follows:

- $ORIGIN directive - sets the domain name that will be appended to any unqualified record as shown in the example below:

```
   $ORIGIN itso.ibm.com
   sample   CNAME  host01
would be equivalent to:
   sample.itso.ibm.com   CNAME host01.ibm.com
```

- $INCLUDE directive - Read and process the file *filename* as if it were included into the file at this point. If origin is specified, the file is processed with $ORIGIN set to that value, otherwise the current $ORIGIN is used. Once the file is read, the origin and current domain revert to the values they were prior to the $INCLUDE. it has the following syntax:

```
   $INCLUDE filename [origin] [comment]
```

- $TTL directive - Set the default Time To Live (TTL) for subsequent records with undefined TTLs. Valid TTLs are of the range 0-2.147.483.647 seconds. It has the following syntax:

```
   $TTL default-ttl [comment]
```

Figure 10-11 shows the contents of the named.for file. It is referenced in the named.boot file you saw in Figure 10-3 as the primary DNS information for the domain itso.ral.ibm.com.

```
;
 $ORIGIN ral.ibm.com.                                                    1
;
 itso    IN   SOA   dns63.itso.ral.ibm.com admin@itso.ral.ibm.com (      2
              1            ; Serial                                      3
              10800        ; Refresh                                     4
              3600         ; Retry                                       5
              604800       ; Expire                                      6
              86400)      ; Minimum ttl                                  7
;
 $ORIGIN itso.ral.ibm.com.                                               8
                   IN      NS      dns63                                 9
                   IN      NS      dns64
;
 _http._tcp.www         SRV  0  0 80   www.itso.ral.ibm.com              10
                        SRV  10 0 8000 www2.itso.ral.ibm.com
;
 localhost         IN   A    127.0.0.1                                   11
 sc63              IN   A    9.12.6.68
 sc64              IN   A    9.12.6.62
 host1             IN   A    9.12.6.67
 host2             IN   A    9.12.6.61
 www2              IN   A    9.179.147.237
 www               IN   A    9.179.147.237
;
 gateway           IN   A    9.12.6.68                                   12
                   IN   A    9.12.6.67
;
 mail              IN   CNAME host1                                      13
 ftp               IN   CNAME host2
```

Figure 10-11 named.for file contents

1 This $ORIGIN control records defines the higher domain zone where the subdomain itso is being defined. Its used in conjunction with the SOA record that follows to resolve the name of `itso.ral.ibm.com.` which will is the domain name of our zone.

2 The SOA record specifies the name of the host that has the domain name server authority for the zone. This name is appended to the domain name specified in the previous $ORIGIN record. The SOA record includes a mailbox address of the user who is responsible for the zone: `admin@itso.ral.ibm.com`. A left parenthesis signals that everything between here and a succeeding right parenthesis should be considered as belonging to the same resource record, despite record boundaries of the master data set. The number enclosed in parenthesis are parameters used to set different values for the zone, as follows

3 the *Serial* parameter is used to identify the serial number of the domain database. is referenced by secondary name servers. Increment the number each time you change the file.The secondary name servers determine when to do a zone transfer based on an increment in this value.

4 The *Refresh* parameter is expressed in seconds. A secondary name server that has transferred this zone from the primary name server should not wait more than this number of seconds before it requests a refresh (a full zone transfer) from the primary name server. Before requesting a zone transfer, the secondary name server checks if the value of the serial field for the zone in question has changed or not. If not, a zone transfer is not necessary.

5 The *Retry* parameter is expressed in seconds. If a secondary name server fails to refresh its copy of resource records, it should wait this number of seconds before it retries the refresh from the primary name server.

6 The *Expire* parameter is expressed in seconds. This is the maximum time a secondary name server should consider its copy of resource data valid. If the secondary name server does not succeed with a zone transfer from the primary name server within this amount of time, it should consider its copy of the resource data obsolete, and stop answering queries for this zone.

7 The *Minimum TTL* parameter is expressed in seconds. Every time a response from this name server is sent out, it contains a time to live (TTL) field, which signifies how long the receiver should be able to consider the response valid. In BIND name servers, this *Minimum TTL* field really represents the Default value if no TTL value has been specified on an individual resource record.

Note: For dynamic WLM resources the TTL value defaults to 0 and can be specified with the *-l* start option.

8 The control entry $ORIGIN appends the string itso.ral.ibm.com. to all the following host names that do not end with a dot ('.').

9 The *NS* (Name Server) records specify the name servers in the zone. Note that *NS* records do not distinguish between primary and secondary name servers. You will later see that in our scenario dns64 is a secondary name server that receives zone transfers from dns63.

10 Specifies the location of services (for example, ftp, http, telnet). This record identifies one or more hosts capable of satisfying the service and protocol represented in the name field of the resource record. The name field for these resource records must follow a unique naming convention. The contents of this RR are:

name priority weight port target

where:

the *name* field must be specified as *Service.Protocol.Name.*:

Service is the symbolic name of the desired service, as defined in Assigned Numbers (1) or locally.

Protocol is typically TCP or UDP

Name is the domain this RR refers to.

Priority is a number in the rage of 0-65535 and represents the priority of this target host. A client must attempt to contact the target host with the lowest-numbered priority. Target hosts with the same priority should be tried in pseudo-random order.

Weight is used for a crude connection balancing mechanism. When selecting a target host among those that have the same priority, the chance of trying this one first should be proportional to its weight. The valid range is 0-65535.

Port is the port on this target host of this service. The valid range is 0-65535.

Target is the domain name of the target host. This name must be a canonical name and not an alias. There must be one or more A records for this name. A target of consisting of a dot (.) means that the service is not available at this domain.

In our file, the SRV records specify the location for the 'http' service using the ‘TCP’ protocol. The first record has a priority of 0, a weight of 0, uses port 80 and the service is provided at host `www.itso.ral.ibm.com`. The second record has a priority of 10 which is lower, a different port and target. A web client capable of using SRV records requesting a site named `http://itso.ral.ibm.com/`, would be directed to www.itso.ral.ibm.com and www2.mycorp.com. The client would be responsible for determining which site to connect first based on priority and then on weight.

11 An `A` record is an address record, naming the host and the IP address of that host.

12 You can also use `A` records to define more than one address to a host. If you do not specify a value in the name field, the value from the preceding record will be used.

13 A `CNAME` record is an alias for a host name.

10.2.6.2 The reverse file (in-addr.arpa file)

Figure 10-12 shows the contents of `itso.rev.v4` reverse file. It is referenced in the named.boot file as the primary DNS information for the domain 6.12.9.in-addr.arpa. Note the inverse syntax used for referencing this in-addr.arpa file in Figure 10-3 on page 325. There must be a single reverse file for each IP address grouping.

```
; $ORIGIN 6.12.9.in-addr.arpa.                             2

 @ IN SOA dns63.itso.com. admin@itso.com. (  1
              1  10800  3600 604800 86400 )
              IN      NS    dns63.itso.com.
              IN      NS    dns64.itso.com.
 68           IN      PTR    dns63.itso.com.        3
 62           IN      PTR    dns64.itso.com.
 67           IN      PTR    host1.itso.com.
 61           IN      PTR    host2.itso.com.
 20           IN      PTR    printserver.itso.com.
```

Figure 10-12 in-addr.arpa file

1 The `@` sign on the SOA record is a special character that indicates the SOA is for the zone named in the named.boot file. It is used as a shorthand method, but it would have been equally as valid to specify the `$ORIGIN` statement that has been commented out at **2**.

3 The value 68 will have the $ORIGIN (implied by the @ sign on the SOA record) appended to form the full resource name of 68.6.12.9.in-addr.arpa, which is a resource name in the in-addr.arpa special domain.

The PTR record type includes, in the data part of the record, the fully qualified name of the host corresponding to the IP address in the name part of the record.

10.2.7 Create the loopback file

The loopback file contains the loopback address. This is the address that a host uses to route queries to itself. The preferred loopback address is 127.0.0.1, although you can configure additional loopback interfaces in the TCPIP Profile. For BIND 4.9.3, DNS will bind to 127.0.0.1 in additionl to the first loopback address configured in TCPIP Profile. Figure 10-13 shows the contents of the itso.lbk.v4 file. It is referenced in the named.boot file as the primary name server information for the domain 0.0.127.in-addr.arpa. domain.

```
0.0.127.in-addr.arpa. IN SOA  ns1.itso.ral.ibm.com. admin@ral.ibm.com (
     1
     10800
     3600
     604800
     86400 )
0.0.127.in-addr.arpa.   IN   NS  dns63.itso.ral.ibm.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.itso.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

Figure 10-13 Loopback file contents

Note: In addition to creating the loopback file, add an address resource record called localhost to the forward domain data file. This record supports proper two-way resolution.

10.2.8 Create the cache file (hints file)

The hints file contains the names and IP addresses of the authoritative root domain name servers. The root name servers contain the names of name servers in the top-level domains such as com, edu, and mil. The name server uses root server information when deciding which name server to contact when it receives a query for a host outside its zone of authority and it does not have the data in its cache.To obtain a hints file, point your Web browser to `ftp://ftp.rs.internic.net` and retrieve the file named.root from the domain subdirectory. Update your hints file on a regular basis.The cache directive in a BIND 4.9.3 boot file specifies the path and name of the hints file.

Note: The hints file does not contain cached data nor does the name server provide other hosts with the information contained in the hints file. A forward-only server is the only type of name server that does not require a hints file.

Figure 10-14 displays the partial contents of the named.ca file.

```
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
 A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
.                       3600000      NS    B.ROOT-SERVERS.NET.
 B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
```

Figure 10-14 Partial contents of a hint file

10.2.9 Starting the DNS server

Once you have finished the previous configuration steps, you will be able to start the domain name server. Basically you can start the `named` process either from a mvs procedure, which has been customized in step 5.2.2, or from the z/OS UNIX shell. If the user choose to use the MVS procedure, a supervisor with an authorized TSO ID can start a name server from the MVS operator's console by starting the customized `named` start procedure.

To execute the named process fro z/OS UNIX, a user with superuser authority can start the name server from the shell by starting z/OS UNIX, then issuing the **`named`** command and, optionally, any parameters. It is also possible to start the server automatically when z/OS UNIX is started by specifying the path and file name of the z/OS UNIX initialization shell script in the /etc/init.options file using the -sc option:

```
 -sc /etc/rc       shell script = /etc/rc
```

Figure 10-15 -sc option definition

The file /etc/rc is the default z/OS UNIX initialization shell script that is executed when z/OS UNIX is started. Information such as the following can be entered in /etc/rc:

```
# Start name server
 |      /usr/lpp/tcpip/sbin/named -b /named/production/named.boot &
```

Figure 10-16 Contents of the rc file

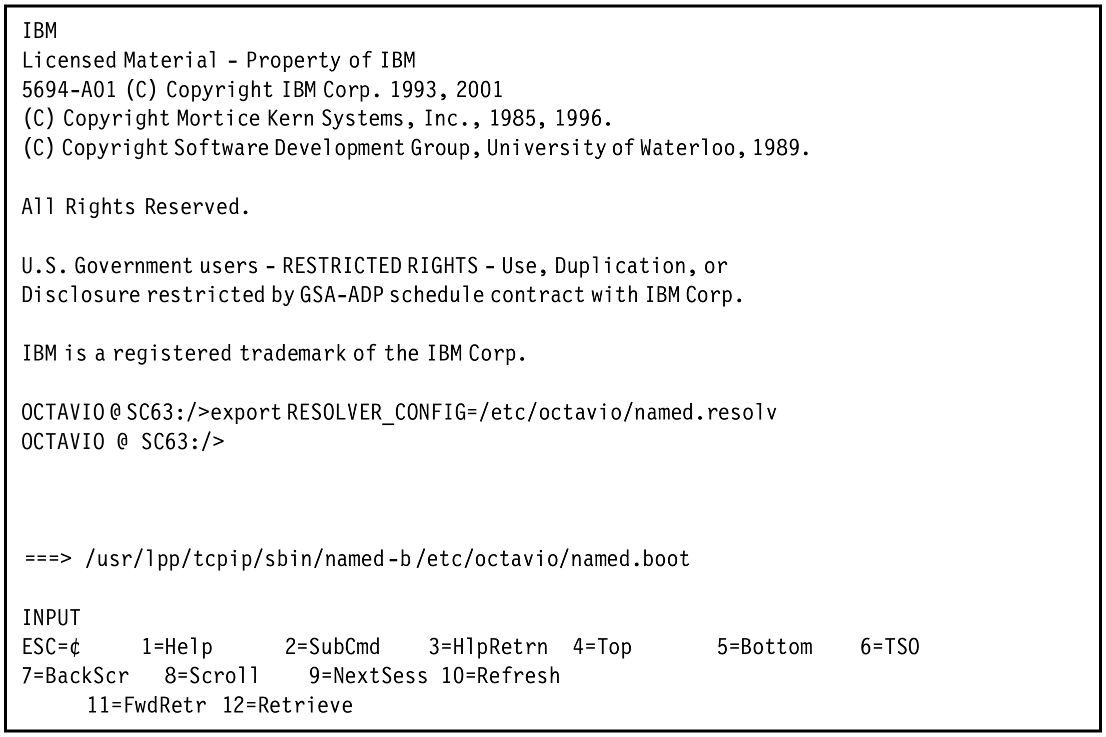
If you use the UNIX System Services /etc/rc method, the actual startup of the name server waits for the completion of TCP/IP initialization. You cannot start the name server with the INETD daemon. The startup must know either via default or via parameters what the boot file name is so that the correct data can be loaded into the DNS.

The following figures will show other method you can use to start the BIND DNS server and the use of export command to define the correct resolv.conf file:

- Starting named using z/OS UNIX console (**omvs** command in TSO command line option) Like any UNIX command you execute each command in the command line of the UNIX console as follows:

```
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2001
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

OCTAVIO @ SC63:/>
OCTAVIO @ SC63:/>


===> export RESOLVER_CONFIG=/etc/octavio/named.resolv

RUNNING
ESC=¢     1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
7=BackScr   8=Scroll    9=NextSess 10=Refresh
      11=FwdRetr  12=Retrieve
```

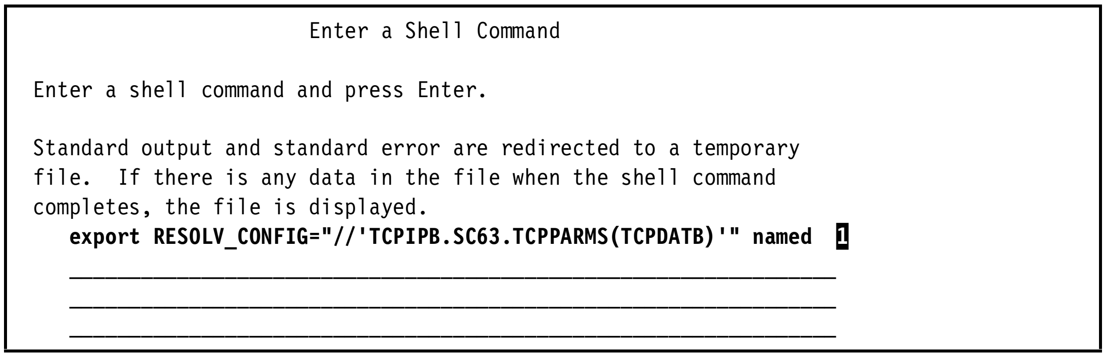
Figure 10-17 Executing export *command from z/OS Unix console*



```
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2001
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

OCTAVIO @ SC63:/>export RESOLVER_CONFIG=/etc/octavio/named.resolv
OCTAVIO @ SC63:/>



===> /usr/lpp/tcpip/sbin/named -b /etc/octavio/named.boot

INPUT
ESC=¢     1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
7=BackScr   8=Scroll    9=NextSess 10=Refresh
      11=FwdRetr  12=Retrieve
```

Figure 10-18 Executing named command from z/OS Unix console

- Starting `named` using ISHELL (**`ish`** in TSO command line option). If you choose this way to start the `named` daemon and you are in a multistack environment then you must use the **`export`** command to define the RESOLV_CONFIG statement that indicates which TCPIP stack you will be using. The following figures show you the commands.



```
                        Enter a Shell Command

 Enter a shell command and press Enter.

 Standard output and standard error are redirected to a temporary
 file.  If there is any data in the file when the shell command
 completes, the file is displayed.
    export RESOLV_CONFIG="//'TCPIPB.SC63.TCPPARMS(TCPDATB)'" named  1
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
```

Figure 10-19 ISHELL command screen

1 In the sample use the export file to point to an MVS file. Once the export command is executed connecting the new resolv file to named process, you can proceed and start the named procedure in the ISHELL command screen as shown:

```
                       Enter a Shell Command

 Enter a shell command and press Enter.

 Standard output and standard error are redirected to a temporary
 file.  If there is any data in the file when the shell command
 completes, the file is displayed.
    /usr/lpp/tcpip/sbin/named -d 11 -b /etc/octavio/named.boot
    _________________________________________________________________
    _________________________________________________________________
    _________________________________________________________________
```

Figure 10-20 Starting named from the Ishell command screen

We customarily started the name server with a procedure from the MVS console. The following is the console log following the startup of the DNS server process at SC63.

```
 S NAMED4
 $HASP100 NAMED4   ON STCINRDR
 IEF695I START NAMED4 WITH JOBNAME NAMED4 IS ASSIGNED TO USER STC, GROUP SYS1
 $HASP373 NAMED4   STARTED
 IEF403I NAMED4 - STARTED - TIME=13.01.26 - ASID=0088 - SC63
                        ---TIMINGS (MINS.)--        ----PAGING COUNTS---
 -JOBNAME  STEPNAME PROCSTEP   RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP
-NAMED4             NAMED      0   36   .00  .00 .00    1290   0      0    0
 EZZ9166I STARTING NAMED, BIND V4
 EZZ6452I NAMED STARTING. @(#) DDNS/NS/NS_MAIN.C, DNS_NS, DNS_R1.1 1.62 9/23/9710:57:21
 EZZ6475I NAMED:  READY TO ANSWER QUERIES.
 -                     --TIMINGS (MINS.)--      ----PAGING COUNTS---
 -JOBNAME STEPNAME PROCSTEP  RC  EXCP  CPU  SRB CLOCK  SERV PG  PAGE  SWAP
 -NAMED4           *OMVSEX  01  541.00.00.04   8049   0      0
  IEF404I NAMED4 - ENDED - TIME=13.01.29 - ASID=0088 - SC63                    1
  -NAMED4   ENDED.  NAME-                    TOTAL CPU TIME= .00
  TOTAL ELAPSED TIME= .04
 $HASP395 NAMED4   ENDED
```

Figure 10-21 named startup messages in the MVS console log

Note: Note that the procedure NAMED4 has ended 1, but you will later see that it has created a child process called NAMED433.

10.2.10 Verifying if the name server has started correctly

Once you had started the named process, you can go to the syslogd hfs file (as defined by the syslogd process in your z/OS UNIX) and check the results of the startup procedure. The messages generated should basically appear as shown in Figure 10-22.

```
/usr/lpp/tcpip/sbin/namedÝ131178¨: EZZ9166I STARTING NAMED, BIND V4
namedÝ131180¨: EZZ6698I name server starting. @(#) ddns/ns/ns_main.c, dns_ns, dns_r1.1
1.62  9/23/9
namedÝ131180¨:  EZZ6701I named established affinity with 'TCPIPB'              1
namedÝ131180¨:  EZZ6540I Static primary zone 'itso.com' loaded (serial 1)        2
namedÝ131180¨:  EZZ6540I Static primary zone '6.12.9.in-addr.arpa' loaded (serial 1)
namedÝ131180¨:  EZZ6540I Static primary zone '0.0.127.in-addr.arpa' loaded (serial 1)
namedÝ131180¨:  EZZ6540I Static cache zone '' loaded (serial 0)
namedÝ131181¨:  EZZ6475I named:  ready to answer queries.
namedÝ131181¨:  EZZ6476I Return from getdtablesize() > FD_SETSIZE
```

Figure 10-22 syslog contents after named is started

Looking at the message EZZ6701 1 in Figure 10-22 above, you can confirm that your server has been associated via the resolver process with the TCPIPB stack. You can also see which level of your customization has been loaded 2: serial 1. You may want to maintain the serial number designation in the files as you customize them in order to understand which version of a file has been loaded. The serial number is also used at secondary name servers to determine whether data needs to be reloaded after a zone transfer. If you discover a mismatch between the data at a secondary name server and that at a primary, the discrepancy could be due to one of the following:

- A view of the secondary name server prior to its having pulled new data from the primary
- The failure of the secondary to pull new data from the primary because a matching serial number at the primary signalled the secondary not to update its data

When the name server is up with no logged errors, ensure that it can accept queries. Ensure that the name server can accept queries locally from both the MVS and z/OS UNIX environments. To be able to do that you must first assure that your userid is using the same resolver configuration files. After the resolver configuration is correct, you can test your server either with the nslookup or dig command. An example using nslookup follows. Issue the following command from both the z/OS UNIX shell and the TSO ready prompt. In this sample, the name 'host1.itso.com.' is used for the search.

```
OCTAVIO @ SC63:/>nslookup host1
```

The result of the command follows:

```
Defaulting to nslookup version 4
Starting nslookup version 4
Server:  dns63.itso.com
Address:  9.12.6.68

Name:    host1.itso.com
Address:  9.12.6.67
```

10.2.11 Reloading the Domain Name Server V4.9.3

To reload data that you may have changed during this lifetime of the name server, you can send a signal to the UNIX System Services shell asking DNS to reread its configuration files. To send the signal from the z/OS UNIX console, execute the following command:

```
kill -HUP ($cat /etc/named.pid)
```

From the ISHELL follow these steps:

1. from the ISHELL select *Tools*
2. From *Tools* select option *1 - Work with Processes.*

3. In the *Work with Processes Signal* screen go to the line where the named process is and enter s
4. In the *Enter Signal Number* screen, confirm you choose the process you want to reload, go to the *Signal* number field and enter the SIGHUP signal number, #1.

These steps will reload the named process implementing your changes. To confirm your name server has been reloaded, you can check the syslogd log file for the following messages:

```
EZZ6557I reloading name server resource information
EZZ6475I named:  ready to answer queries.
```

This command works only at a primary name server. A secondary periodically returns to query the primary for new data, theoretically eliminating the need to reload a secondary with a signal. (The refresh interval is one of the settings in the SOA record.)

The reload process is not designed for dynamic domains, since these are updated via the nsupdate command.

10.2.12 Stopping the DNS server

You can stop the DNS server with the MVS STOP command P NAMED43, which is the name of the startup procedure used to start you name server (it can be any name), followed by the number “3”, which is an extra forking process assigned to BIND V4 during the startup process The advantage of the STOP command is the graceful termination of the name server and the issuing of messages in syslog. Other alternatives are to use the MVS CANCEL command or the OMVS KILL command. The OMVS KILL command can be issued from OMVS or from the NSSIG procedure.

Executing the KILL command in the z/OS UNIX console:

```
kill $(cat /etc/named.pid)
```

From the ISHELL follow these steps:

1. Select *Tools*
2. From *Tools* select 1 - *Work with Processes.*
3. In the *Work with Processes* screen go to the line where the `named` process is and enter `k` (kill)

You can also stop the process using MVS CANCEL command to NAMED43, and this will cause the following messages in the console log:

```
C NAMED43
IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=NAMED43, ASID=007A.
BPXP018I THREAD 106A29D800000000, IN PROCESS 131223, ENDED 085
WITHOUT BEING UNDUBBED WITH COMPLETION CODE 40222000,
AND REASON CODE 00000000.
IEE301I NAMED43          CANCEL COMMAND ACCEPTED
IEF450I NAMED43 STEP1 - ABEND=S222 U0000 REASON=00000000 087
        TIME=09.34.14
```

10.2.13 Implementing a secondary name server DNS64

The next step to perform to implement our domain name server scenario is to configure the secondary name server. A secondary name server can be primary for some zones and secondary for others. Its boot file indicates for which zones it is primary and for which it is secondary. Using a zone transfer process, the secondary server retrieves the files for specified zones from the primary name server that it points to. The secondary stores this information in its own files if the retrieved serial number is higher than the current serial number stored at the secondary (Figure 10-11 on page 329, 5) The secondary obtains the files from the primary name server based upon the refresh interval coded on the SOA record or the resource record (RR) itself.

To implement a secondary name server, follow the same steps used to create the master name server. The difference is that there is no need to create the domain data files (the files containing host-to-address and address-to-host mappings). These files are maintained on the master name server and the secondary (slave) name server transfers this data to its own database. The boot file also has different definitions to state that this server is a slave (secondary) name server.

To show how we define our secondary name server we go through the same steps we followed to build the master name server:

1. Create the boot file of the secondary name server

To create the named.boot file of the secondary name server, copy the named.boot file created for the master name server and alter the contents to reflect this is a secondary server, as shown in Figure 10-23.

```
;  /etc/named.boot
;
;        boot file for name server
;
;
;type    domain                  source file or host
;
directory   /etc/octavio/dnsdata
secondary   itso.com           9.12.6.68       itso.bk.for.v4 1
secondary   6.12.9.in-addr.arpa 9.12.6.68       itso.bk.rev.v4
primary     0.0.127.in-addr.arpa                named.lbk.v4   2
cache     .                                   named.ca.v4
options query-log
```

Figure 10-23 Boot file for the secondary name server

1 This line indicates that this server will be a secondary server and its primary server is located at IP address 9.12.6.68. It also indicates where the data gathered from the primary server will be saved. (itso.bk.for.v4)

2 The server is still considered to be a primary DNS for its loopback address.

The remaining steps are shown here only as an information. For further instructions about each steps, refer to the previous section, implementing the master name server.

2. 10.2.3, “Specify stack affinity (multiple stack environment)” on page 325
3. 10.2.4, “Specify port ownership” on page 326
4. 10.2.5, “Update the name server start procedure (optional)” on page 327
5. 10.2.7, “Create the loopback file” on page 331

6. 10.2.8, “Create the cache file (hints file)” on page 332
7. 10.2.9, “Starting the DNS server” on page 332
8. 10.2.10, “Verifying if the name server has started correctly” on page 335

At a secondary name server, you would see similar console messages about the file loading process. A partial SC65 console log is displayed in Figure 10-24.

```
   +EZZ6475I NAMED:  READY TO ANSWER QUERIES.
 BPXF024I (OCTAVIO) May 22 14:15:09 namedÝ65764¨: EZZ6472I received
 query from 9.12.6.121 for zos12.ral.ibm.com (type=SOA)
 BPXF024I (OCTAVIO) May 22 14:15:09 namedÝ65764¨: EZZ6472I received
 query from 9.12.6.121 for 6.12.9.in-addr.arpa (type=SOA)
 BPXF024I (OCTAVIO) May 22 14:15:13 namedÝ65764¨: EZZ6472I received
 query from 9.12.6.121 for zos12.ral.ibm.com (type=SOA)
 BPXF024I (OCTAVIO) May 22 14:15:20 namedÝ65764¨: EZZ6472I received
 query from 9.12.6.121 for 68.6.12.9.in-addr.arpa (type=PTR)
 BPXF024I (OCTAVIO) May 22 14:15:21 namedÝ65764¨: EZZ6472I received
```

Figure 10-24 Successful zone transfer from SC63 to SC65

If your zone transfers are successful, the files identified in the secondary name server's boot file as the storage for retrieved files can be viewed. The format looks a bit different from your own coding at the primary, but the contents are nonetheless identifiable. For example, below is a portion of the domain file retrieved by SC65, the secondary name server.

```
; BIND version @(#) ddns/ns/named-xfer.c, dns_ns, dns_r1.0 1.30  6/4/97 14:19:25
; zone 'zos12.ral.ibm.com' last serial 0
; from 9.12.6.68 at Wed May 22 14:15:39 2002
$ORIGIN ral.ibm.com.
zos12..IN.SOA.dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
..1 10800 3600 604800 86400)
..IN.NS.dns63.zos12.ral.ibm.com.
..IN.NS.dns64.zos12.ral.ibm.com.
$ORIGIN zos12.ral.ibm.com.
mail..IN.CNAME.host1.zos12.ral.ibm.com.
ftp..IN.CNAME.host2.zos12.ral.ibm.com.
sc63..IN.A.9.12.6.68
..IN.A.9.12.6.67
dns63..IN.A.9.12.6.68
host1..IN.A.9.12.6.67
host2..IN.A.9.12.6.63
localhost.IN.A.127.0.0.1
dns64..IN.A.9.12.6.63
www..IN.A.9.179.147.237
$ORIGIN _tcp.zos12.ral.ibm.com.
_http..IN.SRV.0 0 80 www.zos12.ral.ibm.com.
```

Figure 10-25 Contents of the backup forward file, in SC65, retrieved from SC63

10.3 DNS/WLM - Connection Optimization in a sysplex domain

Officially known as connection optimization, DNS/WLM provides intelligent sysplex distribution of requests through cooperation between the WLM and the DNS server. In DNS/WLM, the DNS will invoke WLM sysplex routing services to determine the best system to service a given client request. For additional information, please refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517.

WLM provides various workload related services: performance administration, performance management and workload balancing, for example. WLM is capable of dynamically assessing resource utilization on all participating hosts within a sysplex.

A DNS server running on a host in the sysplex can take advantage of WLM's knowledge and use it to control how often an address for a particular host in the sysplex is returned on a DNS query. When a sysplex name server queries the WLM for information, it is provided with a weight corresponding to the relative resource availability of each participating host in the sysplex. These weights are used by the name server to control the frequency with which an address will be returned for a given host. If a host in the sysplex is relatively busy, its address will not be returned by the server as often as a less busy host's address. As you might have guessed, this means that you must use host names when accessing an application in the sysplex. The name server is the only place where address selection based upon resource availability can occur. If you use an IP address directly, no workload balancing can occur with DNS/WLM solution.

10.3.1 How load distribution works using DNS/WLM

Before any application can take part in workload balancing (connection optimization), the TCP/IP stack *should* register itself to WLM. It, and only it, can correctly maintain the IP addresses that DNS/WLM will need to resolve a host name to an address. Once the stack has been registered, WLM knows its name and its interface addresses. The SYSPLEX Routing keyword in the IPCONFIG statement in the TCP/IP profile tells the stack to do this. As interfaces are activated and deactivated, the stack keeps WLM informed of the status. Note that if the stack does not register, DNS uses the defined interface addresses but they are never checked for active status.

Note: For each TCP/IP stack within a sysplex, only the first 15 addresses listed in the HOME statement of your profile will be registered to WLM (and passed on to the name server when it requests the information). If an interface is not active, its address will not be forwarded to WLM. When the name server receives these active addresses from WLM, it will accept only the addresses that have a matching address listed in its data file. This requirement for statically defined addresses ensures full administrative control over the workload distribution.

Once the TCP/IP stacks have registered with WLM, the following occurs (see Figure 10-26):

1. When each application becomes active in the sysplex, it registers with WLM using the appropriate group, server, and host (stack) name.
2. The sysplex-enabled name server will query WLM periodically for a list of available applications. WLM returns the names of the applications within each group, the active IP addresses associated with them (obtained from the associated stack name), and a set of workload-related weights.

3. Resource records representing the application's group name are dynamically (and not permanently) added to the name server's data files. These entries will now be treated the same as any hard-coded entries read from the server's data file.
4. When a request to resolve one of these group (server) names comes in, the server will choose an address to return based upon the weighting factor provided from WLM.
5. The next request for the same group name within the sysplex will be given the next address according to the weighting. Depending on the relative resource utilization of the hosts in the sysplex, this could be the same address as retrieved in the previous query, it could be a different address for the same host, or it could be an address for another host in the sysplex.
6. WLM is queried by the name server every 60 seconds (by default) for a new set of addresses and host weights. This can be controlled by the -t parameter at startup of the name server task (daemon).

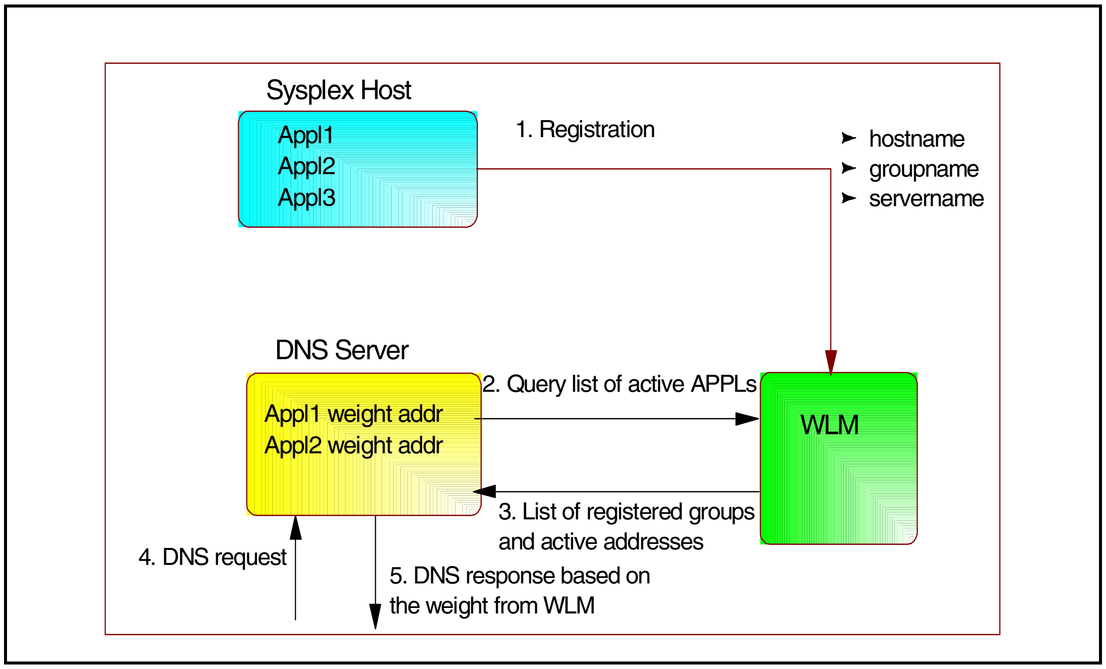


Figure 10-26 WLM and name server working together

If you are familiar with DNS, you might have already noticed that we appear to have crossed a boundary: an application registers with WLM, providing its service name or identification, and then the name server picks up this information and creates a host name entry. The name server dynamically maps an application (actually the group name representing the application) within the sysplex to a host address. While this might be confusing at first, it makes great sense.

If an application goes down, either WLM will be notified by a deregistration command, or else it will detect automatically that the address space has terminated. Then WLM will remove the entry from its tables. When the name server next queries WLM, it will no longer be given that application, group, and hostname. Since we can have multiple applications within a single group, another request for the same group will still succeed if another application registered with the same group is active within the sysplex.

10.3.2 Data returned by the name server

When the sysplex DNS returns information to a DNS requesting data about WLM resources, the sysplex DNS returns a time-to-live (TTL) of 0 so that the local DNS does not cache the results. However, some resolver and name server implementations do not honor a zero TTL, thus reducing the effect of connection optimization during the time they preserve knowledge of cached resources. If you find that there are too many queries on the network for sysplex resources and you wish to choose reduced network traffic over completely optimized connections, you may start the DNS/WLM with a longer TTL value by overriding the default of 0 with the `-l` option.

10.3.3 WLM weights

The DNS/WLM queries the WLM every 60 seconds by default for information regarding resource usage (weights) and available resource addresses. Weights are reflected in the server entries and represent available capacity. The highest weight possible for a server is 64, which indicates the highest capacity available.

The weight 1 of a resource is only visible in a debug trace of the DNS as you see can see in Figure 10-27:

```
Server info from WLM follows for group, TCPIP, with 3 entries:
  Server # 1: Netid = MVS03A, Server = TCPIPA, Weight = 21, Num_addrs = 3
   [172.16.250.3], [9.24.104.113], [9.24.105.126],      1
   host_name = MVS03A
  Server # 2: Netid = MVS28A, Server = TCPIPA, Weight = 21, Num_addrs = 3
   [172.16.252.28], [9.24.104.42], [9.24.105.74],       1
   host_name = MVS28A
  Server # 3: Netid = MVS39A, Server = TCPIPA, Weight = 21, Num_addrs = 2
   [172.16.232.39], [9.24.104.149],                     1
   host_name = MVS39A
 End of WLM Server info
```

Figure 10-27 Partial output of debug trace with WLM weight for MVS images

10.3.4 Static addresses versus registered addresses

The static addresses are those that are defined to the DNS in the sysplex (cluster) domain file. The registered addresses are those that have been defined and are active within the TCP/IP protocol stack. In response to queries, DNS/WLM sends a list of available addresses comprised of the intersection of active addresses registered to the WLM and active addresses that have been defined to the DNS zone files, as shown in Figure 10-28. The idea is to present a list of addresses that are reachable by any host that needs to know. If you have VIPAs configured in your TCP/IP stacks, only VIPA addresses should be statically defined in DNS data files. For a TCP/IP stack within the sysplex, only the first 15 addresses listed in the HOME statement of TCPIP.PROFILE will be registered to WLM (and passed on to the name server when it requests the information). If an interface is not active, its address will not be forwarded to WLM. When the name server receives these active addresses from WLM, it will accept only the addresses that have matching address listed in its data file. This requirement for statically defined addresses ensures full administrative control over the workload distribution.

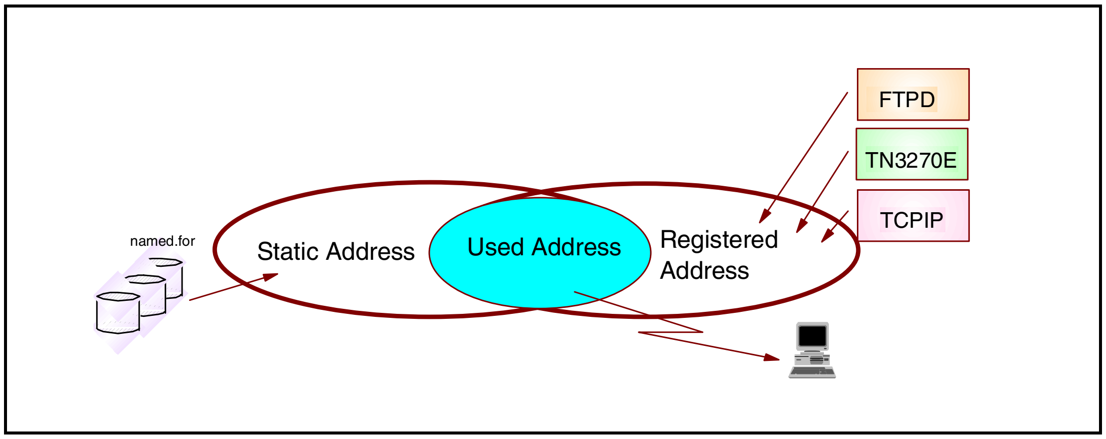


Figure 10-28 DNS addresses used in WLM distribution

The emphasis should be on the words *reachable addresses*. Whether you use static or dynamic routing protocols, you must ensure that any address returned in response to a DNS query can be reached. If you are running static routing in your network without a comprehensive set of static routing definitions on the appropriate platforms, many such addresses could be unreachable. A DNS extraction might present a list of addresses, some of which would not be reachable by every host in the network.

10.3.5 Benefits of DNS/WLM workload distribution

Since the target TCP/IP stack is chosen by the DNS server using the workload information provided by WLM, the workload is balanced in a sysplex based on the current load and system capacity.

Clients use the sysplex name as a server's hostname. In case of TCP/IP stack failure, the connection can be re-established to an appropriate surviving TCP/IP stack within the sysplex.

In general, the benefits of DNS/WLM include:

- Distributes connections in a sysplex based on current load and capacity.
- Distributes load across adapters on a single host.
- Dynamically avoids crashed hosts and servers.
- Dynamically avoids crashed TCP/IP stacks when using sysplex name.
- Highly scalable - new servers may be added without DNS administration.
- Inexpensive to deploy - uses existing technology. No special software/hardware is required.
- Provides for high performance since the distribution is done during hostname resolution.

10.3.6 DNS/WLM limitations

Most of the DNS/WLM solution's limitations arise from its inherent dependence on the Domain Name System. The following lists some of these drawbacks:

- To take advantage of DNS/WLM connection optimization, the clients must be using DNS to resolve addresses.
- Additionally, the DNS server must be implemented within the sysplex. Further, the dynamic naming structure may require client re-configuration.

- The DNS/WLM solution is not applicable to all applications, since application software support is required.
- The DNS/WLM implementation does not distinguish among multiple servers on the same host but using a different port.
- If caching is enabled at other name servers or at hosts and these name servers or hosts ignore the TTL value, full connection optimization is defeated.
- The DNS/WLM can optimize connections only within a single sysplex.
- DNS/WLM is intended primarily for long-lived connections. Although short-lived connections do exploit the potential of DNS/WLM, the added network traffic they generate may outweigh the benefits.

10.3.7 Application and stack registration to WLM

In order for Workload Manager to become aware of an application, the application must register with WLM. There is an assembler macro and a C function available for doing this (IWMSRSRG and IWMDNREG, respectively). Although they have different names, the C function is just a wrapper to call IWMSRSRG. When an application registers, the following information must be passed to WLM:

Group (cluster) name - A generic name used to represent a group of applications running on the sysplex. This can be considered the name of the *service* provided.

Server name - The name of the application running on that particular host in the sysplex. Each application in a group must register with a different server name. Essentially, this is the name of the application instance providing the service.

Hostname - The TCP/IP hostname of the stack associated with the application (server). This can be obtained by issuing the gethostname() call.

When the name server requests a list of registered applications from WLM, the above information is returned for each one, along with a list of active addresses associated with the hostname; that is, all the interfaces that have been successfully activated and have an address assigned via the HOME statement in the TCP/IP profile.

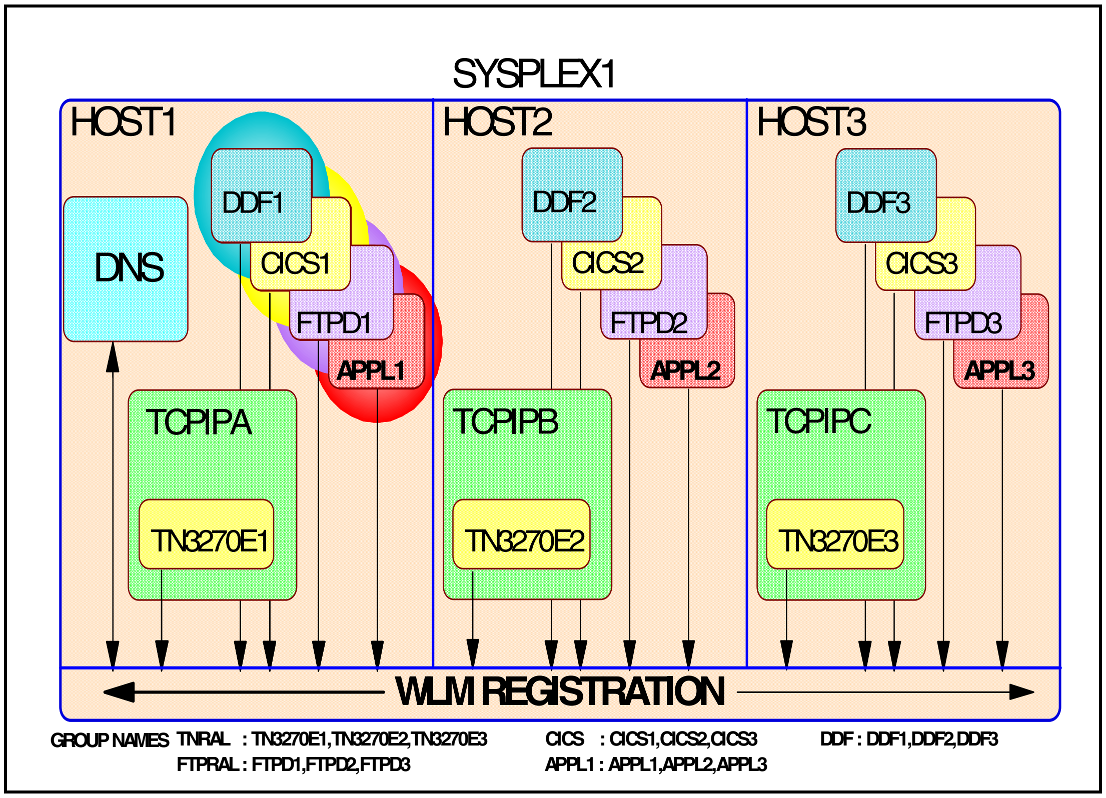


Figure 10-29 Application and Stack registration to WLM

10.3.8 DNS/WLM registration

If you plan to use the connection optimization features of the BIND DNS server that is exploiting WLM, then you need to be aware of several additions to your TCP/IP Profile data set:

```
IPCONFig
  SYSPLEXRouting  1
```

SYSPLEXRouting 1 indicates that this CS for z/OS IP stack participates in a sysplex and should notify the Workload Manager (WLM) of any changes in interface definitions or statuses. This statement allows the stack to register itself and its interfaces with the WLM for connection optimization purposes. SYSPLEXRouting is part of the IPCONFig statements of the PROFILE.TCPIP. If you want Telnet to register with WLM, you need to add WLMClustername and ENDWLMClustername 2 to the TELNETPARMS section of your profile coding:

```
TELNETPARMS
   WLMClustername 2
       TELNET     3
       TN3270     3
       TN3270E    3
       TN03A      3
   ENDWLMClustername  2
;
ENDTELNETPARMS
```

Imbedded between WLMC and ENDWLMC you may specify the names that you would like TELNET or TN3270(E) to go by when they register with the WLM. TELNET **3** and the other names in the WLMC list are known as *group names*. They represent a cluster of equivalent server names in a sysplex environment. The TCP/IP stack and the Telnet server are registered at stack startup if the appropriate definitions have been placed in the PROFILE.TCPIP. Reregistration occurs after every **OBEYFILE** command. You may deregister by stopping the stack or by issuing an **OBEYFILE** command against a PROFILE that has coded NOSYSPLEXrouting or by changing the specifications in the PROFILE between WLMC and ENDWLMC.

The Telnet server can also be deregistered with either of the following commands:

```
V TCPIP,procname,TELNET,QUIESCE
V TCPIP,procname,TELNET,STOP
```

A reregistration is accomplished with:

```
V TCPIP,procname,TELNET,RESUME
```

You may specify up to 16 group names for a TN3270 server.

Other applications besides Telnet can be registered with WLM. A C-interface provides the function to perform this task; an assembler macro (IWMSRSRG) also performs the registration function. Application deregistration capability is also documented.

The best documentation on how to code the WLM registration is in the *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

For FTP, you specify a group name in WLMCLUSTERNAME of the FTP.DATA file. (See Chapter 6, "File Transfer Protocol (FTP)" on page 123.)You may specify up to 16 group names for an FTP server. If you want to change the registration, you have to stop and restart the server.

For CICS, you specify the group names in the LISTENER definition.You may specify up to three group names for a CICS listener. If you want to change the registration, you have to stop and restart the listener.

With regard to the Telnet function itself, the group names by themselves are meaningless. That is, Telnet does not necessarily connect a user with line mode Telnet; that is a function of the Telnet negotiation that occurs when Telnet is requested. Nor does TN3270E by itself mean that the TN3270E server will be invoked; again, this is a function of the client/server Telnet negotiation prior to connection establishment. Nevertheless, we have chosen these names as group names in our WLMCluster definitions just to make it obvious that we are referring to the registration of Telnet group names. A name like FRED would have worked just as well for registration purposes, although it might not have been intuitively obvious what type of server application it represented.

So you may wonder why we chose different group names for the TN3270 or Telnet function. Functionally each group name is not equivalent to every other group name. TN3270 and TN3270E represent group names that allow users to reach the same applications on all four MVS images. TN is a group name that allows users to reach applications that run only on SC63 and SC64. Telnet is a group name that allows users to access only one of the four MVS images; that MVS image, SC64, runs an application that is not available on any of the other three MVS images. If a user entered an incorrect group name, he could be connected with a Telnet that will not allow him to access the application he needs, whether it be directly or indirectly via VTAM cross-domain, cross-network, or APPN services. Figure 10-30 depicts this scenario.

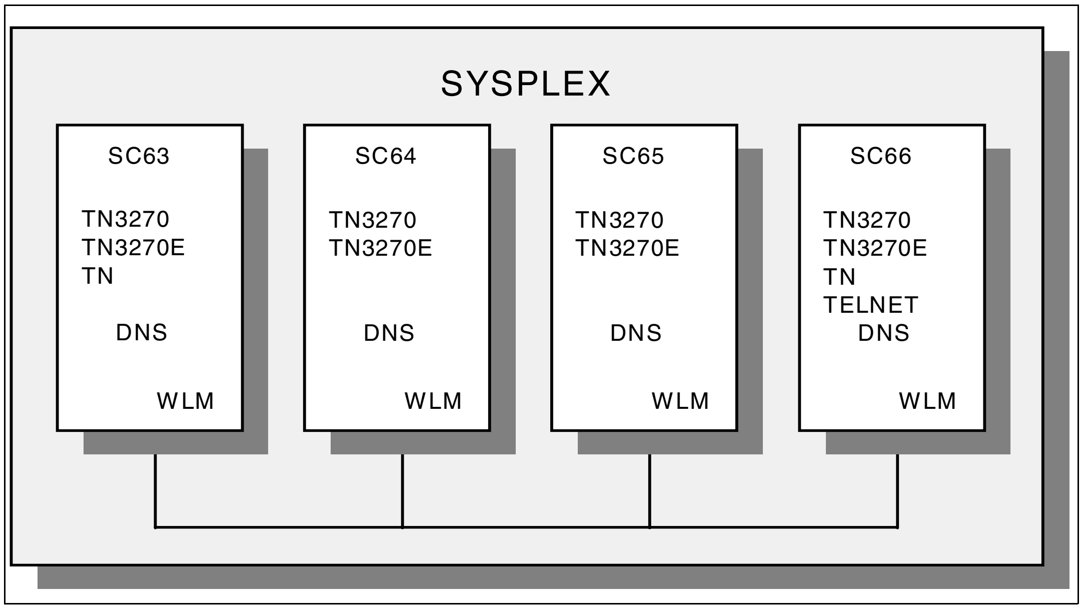


Figure 10-30 Telnet distribution in a sysplex

Telnet clients will be told to issue any of the following commands to reach a Telnet server:

1. `telnet telnet.ralplex1`

   (no dot following ralplex1) so that the fully qualified name can be resolved appropriately
2. `tn3270 tn3270.ralplex1.zos12.ral.ibm.com`
3. `tn3270 tn03a`

The format of the command the end users enter will differ depending on the setting of their TCPIP.DATA information and on the coding at their DNS.

10.3.9 DNS/WLM registration results

Unfortunately there is no generic query available to determine from z/OS WLM the names of resources registered with it. Some applications issue messages about a successful registration:

```
BPXF024I (TCPIPB) Mar  3 17:02:09 ftpd 33554435: EZYFT57I FTP 140
registering with WLM as group = FTPSRV host = SC63
```

Figure 10-31 FTPD successful registration

Other applications have commands available to show the group names registered with WLM:

```
D TCPIP,T03ATCP,T,WLM
EZZ6067I TELNET WLM DISPLAY 204
 PORT
  NUM   WLM CLUSTER NAME          STATUS
-----  ------------------  ---------------------
   23  TELNET              Registered
   23  TN3270              Registered
   23  TN3270E             Registered
3 OF 3 RECORDS DISPLAYED
```

Figure 10-32   Display Telnet group names registration with WLM

If there is no command or message available to check the registration with WLM, you can use the nslookup or any other function that resolves host names to verify the results. Another possibility to be certain if an application has registered is to dump the DNS database. Please review 2.5.6, “Dumping the DNS server cache” on page 36 for a detailed explanation.

10.3.10 Data returned by the name server

When the sysplex DNS returns information to a DNS requesting data about WLM resources, the sysplex DNS returns a Time-to-Live (TTL) of 0 so that the local DNS does not cache the results. However, some resolver and name server implementations do not honor a zero TTL, thus reducing the effect of connection optimization during the time they preserve knowledge of cached resources. If you find that there are too many queries on the network for sysplex resources and you wish to choose reduced network traffic over completely optimized connections, you may start the DNS/WLM with a longer TTL value by overriding the default of 0 with the `-l` option.

10.3.11 Recommendation for DNS/WLM address definition

Based on the discussion so far, we have come to the following conclusions:

1. If you have implemented dynamic routing protocols in your network, limit your statically defined addresses in the sysplex subdomain to the VIPA address and use SOURCEVIPA.
2. If you use dynamic routing protocols throughout your network, but you do not use VIPA at the z/OS IP host, you may still successfully use multiple addresses in your name server forward zone files.
3. If you use static routes in your network, limit the statically defined name server addresses to those that are reachable throughout the network.

Some of the traces you see in this chapter and in the appendixes show that we did not always follow our own recommendations. Expecting dynamic routing to be in place by the time we were ready to test the BIND DNS/WLM, we overdefined our statically defined addresses in the zone files. With static routing still in place, we ended up with some unreachable addresses for a while. When you review our static name server definitions in be aware that they are not necessarily optimal.

10.3.12 Round-robin technique and addresses returned

If the intersection of active addresses and DNS-defined addresses yields multiple potential addresses for a query and if all systems have the same weights, you would expect to see a rotation of the addresses being offered a client. Yet you may test with DNS/WLM and never perceive the phenomenon. This is due to the default for the -t option on the DNS startup. -t represents the amount of time between queries to the WLM about sysplex names, addresses, and weights. When the time specified in -t (default of 60 seconds) expires, DNS resets its list of potential addresses to the order specified in the DNS definition sequence. The default of 60 seconds has been deemed optimal for a production system, because weights can change rapidly; DNS should refresh its knowledge of weights frequently. A great number of connections occur in a production environment in those 60 seconds, and these will receive the benefits of the round-robin address offers. If you set -t to a value greater than 60 seconds, you defeat the purpose of connection balancing by overriding refreshed knowledge about sysplex weights.

However, if you want to see the round-robin effect in action during testing you can temporarily set the value higher than 60 seconds. Multiple (o)nslookups in rapid succession should deliver a list of addresses that rotate with each command.

Comparison of addresses returned

When we had only one active interface eligible for DNS/WLM selection, nslookup from two different workstations on the 9.24.104.0 network (subnet mask = 255.255.255.0) delivered a single entry (Figure 10-33):

```
C:\>nslookup ftpsrv.ralplex1 mvs03
Server:  mvs03-en1.ralplex1.itso.ral.ibm.com
Address:  9.24.105.126
Name:    ftpsrv.ralplex1.itso.ral.ibm.com
Address:  9.24.105.126

C:\>nslookup ftpsrv.ralplex1
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108
Name:    ftpsrv.ralplex1.itso.ral.ibm.com
Address:  9.24.105.126
```

Figure 10-33 nslookup response from different servers

When we had multiple active interfaces eligible for DNS/WLM selection, nslookup delivered the result shown in Figure 10-34:

```
OS2 C:\>nslookup tn3270
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108
Non-authoritative answer:
Name:    tn3270.ralplex1.itso.ral.ibm.com
Addresses:  192.168.251.1, 192.168.252.1, 192.168.236.1, 192.168.221.20
          192.168.109.3, 9.24.105.126
Aliases:  tn3270.itso.ral.ibm.com
```

Figure 10-34 nslookup with multiple active interfaces

All six addresses you see in each of the two screens above are active. They are all reachable addresses if your network employs dynamic routing and has the appropriate connectivity. They are unreachable if your static definitions are not comprehensive.

10.3.13 DNS/WLM TCPDATA considerations

The TCPDATA file (or resolver file) is used by clients to determine, among other things, the stack name they should have affinity to and the domain name that will automatically be appended to their name-based queries. The fully qualified name for mvs03 could end up being mvs03.itso.ral.ibm.com. or mvs03.ralplex1.itso.ral.ibm.com. even if all you entered at the client was ping mvs03. This depends on the resolver file your client is using.

If you are the DNS server for a sysplex domain, you have a decision to make about how you designate the domain name for the TSO or shell client. If you leave the domain name as itso.ral.ibm.com., then every time your client needs to have for example ftpsrv converted into a fully qualified name, it will be resolved into ftpsrv.itso.ral.ibm.com. You may not find the group name ftpsrv under these conditions; however, you would find the thousands of other resources in the domain itso.ral.ibm.com and served by another name server very easily by allowing the default domain to be appended.

Your network users who need to get to the few resources managed by the sysplex domain simply need to change those requests to something like this: **`ftp ftpsrv.ralplex1`**, ensuring that they do not append a period to the request. (The period or dot would indicate that the fully qualified name has been specified and the current domain name should not be appended.) Their client resolver process will expand the two-part name into the fully qualified ftpsrv.ralplex1.itso.ral.ibm.com. To avoid your network users having to specify the long sysplex names, you can simply code CNAME records for your sysplex resources.

On the other hand, if you make the domain name ralplex1.itso.ral.ibm.com, then every time your TSO client needs to find the group name ftpsrv, it will be correctly resolved into ftpsrv.ralplex1.itso.ral.ibm.com. However, to reach the thousands of resources by name that are actually in the itso.ral.ibm.com domain, the TSO or shell client would have to specify the fully qualified name to begin with or would have to rely on your CNAME coding in the name server. (The CNAME coding would spell out the fully qualified name of the resource.)

You have probably figured out that the issue of what domain name to put into the TCPDATA file is one of degree: how many host-based clients are there versus workstation clients? If there are few z/OS-based clients trying to reach resources that are based mostly in the sysplex, then you might decide to use the sysplex subdomain as your TCPDATA domain. If the same clients are trying to reach resources in a completely different domain, then you might decide to use the domain name that represents the greater number of resources.

In our network, we left the TCPDATA file at the host with a domain of itso.ral.ibm.com. for the clients to use and CNAME records to point to the sysplex resources.

10.3.14 Client/server affinity

Some client/server applications require that the client connects to the same server instance after an interruption. This is achieved in the following way:

1. The server uses the new ioctl() function SIOCGSPLXFQDN to get its fully qualified name from the TCP/IP stack.
2. After the connection has been established using only the group name or the sysplex name the server sends its fully qualified name (server_instance.groupname.sysplex_subdomain.domain) to the client.
3. After an interruption, the client uses this fully qualified name to make sure he connects to the same server he was connected to before the interruption.

To enable this function the following definition has to be added to the sysplex name servers loopback file:

```
127.0.0.128 IN PTR ralplex1.itso.ral.ibm.com; Sysplex Loopback Address (SLA)
```

The loopback address range 127.0.0.128-127.0.0.255 has been reserved by IBM for this purpose.

10.3.15 Configuring the DNS server for WLM

In this scenario MVS03 and MVS28 serve as domain name systems in two domains: the parent domain of itso.ral.ibm.com and the sysplex subdomain of ralplex1.itso.ral.ibm.com.

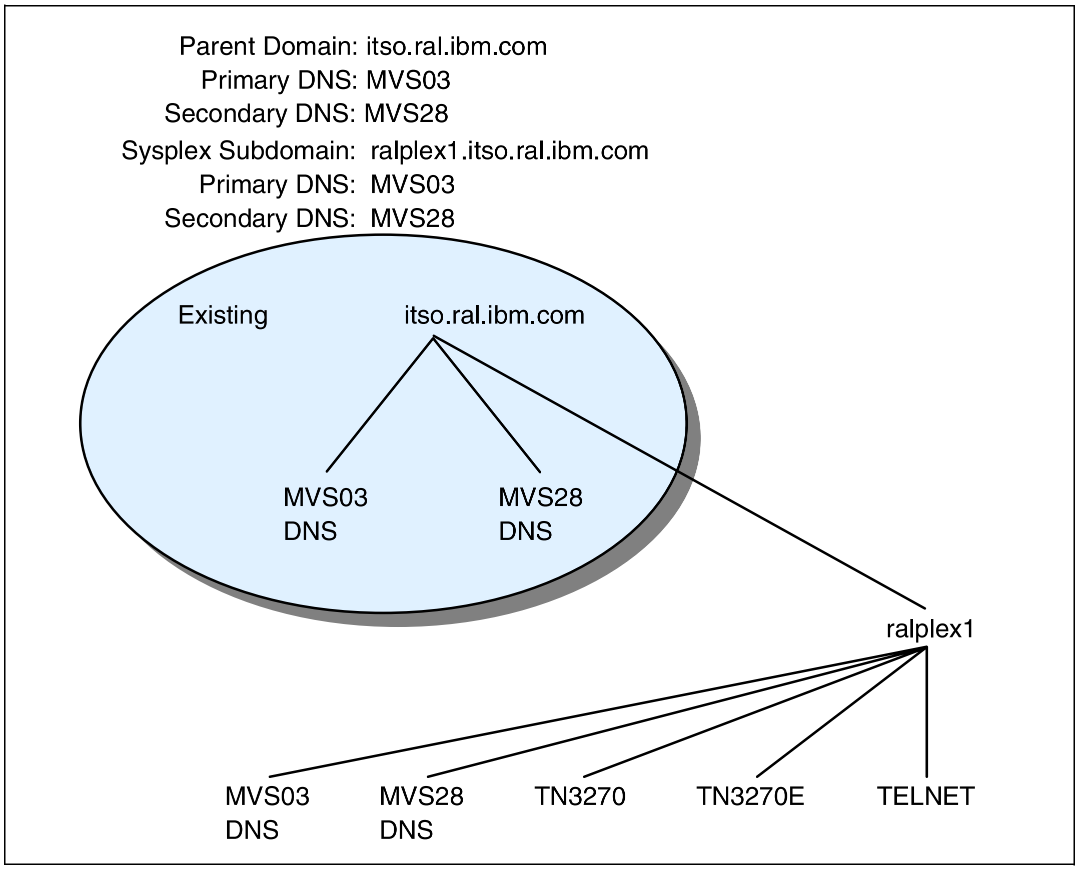


Figure 10-35 DNS/WLM with combined parent and sysplex domains

MVS03 DNBoot file for MVS03: DNS/WLMS/WLM boot file

```
;    /etc/named.boot.wlm2     based on    /etc/named.boot.wlm1
;
;  TYPE        DOMAIN                        FILE OR HOST
directory      /etc/dnsdata
;
; (Following for sysplex name server as both parent + sysplex DNS)
primary      ralplex1.itso.ral.ibm.com  3  named.for.wlm1  cluster 5
primary      itso.ral.ibm.com           4  named.for2
primary      251.168.192.in-addr.arpa        named.wlm2192.168.251
primary      252.168.192.in-addr.arpa        named.wlm2192.168.252
primary      236.168.192.in-addr.arpa        named.wlm2192.168.236
primary      235.168.192.in-addr.arpa        named.wlm2192.168.235
primary      105.24.9.in-addr.arpa           named.wlm29.24.105
primary      104.24.9.in-addr.arpa           named.wlm29.24.104
primary      221.168.192.in-addr.arpa        named.wlm2192.168.221
primary      109.168.192.in-addr.arpa        named.wlm2192.168.109
primary      0.0.127.in-addr.arpa            named.lbk.wlm2
cache       .                              named.ca
forwarders   9.24.104.108
options query-log
```

Figure 10-36 Boot file for MVS03: DNS/WLM

As you would expect, two zone files are listed: one for ralplex1.itso.ral.ibm.com **3** and one for itso.ral.ibm.com **4**. The sysplex subdomain reference uses the `cluster` keyword **5**.

MVS03 DNS/WLM forward files

Here you see the two domain files referenced in the boot file in Figure 10-36.

```
;  /etc/dnsdata/named.for.wlm1 for MVS03 and SYSPLEX
 $ORIGIN itso.ral.ibm.com.
 ralplex1 IN     SOA   8 mvs03.ralplex1.itso.ral.ibm.com. gdente@ralplex1.i
                       5 10800   3600   604800   86400)
 $ORIGIN ralplex1.itso.ral.ibm.com.
                  IN     NS    mvs03
                  IN     NS    mvs28
; OWNER    CLASS   TYPE    RECORD DATA
 localhost    IN      A       127.0.0.1
 mvs03        IN      A  7    9.24.105.126  ;  Ethernet LCS
              IN      A  6    192.168.251.1 ;  Via address
              IN      A       192.168.252.1 ;  SAMEHOST to MVS03C
              IN      A       192.168.236.1 ;  RAS XCF to 28
              IN      A       192.168.235.3 ;  MPC to 25
              IN      A       192.168.221.20;  LICP03 Connection
              IN      A       192.168.109.3 ;  LICCP25 Connection
 mvs03c       IN      A       192.168.252.2 ;  SAMEHOST at MVS03C
 mvs28        IN      A       9.24.105.75   ;  Ethernet LCS at MVS28
              IN      A       192.168.236.2 ;  RAS XCF to 03
              IN      A       192.168.221.24;  LICP03 Connection
              IN      A       192.168.109.1 ;  LICCP25 Connection
```

Figure 10-37 DNS/WLM forward file

As previously noted, this forward file could be trimmed down significantly so that only the VIPA 6 and/or the generally available LAN connection (9.24.105.126) 7 are available as statically defined addresses. As long as full dynamic routing is in place and no firewall interference prevents all addresses from being used, you could leave the full cluster domain file as you see it here. 8 points out that the SOA record for the ralplex1 subdomain is managed using the fully qualified ralplex1 name for mvs03.

The following figures show the remaining files configured in the scenario. The reverse file is shown in Figure 10-38:

```
;  /etc/dnsdata/named.wlm29.24.105 for MVS03-en1 9.24.105.126
;$ORIGIN 105.24.9.in-addr.arpa.
 @    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.ral.ibm.com (
                       2 10800   3600   604800   86400)
      IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
      IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;     LCS entry at MVS03
 126  IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.  1
 126  IN     PTR   mvs03.itso.ral.ibm.com.           2
;     LCS entry at MVS28
 75   IN     PTR   mvs28.ralplex1.itso.ral.ibm.com.  3
 75   IN     PTR   mvs28.itso.ral.ibm.com.           4
```

Figure 10-38 MVS03 DNS/WLM reverse file (in-addr.arpa)

The named addresses belong to two domains and are so identified: 1, 2, 3, 4.

```
;  /etc/dnsdata/named.lbk.wlm2
;
 0.0.127.in-addr.arpa. IN SOA mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.ibm.
                                    5
                                    10800
                                    3600
                                    604800
                                    86400 )
 0.0.127.in-addr.arpa.   IN    NS    mvs03.ralplex1.itso.ral.ibm.com. 5
 0.0.127.in-addr.arpa.   IN    NS    mvs03.itso.ral.ibm.com.          6
 1.0.0.127.in-addr.arpa. IN    PTR   localhost.
```

Figure 10-39 MVS03 DNS/WLM loopback file

Again you see in Figure 10-39 that the loopback address has been associated with both the parent domain and the sysplex subdomain (5, 6).

MVS03 console displays

You will notice in the console displays of the files loading that the cluster file does not appear (Figure 10-40 on page 354) as you might expect in message EZZ6540I.

```
EZZ6699I name server starting. @(#) ddns/ns/ns_main.c, dns_ns, dns_r1.1 1
EZZ6701I named established affinity with 'T03ATCP'
EZZ6540I Static primary zone '251.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '252.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '236.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '235.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '105.24.9.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '104.24.9.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '221.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '109.168.192.in-addr.arpa' loaded (serial 2)
EZZ6540I Static primary zone '0.0.127.in-addr.arpa' loaded (serial 2)
EZZ6540I Static cache zone '' loaded (serial 0)
```

Figure 10-40 Console or SYSLOGD.LOG messages: DNS/WLM loading

This omission is by design. The sysplex domain file is refreshed every 60 seconds, so the decision was made to suppress the load message for any cluster file. Nevertheless, a dump of the DNS server database (OE Signal #2, or SIGINT) will show that the information has been loaded; in addition, if you set debugging at, for example, level 11 (-d 11), the output in named.run also proves that the data has been loaded.

```
;    /etc/named.boot.wlm2     based on    /etc/named.boot
;  TYPE        DOMAIN                       FILE OR HOST
 directory      /etc/dnsdata
; (Following for sysplex name server as both parent/sysplex name server)
 secondary    ralplex1.itso.ral.ibm.com 192.168.236.1  named.for.wlm1 cluster 1
 secondary    itso.ral.ibm.com          192.168.236.1  named.for2 2
 secondary    251.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.251
 secondary    252.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.252
 secondary    236.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.236
 secondary    235.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.235
 secondary    105.24.9.in-addr.arpa     192.168.236.1  named.wlm29.24.105
 secondary    104.24.9.in-addr.arpa     192.168.236.1  named.wlm29.24.104
 secondary    221.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.221
 secondary    109.168.192.in-addr.arpa  192.168.236.1  named.wlm2192.168.109
 primary      0.0.127.in-addr.arpa      named.lbk.wlm2
 cache     .                        named.ca
 forwarders   9.24.104.108
 options query-log
```

Figure 10-41 MVS28 boot file (two forward files)

Again you see in Figure 10-41 that the secondary is to participate in sysplex connection optimization (keyword `cluster` 1). Also note that two domain files will be retrieved in a zone transfer from the primary name server: one for the sysplex domain (1) and one for the parent domain (2).

```
;  /etc/dnsdata/named.lbk.wlm2
;                                    3
0.0.127.in-addr.arpa. IN SOA mvs28.ralplex1.itso.ral.ibm.com. mvs28.ralp
                                   5
                                   10800
                                   3600
                                   604800
                                   86400 )
0.0.127.in-addr.arpa.   IN    NS    mvs28.ralplex1.itso.ral.ibm.com. 4
0.0.127.in-addr.arpa.   IN    NS    mvs28.itso.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN    PTR   localhost.
```

Figure 10-42 MVS28 loopback file

The only changes we make to the loopback file at MVS28 (Figure 10-42) are:

- To identify MVS28 as a member of the ralplex1.itso.ral.ibm.com domain **3**
- To add a loopback record showing the sysplex domain **4**

10.4 Setting up a BIND 9-based Domain Name Server

CS for z/OS V1R2 IP provides a port of the BIND-based version 9 name server to the zSeries platform. It is known as the BIND 9-based name server and it is different from the BIND 4.9.3-based name server that existed in previous CS for OS/390 releases. In a multiple stack (Common INET) environment, the BIND 9 name server is a generic server which does not have stack affinity. This is in contrast to the BIND 4.9.3 name server which does have stack affinity.

In general, the BIND 9 name server may perform slower than the BIND 4.9.3 name server for small zones on simple query/response operations, or you may achieve equivalent throughput as a BIND 4.9.3 name server but with higher CPU consumption, depending on if your system is already CPU constrained. The BIND 9 name server contains extra overhead to support multi-threading, and DNSSEC. For small zones, the extra overhead for multi-threading may cause a performance disadvantage. On the other hand, the multi-threading may improve performance for large zones.

BIND 4.9.3 name servers are unable to answer queries for a period of time during zone transfers. The larger the zone, the more noticeable this may become. Because of the multi-threading, BIND 9 name servers, in contrast, are able to answer queries during zone transfers. Furthermore, BIND 9 name servers are capable of Incremental Zone Transfers, while BIND 4.9.3 name servers are not. Incremental Zone Transfer allows only the changedl information in a zone to be sent to slave name servers instead of the entire zone.

The following steps will show you what is required to implement and run a basic master (primary) domain name server environment, using BIND 9-based DNS:

1. 10.4.1, “Migrating from a BIND V4.9.3 DNS environment” on page 356
2. 10.4.2, “Define your zone” on page 356
3. 10.4.3, “Create a configuration file for your environment (named.conf)” on page 357
4. 10.4.4, “Specify port ownership” on page 359
5. 10.4.5, “Update the name server start procedure (optional)” on page 359
6. 10.4.6, “Create the domain data files” on page 360

7. 10.4.7, “Create the loopback file” on page 364
8. 10.4.8, “Create the cache file (hints file)” on page 365
9. Configure logging
10. Start the name server
11. Verifying if name server is working as expected
12. Define the backup (secondary) domain name server

10.4.1 Migrating from a BIND V4.9.3 DNS environment

The named.boot file in the BIND 4.9.3-based DNS server is replaced with the named.conf file for BIND 9-based DNS servers. Along with the name change is a change in syntax. A migration tool (initiated by specifying a new UNIX command called downstater) is supplied to convert the files from the BIND 4.9.3-based DNS format to the newer BIND 9-based DNS format. The migration tool will not do backwards conversion from the newer version to the older version. In our scenario, since we want to show you how to configure each file we will assume this is a new DNS implementation.

10.4.2 Define your zone

The first thing to do when you need to implement a domain name server in your network environment is to plan and define what you are suppose to control, the boundaries of your domain, which servers are going to control your domain, and how you are going to interact with external domains. By defining these issues, you will be able to draw a good picture of the domain you are going to control, and use it as input to go through all the necessary steps to built your domain name server. A good way to get these definitions is to answer a few questions and draw the scenario you are willing to create.

Our scenario will be defined based on the following definitions

1. Our domain will be named zos12 and it will be a sub domain of ral.ibm.com
2. The host we are going to use as primary server is 9.12.6.68 (sc63)
3. The host we are going to use as secondary server is 9.12.6.64 (sc64)
4. In our scenario all unresolved queries will be forwarded to 9.24.104.108

Based on the answers provided, we are able now to draw our scenario, that, in our case would look like the one showed in Figure 10-43:

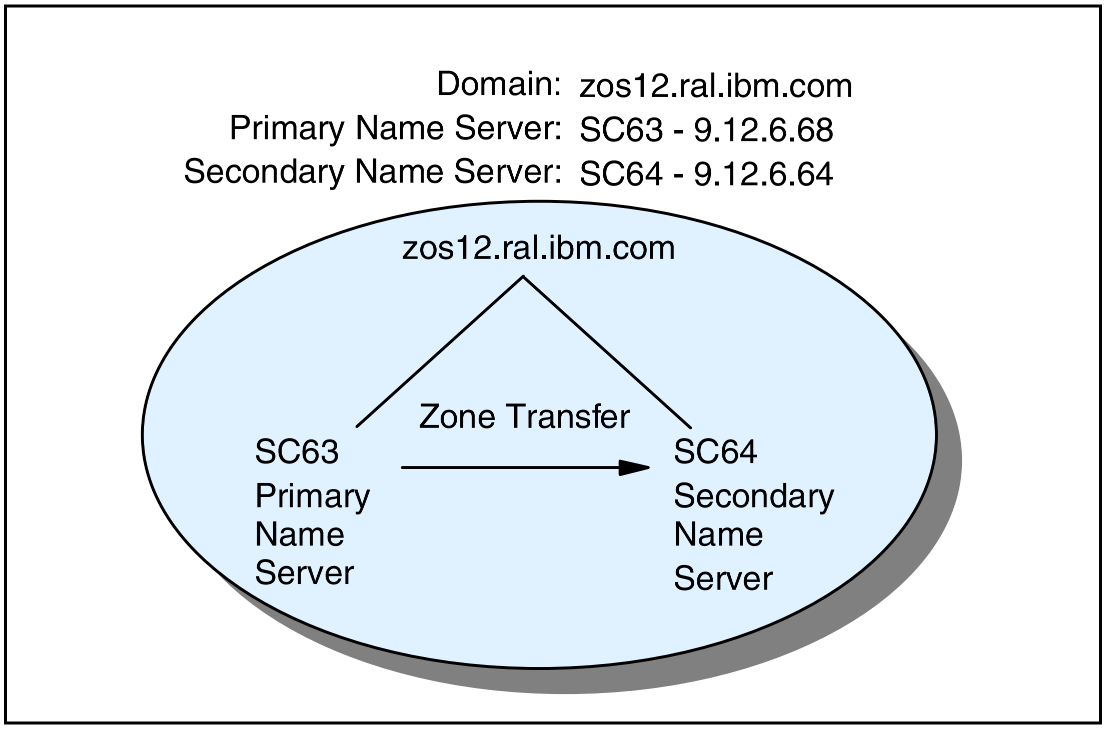

Domain: zos12.ral.ibm.com
Primary Name Server: SC63 - 9.12.6.68
Secondary Name Server: SC64 - 9.12.6.64
zos12.ral.ibm.com
Zone Transfer
SC63 Primary Name Server
SC64 Secondary Name Server

Figure 10-43 DNS basic implementation scenario

The domain is zos12.ral.ibm.com. There is one primary name server platform: SC63. The secondary name server SC64 obtains its information from zone transfers with SC63.

10.4.3 Create a configuration file for your environment (named.conf)

The boot file initializes the name server environment and points to the individual name server definition files and to the options that the name server will provide for each zone it supports. The boot file is referenced in the start procedure used from the MVS console or in the UNIX System Services shell command used to start the domain name system server. If no boot file is referenced in the start procedure or command, the default of /etc/named.boot is assumed.

A BIND 9 configuration consists of statements and comments. Statements end with a semicolon (;). Statements and comments are the only elements that can appear without enclosing braces ([]). Many statements contain a block of substatements, which are also terminated with a semicolon. Figure 10-44 shows the contents of the named.boot files being used in our scenario.

```
 #
 #       conf file for BIND9-based Name Server
 #
 options {                                         1
    directory "/etc/octavio/dnsdata";
};
 logging {                                         2
    category "queries" {
       default_syslog;
};
};
 zone "zos12.ral.ibm.com" in {                     3
    type master;
    file "zos12.for.v9";
};
 zone "6.12.9.in-addr.arpa" in {                   3
    type master;
    file "zos12.rev.v9";
};
 zone "0.0.127.in-addr.arpa" in {                  3
    type master;
    file "zos12.lbk.v9";
};

 zone "." in {                                     4
    type hint;
    file "zos12.ca.v9";
};
 zone "itso.ral.ibm.com" in {                      5
    type forward;
    forward only;
    forwarders {
 9.24.104.108;}
};
```

Figure 10-44 named.conf contents

The contents of this boot file specify the following statements:

1 The *options* statement controls global server configuration options and sets defaults for other statements. It can appear only once in a configuration file. In our sample, we defined only the *directory* parameter to identify the working directory of the server. This will be the default location for all server files identified in this configuration file.

2 The *logging* statement configure the logging options to be used by this name server configuration. In our sample we defined that all queries received will be logged in the syslog daemon default system log.

3 A *zone* statement, associated with the *type* parameter identifies the zone and how this server is related to it. This record defines that this server has a master copy of the data related to the zone and will be able to provide authoritative answers for it. The *file* parameter identifies the file to read for the name-to-IP address, IP address-to-name, and loopback mappings.

4 This *zone* statement has a *type* parameter *hint* defined identifies the root level domain, identified by a dot (.), and the file in which the IP address of the root DNS servers can be found. The cache file is also known as the hints file.

5 This zone statement has a type of forward and is a way to configure forwarding on a per-domain basis. A zone statement of type forward can contain a forward or forwarders statement, which will apply to queries within the domain given by the zone name. If no forwarders statement is present or an empty list for forwarders is given, then no forwarding will be done for the domain, cancelling the effects of any forwarders in the options statement. Thus if you want to use this type of zone to change the behavior of the global forward option (that is, forward first to, then forward only, or vice versa, but want to use the same servers as set globally) you need to respecify the global forwarders.

10.4.4 Specify port ownership

By default, the name server uses a single port (53) for TCP and UDP sessions. DNS servers can share port 53 if they connect to clients or other servers on different IP addresses. A BIND 9 server can specify IP addresses to listen on and from which to send queries, notifies and zone transfers in its configuration file.

To specify port ownership when using the named start procedure for BIND 9, add the following statements to the PROFILE.TCPIP data set:

```
PORT
  53 TCP NAMED91              1
  53 UDP NAMED91
```

Figure 10-45 Port ownership definition

1 The procedure name is NAMED9.

If you decide to start the Name Server using the z/OS UNIX shell, the port can be generically reserved to UNIX applications, as shown:

```
PORT
  53 TCP OMVS;reserved to z/OS UNIX
  53 UDP OMVS;reserved to z/OS UNIX
```

Figure 10-46 Ports reserved to z/OS UNIX

10.4.5 Update the name server start procedure (optional)

When choosing to start the name server from MVS, create a start procedure. This is not necessary if the name server is started from the z/OS UNIX shell. Move the sample start procedure, SEZAINST(NAMED9), to a recognized PROCLIB. Specify name server parameters and change the data set names as required to suit local configuration. The boot file path (for BIND 9-based DNS) can also be changed as shown in the sample start procedures shown in Figure 10-47 below:

```
//NAMED    PROC C='/etc/octavio/named.conf'
//NAMED    EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/named -c &C '                 1
//*STDENV   DD PATH='/etc/named.env',                                  2
//*            PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
```

Figure 10-47 BIND 9 startup procedure

1 The parameter that identifies the configuration file has been changed to *-c*. The older versions used *-b* to define the same file.

2 Since BIND9 has no stack affinity there is no need to define a STDENV DD card. It is an optional card.

10.4.6 Create the domain data files

The domain data files contain information about a domain, such as the IP addresses and names of the hosts in the domain for which the master name server is authoritative. The forward domain data file contains entries that provide forward mapping (host names-to-IP addresses for each host system in the zone) as well as additional information about system resources. The reverse domain data file contains entries that provide reverse mapping (IP addresses-to-host names)

10.4.6.1 The forward file

The forward files are identified in the /etc/named.boot and contain the name-to-IP address mappings. Every zone must have a separate forward file. If your DNS server services three zones, then you must have three separate forward files. These files are stored in HFS format and their UNIX permission bits must allow name server read/write access. The entries in the forward files are defined using a special format that is defined by RFC1034. These entries are called resource records. All resource records (RR) are written in the file with the following format:

```
name ttl address_class record_type record_data
```

The record_type and record_data fields are the only required fields. The name, ttl and address_class have defaults that may be set if they are not specified. The record_type indicates the type of resource record which defines the record data format. Valid resource record types include, but are not limited to, the following:

Table 10-2 DNS resource record types

DNS Resource Record type	**Record Type description**
SOA	Start Of Authority for the stated Zone
A	Name -to-IP address translation record
SRV	Service-to-Location definition
PTR	IP address-to-Name translation record
NS	Name of the authoritative DNS server for the stated Zone

DNS Resource Record type	Record Type description
CNAME	alias name of a stated host

You also can include in the zone files other records, called master file directives, to define some specific values, as follows:

- *$ORIGIN* directive - sets the domain name that will be appended to any unqualified record as shown in the example below:

```
$ORIGIN itso.ibm.com
sample   CNAME  host01
would be equivalent to:
sample.itso.ibm.com   CNAME host01.ibm.com
```

- *$INCLUDE* directive - Read and process the file filename as if it were included into the file at this point. If origin is specified, the file is processed with $ORIGIN set to that value, otherwise the current $ORIGIN is used. Once the file is read, the origin and current domain revert to the values they were prior to the $INCLUDE. it has the following syntax:

```
$INCLUDE filename [origin] [comment]
```

- *$TTL* directive - Set the default Time To Live (TTL) for subsequent records with undefined TTLs. Valid TTLs are of the range 0-2.147.483.647 seconds. It has the following syntax:

```
$TTL default-ttl [comment]
```

Note: Since BIND 8.2, you must use the $TTL directive to set the default TTL for the zone. If you do not define it, you will receive this message:

```
No default TTL set using SOA minimum instead
```

Figure 10-48 shows the contents of the named.for file. It is referenced in the named.conf file you saw in Figure 10-44 as the primary DNS information for the domain zos12.ral.ibm.com.

```
; Default TTL value
 $TTL 86400                                                               1
 $ORIGIN ral.ibm.com.                                                     2
 zos12              IN      SOA   dns63.zos12 admin.zos12 (               3
             1            ; Serial (incriminated when database is changed  4
             10800        ; Refresh (slave will check every 3 hours       5
             3600         ; Retry  (retry every hour after refresh failure 6
             604800       ; Expire (slave gives up retry after 1 week     7
             86400)       ; Leg. Cache (cache NXDOMAIN/RRSET responses 1 day 8
$ORIGIN zos12.ral.ibm.com.                                                9
                    IN      NS      dns63                                 10
                    IN      NS      dns64
  _http._tcp              SRV  0  0 80   www.zos12.ral.ibm.com.           11
  localhost          IN   A    127.0.0.1                                  12
  dns63              IN   A    9.12.6.68
  dns64              IN   A    9.12.6.64
  host1              IN   A    9.12.6.67
  host2              IN   A    9.12.6.63
  www                IN   A    9.179.147.237
  sc63               IN   A    9.12.6.67                                      13
                     IN   A    9.37.6.68
  mail               IN   CNAME host1                                         14
  ftp                IN   CNAME host2
```

Figure 10-48 Forward master file definitions

1 The *$TTL* directive must be defined and provides the time-to-live value. If *named* finds a $TTL directive it follows TTL semantics defined in RFC 2308, which states that records not explicitly setting a TTL inherit the TTL from the $TTL value. If there is no $TTL set, it follows TTL semantics from RFCs 1035 and 1035, which state that records with no explicit TTL inherit one from the previous record. This implies that to follow RFC 1034/1035 semantics, the SOA RR must set its TTL value. For simplicity, it is recommended that you always specify a $TTL value. This line sets the default TTL for all records to 86400 seconds (one day).

2 This *$ORIGIN* control records defines the higher domain zone where the subdomain itso is being defined. Its used in conjunction with the SOA record that follows to resolve the name of zos12.ral.ibm.com, which will is the domain name of our zone.

3 The *SOA* record specifies the name of the host that has the domain name server authority for the zone. This name is appended to the domain name specified in the previous $ORIGIN record. The SOA record includes a mailbox address of the user who is responsible for the zone: admin@zos12.ral.ibm.com. A left parenthesis signals that everything between here and a succeeding right parenthesis should be considered as belonging to the same resource record, despite record boundaries of the master data set. The number enclosed in parenthesis are parameters used to set different values for the zone, as follows:

4 The *Serial* parameter is used to identify the serial number of the domain database. is referenced by secondary name servers. Increment the number each time you change the file.The secondary name servers determine when to do a zone transfer based on an increment in this value.

5 The *Refresh* parameter is expressed in seconds. A secondary name server that has transferred this zone from the primary name server should not wait more than this number of seconds before it requests a refresh (a full zone transfer) from the primary name server. Before requesting a zone transfer, the secondary name server checks if the value of the serial field for the zone in question has changed or not. If not, a zone transfer is not necessary.

6 The *Retry* parameter is expressed in seconds. If a secondary name server fails to refresh its copy of resource records, it should wait this number of seconds before it retries the refresh from the primary name server.

7 The *Expire* parameter is expressed in seconds. This is the maximum time a secondary name server should consider its copy of resource data valid. If the secondary name server does not succeed with a zone transfer from the primary name server within this amount of time, it should consider its copy of the resource data obsolete, and stop answering queries for this zone.

8 The meaning of the last *SOA* value has changed from BIND v4 to V9. It now represents length of time other servers should cache negative responses from this zone. This line sets that value to 86400 seconds (one day).

9 The control entry *$ORIGIN* appends the string zos12.ral.ibm.com. to all the following host names that do not end with a dot ('.').

10 The *NS* (Name Server) records specify the name servers in the zone. Note that NS records do not distinguish between primary and secondary name servers. You will later see that in our scenario dns64 is a secondary name server that receives zone transfers from dns63.

11 Specifies the location of services (for example, ftp, http, telnet). This record identifies one or more hosts capable of satisfying the service and protocol represented in the name field of the resource record. The name field for these resource records must follow a unique naming convention. The contents of this RR are as follows:

name priority weight port target where:

the *name* field must be specified as Service.Protocol.Name.:

Service is the symbolic name of the desired service, as defined in Assigned Numbers (1) or locally.

Protocol is typically TCP or UDP

Name is the domain this RR refers to.

Priority is a number in the rage of 0-65535 and represents the priority of this target host. A client must attempt to contact the target host with the lowest-numbered priority. Target hosts with the same priority should be tried in pseudo-random order.

Weight is used for a crude connection balancing mechanism. When selecting a target host among those that have the same priority, the chance of trying this one first should be proportional to its weight. The valid range is 0-65535.

Port is the port on this target host of this service. The valid range is 0-65535.

Target is the domain name of the target host. This name must be a canonical name and not an alias. There must be one or more A records for this name. A target of consisting of a dot (.) means that the service is not available at this domain.

In our file, the SRV records specify the location for the 'http' service using the 'tcp' protocol. The first record has a priority of 0, a weight of 0, uses port 80 and the service is provided at host www.itso.ral.ibm.com. The second record has a priority of 10 which is lower, a different port and target. A web client capable of using SRV records requesting a site named `http://itso.ral.ibm.com/`, would be directed to `www.itso.ral.ibm.com` and `www2.mycorp.com`. The client would be responsible for determining which site to connect first based on priority and then on weight.

12 An *A* record is an address record, naming the host and the IP address of that host.

13 You can also use *A* records to define more than one address to a host. If you do not specify a value in the name field, the value from the preceding record will be used.

14 A *CNAME* record is an alias for a host name.

10.4.6.2 The reverse file (in-addr.arpa file)

Figure 10-49 shows the contents of zos12.rev.v9 reverse file. It is referenced in the named.conf file as the primary DNS information for the domain 6.12.9.in-addr.arpa. Note the inverse syntax used for referencing this in-addr.arpa file in Figure 10-44 on page 358. There must be a single reverse file for each IP address grouping.

```
;
 $TTL 86400
 $ORIGIN 12.9.in-addr.arpa.
 6  IN SOA dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
              1  10800  3600 604800 86400)
              IN      NS    dns63.zos12.ral.ibm.com.
              IN      NS    dns64.zos12.ral.ibm.com.
 $ORIGIN 6.12.9.in-addr.arpa.
 68           IN      PTR    dns63.zos12.ral.ibm.com.                    1
 62           IN      PTR    dns64.zos12.ral.ibm.com.
 $GENERATE 3-6 $PTR host$.zos12.ral.ibm.com.                             2
; The following records are generated by the above $GENERATE directive.
;3           IN      PTR    host3.zos12.ral.ibm.com.
;4           IN      PTR    host4.zos12.ral.ibm.com.
;5           IN      PTR    host5.zos12.ral.ibm.com.
;6           IN      PTR    host6.zos12.ral.ibm.com.
 67           IN      PTR    host1.zos12.ral.ibm.com.
 61           IN      PTR    host2.zos12.ral.ibm.com.
```

Figure 10-49 Reverse master file sample

1 The *PTR* record type includes, in the data part of the record, the fully qualified name of the host corresponding to the IP address in the name part of the record.

2 *$GENERATE* is a v9-specific directive that is useful for creating a series of records that differ only by an tolerator. This line prompts the name server to create the records listed below upon zone load. The syntax and a brief explanation over the parameters follows:

$GENERATE range hs type rhs [comment]

range - This can be one of two forms: start-stop or start-stop/step. If the first form is used then step is set to 1. All of start, stop and step must be positive.

hs - This parameter describes the owner name of the resource records to be created. Any single $symbols within the *lhs* side are replaced by the interactor value.

type - At present the only supported types are PTR, CNAME, NS, A, and AAAA. In our sample we defined as PTR

rhs - A domain name. It is processed similarly to *lhs*.

Note: The *$GENERATE* directive is a BIND extension and not part of the standard zone file format.

10.4.7 Create the loopback file

The loopback file contains the loopback address. This is the address that a host uses to route queries to itself. BIND 9 model requires the availability to bind onto loopback address 127.0.0.1. Figure 10-50 shows the contents of the loopback file.

```
$TTL 86400
0.0.127.in-addr.arpa. IN SOA  dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com (
     1
     10800
     3600
     604800
     86400)

0.0.127.in-addr.arpa.   IN   NS  dns63.zos12.ral.ibm.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.zos12.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

Figure 10-50 Loopback file contents

Note: In addition to creating the loopback file, add an address resource record called localhost to the forward domain data file. This record supports proper two-way resolution.

10.4.8 Create the cache file (hints file)

The hints file contains the names and IP addresses of the authoritative root domain name servers. The root name servers contain the names of name servers in the top-level domains such as com, edu, and mil. The name server uses root server information when deciding which name server to contact when it receives a query for a host outside its zone of authority and it does not have the data in its cache. To obtain a hints file, point your Web browser at ftp://ftp.rs.internic.net and retrieve the file named.root from the domain subdirectory. Update your hints file on a regular basis.The zone statement with a type *hints* parameter definition in a BIND 9 conf file specifies the path and name of the hints file. Figure 10-51 displays the partial contents of the named.ca file.

```
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
 A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
 B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
```

Figure 10-51 Partial contents of a cache file

Note: The hints file does not contain cached data nor does the name server provide other hosts with the information contained in the hints file. A forward-only server is the only type of name server that does not require a hints file.

10.4.9 Configuring logging

A wide variety of logging options for the nameserver can be configured via the logging statement. Its channel phrase associates output methods, format options and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged. Only one logging statement is used to define as many channels and categories as are wanted. If there is no logging statement, the logging configuration will be:

```
logging {
category "default" {"default_syslog"; "default_syslog";};
};
```

In BIND 9, the logging configuration is only established when the entire configuration file has been parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels. Therefore, if started from a procedure, the logging messages will be written to syslogd. If started from z/OS UNIX, the logging messages may be written to '*named.run*' if started with the -d option, in addition to syslog.

All log output goes to one or more channels; you can make as many of them as you want. Every channel definition must include a clause that says whether messages selected for the channel go to a file, to a particular syslog facility, or are discarded. It can optionally limit the message severity level that will be accepted by the channel (the default is info), and whether to include a named -generated time stamp, the category name and/or severity level (the default is not to include any).

A example of a channel definition in the named.conf file is shown in Figure 10-52:

```
logging {
   channel "main_log" {                              1
   file "/etc/octavio/named.run" versions 2 size 20M;  2
   print-time yes;                                     3
   print-category yes;                                 4
   print-severity yes;                                 5
   severity dynamic;                                   6
               };
```

Figure 10-52 Channel definition

1 The *channel* phrase defines an output destination to log DNS generated messages. You can create as many channels as you want and reference them in your *category* phrase.

2 The *file* option defines the file name where all messages directed to this channel should be sent. It also defines the size of the file (*size* parameter) and how many files to create in a round robin fashion, before it reuses the first file (*version* parameter). If you want this channel not to log any messages, you can use the *null* option instead of *file.* The word *null* in the destination option for the channel will cause all messages sent to it to be discarded; in that case, other options for the channel are meaningless.

3 The *print-time* clause defines whether you want to append the date and time to the messages being logged. The default option is no.

4 The *print-category* clause defines whether you want to append to the messages being received the category that generated them. The default is no.

5 The *print-severity* clause defines whether you want to append their severity to the messages being received. The default is no.

6 The *severity* clause works like syslog's priorities, except that it can also be used if you are writing straight to a file rather than using syslog. Messages which are not at least of the severity level given will not be selected for the channel; messages of higher severity levels will be accepted. The normal options to be defined here are *info*, *debug*, and *dynamic*. Use *debug* to define the detail level you want to log in this channel: up to 99, which is the most detailed debug level. Using the *dynamic* option, the debug detail level is defined by the *named -d* start option value. Figure 10-53 shows partial contents of a log file.

```
May 22 12:23:36.735 general: debug 1: now using logging configuration from config file
May 22 12:23:36.738 general: debug 1: load_configuration: success
May 22 12:23:36.738 general: debug 1: dns_zone_load: zone 0.0.127.in-addr.arpa/IN:
May 22 12:23:36.741 general: debug 1: dns_zone_load: zone 6.12.9.in-addr.arpa/IN: start
May 22 12:23:36.742 general: debug 1: zone_timer: zone 0.0.127.in-addr.arpa/IN: enter
May 22 12:23:36.742 resolver: debug 1: createfetch: dns64.zos12.ral.ibm.com. A
```

Figure 10-53 Partial contents of a log file.

There are four predefined channels that are used for named's default logging, as follows:

```
      channel "default_syslog" {
         syslog daemon;                  // end to syslog's daemon
                                         // facility
         severity info;                  // only send priority info
                                         // and higher
};
      channel "default_syslog" {
           file "named.run";             // write to named.run in
                                         // the working directory
             severity dynamic            // log at the server's
                                         // current debug level
 };
      channel "default_stderr" {              // writes to stderr
          file "<stderr>";                      // this is illustrative only;
                                                // there's currently no way of
                                                // specifying an internal file
                                                // descriptor in the
                                                // configuration language.
          severity info;                        // only send priority info
                                                // and higher
};
      channel "null" {
         null;                                  // toss anything sent to
                                                // this channel
};
```

Once a channel is defined, it cannot be redefined. Thus, you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined. There are many categories, so you can send the logs you want to see wherever you want, without seeing logs you do not want. If you don't specify a list of channels for a category, then log messages in that category will be sent to the default category instead. If you do not specify a default category, the following "default" is used:

```
category "default" {"default_syslog"; "default_stderr";};
```

You can use the category phrase to direct the messages being generated to each category to one or more channels, as shown in the following examples:

```
category client   {main_log;};
category config   { main_log; };
category "database" { main_log; };
category dispatch { main_log; };
category dnssec { security_log; main_log; };
category security { security_log; main_log; };
category update   { main_log; };
category queries  { query_log;};
category lame-servers { query_log; main_log; };
category xfer-in  { "transfer_log"; };
category xfer-out { "transfer_log"; };
category default  { main_log; };
};
```

More details about the available categories and brief descriptions of the types of log information they contain can be found in the *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

10.4.10 Starting the DNS server

Once you have completed the previous configuration steps, you will be able to start the Domain Name Server. Basically, you can start the named process either from an MVS procedure, which has been customized in step 5.2.2, or from the z/OS UNIX shell. If the user choose to use the MVS procedure, a supervisor with an authorized TSO ID can start a name server from the MVS operator's console by starting the customized named start procedure.

```
S NAMED9
```

To execute the named process fro z/OS UNIX, a user with superuser authority can start the Name Server from the shell by starting z/OS UNIX, then issuing the named command and, optionally, any parameters. It is also possible to start the server automatically when z/OS UNIX is started by specifying the path and file name of the z/OS UNIX initialization shell script in the /etc/init.options file using the -sc option as shown in Figure 10-54.

```
-a  9999                          timeout = 9999 seconds
-t  1                             terminate shell = yes
-sc /etc/rc                       shell script = /etc/rc
-e  TZ=EST5EDT                    TZ environment variable
```

Figure 10-54   Contents of the init.options file

The file /etc/rc is the default z/OS UNIX initialization shell script that is executed when z/OS UNIX is started. Information such as the following can be entered in /etc/rc:

```
# Start name server
 |       /usr/lpp/tcpip/sbin/named -c /named/production/named.conf &
```

Figure 10-55   Contents of /etc/rc file

If you use the UNIX System Services /etc/rc method, the actual startup of the name server waits for the completion of TCP/IP initialization. You cannot start the name server with the INETD daemon. The startup must know either via default or via parameters what the boot file name is so that the correct data can be loaded into the DNS.

You can use the same syntax to start named using z/OS UNIX console (omvs command in TSO command line option). Like any UNIX command you execute each command in the command line of the UNIX console as follows:

```
===> /usr/lpp/tcpip/sbin/named -c /etc/octavio/named.conf
```

Figure 10-56 Starting named v9 from z/OS UNIX command line

The same command could also be issued from the ISHELL command panel. To get to this panel, follow these steps from the TSO/PDF option 6 (command line):

1. In TSO - *ISPF Primary Options Menu* - enter option `6` - (commands)
2. In TSO - *ISPF Command Shell* - enter the command **`ish`** - (UNIX System Services ISPF shell)
3. In the *UNIX System Services ISPF Shell* screen - go to top line and choose **Tools**.This will open a box with the available tools. Choose option *2* - **Run Shell Command**
4. In the Enter a Shell Command panel, enter the named startup command as shown:

```
                       Enter a Shell Command

Enter a shell command and press Enter.

Standard output and standard error are redirected to a temporary
file.  If there is any data in the file when the shell command
completes, the file is displayed.
   /usr/lpp/tcpip/sbin/named -c /etc/octavio/named.conf
   ________________________________________________________________
   ________________________________________________________________
   ________________________________________________________________






 F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel
```

Figure 10-57 Executing named from UNIX System Services ISPF Shell

The usual way to start named in z/OS V1R2 is to execute a procedure from the MVS console. The following is the console log following the startup of the DNS server process at SC63.

```
S NAMED9
$HASP100 NAMED9   ON STCINRDR
IEF695I START NAMED9   WITH JOBNAME NAMED9   IS ASSIGNED TO USER STC
 , GROUP SYS1
$HASP373 NAMED9   STARTED
IEF403I NAMED9 - STARTED - TIME=17.06.08 - ASID=0065 - SC63
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-NAMED9            NAMED       00     45  .00  .00  .00   1881
 0      0      0      0     0
+EZZ9095I STARTING NAMED, BIND 9.1.1
-                                         --TIMINGS (MINS.)--
   ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV
PG   PAGE   SWAP    VIO SWAPS
-NAMED9            *OMVSEX     00    620  .00  .00  .03  16797
 0      0      0      0     0
IEF404I NAMED9 - ENDED - TIME=17.06.10 - ASID=0065 - SC63
-NAMED9   ENDED.  NAME-                     TOTAL CPU TIME= .00
TOTAL ELAPSED TIME= .03
$HASP395 NAMED9   ENDED
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=0065.
+EZZ9130I NAMED, BIND 9.1.1 IS RUNNING
```

Figure 10-58 Named V9 startup messages in the MVS Console Log

Note: Note that the procedure NAMED9 has ended, but it creates a child process called NAMED91. This process is used to stop named V9 as we will see later in this chapter.

10.4.11 Verifying that the name server has started correctly

In BIND 9, the logging configuration is only established when the entire configuration file has been parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels. Therefore, if started from a procedure, the logging messages will be written to syslogd. If started from z/OS UNIX, the logging messages may be written to 'named.run' if started with the -d option, in addition to syslog. Once you had started the named process, you can go to the logging files you defined in your named.conf configuration file and check the results of the startup procedure. The messages generated should basically appear as shown in Figure 10-59.

```
EZZ9171I LPAR mode detected. Using 2 CPUs for -n option                    1
EZZ9547I starting named, BIND 9.1.1 -d 1 -c /etc/octavio/named.conf          2
EZZ9217I Running non-swappable
EZZ9540I using 2 CPUs
EZZ9126I loading configuration from '/etc/octavio/named.conf'
EZZ8842I the default for the 'auth-nxdomain' option is now 'no'
EZZ9052I no IPv6 interfaces found
EZZ9046I listening on IPv4 interface OSA22E0      , 9.12.6.67#53             3
EZZ9046I listening on IPv4 interface VIPL090C0644  , 9.12.6.68#53
```

Figure 10-59 Named V9 startup messages in named.run file

1 Because the BIND 9 name server is multi-threaded, it can take advantage of any additional processors you add to the system. The BIND 9 name server will detect the number of logical CPUs configured for the system (if not running partitioned) or LPAR (if running partitioned), and create additional worker threads accordingly. You can define how many processors you are going to use specifying the option *-n* in *named* command.

2 Message `EZZ9547I` shows to you the parms you used to start named and also the BIND version you are about to start

3 A BIND V9 server can specify IP address to listen on and from which to send queries, notifies and zone transfers in its configuration file. During the startup process, named tries to bind to each interface you configured. Since BIND V9 has no stack affinity, the default is to try all defined interfaces in all TCPIP stacks. To specify which interfaces we want, use option *listen-on* in the named.conf file.

After these messages, named initializes the logging system and all remaining messages are directed to the configured logging files.

When the name server is up with no logged errors, ensure that it can accept queries. Ensure that the name server can accept queries locally from both the MVS and z/OS UNIX environments. To be able to do that you must first assure that your userid is using the same resolver configuration files. After the resolver configuration is correct, you can test your server either with the nslookup or dig command. An example using nslookup follows. Issue the following command from both the z/OS UNIX shell and the TSO ready prompt. In this example, the name 'host1.zos12.ral.ibm.com.' is used for the search.

```
nslookup host1
```

The result of the command follows in Figure 10-60.

```
OCTAVIO @ SC63:/>nslookup host1
Defaulting to nslookup version 4
Starting nslookup version 4
Server:  dns63.zos12.ral.ibm.com
Address:  9.12.6.68

Name:    host1.zos12.ral.ibm.com
Address:  9.12.6.67
```

Figure 10-60 nslookup command response

10.4.12 Reloading BIND V9

To reload data that you may have changed during this lifetime of the name server, you can send a signal to the UNIX System Services shell asking DNS to reread its configuration files. To send the signal from the z/OS UNIX console, execute the following command:

```
kill -HUP ($cat /etc/named.pid)
```

You can execute also a signal command from the ISHELL command panel. To get to this panel, follow these steps from the TSO/PDF option 6 (command line):

1. In TSO - *ISPF Primary Options Menu* - enter option `6` - (commands)
2. In TSO - *ISPF Command Shell* - enter the command **`ish`** - (UNIX System Services ISPF shell)

3. In the *UNIX System Services ISPF Shell* screen - go to top line and choose **Tools**.This will open a box with the available tools. Choose option 1 - **Work with processes**
4. In the *Work with Processes Signal* screen go to the line where the named process is and enter `s`
5. In the *Enter Signal Number* screen, confirm you choose the process you want to reload, go to the Signal number field and enter the SIGHUP signal number, `#1.`

These steps will reload the named process implementing your changes. To confirm your name server has been reloaded, you can check the main_log file you defined for the following messages:

```
general: info: EZZ9126I loading configuration from '/etc/octavio/named.conf'
config: warning: EZZ8842I the default for the 'auth-nxdomain' option is now 'no'
network: info: EZZ9052I no IPv6 interfaces found
```

This command works only at a primary name server. A secondary periodically returns to query the primary for new data, theoretically eliminating the need to reload a secondary with a signal. (The refresh interval is one of the settings in the SOA record.)

The reload process is not designed for dynamic domains, since these are updated via the nsupdate command.

10.4.13 Stopping the DNS server

You can stop the DNS server with the MVS STOP command P NAMED91, which is the name of the startup procedure used to start you name server (it can be any name), followed by the number "1", which is an extra forking process assigned to BIND V9 during the startup process The advantage of the STOP command is the graceful termination of the name server and the issuing of messages in SYSLOG. Other alternatives are to use the MVS CANCEL command or the OMVS **`kill`** command. The OMVS **`kill`** command can be issued from OMVS or from the NSSIG procedure.

Executing the **`kill`** command in the z/OS UNIX console:

```
kill $(cat /etc/named.pid)
```

You can execute also the **`kill`** command from the ISHELL command panel. To get to this panel, follow these steps from the TSO/PDF option 6 (command line):

1. In TSO - *ISPF Primary Options Menu* - enter option `6` - (commands)
2. In TSO - *ISPF Command Shell* - enter the command **`ish`** - (UNIX System Services ISPF shell)
3. In the *UNIX System Services ISPF Shell* screen - go to top line and choose **Tools**.This will open a box with the available tools. Choose option 1 - **Work with processes**
4. In the *Work with Processes* screen go to the line where the *named* process is and enter `k`

You can also stop the process using the MVS **`CANCEL`** command to NAMED43, and this will cause the following messages in the console log:

```
C NAMED91
IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=NAMED91, ASID=0056.
BPXP018I THREAD 106954D000000001, IN PROCESS 50397272, ENDED 764
WITHOUT BEING UNDUBBED WITH COMPLETION CODE 00222000,
AND REASON CODE 00000000.
BPXP018I THREAD 1069615800000000, IN PROCESS 50397272, ENDED 768
WITHOUT BEING UNDUBBED WITH COMPLETION CODE 40222000,
AND REASON CODE 00000000.
IEE301I NAMED91          CANCEL COMMAND ACCEPTED
IEF450I NAMED91 STEP1 - ABEND=S222 U0000 REASON=00000000 770
        TIME=11.27.34
```

Figure 10-61 Cancel command messages

10.4.14 Implementing a secondary name server DNS64

The next step to perform to implement our domain name server scenario is to configure the secondary name server. A secondary name server can be primary for some zones and secondary for others. Its boot file indicates for which zones it is primary and for which it is secondary. Using a zone transfer process, the secondary server retrieves the files for specified zones from the primary name server that it points to. The secondary stores this information in its own files if the retrieved *serial numbe*r is higher than the current serial number stored at the secondary (Figure 10-48 on page 361, 4). The secondary obtains the files from the primary name server based upon the *refresh interval* coded on the SOA record or the resource record (RR) itself.

To implement a secondary name server, follow the same steps used to create the master name server. The difference is that there is no need to create the domain data files (the files containing host-to-address and address-to-host mappings). These files are maintained on the master name server and the secondary (slave) name server transfers this data to its own database. The configuration file also has different definitions to state that this server is a a slave (secondary) name server.

To show how we define our secondary name server, we'll go through the same steps we followed to build the master name server.

10.4.14.1 Create the configuration file of the secondary name server

The best approach to create the named.conf file of the secondary name server, is to copy the named.conf file created for the master name server and alter the contents to reflect this is a secondary server, as shown:

```
options {
   pid-file "/etc/named.pid" ;
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.63;9.12.6.64; };                               1
};
logging {
   channel main_log {                                                2
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   severity dynamic;
                    };
   category xfer-in  { main_log; };                                  3
   category xfer-out { main_log; };
   };
zone "zos12.ral.ibm.com" in {                                        4
   type slave;
   file "zos12.for.bak.v9";
   masters { 9.12.6.68; };
};
zone "6.12.9.in-addr.arpa" in {
   type slave;
   file "zos12.rev.bak.v9";
   masters { 9.12.6.68; };
};
zone "0.0.127.in-addr.arpa" in {                                     5
   type master;
   file "zos12.lbk.v9";
};
zone "." in {
   type hint;
   file "zos12.ca.v9";
};
```

Figure 10-62 Contents of named.conf for the slave DNS

1 The listen-on parameter takes an optional port and an address_match_list. The server will listen on all interfaces allowed by the address match list. If a port is not specified, port 53 is used.

2 A channel phrase in the logging statement associates output methods, format options and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged. All log output goes to one or more channels; you can make as many of them as you want. In our config file we created a channel called main_log that will send all log output to the file named_main.log.

3 A category phrase selects how various classes of messages are logged. Use with channel name. Because many categories exist, you can send the logs you want to see wherever you want and avoid seeing logs you do not want. In this configuration file, we want to log only the messages generated during transfer processes between the master and the slave DNS.

4 This zone statement defines that this DNS is a slave for the indicated zone, will transfer definitions from a Master DNS pointed by the *masters* statement and will save this definitions in the file defined by the statement *file*

5 The server is still considered to be a primary DNS for its loopback address.

The remaining steps are shown here only for your information. For further instructions about each step, refer to the previous section, implementing the master name server.

1. Specify port ownership
2. Update the name server start procedure (optional)
3. Create the loopback file
4. Create the hints (cache) file
5. Start the secondary name server
6. Verifying if secondary name server is working as expected

The remaining steps are shown here only for your information. For further instructions about each step, refer to the previous section, implementing the master name server.

At a secondary name server, you would see similar console messages about the file loading process. You also should be able to see messages indicating a successful transfer of file contents between master and slave DNS. Figure 10-63 shows a partial contents of the main_log file.

```
EZZ9156I transfer of 'zos12.ral.ibm.com' from 9.12.6.68#53: end of transfer
EZZ9156I transfer of '6.12.9.in-addr.arpa' from 9.12.6.68#53: end of transfer
```

Figure 10-63 Successful zone transfer from Master DNS

If your zone transfers are successful, the files identified in the secondary name server's boot file as the storage for retrieved files can be viewed. The format looks a bit different from your own coding at the primary, but the contents are nonetheless identifiable. For example, below is a portion of the domain file retrieved by DNS64, the secondary name server.

```
$ORIGIN .
$TTL 86400 ; 1 day
zos12.ral.ibm.com IN SOA dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com (
    1        ; serial
    10800    ; refresh (3 hours)
    3600     ; retry (1 hour)
    604800   ; expire (1 week)
    86400    ; minimum (1 day)
  )
   NS dns63.zos12.ral.ibm.com.
   NS dns64.zos12.ral.ibm.com.
$ORIGIN zos12.ral.ibm.com.
_http._tcp  SRV 0 0 80 www
dns63   A 9.12.6.68
dns64   A 9.12.6.64
ftp   CNAME host2
host1   A 9.12.6.67
host2   A 9.12.6.63
localhost  A 127.0.0.1
mail   CNAME host1
sc63   A 9.12.6.67
       A 9.12.6.68
www   A 9.179.147.237
```

Figure 10-64 Contents of the backup domain file after the zone transfer operation

10.4.15 BIND 9 name server advanced topics

The following topics describe the new funcionality implemented in CS for z/OS V1R2 IP with the BIND 9-Based Name Server:

- 10.4.15.1, “Multiple TCP/IP stack (Common INET) considerations” on page 376
- 10.4.15.2, “Dynamic update” on page 377
- 10.4.15.3, “Incremental zone transfers (IXFR)” on page 377
- 10.4.15.4, “Split DNS” on page 377
- 10.4.15.5, “TSIG” on page 380
- 10.4.15.6, “DNSSEC” on page 382
- 10.4.15.7, “IPv6 support in BIND 9” on page 384

10.4.15.1 Multiple TCP/IP stack (Common INET) considerations

The BIND 9 name server is a generic server which does not have stack affinity. This is in contrast to the BIND 4.9.3 name server which does have stack affinity. This has certain implications. If you wish to run multiple BIND 9 name servers, you must divide the interfaces between the name servers with the *listen-on* named.conf file option. For example, you may want one stack serviced by one BIND 9 name server, and a second stack serviced by a second BIND 9 name server. Each name server would contain only the IP addresses of the assigned stack in its listen-on option **1**. Figure 10-65 shows how to configure it in the named.conf file:

```
options {
   pid-file "/etc/named.pid" ;
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.67;9.12.6.68; }; 1
```

Figure 10-65 named.conf file with the listen-on definition

Be aware that once you start a TCP/IP stack, all of the adapters may not be active immediately, and therefore will not be usable by the name server immediately. When the name server is started manually or restarted automatically by stack bring up or bring down, it immediately queries the available TCP/IP stacks for active adapters. Often times, it will take some time for all of the adapters to become active (this is independent of the name server). The name server will re-query the stacks every minute, by default, for any changes in the active/inactive status of adapters and then make use of them once they are active. The one minute interval can be lengthened by the *interface-interval* named.conf file option if desired but this is not recommended.

By default, the name server will unpredictably choose one adapter from any of the active stacks to use when it must communicate with other name servers. If some adapters do not have the capability to route into the network, you may see unpredictable results on name server queries. This unpredictable behavior can be eliminated by making use of the *query-source* option in the named.conf file. The *query-source* option should specify an adapter address that will always have network routing capability. The query-source option then places a dependency on the stack that owns that address to be active. If the owning TCP/IP stack of the query-source option address is taken down, the name server will end, since it will no longer have a way to communicate with the network, and thus, with other name servers.

10.4.15.2 Dynamic update

Dynamic update is the term used to refer to the ability, under certain specified conditions, to add, modify or delete records or RRsets in the master zone files. Dynamic update is fully described in *RFC 2136*. Dynamic update is enabled on a zone-by-zone basis, by including an *allow-update* or *update-policy* clause in the zone statement. Preferably, use *TSIG* security between the nsupdate utility and the targeted name server. BIND 9 name server configuration processing messages will remind you when nsupdate authorization is only based on client IP address. Updating of secure zones (zones using DNSSEC) is modelled after the *simple-secure-update* proposal, a work in progress in the DNS Extensions working group of the IETF. (See http://www.ietf.org/html.charters/dnsext-charter.html for information about the DNS Extensions working group.) SIG and NXT records affected by updates are automatically regenerated by the server using an online zone key. Update authorization is based on transaction signatures and an explicit server policy.

The zone files of dynamic zones must not be edited by hand. If the zone file of a dynamic zone is edited by hand, corrupt .jnl files can result and all changes not written to the zone file may be lost. The zone file on disk at any given time may not contain the latest changes performed by dynamic update. The zone file is written to disk only periodically, and changes that have occurred since the zone file was last written to disk are stored only in the zone's journal (.jnl) file. Depending on signal or `rndc` stop options, BIND 9 name server may or may not update the zone file. Therefore, editing the zone file manually is unsafe even when the server has been shut down.

10.4.15.3 Incremental zone transfers (IXFR)

The incremental zone transfer (IXFR) protocol is a way for slave servers to transfer only changed data, instead of having to transfer the entire zone. The IXFR protocol is documented in *RFC 1995*. When acting as a master, BIND 9 supports IXFR for those zones where the necessary change history information is available. These include master zones maintained by dynamic update and slave zones whose data was obtained by IXFR, but not manually maintained master zones nor slave zones obtained by performing a full zone transfer (AXFR). When acting as a slave, BIND 9 will attempt to use IXFR unless it is explicitly disabled. For more information about disabling IXFR, see the description of the request-ixfr clause of the server statement.

10.4.15.4 Split DNS

Setting up different views, or visibility, of DNS space to internal and external resolvers is usually referred to as a split DNS setup. There are several reasons an organization would want to set up its DNS this way. One common reason for setting up a DNS system this way is to hide internal DNS information from external clients on the Internet. Another common reason for setting up a split DNS system is to allow internal networks that are behind filters or in *RFC 1918* space (reserved IP space, as documented in *RFC 1918*) to resolve DNS on the Internet. Split DNS can also be used to allow mail from outside back in to the internal network.

For example, a company named Example, Inc. (example.com) has several corporate sites that have an internal network with reserved Internet Protocol (IP) space and an external demilitarized zone (DMZ), or outside section of a network, that is available to the public.

Example, Inc. wants its internal clients to be able to resolve external hostnames and to exchange mail with people on the outside. The company also wants its internal resolvers to have access to certain internal-only zones that are not available at all outside of the internal network.

In order to accomplish this, the company will set up two sets of name servers. One set will be on the inside network (in the reserved IP space) and the other set will be on bastion hosts, which are proxy hosts that can talk to both sides of its network, in the DMZ.

The internal servers will be configured to forward all queries, except queries for site1.internal, site2.internal, site1.example.com, and site2.example.com, to the servers in the DMZ. These internal servers will have complete sets of information for site1.example.com, site2.example.com, site1.internal, and site2.internal.

To protect the site1.internal and site2.internal domains, the internal name servers must be configured to disallow all queries to these domains from any external hosts, including the proxy hosts.

The external servers, which are on the proxy hosts, will be configured to serve the public version of the site1 and site2.example.com zones. This could include things such as the host records for public servers (www.example.com and ftp.example.com), and mail exchange (MX) records (a.mx.example.com and b.mx.example.com).

In addition, the public site1 and site2.example.com zones should have special MX records that contain wildcard (*) records pointing to the proxy hosts. This is needed because external mail servers do not have any other way of looking up how to deliver mail to those internal hosts.

With the wildcard records, the mail will be delivered to the proxy host, which can then forward it on to internal hosts.

Here's an example of a wildcard MX record:

```
*   IN MX 10 external1.example.com.
```

Now that they accept mail on behalf of anything in the internal network, the proxy hosts will need to know how to deliver mail to internal hosts. In order for this to work properly, the resolvers on the proxy hosts will need to be configured to point to the internal name servers for DNS resolution. Queries for internal hostnames will be answered by the internal servers, and queries for external hostnames will be forwarded back out to the DNS servers on the proxy hosts. In order for all this to work properly, internal clients will need to be configured to query only the internal name servers for DNS queries. This could also be enforced via selective filtering on the network. If everything has been set properly, Example, Inc.'s will be allow the following:

- Its internal clients will now be able to:
  - Look up any hostnames in the site1 and site2.example.com zones.
  - Look up any hostnames in the site1.internal and site2.internal domains.
  - Look up any hostnames on the Internet.
  - Exchange mail with internal and external people.
- Hosts on the Internet will be able to:
  - Look up any hostnames in the site1 and site2.example.com zones.
  - Exchange mail with anyone in the site1 and site2.example.com zones.

Figure 10-66 on page 379 shows the internal server configuration for the setup we just described above. Note that this is not the complete file.

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };
acl externals { proxy-ips-go-here; };
options {
       ...
       ...
         forward only;
         forwarders {                        // forward to external servers
       proxy-ips-go-here;
     };
      allow-transfer { none; };               // sample allow-transfer (no one)
      allow-query { internals; externals; };  // restrict query access
      allow-recursion { internals; };         // restrict recursion
     ...
     ...
};
 zone "site1.example.com" {                   // sample slave zone
       type master;
       file "m/site1.example.com";
       forwarders {};                         // do normal iterative
                                               // resolution (do not forward)
       allow-query { internals; externals; };
       allow-transfer { internals; };
};
 zone "site2.example.com" {
       type slave;
       file "s/site2.example.com";
       masters { 172.16.72.3; };
       forwarders { };
       allow-query { internals; externals; };
       allow-transfer { internals; };
};
 zone "site1.internal" {
       type master;
       file "m/site1.internal";
       forwarders { };
       allow-query { internals; };
       allow-transfer { internals; }
};
 zone "site2.internal" {
       type slave;
       file "s/site2.internal";
       masters { 172.16.72.3; };
       forwarders { };
       allow-query { internals };
       allow-transfer { internals; }
};
```

Figure 10-66 named.conf file for the internal server

This shows the internal server configuration for the setup we just described above. Note that this is not the complete file.

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };
acl externals { proxy-ips-go-here; };
       options {
    ...
    ...
      allow-transfer { none; };                  // sample allow-transfer (no one)
      allow-query { internals; externals; };     // restrict query access
      allow-recursion { internals; externals; }; // restrict recursion
    ...
    ...
};
 zone "site1.example.com" {                     // sample master zone
     type master;
     file "m/site1.foo.com";
     allow-query { any; };
     allow-transfer { internals; externals; };
};
 zone "site2.example.com" {                      // sample slave zone
     type slave;
     file "s/site2.foo.com";
     masters { another_proxy_host_maybe; };
     allow-query { any; };
     allow-transfer { internals; externals; }
};
```

Figure 10-67 External server named.conf definitions

Figure 10-68 on page 380 shows the resolf.conf file definitions in the proxy server configuration for the setup we just described above. Note that this is not the complete file.

```
In the resolv.conf (or equivalent) on the proxy host(s):
   search ...
   nameserver 172.16.72.2
   nameserver 172.16.72.3
   nameserver 172.16.72.4
```

Figure 10-68 resolv.conf file definitions in the proxy server

10.4.15.5 TSIG

This is a short guide to setting up Transaction SIGnatures (TSIG) based transaction security in BIND. It describes changes to the configuration file as well as what changes are required for different features, including the process of creating transaction keys and using transaction signatures with BIND. BIND primarily supports TSIG for server to server communication. This includes zone transfer, notify, and recursive query messages. Resolvers on other platforms which are based on newer versions of BIND 8 have limited support for TSIG.

TSIG might be most useful for dynamic update. A master server for a dynamic zone should use access control to control updates, but IP-based access control is insufficient. Key-based access control is far superior. The nsupdate program supports TSIG via the -k and -y command line options.

To implement TSIG in your environment, follow these steps:

- “Generate shared keys for each pair of hosts” on page 381
- “Copying the shared secret to both machines” on page 381

- “Informing the servers of the key's existence” on page 381
- “Instructing the server to use the key” on page 381y
- “TSIG key based access control” on page 382
- “Errors” on page 382

Generate shared keys for each pair of hosts

A shared secret is generated to be shared between host1 and host2. An arbitrary key name is chosen: "host1-host2.". The key name must be the same on both hosts. The shared secret can be generated either automatically or manually, as follows:

- *Automatic Generation*: The following command will generate a 128-bit (16-byte) HMAC-MD5 key as described above. Longer keys are better, but shorter keys are easier to read. Note that the maximum key length is 512 bits; keys longer than that will be digested with MD5 to produce a 128 bit key.

  ```
  dnssec-keygen -a hmac-md5 -b 128 -n HOST host1-host2.
  ```

The key is in the file Khost1-host2.+157+00000.private. Nothing directly uses this file, but the base-64 encoded string following "Key:" can be extracted from the file and used as a shared secret:

```
Key: La/E5CjG9O+os1jq0a2jdA==
```

The string "La/E5CjG9O+os1jq0a2jdA==" can be used as the shared secret.

- *Manual Generation*: The shared secret is simply a random sequence of bits, encoded in base-64. Most EBCDIC strings are valid base-64 strings (assuming the length is a multiple of 4 and only valid characters are used), so the shared secret can be manually generated. Also, a known string can be run through mmencode or a similar program to generate base-64 encoded data.

Copying the shared secret to both machines

This is beyond the scope of DNS. A secure transport mechanism should be used. This could be secure FTP, ssh, telephone, etc.

Informing the servers of the key's existence

Imagine host1 and host2 are both servers. The following is added to each server's named.conf file:

```
    key host1-host2. {
    algorithm hmac-md5;
    secret "La/E5CjG9O+os1jq0a2jdA==";
  };
```

The algorithm, hmac-md5, is the only one supported by BIND. The secret is the one generated above. Since this is a secret, it is recommended that either named.conf be non-world readable, or the key directive be added to a non-world readable file that is included by named.conf. At this point, the key is recognized. This means that if the server receives a message signed by this key, it can verify the signature. If the signature succeeds, the response is signed by the same key.

Instructing the server to use the key

Since keys are shared between two hosts only, the server must be told when keys are to be used. The following is added to the named.conf file for host1, if the IP address of host2 is 10.1.2.3:

```
   server 10.1.2.3 {
     keys { host1-host2.; };
 };
```

Multiple keys may be present, but only the first is used. This directive does not contain any secrets, so it may be in a world-readable file. If host1 sends a message that is a request to that address, the message will be signed with the specified key. host1 will expect any responses to signed messages to be signed with the same key. A similar statement must be present in host2's configuration file (with host1's address) for host2 to sign request messages to host1.

TSIG key based access control

BIND allows IP addresses and ranges to be specified in ACL definitions and allow-{query | transfer | update} directives. This has been extended to allow TSIG keys also. The above key would be denoted key host1-host2. An example of an allow-update directive would be:

```
allow-update { key host1-host2. ;};
```

This allows dynamic updates to succeed only if the request was signed by a key named "host1-host2.".

Errors

The processing of TSIG signed messages can result in several errors. If a signed message is sent to a non-TSIG aware server, a *FORMERR* will be returned, since the server will not understand the record. This is a result of misconfiguration, since the server must be explicitly configured to send a TSIG signed message to a specific server. If a TSIG aware server receives a message signed by an unknown key, the response will be unsigned with the TSIG extended error code set to *BADKEY.* If a TSIG aware server receives a message with a signature that does not validate, the response will be unsigned with the TSIG extended error code set to *BADSIG*. If a TSIG aware server receives a message with a time outside of the allowed range, the response will be signed with the TSIG extended error code set to *BADTIME*, and the time values will be adjusted so that the response can be successfully verified. In any of these cases, the message's return code is set to NOTAUTH.

10.4.15.6 DNSSEC

Cryptographic authentication of DNS information is possible through the DNS Security (DNSSEC) extensions, defined in *RFC 2535*. This section describes the creation and use of DNSSEC signed zones.

The set of dnssec- tools rely on a /dev/random device for the entropy it needs to generate cryptographically strong keys. If RSA keys are used, only dnssec-keygen requires random data. z/OS UNIX does not include such a device, but the tools provide alternate methods of providing them with random data. The user can specify a file containing random data or can provide random data via the keyboard. To specify a file, use the -r random data file option on the tool command line. The dnssec- tools use the timing between keystrokes as the source of entropy. As such, TN3270 terminal emulation is not the ideal interface. Setting up a VT100 terminal session is a better solution. Refer to "Configuring the z/OS UNIX Telnet Server (otelnetd)" in topic 2.2.2 for more information on setting up otelnetd.

In order to set up a DNSSEC secure zone, there are a series of steps which must be followed. z/OS ships with several tools that are used in this process, which are explained in more detail below. In all cases, the "-h" option prints a full list of parameters. Note that the DNSSEC tools require the keyset and signedkey files to be in the working directory. There must also be communication with the administrators of the parent and/or child zone to transmit keys and signatures. A zone's security status must be indicated by the parent zone for a DNSSEC capable resolver to trust its data. For other servers to trust data in this zone, they must either be statically configured with this zone's zone key or the zone key of another zone above this one in the DNS tree using the trusted-keys statement.

To implement DNSSEC, follow these steps:

- “Generating keys” on page 383
- “Creating a keyset” on page 383
- “Signing the child's keyset” on page 383
- “Signing the zone” on page 384
- “Configuring servers” on page 384

Generating keys

The **dnssec-keygen** program is used to generate keys. A secure zone must contain one or more zone keys. The zone keys will sign all other records in the zone, as well as the zone keys of any secure delegated zones. Zone keys must have the same name as the zone, a name type of ZONE, and must be usable for authentication. The following command will generate a 768-bit RSA key for the child.example zone:

```
dnssec-keygen -a RSA -b 768 -n ZONE child.example
```

Two output files will be produced: *Kchild.example.+001+12345.key* and *Kchild.example.+001+12345.private* (where 12345 is an example of a key tag). The key file names contain the key name (child.example.), algorithm (3 is DSA, 1 is RSA, etc.), and the key tag (12345 in this case). The private key (in the .private file) is used to generate signatures, and the public key (in the .key file) is used for signature verification. To generate another key with the same properties (but with a different key tag), repeat the above command. The public keys should be inserted into the zone file with *$INCLUDE* statements, including the .key files.

Creating a keyset

The **dnssec-makekeyset** program is used to create a key set from one or more keys. Once the zone keys have been generated, a key set must be built for Transmission to the administrator of the parent zone, so that the parent zone can sign the keys with its own zone key and correctly indicate the security status of this zone. When building a key set, the list of keys to be included and the TTL of the set must be specified, and the desired signature validity period of the parent's signature may also be specified. The list of keys to be inserted into the key set may also include non-zone keys present at the top of the zone. **dnssec-makekeyset** may also be used at other names in the zone.

The following command generates a key set containing the above key and another key similarly generated, with a TTL of 3600 and a signature validity period of 10 days starting from now.

```
dnssec-makekeyset -t 3600 -e +8640 Kchild.example.+001+12345
Kchild.example.+001+23456
```

One output file is produced: *keyset-child.example*. This file should be transmitted to the parent to be signed. It includes the keys, as well as signatures over the key set generated by the zone keys themselves, which are used to prove ownership of the private keys and encode the desired validity period.

Signing the child's keyset

The **dnssec-signkey** program is used to sign one child's keyset. If the child.example zone has any delegations which are secure, for example, grand.child.example, the child.example administrator should receive keyset files for each secure subzone. These keys must be signed by this zone's zone keys. The following command signs the child's key set with the zone keys:

```
dnssec-signkey keyset-grand.child.example. Kchild.example.+001+12345
Kchild.example.+001+23456
```

One output file is produced: *signedkey-grand.child.example.*. This file should be both transmitted back to the child and retained. It includes all keys (the child's keys) from the keyset file and signatures generated by this zone's zone keys.

Signing the zone

The **dnssec-signzone** program is used to sign a zone. Any signedkey files corresponding to secure subzones should be present, as well as a signedkey file for this zone generated by the parent (if there is one). The zone signer will generate NXT and SIG records for the zone, as well as incorporate the zone key signature from the parent and indicate the security status at all delegation points. The following command signs the zone, assuming it is in a file called zone.child.example. By default, all zone keys which have an available private key are used to generate signatures.

```
dnssec-signzone -o child.example zone.child.example
```

One output file is produced: *zone.child.example.signed*. This file should be referenced by named.conf as the input file for the zone.

Configuring servers

Data is not verified on load in BIND 9, so zone keys for authoritative zones do not need to be specified in the configuration file. The public key for any security root must be present in the configuration file's trusted-keys statement.

10.4.15.7 IPv6 support in BIND 9

BIND 9 fully supports all currently defined forms of IPv6 name to address and address to name lookups. For forward lookups, BIND 9 supports both A6 and AAAA records. The use of AAAA records is deprecated, but it is still useful for hosts to have both AAAA and A6 records to maintain backward compatibility with installations where AAAA records are still used. In fact, the stub resolvers currently shipped with most operating system support only AAAA lookups, because following A6 chains is much harder than doing A or AAAA lookups. For IPv6 reverse lookups, BIND 9 supports the new bitstring format used in the ip6.arpa domain, as well as the older, deprecated nibble format used in the ip6.int domain

Address lookups using AAAA records

The AAAA record is a parallel to the IPv4 A record. It specifies the entire address in a single record. For example:

```
 $ORIGIN example.com.
host  3600 IN AAAA 3ffe:8050:201:1860:42::1
```

While their use is deprecated, they are useful to support older IPv6 applications. They should not be added where they are not absolutely necessary.

Address lookups using A6 records

The A6 record is more flexible than the AAAA record, and is therefore more complicated. The A6 record can be used to form a chain of A6 records, each specifying part of the IPv6 address. It can also be used to specify the entire record as well. For example, this record supplies the same data as the AAAA record in the previous example:

```
$ORIGIN example.com.
host  3600 IN A6 0 3ffe:8050:201:1860:42::1
```

A6 Chains: A6 records are designed to allow network renumbering. This works when an A6 record only specifies the part of the address space the domain owner controls. For example, a host may be at a company named "company." It has two ISPs which provide IPv6 address space for it. These two ISPs fully specify the IPv6 prefix they supply.

- In the company's address space:

```
$ORIGIN example.com.
host  3600 IN A6 64 0:0:0:0:42::1 company.example1.net.
host  3600 IN A6 64 0:0:0:0:42::1 company.example2.net.
```

- ISP1 will use:

```
$ORIGIN example1.net.
company  3600 IN A6 0 3ffe:8050:201:1860::
```

- ISP2 will use:

```
$ORIGIN example2.net.
company  3600 IN A6 0 1234:5678:90ab:fffa::
```

When host.example.com is looked up, the resolver (in the caching name server) will find two partial A6 records, and will use the additional name to find the remainder of the data.

Note: A6 chain resolution is only performed when one name server requests an A6 record from another name server. A6 chain resolution will not be performed if a client requests an A6 record.

A6 Records for DNS servers: When an A6 record specifies the address of a name server, it should use the full address rather than specifying a partial address. For example:

```
$ORIGIN example.com.
@  14400  IN NS
ns0
  14400  IN NS
ns1
ns0  14400  IN A6  0
3ffe:8050:201:1860:42::1
ns1  14400  IN A
192.168.42.1
```

It is recommended that IPv4-in-IPv6 mapped addresses not be used. If a host has an IPv4 address, use an A record, not an A6, with ::ffff:192.168.42.1 as the address.

10.5 Securing your DNS environment

z/OS V1R2 Communications Server has introduced several important security features that help you protect your name server, as we already shown in “BIND 9 name server advanced topics” on page 376. These features are particularly important if your name server is running on the Internet, but they're also useful on purely internal name servers. The purpose of these section is to show you, step by step, the measures you could take to secure the implementation scenario we created in the section 10.4, “Setting up a BIND 9-based Domain Name Server” on page 355. The security implementation scenario will follow these steps:

1. 10.5.1, “Restricting queries” on page 386
2. 10.5.2, “Preventing unauthorized zone transfers” on page 386
3. 10.5.3, “Creating a transaction signature between master and slave” on page 387
4. 10.5.4, “Signing your zone” on page 388

10.5.1 Restricting queries

Before BIND 4.9, administrators had no way to control who could look up names on their name servers. That makes a certain amount of sense; the original idea behind DNS was to make information easily available all over the Internet. The Internet is not such a friendly place anymore, though. In particular, people who run Internet firewalls may have a legitimate need to hide certain parts of their namespace from most of the world while making it available to a limited audience.

The BIND 9 *allow-query* substatement lets you apply an IP address-based access control list to queries. The access control list can apply to queries for data in a particular zone or to any queries received by the name server. In particular, the access control list specifies which IP addresses are allowed to send queries to the server.

The global form of the allow-query substatement looks like this:

```
options {
      allow-query { address_match_list; };
};
```

So to restrict our name server to answering queries only from our internal network, *zos12.ral.ibm.com*, we'd use:

```
options {
      allow-query { 9.12.6/24;};
};
```

BIND 9 also allow you to apply an access control list to a particular zone. In this case, just use *allow-query* as a substatement to the *zone* statement for the zone you want to protect:

```
acl "zos12-net" { 9.12.6/24; };
 zone "zos12.ral.ibm.com" {
      type master;
      file "zos12.for.v9";
      allow-query { "zos12-net"; };
};
```

Any kind of authoritative name server, master or slave, can apply an access control list to the zone. Zone-specific access control lists take precedence over a global ACL for queries in that zone. The zone-specific access control list may even be more permissive than the global ACL. If there's no zone-specific access control list defined, any global ACL will apply.

Any attempt of querying a record coming from an unauthorized host will not receive an answer, and the request will timeout with no further notice.

10.5.2 Preventing unauthorized zone transfers

Even more important than controlling who can query your name server is ensuring that only your real slave name servers can transfer zones from your name server. Users on remote hosts that can query your name server's zone data can only look up records (e.g., addresses) for domain names they already know, one at a time. Users who can start zone transfers from your server can list all of the records in your zones.

BIND 8 and BIND 9's *allow-transfer* substatement let administrators apply an access control list to zone transfers. *allow-transfer* restricts transfers of a particular zone when used as a *zone* substatement, and restricts all zone transfers when used as an *options* substatement. It takes an address match list as an argument.

The slave server for our *zos12.ral.ibm.com* zone have the IP addresses 9.12.6.63 (sc64.zos12.ral.ibm.com). The following zone statement:

```
zone "zos12.ral.ibm.com" in {
    type master;
    file "zos12.for.v9";
    allow-query { "zos12-net";};
    allow-transfer { 9.12.6.63; };
```

allows only this slave to transfer *zos12.ral.ibm.com* records from the primary master name server. Note that because the default for BIND 9 is to allow zone transfer requests from any IP address, and because hackers can just as easily transfer the zone from your slaves, you should probably also have a zone statement like this on your slaves:

```
zone "zos12.ral.ibm.com" in {
   type slave;
   file "zos12.for.bak.v9";
   masters { 9.12.6.68; };
   allow-transfer {none; };
};
```

BIND 9 also let you apply a global access control list to zone transfers. This applies to any zones that don't have their own explicit access control lists defined as zone substatements. For example, we might want to limit all zone transfers to our internal IP addresses:

```
options {
      allow-transfer { 9.12.6/24; };
};
```

10.5.3 Creating a transaction signature between master and slave

The next step to increase our secure environment is to use TSIG to let you restrict zone transfers to slave name servers that include a correct transaction signature with their request. On the master name server, generate a key, *sc63-sc64*, using the command *dnssec-keygen* as follows:

```
OCTAVIO @ SC63:/>dnssec-keygen -a hmac-md5 -b 128 -n HOST sc63-sc64
start typing:
...............................
...........................
stop typing.
Ksc63-sc64.+157+35544
```

This generates two key files:

```
Ksc63-sc64.+157+35544.key
Ksc63-sc64.+157+35544.private
```

The contents of these files is shown in Figure 10-69:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: 8kOIjL3m4sSO13DBNs6S6w==
```

Figure 10-69 Contents of file Ksc63-sc64.+157+37860.private

Get the key you generated, insert it in a key statement and then specify the key in the address match list as shown next:

```
key sc63-sc64 {
algorithm "hmac-md5";
secret "8kOIjL3m4sSO13DBNs6S6w==";
};
server 9.12.6.67 {
keys { sc63-sc64; };
};
zone "zos12.ral.ibm.com" in {
    type master;
    file "zos12.for.v9";
    allow-query { "zos12-net";};
    allow-transfer { key dns63-dns64.; };
};
```

On the slave's end, you need to configure the slave to sign zone transfer requests with the same key:

```
key sc63-sc64 {
algorithm "hmac-md5";
secret "8kOIjL3m4sSO13DBNs6S6w==";
};
server 9.12.6.68 {
keys { sc63-sc64; };
};
zone "zos12.ral.ibm.com" {
type slave;
masters { 9.12.6.68; };
file "zos12.for.bak.v9";
};
```

For a primary master name server accessible from the Internet, you probably want to limit zone transfers to just your slave name servers. You probably don't need to worry about name servers inside your firewall, unless you're worried about your own employees listing your zone data.

10.5.4 Signing your zone

The last and higher level of security to implement in our DNS environment is to use DNSSEC to sign your zone. We'll show you how we signed our DNS scenario, which is the zone called *zos12.ral.ibm.com*. All process will be done using the BIND 9 tools, as follows:

10.5.4.1 Generating your key pair

First, we generated a key pair for zone zos12.ral.ibm.com using dnssec-key:

```
OCTAVIO @ SC63:>dnssec-keygen -a DSA -b 512 -n ZONE zos12.ral.ibm.com
start typing:
...............................
...........................
stop typing.
Kzos12.ral.ibm.com.+003+09520
```

We ran dnssec-keygen in our name server's working directory. That's mostly for convenience: the zone data files are in this directory, so we won't need to use full pathnames as arguments. If we want to use dynamic update with DNSSEC, though, we'd need the keys in the name server's working directory.

Recall dnssec-keygen's options from the TSIG section of this chapter:

-a The cryptographic algorithm to use, in this case RSA. We could also have used DSA, but RSA is more efficient.

-b The length of the keys to generate, in bits. RSA keys can be anywhere from 512 to 2000 bits long. DSA keys can be 512 to 1024 bits long, as long as the length is divisible by 64.

-n The type of key. DNSSEC keys are always ZONE keys.

The only non-option argument is the domain name of the zone, zos12.ral.ibm.com. The dnssec-keygen program prints the basename of the files it's written the keys to. The numbers at the end of the basename (+003 and +09520), as we explained in the TSIG section, are the key's DNSSEC algorithm number as used in the KEY record (003 is DSA/MD5), and the key's fingerprint (+09520), used to distinguish one key from another when multiple keys are associated with the same zone.

The public key is written to the file *basename.key* (Kzos12.ral.ibm.com.+003+09520.key). The private key is written to the file *basename.private* (Kzos12.ral.ibm.com.+003+09520.key). Remember to protect the private key; anyone who knows the private key can forge signed zone data. dnssec-keygen does what it can to help you: it makes the .private file readable and writable only by the user who runs it.

10.5.4.2 Sending your keys to be signed (optional)

Next, we sent our KEY record to the administrator of our parent zone to sign. BIND 9 includes a program to package up the key for transmission, *dnssec-makekeyset*:

```
# > dnssec-makekeyset -t 172800 Kzos12.ral.ibm.com.+003+09520.key
```

After executed *dnssec-makekeyset* created a file called keyset-zos12.ral.ibm.com.. Its contents are shown in Figure 10-70.

```
$ORIGIN .
$TTL 172800 ; 2 days
zos12.ral.ibm.com IN KEY 256 3 3 (
    AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbEERPh9rWy
    MlQbHuUBXTHdU/wsIvBr7omnRUXvLQxXEF4mLUnILL7j
    G5yncdRBo1KR7XTDkzGI9CW4qU4UGpH98QnbnZupGXDw
    nrHs8pK7stgssfjniShyUAbzVVnSxCtCHC7EQbnvIKdf
    jI5pydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ/+lh
    TjUKTZGYFDc57i6yMPNErsP8DI3GwQ41ONaP2OtCGMTJ
    YtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
   SIG KEY 3 4 172800 20020626173942 (
    20020527173942 9520 zos12.ral.ibm.com.
    AHUWn73ryAc3TInk8OlCso/1Mb3QFO2026Xop+8nibcK
    WrtY24+DJsE= )
```

Figure 10-70 Contents of keyset-zos12.ral.ibm.com. file

The -t option takes a TTL for the records to submit. This serves as a suggestion to your parent zone's administrator of the TTL (in seconds) you'd like for your record. They may ignore it, of course. The SIG record actually contains a signature covering your zone's KEY record, generated with your zone's private key. That proves you really have the private key that corresponds to the public key in the KEY record--you're not just submitting a KEY record you found on the street.

Then we sent our file off to our parent zone's administrators to sign. Since the message included proof of our identity, they signed it with the dnssec-signkey program:

```
# dnssec-signkey keyset-zos12.ral.ibm.com Kral.ibm.com.+003+64185.private
```

and sent the resulting file, signedkey-zos12.ral.ibm.com, back to us. Its contents are shown in Figure 10-71.

```
$ORIGIN .
$TTL 172800 ; 2 days
zos12.ral.ibm.com IN KEY 256 3 3 (
    AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbEERPh9rWy
    MlQbHuUBXTHdU/wsIvBr7omnRUXvLQxXEF4mLUnILL7j
    G5yncdRBo1KR7XTDkzGI9CW4qU4UGpH98QnbnZupGXDw
    nrHs8pK7stgssfjniShyUAbzVVnSxCtCHC7EQbnvIKdf
    jI5pydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ/+lh
    TjUKTZGYFDc57i6yMPNErsP8DI3GwQ41ONaP2OtCGMTJ
    YtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
   SIG KEY 3 4 172800 20020626173942 (
    20020527173942 64185 ral.ibm.com.
    AEvxWeXonAFwpaQp9nYhI5GDqafcHSTblwL1oAh+841m
    4FZzImMpJF4= )
```

Figure 10-71 signedkey-zos12.ral.ibm.com. contents

If we did not care about getting our KEY record signed, we could have skipped this step. But then only name servers with a trusted-keys entry for zos.ral.ibm.com could verify our data.

10.5.4.3 Signing your zone

Before signing our zone, we had to include a reference to the key file into the our zone data file (zos12.for.v9) using the *$INCLUDE* statement:

```
$INCLUDE Kzos12.ral.ibm.com.+003+09520.key
```

This gives the signer program the information it needed to know which key to use to sign the zone. It automatically finds and includes the contents of *signedkey-zos12.ral.ibm.com*.

Then we signed the zone with *dnssec-signzone*:

```
# dnssec-signzone -o zos12.ral.ibm.com. zos12.for.v9
```

We used the -o option to specify the origin in the zone data file, because dnssec-signzone doesn't read named.conf to determine which zone the file describes. The only non-option argument is the name of the zone data file.

This produces a new zone data file, zos12.for.v9.signed. A partial contents of this file is shown in Figure 10-72.

```
; File written on Mon. May 27 14:26:46 2002
; dnssec_signzone version 9.1.1
 zos12.ral.ibm.com. 86400 IN SOA dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
      4        ; serial
      10800    ; refresh (3 hours)
      3600     ; retry (1 hour)
      604800   ; expire (1 week)
      86400    ; minimum (1 day)
    )
    86400 SIG SOA 3 4 86400 20020626182646 (
      20020527182646 9520 zos12.ral.ibm.com.
      AF2UFK1AP5eTSlpvgayWUN84rvREMzM3paa3
      /KJwdJ/EZOvnAuIAuc4= )
    86400 NS dns63.zos12.ral.ibm.com.
   86400 NS dns63.zos12.ral.ibm.com.
   86400 NS dns64.zos12.ral.ibm.com.
   86400 SIG NS 3 4 86400 20020626182646 (
     20020527182646 9520 zos12.ral.ibm.com.
     AH1qWnUz5L9xVKBEOc6twGTXyp6BKLPtNRJo
     PEhuc2GXBvgHhYwssj4= )
   86400 KEY 256 3 3 (
     AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbE
     ERPh9rWyMlQbHuUBXTHdU/wsIvBr7omnRUXv
     LQxXEF4mLUnILL7jG5yncdRBo1KR7XTDkzGI
     9CW4qU4UGpH98QnbnZupGXDwnrHs8pK7stgs
     sfjniShyUAbzVVnSxCtCHC7EQbnvIKdfjI5p
     ydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ
     /+lhTjUKTZGYFDc57i6yMPNErsP8DI3GwQ41
     ONaP2OtCGMTJYtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
   172800 SIG KEY 3 4 172800 20020626173942 (
     20020527173942 64185 ral.ibm.com.
     AEvxWeXonAFwpaQp9nYhI5GDqafcHSTblwL1
     oAh+841m4FZzImMpJF4= )
```

Figure 10-72 Partial contents of zos12.for.v9.signed file

Finally, we changed the zone statement in named.conf so that named would load the new zone data file:

```
zone "zos12.ral.ibm.com" in {
    type master;
    file "zos12.for.v9.signed";
    allow-query { "zos12-net";};
    allow-transfer { key dns63-dns64.; };
    allow-update { 9.12.6.67;9.12.6.68;9.12.6.63; };
};
```

Then we reloaded the zone using **rndc reload** command and checked syslog.

10.5.4.4 DNSSEC and dynamic update

dnssec-signzone is not the only way to sign zone data. The BIND 9 name server is capable of signing dynamically updated records on the fly. As long as the private key for a secure zone is available in the name server's working directory (in the correctly named .private file), a BIND 9 name server signs any records that are added via dynamic update. If any records are added to or deleted from the zone, the name server adjusts (and re-signs) the neighboring NXT records, too. The following steps show how it is done:

- We first query the name server about a host called newhost using *dig* command:

```
/> dig +dnssec newhost.zos12.ral.ibm.com.
```

The result of the command will be:

```
OCTAVIO @ SC63:/>dig +dnssec newhost.zos12.ral.ibm.com
Allocated socket 5, type udp

; <<>> DiG 9.1.1 <<>> +dnssec newhost.zos12.ral.ibm.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 49597
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version:   0, udp=  4096
;; QUESTION SECTION:
;newhost.zos12.ral.ibm.com.     IN      A

;; AUTHORITY SECTION:
zos12.ral.ibm.com.      86400   IN      SOA     dns63.zos12.ral.ibm.com.
admin.zos12.ral.ibm.com.
zos12.ral.ibm.com.      86400   IN      SIG     SOA 3 4 86400 20020626182646
20020527182646 9520 z
gayWUN84rvREMzM3paa3/KJwdJ/EZOvnAuIAuc4=
mail.zos12.ral.ibm.com. 86400   IN    NXT    sc63.zos12.ral.ibm.com. CNAME SIG NXT
mail.zos12.ral.ibm.com. 86400   IN      SIG     NXT 3 5 86400 20020626182646
20020527182646 9520 z
;; Query time: 2 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Mon May 27 15:23:29 2002
;; MSG SIZE  rcvd: 640
```

Note that we trimmed the output a little. Notice the mail.zos12.ral.ibm.com NXT record, which is the last record in our domain file, indicating that the domain name doesn't exist. Now we'll use nsupdate to add an address record for *newzone.zos12.ral.ibm.com*:

```
/>nsupdate -V v9
> update add newhost.zos12.ral.ibm.com. 3600 IN A 9.12.6.61
```

Now, we look up newhost.zos12.ral.ibm.com again:

```
% dig +dnssec newhost.zos12.ral.ibm.com.
OCTAVIO @ SC63:/>dig +dnssec newhost.zos12.ral.ibm.com
Allocated socket 5, type udp

; <<>> DiG 9.1.1 <<>> +dnssec newhost.zos12.ral.ibm.com
;; global options:  printcmd
; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11973
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 9
;; QUESTION SECTION:
;newhost.zos12.ral.ibm.com.     IN      A
;; ANSWER SECTION:
newhost.zos12.ral.ibm.com. 3600    IN      A       9.12.6.61
newhost.zos12.ral.ibm.com. 3600    IN      SIG     A 1 3 3600 20010215195456
20010116185456 27791 zos12.ral.ibm.com.
C/JXdCLUdugxN91vODZuUDTusi2XNNttb4bdB2nBujLxjwwPAf/D5MJz
//cDtuZ3X+uYzhkN8MDROqOwUQuQSA==

;; AUTHORITY SECTION:
```

```
zos12.ral.ibm.com.      86400   IN      SOA     dns63.zos12.ral.ibm.com.
admin.zos12.ral.ibm.com.
zos12.ral.ibm.com.      86400   IN      SIG     SOA 3 4 86400 20020626182646
20020527182646 9520 z
gayWUN84rvREMzM3paa3/KJwdJ/EZOvnAuIAuc4=
mail.zos12.ral.ibm.com. 86400  IN    NXT   sc63.zos12.ral.ibm.com. CNAME SIG NXT
mail.zos12.ral.ibm.com. 86400   IN      SIG     NXT 3 5 86400 20020626182646
20020527182646 9520 z
;; Query time: 2 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Mon May 27 15:23:29 2002
;; MSG SIZE  rcvd: 640
```

Again, we trimmed the output a little. Now, not only was an address record generated, but there is also a SIG record generated from movie.edu's private key. As impressive as this is, you should be careful when allowing dynamic updates to secure zones. You should make sure that you use strong authentication (e.g., TSIG) to authenticate the updates, or you will give a hacker an easy backdoor to use to modify your "secure" zone. And you should ensure you have enough horsepower for the task: normally, dynamic updates do not take much to process. But dynamic updates to a secure zone require NXT recalculation and, more significantly, asymmetric encryption (to calculate new SIG records), so you should expect your name server to take longer and need more resources to process them.

10.6 Running DNS in BIND 9 and BIND 4.9.3 simultaneously

Because each name server version supports some functions that the other does not, you may wish to run both name servers on the same TCP/IP stack. This is accomplished by having each name server listen on a different set of interfaces. When running the name server in both BIND 9 and BIND 4.9.3 modes, the same requirements and restrictions of running them separately apply.

10.6.1 Compatibility considerations

Zone transfers

It is not recommended to have a BIND 4.9.3- DNS act as a slave name server to a BIND 9 master. However, if required, it can be done under certain conditions. If the BIND 9 master contains any resource records (RRs) that BIND 4.9.3-DNS does not understand, the zone will fail to load. A BIND 9 nameserver with BIND 4 slaves should specify the 'transfer-format one-answer' option in its named.conf. Otherwise, any transfer will also fail.

BIND 4.9.3 does not understand NOTIFY, therefore, if BIND 4.9.3 is running as a slave name server to a BIND 9 master, the DNS Change Notification protocol will not work. The standard method of zone transfers applies, where the slave periodically polls the master for an updated SOA serial number.

Queries

BIND 4.9.3 can participate in DNS queries when BIND 9 name servers are in the DNS tree structure, even when the queries are for RR types that BIND 4.9.3 does not understand. For example, if a resolver is pointed to a BIND 4.9.3-DNS and is asked for an AAAA record (an RR type that BIND 4.9.3 name server does not understand), BIND 4.9.3-DNS will recursively query other (possibly BIND 9) name servers and return the answer to the client. The BIND 4.9.3-DNS will not cache the response (in the case of RR types it does not understand), which may mean a little more network traffic. This caching issue may or may not be significant depending on your particular network traffic patterns.

Dynamic update

BIND 4.9.3 version ofdynamic update is incompatible with the BIND 9 version. For dynamic update on BIND 4.9.3-DNS, use nsupdate with -V v4 start option. For dynamic update on BIND 9-DNS, use nsupdate with -V v9 start option. Additionally, only the DHCP server on z/OS and OS/2 can successfully dynamically update the BIND 4.9.3-DNS.

DNSSEC

BIND 4.9.3 does not support DNSSEC.

TSIG

BIND 4.9.3 does not support TSIG security which may be used on queries, update and zone transfers on BIND 9 name servers.

DNS/WLM (Sysplex Connection balancing)

Not supprorted by a name server running in BIND 9 mode.

IPV6 Support

BIND 4.9.3 name server does not fully suport IPv6.

Stack Affinity

The BIND 9 name server is a generic server, unlike the BIND 4.9.3 name server which has stack affinity. If stack affinity is desired for the BIND 9 name server, use the _BPXK_SETIBMOPT_TRANSPORT environment variable.

NOTIFY

This function notifies the slave of a change in the master. It is not supported by a BIND 4.9.3 name server.

10.6.2 Implementation procedure

To implement the scenario shown in, see section 10.2, “Setting up a BIND 4.9.3-based Domain Name Server” on page 323 and section 10.4, “Setting up a BIND 9-based Domain Name Server” on page 355 for details about the steps to follow to implement each one of them, making sure to follow these tasks to allow both BINDs to run simultaneously:

10.6.2.1 Specify port ownership

Allow the name server port (53) to be shared by removing any PORT reservations for port 53 or else reserve PORT 53 for TCP for both jobnames. Ensure that PORT 53 TCP port reservations have a suffix of 2 for BIND 4.9.3 and a suffix of 1 for BIND 9 if they are started from a procedure. To implement our scenario, specify one of the following:

a. reserve TCP port 53 to each procedure:

```
PORT 53 TCP NAMED42 ;BIND 4.9.3
PORT 53 TCP NAMED91 ;BIND 9
```

Note: UDP PORT reservation for multiple jobnames is not allowed.

b. Reserve TCP and UDP port 53 to OMVS:

```
PORT 53 TCP OMVS ; Reserved to OMVS
PORT 53 UDP OMVS ; Reserved to OMVS
```

This might be the best choice if you are going to start the name servers from z/OS UNIX.

10.6.2.2 Bind each name server to its own set of IP interfaces

- Bind the BIND 9 name server to the set of interfaces you wish to be serviced by the BIND 9 name server using the 'listen-on{}' 1 option in the named.conf file.

```
options {
directory "/etc/octavio/dnsdata";
1 listen-on { 9.12.6.68; };
```

- Bind the BIND 4.9.3 name server to the interface you wish to be serviced by the BIND 4.9.3 name server using the BIND option on the PORT statement in the TCPIP.PROFILE for the name server's port. This should use the job name of the BIND 4.9.3 name server. 1

```
PORT
    20 TCP OMVS NOAUTOLOG    ; FTP SERVER DATA PORT
    21 TCP FTPDB1            ; FTP SERVER CONTROL PORT
    53 TCP NAMED91           ; RESERVE PORT TO BIND 9 DNS
 1  53 TCP NAMED42 BIND 9.12.6.67 ;RESERVE PORT TO BIND 4.93 DNS
```

10.6.2.3 Assign unique job names

Assign unique job names to the two name servers and correlate those names with the job names used in the previous steps. Ensure that the jobnames match those on the PORT statement, with the exception of the numerical suffix. Port reservation will not work if named is started from z/OS UNIX, unless the ports are reserved with the name OMVS. In our scenario we use *NAMED4* and *NAMED9* to specify each BIND's procedure.

10.6.2.4 Store the name server process IDs (PIDs) in unique files

Configure the BIND 9 name server to store the process ID (PID) in a file other than that one used for the BIND 4.9.3 name server (/etc/named.pid). This is done with the 'pid-file' named.conf file option. 1

```
options {
pid-file "/etc/named.v9.pid";  1
    directory "/etc/octavio/dnsdata";
    listen-on { 9.12.6.68; };
};
```

10.6.2.5 Change the loopback address for BIND 4.9.3 Name Server

BIND 9 and BIND 4 server modes coexistence on the same host requires different loopback addresses. If BIND 9 and BIND 4.9.3 are to be started on the same host, once BIND 9 mode is started, BIND 4 mode server will have to use a different loopback address. BIND 4 server master forward and loopback zone files should define A and PTR resource records, respectively, for the appropriate loopback addresses, as shown:

a. In the forward file:

```
localhost          IN   A    127.0.0.2
```

b. In the loopback file:

```
0.0.127.in-addr.arpa.   IN   NS  dns63.wlm.zos12.ral.ibm.com.
2.0.0.127.in-addr.arpa. IN   PTR localhost.
```

10.6.2.6 Forward the queries to the correct name server

When you implement an environment where you have 2 Name Servers working simultaneously you need to be sure your clients are going to reach the right server, by configure clients' resolver configuration data set or HFS file so that it points to the interface or interfaces of the desired name server. This is typically specified by the NSINTERADDR statement, or an equivalent statement. Another way to do it is to forward the queries that does not belong to one server to the right by defining the forwarder option on the options statement as shown in Figure 10-73:

```
 options {
    pid-file "/etc/named.v9.pid";
    directory "/etc/octavio/dnsdata";
    listen-on { 9.12.6.68; };
    forwarders { 9.12.6.67; };
};
```

Figure 10-73 Forwarders definition in named.conf file

10.6.2.7 Verifying the implementation

After you start both name servers, you can check if everything is up as planned, using these options:

- Check the startup messages for both servers in the console log as shown in Figure 10-74 and Figure 10-75:

```
$HASP100 NAMED4   ON STCINRDR
IEF695I START NAMED4   WITH JOBNAME NAMED4   IS ASSIGNED TO
$HASP373 NAMED4   STARTED
EZZ9166I STARTING NAMED, BIND V4
EZZ6697I NAMED STARTING
EZZ6475I NAMED:  READY TO ANSWER QUERIES.
IEF404I NAMED4 - ENDED - TIME=16.23.47 - ASID=006C - SC63
$HASP395 NAMED4   ENDED
```

Figure 10-74 Startup messages for named V4 (partial contents).

```
S NAMED9
$HASP100 NAMED9   ON STCINRDR
IEF695I START NAMED9   WITH JOBNAME NAMED9   IS ASSIGNED TO
$HASP373 NAMED9   STARTED
+EZZ9095I STARTING NAMED, BIND 9.1.1
IEF404I NAMED9 - ENDED - TIME=16.23.41 - ASID=006C - SC63
$HASP395 NAMED9   ENDED
+EZZ9130I NAMED, BIND 9.1.1 IS RUNNING
```

Figure 10-75 Startup messages for named V9 (partial contents)

- Confirm that each Server has opened the interface they were allowed to bind, by using the command **`netstat conn`**. The partial results of the command shown in Figure 10-76.

```
 D TCPIP,TCPIPB,N,CONN
 EZZ2500I NETSTAT CS V1R2 TCPIPB 866
 USER ID  CONN     LOCAL SOCKET          FOREIGN SOCKET         STATE
NAMED42  0000062B 9.12.6.67..53         0.0.0.0..0             LISTEN
NAMED91  00000628 9.12.6.68..53         0.0.0.0..0             LISTEN
NAMED42  0000062C 0.0.0.0..53           *..*                   UDP
NAMED91  00000627 9.12.6.68..53         *..*                   UDP
```

Figure 10-76 Partial results of netstat conn command

- Check that the queries are being answered by the correct servers using nslookup in the host as follows:

  a. To see if the BIND 4 is accepting queries, enter the nslookup with the server_address parameter and the host_name you want to check:

```
OCTAVIO @ SC63:/>nslookup host1.wlm.ral.ibm.com. 9.12.6.67
Defaulting to nslookup version 4
Starting nslookup version 4
Server:  host1.wlm.ral.ibm.com
Address:  9.12.6.67
Name:    host1.wlm.ral.ibm.com
Address:  9.12.6.67
```

  b. To see if the BIND 9 is accepting queries, the nslookup with the server_address parameter and the host_name you want to check:

```
OCTAVIO @ SC63:/>nslookup -V=v9 host1.zos12.ral.ibm.com 9.12.6.68
Running nslookup version 9
Allocated socket 5, type udp
Server:         9.12.6.68
Address:        9.12.6.68#53
Name:   host1.zos12.ral.ibm.com
Address: 9.12.6.67
```

  c. Check if the forwarder is sending your query to the right server, assuming that your workstation uses BIND9 as primary DNS:

```
OCTAVIO @ SC63:/>nslookup -V=v9 host1.wlm.ral.ibm.com 9.12.6.68
Running nslookup version 9
Allocated socket 5, type udp
Server:         9.12.6.68
Address:        9.12.6.68#53

Non-authoritative answer:    1
Name:   host1.wlm.ral.ibm.com
Address: 9.12.6.67
```

  Looking at the resulting messages, you can confirm that you are receiving your answer from the other server, as expected 1.

10.7 DNS tools

There are several diagnostics, administrative and monitoring tools available to the system administrator for controlling and debugging the nameserver daemon. z/OS V1R2 Communications Server has improved some tools and implemented new functions to enhance the control over a Domain Name Server. In the next sections, we are going to describe the most used ones.

10.7.1 Administrative tools

This section describes the Domain Name System (DNS) administrative tools you can use to control, maintain and verify domain name servers, resolvers, and resource records. z/OS V1R2 Communication Server implemented the following commands:

- **`nsupdate`** -
- dnssec commands (**`keygen`**, **`makekeyset`**, **`signkey`**, and **`signzone`**) related to DNS security
- **`rndc`** command to remotely control a name server
- **`dnsmigrate`** command to convert named.boot file syntax into named.conf file syntax

10.7.1.1 Using the z/OS UNIX nsupdate command

You can use **`nsupdate`** to create and execute DNS update operations on a host record as defined in RFC 2136 (for DNS 9) or RFC 2065 (for DNS 4.9.3) to a name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Note: Zones that are under dynamic control via nsupdate or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

- BIND v4

You can use nsupdate command in an interactive fashion (where you are prompted through a series of subcommands and associated input values), or if you know the sequence of operations and input values beforehand, you can use nsupdate in batch mode and specify a subcommand sequence in the -s command line parameter. The search order locations and order of priority from which the values for version BIND 4.9.3 nsupdate options can be specified are:

  – nsupdate command options

  – nsupdaterc file in the home directory

  – environment variable (LOCALDOMAIN)

- BIND v9

  The resource records for nsupdate using BIND 9 that are dynamically added or removed with nsupdate have to be in the same zone. Requests are sent to the zone's master server. This is identified by the MNAME field of the zone's SOA record. Batch mode is indirectly provided when input is redirected to a file, such as

  ```
  nsupdate -V v9 < /tmp/update.zone.
  ```

  A valid series of nsupdate commands must be in the file, one command per line, prior to nsupdate invocation.

  BIND 9 DNS uses the z/OS application's search order to find TCPIP.DATA statements. It uses the following directives from the resolver configuration file:

  – nameserver/nsinteraddr

  – options ndots:n

  – search domain/domainorigin

The examples below show how nsupdate with BIND 9 could be used to insert and delete resource records from the example.com zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for example.com.

```
# nsupdate
> update delete host1.zos12.ral.ibm.com.com A
> update add newhost.zos12.ral.ibm.com 86400 A 9.12.6.67
>
```

Any *A* records for host1.zos12.ral.ibm.com are deleted, and an A record for newhost.zos12.ral.ibm.com at IP address 9.12.6.67 is added. The newly added record has a 1-day TTL (86400 seconds).

10.7.1.2 Using the z/OS UNIX dnssec-keygen command

The **dnssec-keygen** command generates keys for DNSSEC, Secure DNS, as defined in RFC 2535. It also generates keys for use in Transaction Signatures (TSIG) which is defined in RFC 2845. The dnssec-keygen command can only be run from the z/OS UNIX shell. If dnssec-keygen is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

To generate a 768-bit DSA key for the domain example.com, the following command would be issued:

```
# dnssec-keygen -a DSA -b 768 -n ZONE example.com Kexample.com.+003+26160
```

dnssec-keygen has printed the key identification string Kexample.com.+003+26160,indicating a DSA key with identifier 26160. It will also have created the files Kexample.com.+003+26160.key and Kexample.com.+003+26160.private containing the public and private keys for the generated DSA key.

10.7.1.3 Using the z/OS UNIX dnssec-makekeyset command

The **dnssec-makekeyset** command, used to configure the BIND 9 DNS DNSSEC feature, creates a key set file. A key set contains all of the keys containing KEY and SIG records for some zone which can then be signed by the zone's parent, if the parent zone is DNSSEC-aware. The **dnssec-makekeyset** command may only be run from the z/OS UNIX shell. If dnssec-makekeyset is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

The following command generates a key set for the DSA key for example.com.

```
# dnssec-makekeyset -t 86400 -s 20000701120000 -e +2592000 Kexample.com.+003+26160
```

dnssec-makekeyset creates a file called keyset-example.com. containing a SIG and KEY record for example.com. These records will have a TTL of 86400 seconds (1 day). The SIG record becomes valid at noon UTC on July 1st 2000 and expires 30 days (2592000 seconds) later.

The DNS administrator for example.com could then send keyset-example.com. to the DNS administrator for .com to sign the resource records in the file. This assumes that the .com zone is DNSSEC-aware and the administrators of the two zones have some mechanism for authenticating each other and exchanging the keys and signatures securely.

10.7.1.4 Using the z/OS UNIX dnssec-signkey command

The **dnssec-signkey** command, used to configure DNSSEC security features for a BIND 9 name server, signs one child's keyset with the parent zone's private key. The dnssec-signkey command may only be run from the z/OS UNIX shell.

If **dnssec-signkey** is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments. The DNS administrator for a DNSSEC-aware .com zone would use the following command to make **dnssec-signkey** sign the keyset file for example.com:

```
# dnssec-signkey keyset-example.com. Kcom.+003+51944
```

where `Kcom.+003+51944` was a key file identifier produced when **dnssec-keygen** generated a key for the .com zone. **dnssec-signkey** produces a file called *signedkey-example.com*. which has the keys for example.com signed by the com zone's zone key.

10.7.1.5 Using the z/OS UNIX dnssec-signzone command

The **dnssec-signzone** command, used to configure the DNSSEC security feature for BIND 9 name servers, signs zones with the keys generated by **dnssec-keygen**. By signing zones with a private key, users of that data which have the public key or can securely obtain the public key can be assured that the data is authentic. The **dnssec-signzone** command may only be run from the z/OS UNIX shell.

The following example shows how dnssec-signzone could be used to sign the zone file. The zone file for this zone is example.com, which is the same as the origin, so there is no need to use the -o option to set the origin. This zone file contains the keyset for example.com that was created by dnssec-makekeyset. The zone's keys were either appended to the zone file or incorporated using a $INCLUDE statement. If there was a signedkey file from the parent zone (signedkey-example.com.), it should be present in the current directory. This allows the parent zone's signature to be included in the signed version of the example.com zone.

```
# dnssec-signzone example.com Kexample.com.+003+26160
```

dnssec-signzone will create a file called example.com.signed, the signed version of the example.com zone. This file can then be referenced in a zone{} statement in /etc/named.conf so that it can be loaded by the name server.

10.7.1.6 Using the z/OS UNIX dnsmigrate command

The **dnsmigrate** command is a migration aid that will convert *named.boot* files for the BIND.4.9.3 mode, into *named.conf* files suitable for the BIND.9 mode. The dnsmigrate command may only be run from the z/OS UNIX shell.

The following command, using **dnsmigrate** without parameters, converts the default input file (/etc/named.boot) and writes the results to the default output file (/etc/named.conf).

```
dnsmigrate
```

The following converts the specified input file (/tmp/named.boot) to the specified output file (/tmp/named.conf). Note that it does not matter what order the input and output file options are used.

```
dnsmigrate -o /tmp/named.conf -i /tmp/named.boot
```

10.7.1.7 Using the z/OS UNIX rndc command

Remote Name Daemon Control (rndc) command allows the system administrator to control the operation of a name server. If **rndc** is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments. **rndc** may only be used with the v9 (BIND9) name server and will not function with the v4 (BIND.4.9.3) name server

The function of **rndc** may be used as a secure remote client to control the name server. Some local UNIX signal functions for the name server can be replaced by equivalent **rndc** functions. The name server and **rndc** communicates over a TCP connection, sending commands authenticated with digital transaction signatures *(TSIG)*. This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server. Therefore, **rndc** and the name server must be configured with a *'shared-secret'*. This shared secret is a TSIG key, which may be generated with the **dnssec-keygen** utility. The only supported encryption algorithm for rndc is HMAC-MD5, which uses a shared secret on each end of the connection. The functions available are:

- Reload configuration file and zones.
- Write server statistics to the statistics file.
- Toggle query logging.
- Dump the current contents of the cache.
- Stop the server.

If you run rndc without any options it will display a usage message as follows:

```
rndc [-c config] [-s server] [-p port] [-y key] [-z zone] [-v view] command [command ...]
```

A configuration file is required, because all communication with the server is authenticated with digital signatures that rely on a shared secret, and there is no way to provide that secret other than with a configuration file. The default location for the rndc configuration file is /etc/rndc.conf, but an alternate location can be specified with the -c option. The format of the configuration file is similar to that of named configuration file, but limited to only three statements, the *options*, *key* and *server* statements. These statements are what associate the secret keys to the servers with which they are meant to be shared. The order of statements is not significant.

The *options* statement has two clauses: *default-server* and *default-key*. If *default-server* is specified, the value takes a host name or address argument and represents the server that will be contacted if no *-s* option is provided on the command line. Alternatively, *default-key* takes the name of key as its argument, as defined by a key statement. In future releases, a *default-port* clause will be added to specify the port to which **rndc** should connect.

The *server* statement uses the *key* clause to associate a key-defined key with a server. The argument to the *server* statement is a host name or address (addresses must be enclosed in double quotes). The argument to the *key* clause is the name of the key as defined by the *key* statement. A *port* clause will be added to a future release to specify the port to which **rndc** should connect on the given server.

A implement a sample minimal configuration to allow the local host to control the local Domain Name Server using a key named rndc-dns63, follow these steps:

1. Generate the shared secret key (a TSIG key) using the command **dnssec-keygen:**

   ```
   dnssec-keygen -a hmac-md5 -b 128 -n HOST rncd-dns63
   ```

the dnssec-keygen command generates 2 files:

```
Krndc-dns63.+157+22612.key
Krndc-dns63.+157+22612.private
```

2. Open the file *Krndc-dns63.+157+22612.private* and copy the generated key 1:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: iRIgqf6MhdHFCKl2FzqkgA== 1
```

3. Configure the rndc.conf file on the host you whish to run **rndc** from. the contents of the file should be the key parameter, the server parameter and the options parameter, which defines the default key and server, as follows:

```
 key rndc-dns63 {
      algorithm "hmac-md5";
   1  secret "iRIgqf6MhdHFCKl2FzqkgA==";};
 server dns63.zos12.ral.ibm.com {  2
       key rndc-dns63;};
 options { default-server localhost;
           default-key rndc-dns63; };
copy the key generated in step 1 as shown in 1
The server 2 parameter defines which server you want to contact and which key you are
going to use to contact it.
```

4. Add a *key{}* statement to the name server's named.conf file describing the key. you must use the same key generated in the rndc host and implemented in the rndc.conf file at that host:

```
 key rndc-dns63 {
      algorithm "hmac-md5";
      secret "iRIgqf6MhdHFCKl2FzqkgA==";};
```

5. Add a *control{}* statement to the name server's named.conf file as follows:

```
 controls {
      inet 127.0.0.1 allow {localhost;} keys {rndc-dns63;
};};
to connect to 127.0.0.1 port 953, using the key rndc-dns63 to authenticate the rndc
command. this key must match the one configured in rndc.config file.
```

6. reload the Name Server and test the command **rndc:**

```
rndc -c /etc/rndc.conf reload
```

10.7.2 DNS diagnostic tools

This section describes the available tools to diagnose a DNS problem:

- 10.7.2.1, "Checking messages sent to the operators console" on page 402
- "Checking syslog messages" on page 403
- Using Name Server Signals to Diagnose BIND 4.9.3-DNS Problems
- Using rndc to Diagnose BIND 9 Problems
- Using nslookup to Diagnose Problems
- Using dig to Diagnose Problems

10.7.2.1 Checking messages sent to the operators console

Messages displayed on the operators console indicate the status of your DNS. Messages fall into the following categories:

- Name server initialization
- Name server initialization failure
- Name server initialization complete
- Name server termination
- Assertion failures (unexpected errors) (v9 only)

Regularly check console messages to identify problems.

```
 S NAMED4
 $HASP100 NAMED4   ON STCINRDR
 IEF695I START NAMED4   WITH JOBNAME NAMED4   IS ASSIGNED TO USER STC
  , GROUP SYS1
 $HASP373 NAMED4   STARTED
 IEF403I NAMED4 - STARTED - TIME=11.12.25 - ASID=0066 - SC63
 -                                         --TIMINGS (MINS.)--
 EZZ9166I STARTING NAMED, BIND V4
 EZZ6452I NAMED STARTING. @(#) DDNS/NS/NS_MAIN.C, DNS_NS, DNS_R1.1 1.
 +EZZ6475I NAMED:  READY TO ANSWER QUERIES.
```

Figure 10-77 BIND 4.9.3 startup messages in the operator console

```
 S NAMED9
 $HASP100 NAMED9   ON STCINRDR
 IEF695I START NAMED9   WITH JOBNAME NAMED9   IS ASSIGNED TO USER STC
  , GROUP SYS1
 $HASP373 NAMED9   STARTED
 IEF403I NAMED9 - STARTED - TIME=14.10.15 - ASID=0066 - SC63
+EZZ9095I STARTING NAMED, BIND 9.1.1
 IEF404I NAMED9 - ENDED - TIME=14.10.17 - ASID=0066 - SC63
 -NAMED9   ENDED.  NAME-                     TOTAL CPU TIME= .00
```

BIND 9 DNS startup messages in tho operator console

10.7.2.2 Checking syslog messages

Error messages may also be displayed in the syslog output file, which is pointed to by the syslog configuration file. (/etc/syslog.conf is the default configuration file.) For BIND 9, refer to 10.4.9, “Configuring logging” on page 366. For the BIND 9 name server, initial startup messages go to syslog, later messages will be directed to other defined or default logs according to logging statements found or implied in the configuration file. Error, debug and informational messages can be written to the name server's logging files.

```
 /usr/lpp/tcpip/sbin/namedÝ83951713¨: EZZ9166I STARTING NAMED, BIND V4
 EZZ6452I named starting. @(#) ddns/ns/ns_main.c, dns_ns, dns_r1.1 1.62  9/23/97 10:57:2
 EZZ6701I named established affinity with 'TCPIPB'
 EZZ6540I Static primary zone 'zos12.ral.ibm.com' loaded (serial 1)
 EZZ6540I Static primary zone '6.12.9.in-addr.arpa' loaded (serial 1)
 EZZ6540I Static primary zone '0.0.127.in-addr.arpa' loaded (serial 1)
 EZZ6540I Static cache zone '' loaded (serial 0)
 EZZ6475I named:  ready to answer queries.
 EZZ6476I Return from getdtablesize() > FD_SETSIZE
```

Figure 10-78 BIND 4.9.3 DNS startup messages in the syslog file

```
EZZ9171I LPAR mode detected. Using 2 CPUs for -n option
EZZ9547I starting named, BIND 9.1.1 -d 1 -c /etc/octavio/named.co
EZZ9095I STARTING NAMED, BIND 9.1.1
EZZ9217I Running non-swappable
EZZ9540I using 2 CPUs
EZZ9126I loading configuration from '/etc/octavio/named.conf'
EZZ9542I zone 'zos12.ral.ibm.com' allows updates by IP address, w
EZZ9542I zone '6.12.9.in-addr.arpa' allows updates by IP address,
EZZ9542I zone '0.0.127.in-addr.arpa' allows updates by IP address
EZZ8842I the default for the 'auth-nxdomain' option is now 'no'
EZZ9052I no IPv6 interfaces found
EZZ9046I listening on IPv4 interface OSA22E0         , 9.12.6.67#
EZZ9046I listening on IPv4 interface VIPL090C0644    , 9.12.6.68#
EZZ9111I command channel listening on 127.0.0.1#953
EZZ9130I NAMED, BIND 9.1.1 IS RUNNING
```

Figure 10-79 BIND 9 DNS startup messages in the syslog file

10.7.2.3 Using nslookup to diagnose problems

The z/OS UNIX nslookup is a program used to query Internet domain name servers. nslookup has two modes: interactive and non-interactive. It also has two versions in z/OS UNIX: v4 and v9, where v4 gives the legacy z/OS UNIX onslookup function, and v9 gives the BIND 9 version of nslookup. Use the interactive mode to query name servers for information about various hosts and domains or to display a list of hosts (BIND 4) in a domain. Non-interactive mode is used to display just the name and requested information for a host or domain.

The z/OS UNIX onslookup/nslookup command enables you to perform the following tasks from the z/OS UNIX environment:

- Identify the location of name servers
- Examine the contents of a name server database
- Establish the accessibility of name servers

To display a list of options, enter the following from the command line:

```
onslookup -h
```

Note: The *onslookup* command is a synonym for the *nslookup* command in the z/OS UNIX shell. The *nslookup* command syntax is the same as that for the *onslookup* command. The *nslookup* command may be run from the z/OS UNIX shell or from TSO; however, only the legacy TSO version of *NSLOOKUP* is available from TSO.

```
OCTAVIO @ SC63:/>nslookup host1
Defaulting to nslookup version 4
Starting nslookup version 4
Server:  dns63.zos12.ral.ibm.com
Address:  9.12.6.68
Name:    host1.zos12.ral.ibm.com
Address:  9.12.6.67
```

Figure 10-80 nslookup non-interactive command (using the default version 4)

```
OCTAVIO @ SC63:/>nslookup -V=v9 host1
Running nslookup version 9
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-silŸent¨' option to prevent this message from appearing.
Allocated socket 5, type udp
Server:         9.12.6.68
Address:        9.12.6.68#53

Name:   host1.zos12.ral.ibm.com
Address: 9.12.6.67
```

Figure 10-81 nslookup non-interactive mode command (version 9)

```
OCTAVIO @ SC63:/>nslookup
Defaulting to nslookup version 4
Starting nslookup version 4
Default Server:  dns63.zos12.ral.ibm.com
Address:  9.12.6.68

> root
Default Server:  a.root-servers.net
Address:  198.41.0.4

>
```

Figure 10-82 nslookup interactive mode (version 4)

```
> exit
OCTAVIO @ SC63:/>nslookup -V=v9
Running nslookup version 9
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-silŸent¨' option to prevent this message from appearing.
Default server: 9.12.6.68
Address: 9.12.6.68#53
> lserver
Default server: 9.12.6.68
Address: 9.12.6.68#53
>
```

Figure 10-83 nslookup interactive mode (version 9)

10.7.2.4 Using the z/OS UNIX dig command

The domain information groper (dig) is a command line tool that can be used to gather information from the Domain Name System servers. The dig command has two modes: simple interactive mode for a single query, and batch mode, which executes one query for each in a list of several query lines. All query options are accessible from the command line. Dig is a program for querying Domain Name Servers, which enables you to:

- Exercise name servers
- Gather large volumes of domain name information
- Execute simple domain name queries
- Execute multiple lookups from the command line

You can use dig in several methods:

- Command line

  All options are specified on the invoking command line, as shown in Figure 10-84.

```
OCTAVIO @ SC63:/>dig host1.zos12.ral.ibm.com.
Allocated socket 5, type udp

; <<>> DiG 9.1.1 <<>> host1.zos12.ral.ibm.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49597
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;host1.zos12.ral.ibm.com.       IN      A

;; ANSWER SECTION:
host1.zos12.ral.ibm.com. 86400  IN      A       9.12.6.67

;; AUTHORITY SECTION:
zos12.ral.ibm.com.      86400   IN      NS      dns63.zos12.ral.ibm.com.
zos12.ral.ibm.com.      86400   IN      NS      dns64.zos12.ral.ibm.com.

;; ADDITIONAL SECTION:
```

Figure 10-84 Using dig command line option

- Batch mode

  A group of queries are placed in a file and executed by a single invocation of dig using the -f filename option. The filename contains complete queries, one per line. The keyword dig is not used within a batch file when specifying queries. Blank lines are ignored, and lines beginning with a # character or a semicolon (;) in the first column are comment lines. Figure 10-86 shows the result of a dig command executed using a file named queries.file as parameter. the contents of the queries.file is shown in Figure 10-85.

```
 EDIT       /etc/octavio/queries.file
 Command ===>
 ****** ******************************************
 000001 host1
 000002 dns63
 ****** ******************************************
```

Figure 10-85 Contents of the file pointed by the dig -f batch command

```
OCTAVIO @ SC63:/>dig -f /etc/octavio/queries.file
Allocated socket 6, type udp
; <<>> DiG 9.1.1 <<>> host1
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 49597
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;host1.                        IN      A
;; AUTHORITY SECTION:
.                       10369   IN      SOA     A.ROOT-SERVERS.NET.
NSTLD.VERISIGN-GRS.COM. 2002052500 1800 900 604800 86400
;; Query time: 7 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Sat May 25 15:09:56 2002
;; MSG SIZE  rcvd: 98

Allocated socket 7, type udp
; <<>> DiG 9.1.1 <<>> dns63
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 41218
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;dns63.                 IN    A
;; AUTHORITY SECTION:
.                       10800   IN      SOA     A.ROOT-SERVERS.NET.
NSTLD.VERISIGN-GRS.COM. 2002052500 1800 900 604800 86400
;; Query time: 239 msec
;; Query time: 239 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Sat May 25 15:09:57 2002
;; MSG SIZE  rcvd: 98
```

Figure 10-86 dig command in batch mode

- Multiple queries

  The BIND 9 implementation of dig supports specifying multiple queries on the command line (in addition to supporting the -f batch file option). Each of those queries can be supplied with its own set of flags, options and query options. In multiple queries, query1, query2, and so on represent an individual query in the command-line syntax. Each consists of any of the standard options and flags, the name to be looked up, an optional query type and class and any query options that should be applied to that query, as shown in Figure 10-87

Note: When entered on z/OS UNIX shell command line, long dig commands may be broken up in segments entered with a terminating backlashes (\) except for the last segment.

```
; <<>> DiG 9.1.1 <<>> host1.zos12.ral.ibm.com. host2.zos12.ral.ibm.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49597
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;host1.zos12.ral.ibm.com.       IN      A
;; ANSWER SECTION:
host1.zos12.ral.ibm.com. 86400  IN      A       9.12.6.67
;; Query time: 2 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Sat May 25 15:35:19 2002
;; MSG SIZE  rcvd: 129

Allocated socket 5, type udp
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41218
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; ANSWER SECTION:
host2.zos12.ral.ibm.com. 86400  IN      A       9.12.6.63
;; Query time: 1 msec
;; SERVER: 9.12.6.68#53(9.12.6.68)
;; WHEN: Sat May 25 15:35:19 2002
;; MSG SIZE  rcvd: 129

OCTAVIO @ SC63:/>
```

Figure 10-87 dig command with multiple queries (partial screen)

10.7.2.5 Using name server signals to Diagnose BIND 4.9.3-DNS

You can use name server signals to send messages to a BIND 4.9.3 or a BIND 9 DNS name server. Note that some of the signals may have different consequences if a signal is sent to a BIND 4.9.3 name server instead of a BIND 9 name server. These signals control various functions that can be used to diagnose problems. The signals valid for the Domain Name System server are SIGHUP, SIGINT, SIGABRT, SIGUSR1, SIGUSR2, and SIGWINCH. A sample job stream in SEZAINST named NSSIG allows you to issue signal s to the name server from the MVS operator console:

```
s nssig,sig=hup
```

The documentation for issuing signals with the name server is in *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 and in Figure 10-88 you see a copy of the NSSIG JCL.

```
//NSSIG PROC SIG=''
//NSSIG EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//             PARM='SH kill -s &SIG $(cat /etc/named.pid)'
//* STDIN and STDOUT are both defaulted to /dev/null
//STDERR      DD PATH='/etc/log',PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

Figure 10-88 NSSIG sample from SEZAINST

When you are using a BIND 4.9.3-based Nameserver you can use signals to execute a number of the following tasks.

Displaying the DNS active sockets

Once the DNS server process has started, you can display the active sockets with an **onetstat -a** display. Also, you can use the **-c** and the **-s** options of onetstat to display the active sockets. In Figure 10-89 is the result of an onetstat display. and you can observe that DNS has opened a connection with each interface 1 defined in the named.conf's *listen-on* option using the default port 53.

```
OCTAVIO @ SC63:/>onetstat -a
MVS TCP/IP onetstat CS V1R2      TCPIP Name: TCPIPB          13:54:21
User Id  Conn     Local Socket           Foreign Socket        State
-------  ----     ------------           --------------        -----
FTPDB1   00001F31 0.0.0.0..21            0.0.0.0..0            Listen
NAMED91  00002C7B 9.12.6.68..53          0.0.0.0..0            Listen
NAMED91  00002C79 9.12.6.67..53          0.0.0.0..0            Listen
TCPIPB   00000014 0.0.0.0..23            0.0.0.0..0            Listen
TCPIPB   00000013 127.0.0.1..1025        127.0.0.1..1026       Establsh
NAMED91  00002C78 9.12.6.67..53          *..*                  UDP    1
NAMED91  00002C7A 9.12.6.68..53          *..*                  UDP
OCTAVIO @ SC63:/>
```

Figure 10-89 Display of active sockets: onetstat -a

In Figure 10-90, you see the result of a different onetstat display.

```
OCTAVIO @ SC63:/>onetstat -s
MVS TCP/IP onetstat CS V1R2      TCPIP Name: TCPIPB          14:26:55
Sockets interface status:
Type   Bound to              Connected to          State    Conn
====   ========              ============          =====    ====
Name: FTPDB1    Subtask: 007F9030
Stream 0.0.0.0..21           0.0.0.0..0            Listen   00001F31
Name: NAMED91   Subtask: 007E2BC8
Dgram  9.12.6.68..53         *..*                  UDP      00002C7A
Dgram  9.12.6.67..53         *..*                  UDP      00002C78
Stream 9.12.6.67..53         0.0.0.0..0            Listen   00002C79
Stream 9.12.6.68..53         0.0.0.0..0            Listen   00002C7B
```

Figure 10-90 onetstat -s display after starting DNS

Dumping the DNS server cache

When you start the DNS server process it will read all the ZONE files and place the information in memory. This memory database will get updated with entries that it learns from other DNS servers because of the recursive searching that may go on between DNS servers.

When using BIND 4.9.3-based Nameserver, you have the ability to dump this memory table by sending a signal to the DNS server. The `SIGINT` signal dumps the name server memory database in the HFS file /tmp/named_dump.db. You can issue the signal through the ISPF ISHELL panel or you can issue the command in the UNIX System Services shell by entering:

```
kill -INT $(cat /etc/named.pid)
```

The process ID of the named daemon is stored in the /etc/named.pid file at the named startup. Alternatively you might enter the PID directly:

```
kill -INT 402653187
```

You can obtain the PID with an UNIX System Services shell command **`ps -e`** or with the **`D OMVS,A=ALL`** console command, as you see in Figure 10-91 **1**. Note the name of the executed program when using BPXBATCH **2**.

```
  D OMVS,A=ALL
  BPXO040I 10.52.17 DISPLAY OMVS 659
  OMVS     000F ACTIVE          OMVS=(ZF)
  USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
  OMVSKERN BPXOINIT 003C          1          0 MRI--- 08.27.21     19.73
    LATCHWAITPID=         0 CMD=BPXPINPR
    SERVER=Init Process                    AF=    0 MF=00000 TYPE=FILE
  STC      MVSNFSC5 002A   16842757          1 1A---- 08.27.42     .84
    LATCHWAITPID=         0 CMD=BPXVCMT
  STC      MVSNFSC5 002A   16842759          1 1R---- 08.27.40     .84
    LATCHWAITPID=         0 CMD=BPXVCLNY
  STC      MVSNFSC5 002A   67174409          1 1R---- 08.27.44     .84
    LATCHWAITPID=         0 CMD=GFSCMAIN
  TCPIPMVS FTPDC1   004C   33619978          1 1FI--- 17.39.21     .09
    LATCHWAITPID=         0 CMD=FTPD
  TCPIPMVS TCPIP MVS 0020   16842763          1 MR---B 08.29.13    594.88
    LATCHWAITPID=         0 CMD=EZBTCPIP
  STC      NAMED43  0051      65645  1       1 1F---- 10.51.41     .02
    LATCHWAITPID=         0 CMD=/usr/lpp/tcpip/sbin/named4 -b /etc/octav  2
```

Figure 10-91 D OMVS,A=ALL: process ID

See Figure 10-92 for a partial copy of the /tmp/named_dump.db file that was created when you issued command *kill -INT $(cat /etc/named.pid)*.

```
; Dumped at Sat May 25 15:13:12 2002
;; ++zone table++
; zos12.ral.ibm.com (type 1, class 1, source zos12.for.v4)
; time=1022346871, lastupdate=1022091095, serial=1,
; refresh=10800, retry=3600, expire=604800, minimum=86400
; ftime=1022091095, xaddr=Ý0.0.0.0¨, state=0041, pid=0
; Note: NT=milliseconds for any A RR which we've used as a name
; --- Cache & Data ---
$ORIGIN .
 518359 IN NS A.ROOT-SERVERS.NET. ;Cr=auth Ý192.33.4.12¨
 518359 IN NS B.ROOT-SERVERS.NET. ;Cr=auth Ý192.33.4.12¨
$ORIGIN ral.ibm.com.
zos12  IN SOA dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
 1 10800 3600 604800 86400 ) ;Cl=4
 IN NS dns63.zos12.ral.ibm.com. ;Cl=4
 IN NS dns65.zos12.ral.ibm.com. ;Cl=4
$ORIGIN zos12.ral.ibm.com.
mail   IN CNAME host1.zos12.ral.ibm.com. ;Cl=4
ftp    IN CNAME host2.zos12.ral.ibm.com. ;Cl=4
sc63   IN A 9.12.6.67 ;Cl=4
```

Figure 10-92 Partial contents of tmp/named_dump.db (from a SIGINT to DNS process)

DNS statistics

When you are using BIND 4.9.3, you can obtain DNS statistics by using the signal #3 (ABRT), available either from the ISHELL selection menus or by issuing a **`kill -3 $(cat /etc/named.pid)`** from the shell. The data is stored in /tmp/named.stats as shown in Figure 10-93.

```
+++ Statistics Dump +++ (1022340539) Sat May 25 15:28:59 2002
992 time since boot (secs)
992 time since reset (secs)
0 Unknown query types
4 A queries
4 SOA queries
++ Name Server Statistics ++
(Legend)
 RQ RR RIQ RNXD RFwdQ
 RFwdR RDupQ RDupR RFail RFErr
 RErr RTCP RAXFR RLame ROpts
 SSysQ SAns SFwdQ SFwdR SDupQ
 SFail SFErr SErr RNotNsQ SNaAns
 SNXD
(Global)
 8 11 0 4 4  10 0 0 0 0  0 1 0 0 0  1 3 4 10 11  0 0 0 5 0  0
Ÿ9.12.6.68¨
 4 0 0 0 4  6 0 0 0 0  0 0 0 0 0  0 0 0 4 0  0 0 0 4 0  0
```

Figure 10-93 Name server statistics (partial contents)

Tracing the name server

With BIND 4.9.3-based name server you can also use a signal to start, increase the debug level and stop a trace. As shown in Figure 10-94, the Signal #16 (USR1) **1** begins a trace of the DNS server processes. Signal #17 (USR2) **2** terminates the trace. The data is placed in /tmp/named.run as seen in Figure 10-95.

```
                           Work with Processes
 +______________________________________________________________________
 |                           Enter Signal Number
 |
 | Process ID.....: 16777232
 | Command......: /usr/lpp/tcpip/sbin/named
 | Signal number.... 16 1
 |
 | Some of the common signal numbers are:
 |    1 SIGHUP  hang-up                  2  17 SIGUSR2 application defined
 |    3 SIGABRT abnormal termination       19 SIGCONT continue
 |    7 SIGSTOP stop                       20 SIGCHLD child
 |    9 SIGKILL kill                       21 SIGTTIN ctty background rea
 |   13 SIGPIPE write with no readers      22 SIGTTOU ctty background wri
 |   14 SIGALRM alarm                      23 SIGIO   I/O completion
 |   15 SIGTERM termination                24 SIGQUIT quit
 |1  16 SIGUSR1 application defined        25 SIGTSTP interactive stop
 |
 |
 |  F1=Help       F3=Exit       F6=Keyshelp   F12=Cancel
 +______________________________________________________________________
```

Figure 10-94   Using ISHELL to send a signal to start and stop a trace process.

```
Debug turned ON, Level 1
Debug turned ON, Level 2
Debug turned ON, Level 3
Debug turned ON, Level 4
Debug turned ON, Level 5
Debug turned ON, Level 6
Debug turned ON, Level 7
Debug turned ON, Level 8
Debug turned ON, Level 9
datagram from [127.0.0.1].1135, fd 11, len 39; now Thu Feb  5 16:17:01 19
ns_req(from=[127.0.0.1].1135)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1
;; flags: rd; Ques: 1, Ans: 0, Auth: 0, Addit: 0
;; QUESTIONS:
;;.fred.itso.ral.ibm.com, type = A, class = IN
;; ...truncated
req: nlookup(fred.itso.ral.ibm.com) id 1 type=1 class=1
req: found 'fred.itso.ral.ibm.com' as 'fred.itso.ral.ibm.com' (cname=0)
```

Figure 10-95   /tmp/named.run output

The debug level was raised to 9 in the trace because we issued the USR1 signal (#16) multiple times. We really wanted a debug level of 11, the recommended (and highest allowable) setting. Attempting to reach debug level 11 via the ISHELL TOOLS menu can be tedious, as you see. We recommend instead short cutting the ISHELL process and setting the debug trace level at 11 by one of two methods:

1. Use `-d 11` on the named start command or in your JCL.
2. Issue the `kill -USR1 ($cat /etc/named.pid)` multiple times from the shell with the help of the retrieve key.

Resolver tracing is available with the name server and is enabled as usual in the TCPDATA file. The output goes to the MVS console or to the syslogd.log if you have started the syslog daemon.

Note: The signals showed doesn't provide the results if used with BIND V9-based Name Server. with BIND 9 you must use `rncd` command to execute the same process.

10.7.2.6 Using rndc to diagnose BIND 9 problems

`rndc` may be used to dump the name servers cache to a file with the *dumpdb* parameter. Also, the *reload* parameter may be used similar to the SIGHUP signal for increasing the amount of information written to the logging files. The *querylog* and *stats* options may also be helpful.

Remote Name Daemon Control (`rndc`) command allows the system administrator to control the operation of a name server. If rndc is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments. rndc may only be used with the v9 (BIND9) name server and will not function with the v4 (BIND.4.9.3) name server

The function of rndc may be used as a secure remote client to control the name server. Some local UNIX signal functions for the name server can be replaced by equivalent rndc functions. The name server and rndc communicates over a TCP connection, sending commands authenticated with digital transaction signatures (TSIG). This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server. Therefore, rndc and the name server must be configured with a 'shared-secret'. This shared secret is a TSIG key, which may be generated with the dnssec-keygen utility. The only supported encryption algorithm for rndc is HMAC-MD5, which uses a shared secret on each end of the connection.

To allow your host to execute the rndc command, follow the steps defined in section "Using the z/OS UNIX rndc command" on page 400.

Part 5

Utility applications

In this part, we introduce some of the utility applications shipped with Communications Server for z/OS IP. These include applications suchs as inetd, netstat, and syslogd that make it easier to operate in the TCP/IP environment.

11

InetD

InetD is a generic listener. It can be used by any server that does not have its own listener. InetD supports various servers, such as the z/OS UNIX telnet server, the z/OS UNIX REXECD server, the z/OS UNIX RSH server, and the rlogin server.

This chapter contains the following sections:

11.1 InetD configuration

The InetD configuration information is by default placed in /etc/inetd.conf. If you need to specify server run-time options for the servers that are started via InetD, you have to do so in the /etc/inetd.conf file.

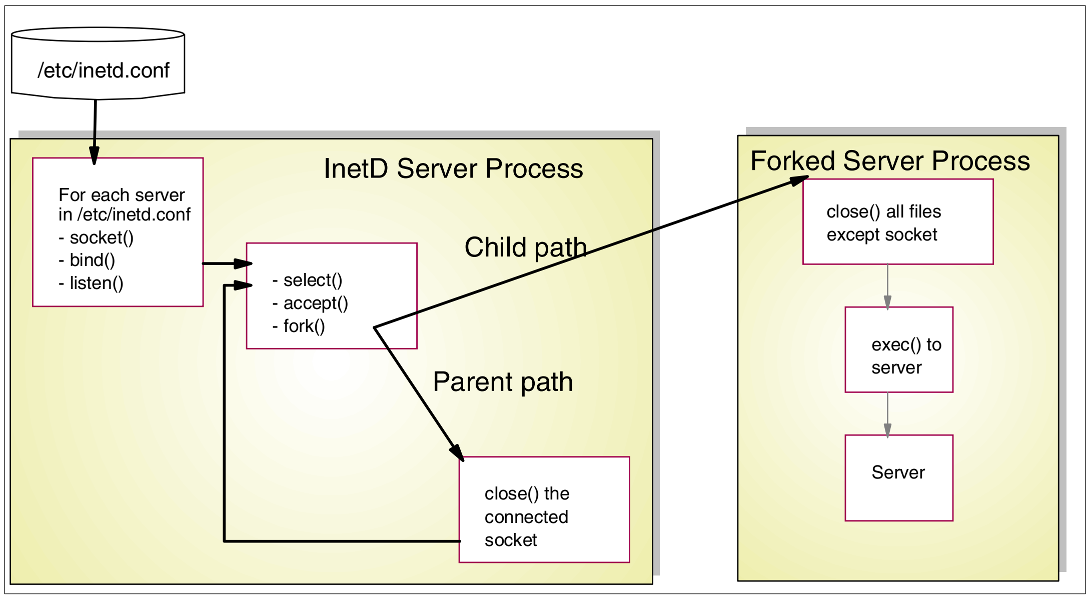


Figure 11-1 InetD generic listener

The /etc/inetd.conf file describes to the InetD daemon what servers to manage. InetD looks up the ETC.SERVICES file to find the port numbers for its defined servers and listens on these ports. If the InetD daemon receives a request on one of these sockets, it determines which service corresponds to that socket and then invokes the appropriate server.

The /etc/inetd.conf configuration file that was used in our sample setup looks like the following:

```
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
#
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -l
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV -d -r
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
uucp     stream tcp nowait OMVSKERN /usr/sbin/uucpd uucpd -10
```

As InetD opens sockets and issues listen() calls on these sockets, you need to pay attention to when you start InetD and when you need to restart InetD.

You have three different choices to start InetD:

- Using a small shell script like the following:

```
#
# Start the INETD daemon
#

export _BPX_JOBNAME='INETD'
export _CEE_RUNOPTS='ALL31(ON)'
/usr/sbin/inetd /etc/inetd.conf &

echo -- /u/silviar/restart_inetd.sh script executed, `date`
```

Figure 11-2 Sample shell script to start the InetD server

- Including the command to start the InetD in /etc/rc file, thus starting it automatically.
- Using a started procedure. In our test environment, this method was chosen.

  On OS/390 V2R7 the InetD module has been moved to the HFS. InetD has to be started with BPXBATCH.

```
//INETD    EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//         PARM='PGM /usr/sbin/inetd /etc/inetd.conf'
//STDOUT   DD PATH='/tmp/inetd.stdout',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR   DD PATH='/tmp/inetd.stderr',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDENV   DD DSN=TCPIP.TCPPARMS(INETDENV),DISP=SHR    1
```

Figure 11-3 Sample InetD started procedure

1 The STDENV DD statement can be used to define the environment variables for the shell in which the InetD runs. Figure 11-4 shows the variable settings we used.

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA      2
TZ=EST5EDT
```

Figure 11-4 The environment variable setting fot InetD

2 If you want InetD to bind only one TCP/IP stack, the environment variable `_BPXK_SETIBMOPT_TRANSPORT` has to be configured to direct InetD to this particular TCP/IP stack.

Note: If there is no TCP/IP active when InetD is started, it will try to connect to TCP/IP every three minutes. If the stack goes down, InetD will also try to re-establish the connection. This behavior is different from other servers.

11.2 Internet services supported internally by InetD

The InetD daemon supports various server functions internally. The services supported internally by the inetd daemon are generally used for debugging. They include the following internal services:

- `echo`, which returns data packets to a client host.
- `discard`, which discards received data packets.
- `chargen`, which discards received data packets and sends predefined or random data.
- `daytime`, which sends the current date and time in user-readable form.
- `time`, which sends the current date and time in machine-readable form.

Since all internal servers support both TCP and UDP protocols, they can cooperate with the clients that support only one of two transport protocols. For example, the time client provided in AIX, the **`setclock`** command, supports TCP protocol only, while the existing TIMED server shipped with CS for z/OS IP runs on the UDP transport only, so they will never be able to communicate with each other. By using the time service implemented inside InetD, the **`setclock`** command can retrieve the current time from a z/OS system and set it for the local system.

To activate the internal services you will have to update both /etc/inetd.conf and /etc/services as shown in Figure 11-5 and Figure 11-6 on page 421.

```
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
otelnet  stream tcp nowait OMVSKERN /usr/sbin/otelnetd otelnetd -m
shell    stream tcp nowait OMVSKERN /usr/sbin/orshd orshd -LV -d -r
login    stream tcp nowait OMVSKERN /usr/sbin/rlogind rlogind -m
exec     stream tcp nowait OMVSKERN /usr/sbin/orexecd orexecd -LV
#
echo      stream tcp nowait OMVSKERN internal
echo      dgram  udp wait   OMVSKERN internal
discard   stream tcp nowait OMVSKERN internal
discard   dgram  udp wait   OMVSKERN internal
chargen   stream tcp nowait OMVSKERN internal
chargen   dgram  udp wait   OMVSKERN internal
daytime   stream tcp nowait OMVSKERN internal
daytime   dgram  udp wait   OMVSKERN internal
time      stream tcp nowait OMVSKERN internal
time      dgram  udp wait   OMVSKERN internal
```

Figure 11-5 InetD configuration for the internal services

All internal services can run over both TCP and UDP protocols, so you should configure two entries for each service. The server program field should be internal for these services.

```
 1            2   3
echo          7/tcp
echo          7/udp
discard       9/tcp           sink null
discard       9/udp           sink null
daytime       13/tcp
daytime       13/udp
chargen       19/tcp          ttytst source
chargen       19/udp          ttytst source
time          37/tcp          timserver
time          37/udp          timserver
```

Figure 11-6 /etc/services configuration for the internal services

The socket and protocol have to be defined in the service definition files (/etc/services) for each service. The configured values will be an official Internet service name 1, the socket port number 2, and the transport protocol 3 used for the service, TCP or UDP.

If these ports have been reserved in TCPIP.PROFILE, you will have to update the definitions to allow InetD to bind to them. Reserving with OMVS is recommended, because if InetD starts with the `-d` option (debug option), it will not fork a child process, then those ports would need to be reserved for InetD. Whereas, without this option, InetD will create a child process with the name of INETD1 that will bind to the specified ports. In this scenario, you have to configure the PORT statements with the procedure name INETD1.

If you have the PORT statements configured with the user name of OMVS, InetD will work correctly regardless of the `-d` configuration.

Figure 11-7 is the report from the **`D TCPIP,,N,CONN`** command. You will see that InetD has bound to the ports for the internal services.

```
D TCPIP,TCPIPA,N,CONN
EZZ2500I NETSTAT CS V2R10 TCPIPA 524
USER ID  CONN     LOCAL SOCKET           FOREIGN SOCKET         STATE
                :
INETD1   000106A7 0.0.0.0..7             0.0.0.0..0             LISTEN
INETD1   000106A8 0.0.0.0..9             0.0.0.0..0             LISTEN
INETD1   000106AB 0.0.0.0..37            0.0.0.0..0             LISTEN
INETD1   000106AA 0.0.0.0..13            0.0.0.0..0             LISTEN
INETD1   000106A9 0.0.0.0..19            0.0.0.0..0             LISTEN
                :
INETD1   000106AC 0.0.0.0..7             *..*                   UDP
INETD1   000106AD 0.0.0.0..9             *..*                   UDP
INETD1   000106AF 0.0.0.0..13            *..*                   UDP
INETD1   000106AE 0.0.0.0..19            *..*                   UDP
INETD1   000106B0 0.0.0.0..37            *..*                   UDP
46 OF 46 RECORDS DISPLAYED
```

Figure 11-7 Report from NETSTAT CONN

12

Netstat

The TSO **NETSTAT**, z/OS UNIX **onetstat** (USS **onetstat**), and MVS **Netstat** console commands provide information about the status of the local host, including information about TCP/IP connections, network clients, gateways, and devices. The enhancements to Netstat in CS z/OS V1R2.0 include filtering, performance counters, and restricted access to netstat commands. It is important to remember that most, although not *all* of the commands are universal between the three environments. For detailed information on the syntax, options, and explanation of the command results, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781.

This chapter contains the following sections:

12.1 TSO Netstat

The TSO **Netstat** is issued as a TSO command. The TSO **Netstat** command is used to display the network status on the local host. Figure 12-2 produces the results from the following command:

```
onetstat -c
```

Note: **netstat** is a synonym for the **onetstat** command in the z/OS UNIX shell. The **netstat** command syntax is the same as that for the **onetstat** command.

```
MVS TCP/IP NETSTAT CS V1R2       TCPIP NAME: TCPIPMVS         20:08:46
Home address list:
Address          Link             Flg
-------          ----             ---
 9.12.6.9         OSA22E0LNK       P
 172.16.233.209   EZASAMEMVS
 172.16.233.209   IQDIOLNKAC10E9D1
 172.16.233.209   EZAXCF63
 127.0.0.1        LOOPBACK
```

Figure 12-1 TSO netstat home

12.2 USS onetstat

The z/OS UNIX **onetstat** command is used to display the network status of the local host.

```
MVS TCP/IP onetstat CS V1R2       TCPIP Name: TCPIPB          15:43:33
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
DB7KDIST 0000001F 0.0.0.0..33741         0.0.0.0..0             Listen
DB7KDIST 0000001B 0.0.0.0..33740         0.0.0.0..0             Listen
PAGENT   0000073C 10.1.1.5..1701         10.1.1.6..1700         Establsh
PAGENT   0000073A 10.1.1.5..1701         10.1.1.5..1700         Establsh
PAGENT   00000734 0.0.0.0..1700          0.0.0.0..0             Listen
PAGENT   00000743 10.1.1.5..1701         10.1.1.4..1700         Establsh
PAGENT   0000073B 10.1.1.5..1700         10.1.1.5..1701         Establsh
TCPIPB   0000000B 127.0.0.1..1025        0.0.0.0..0             Listen
TCPIPB   00000018 0.0.0.0..2364          0.0.0.0..0             Listen
TCPIPB   00000017 0.0.0.0..23            0.0.0.0..0             Listen
TCPIPB   00000015 127.0.0.1..1026        127.0.0.1..1025        Establsh
TCPIPB   00000016 127.0.0.1..1025        127.0.0.1..1026        Establsh
NAMED91  0000084D 0.0.0.0..10009         *..*                   UDP
NAMED91  0000084B 9.12.6.63..53          *..*                   UDP
```

Figure 12-2 onetstat -c

12.3 MVS netstat console command

The Display TCPIP,,NETSTAT command is issued from an operator's console to request NETSTAT information. Figure 12-3 displays the output from the following console command:

```
D TCPIP,TCPIPB,N,AR
```

```
000290  D TCPIP,TCPIPB,N,AR
000090  EZZ2500I NETSTAT CS V1R2 TCPIPB 680
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.63
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.68
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.67
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.62
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.61
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.3
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.6
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.122
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.121
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.11
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.10
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.5
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.4
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.9
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.8
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.1
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.2
000090  LINK: OSA22E0           ETHERNET: 0006296C3650
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.75
000090  LINK: OSA22E0           ETHERNET: 10005A995C46
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.78
000090  LINK: OSA22E0           ETHERNET: 0004AC57456D
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.7
000090  LINK: OSA22E0           ETHERNET: 0004ACEE2208
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.155
000090  LINK: OSA22E0           ETHERNET: 000255E45AD7
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.30
000090  LINK: OSA22E0           ETHERNET: 0006296CA584
000090  QUERYING ARP CACHE FOR ADDRESS 9.12.6.92
000090  LINK: OSA22E0           ETHERNET: 0007858565C2
000090  23 OF 23 RECORDS DISPLAYED
```

Figure 12-3   MVS netstat console command for ARP cache information

12.4 Netstat enhancements in z/OS V1R2.0

The following enhancements were added to the `netstat` command in z/OS V1R2.0:

- Filter enhancements
- Performance counters
- Restricted access to netstat commands

For more information on the netstat enhancements, you can refer to *z/OS V1R2.0 CS: IP Migration,* GC31-8773.

12.4.1 Filter enhancements

CS for z/OS V1R2 IP includes enhancements to the netstat commands to allow a choice to include or exclude the TN3270 server connections from the netstat ALL/-A, ALLCONN/-a, CONN/-c, BYTEINFO/-b, CLIENTS/-e, and SOCKETS/-s reports. The following netstat entries include new and changed filter options:

- **TSO NETSTAT command:** ALL, ALLCONN, BYTEINFO, CLIENTS, CONN, and SOCKETS options
- **UNIX onetstat/netstat command:** -A, -a, -b, -c, e, and -s options
- **MVS D TCPIP,,NETSTAT command:** ALLCONN, BYTEINFO, CONN, and SOCKETS options

In addition, the existing filter support for CLIENT/-E, IPADDR/-I, and PORT/-P is enhanced to work for netstat SOCKETS/-s so that the netstat SOCKETS/-s report can provide the response on the specified client name, IP address, or port number. The existing filter support for IPADDR/-I, and PORT/-P is enhanced to work for netstat ALL/-A so that the netstat ALL/-A report can provide the response on the specified IP address, or port number.

12.4.2 Performance counters

z/OS V1R2 Communications Server enhances the netstat commands to show performance characteristics and identify performance problems.The following netstat entries include new or changed performance options:

- **TSO `NETSTAT` command:** ALL, DEVLINKS, HELP, and STATS options
- **UNIX `onetstat/netstat` command:** -A, -d, -S, and -? options
- **UNIX `onetstat/netstat` command:** -A, -d, -S, and -? options

12.4.3 Restricting access to Netstat commands

z/OS V1R2 Communications Server provides a new way to control access to the `netstat` command at both the overall command level and command option level. You can permit or disallow user access to specific netstat options or resources. This function only applies to TSO and UNIX shell `netstat` command users.

Implementing netstat access control

The procedure for implementing netstat access control is to define the security product resource name, EZB.NETSTAT.mvsname.tcpprocname.option, in the SERVAUTH class. Ensure that the SERVAUTH class is active and RACLISTed. Permit users READ access to the resource name. For more information on the implementation, please refer to *zz/OS V1R2.0 CS: IP System Administrator's Commands,* SC31-8781 and *z/OS V1R2.0 CS: IP Configuration Guide,* SC31-8775.

13

ONC/RPC port mapper

Port mapper converts RPC program numbers into Internet port numbers.

It comes in two flavors:

- z/OS UNIX port mapper
- z/OS port mapper

The ONC/RPC port mapper was supplied with the OS/390 TCP/IP OpenEdition as a UNIX System Services application program.

To support ONC/RPC application programs that execute in the OpenEdition environment, you may use either the z/OS UNIX version of the port mapper or the non-z/OS UNIX port mapper (PORTMAP). In this chapter, we will address the z/OS UNIX port mapper only. For more information about the ONC/RPC port mapper, please refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

This chapter contains the following sections:

- 13.1, “The z/OS port mapper” on page 430
- 13.2, “The non-z/OS UNOIX port mapper” on page 433

13.1 The z/OS port mapper

Following are examples of how to operate the z/OS UNIX port mapper.

The port reservations in the TCP/IP for MVS PROFILE data set must tell OMVS to start the UNIX System Services version of the port mapper:

```
PORT
   111 UDP OMVS                 ; OE Portmapper Server
   111 TCP OMVS                 ; OE Portmapper Server
```

To start the port mapper as a UNIX System Services socket application, you may either start it from the shell or as a started task in MVS.

13.1.1 Starting the port mapper from the z/OS UNIX shell

This is how to start the port mapper from OMVS with displays of the result:

```
OEUSER:/u/oeuser: >cd /bin
OEUSER:/bin: >su            1
OEUSER:/bin: >oportmap &    2
[1]     67108889
OEUSER:/bin: >
[1] + Done oportmap &       3
OEUSER:/bin: >exit          4
OEUSER:/bin: >


 D TCPIP,T03ATCP,N,SOCK

 EZZ2500I NETSTAT CS V2R5 T03ATCP 741
 SOCKETS INTERFACE STATUS:
 TYPE   BOUND TO               CONNECTED TO          STATE    CONN
 NAME: OEUSER5   SUBTASK: 007DB3E0          5
 DGRAM  0.0.0.0..111           *..*                   UDP      00191
 STREAM 0.0.0.0..111           0.0.0.0..0             LISTEN   00192
        ..
17 OF 17 RECORDS DISPLAYED


 D OMVS,A=ALL

 BPXO040I 19.17.37 DISPLAY OMVS 743
 OMVS     000E ACTIVE         OMVS=(03)
 USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
 OMVSKERN BPXOINIT 0025          1         0 MKI   10.34.39       .346
   LATCHWAITPID=        0 CMD=BPXPINPR
   SERVER=Init Process                   AF=    0 MF=65535 TYPE=FILE
        ..
OEUSER   OEUSER5  003B  134217752       6  1 1FI   19.14.32       .126
   LATCHWAITPID=        0 CMD=oportmap
    ..
```

Figure 13-1 The port mapper started via OMVS

The following apply to Figure 13-1:

1 Switch to a superuser.

2 Start oportmap as a background process.

3 Oportmap has forked a child process.

4 Exit from the superuser mode. The user name here is OEUSER5.

5, 6 The forked process.

13.1.2 Starting the port mapper from a started task

This section illustrates how to start the port mapper from a started task.

The sample started procedure can be found in TCPIP.SEZAINST(OPORTPRC). We renamed it to PORTMAP as follows:

```
//PORTMAP  PROC
//*
//PORTMAP  EXEC PGM=OPORTMAP,REGION=4096K,TIME=1440,
//           PARM=('POSIX(ON),ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPA")',
//           '/')
//*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*        PEND
```

Following is the STC **START** command and its results:

```
S PORTMAP

$HASP100 PORTMAP  ON STCINRDR
IEF695I START PORTMAP  WITH JOBNAME PORTMAP  IS ASSIGNED TO USER
OMVSKERN, GROUP OMVSGRP
$HASP373 PORTMAP  STARTED
-                                                --TIMINGS (MINS.)--
        ----PAGING COUNTS---
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB  CLOCK
SERV  PG  PAGE  SWAP   VIO SWAPS
-PORTMAP           PORTMAP     00     12     28    .00    .00     .3
5052   0     0     0     0     0
-PORTMAP  ENDED.  NAME-                    TOTAL TCB CPU TIME=   .00
TOTAL ELAPSED TIME=    .3
$HASP395 PORTMAP  ENDED


D TCPIP,TO3ATCP,N,SOCK

EZZ2500I NETSTAT CS V2R5 TO3ATCP 436
SOCKETS INTERFACE STATUS:
TYPE   BOUND TO               CONNECTED TO           STATE    CONN
NAME: PORTMAP1  SUBTASK: 007DB438
DGRAM  0.0.0.0..111           *..*             1      UDP      00099
STREAM 0.0.0.0..111           0.0.0.0..0       1      LISTEN   0009A
 ..
11 OF 11 RECORDS DISPLAYED


D OMVS,A=ALL

BPXO040I 16.14.01 DISPLAY OMVS 434
OMVS     000E ACTIVE         OMVS=(03)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
OMVSKERN BPXOINIT 0025          1          0 MKI   10.34.39       .282
  LATCHWAITPID=        0 CMD=BPXPINPR
  SERVER=Init Process                  AF=    0 MF=65535 TYPE=FILE
   ..
OMVSKERN PORTMAP1 0038  771751955      2  1 1FI   16.12.52       .122
  LATCHWAITPID=        0 CMD=OPORTMAP
```

Figure 13-2 The z/OS UNIX port mapper started via started task

The following apply to Figure 13-2:

1 The forked process PORTMAP1 is now listening on port 111.

2 The new address space.

13.2 The non-z/OS UNOIX port mapper

The non-z/OS UNIX port mapper may also be used. We found the instructions in *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775, to be adequate for getting the non-z/OS UNIX port mapper up and running. We will not repeat that chapter. Just make sure the PORT statement's *user* and the started task member name are the same. In our tests, we used the name PORTMAP.

In the SEZAINST data set, you will find the catalogued procedure to run the non-z/OS UNIX port mapper. It is named PORTPROC and only needs the STEPLIB and SYSTCPD DD statements to be adapted to your environment.

14

syslogd

syslogd is a server process in UNIX System Services that logs application messages, user messages, authorization violation messages, and trace data. Prior to IBM Communications Server for OS/390 V2R10 IP, there were two versions of syslogd. One version shipped with CS for OS/390 IP and the other shipped with OS/390 Security Server. Currently there is one syslogd for z/OS and it ships with CS for z/OS IP. If you are using the version of syslogd that shipped with Security Server, you must migrate to Communications Server for z/OS IP version of syslogd. Refer to *z/OS V1R2.0 CS: IP Migration*, GC31-8773 guide for assistance.

This chapter contains the following sections:

14.1 z/OS UNIX syslogd overview

The processing of syslogd is controlled by a configuration file (etc/syslogd.conf). The configuration file contains entries for different logging facility names, priority codes, log file names or destinations. For example, an entry indicating the facility name of `mail`, a priority code of `err` and a destination file path and name of /tmp/mail.log, will record mail server messages in that file. The configuration file can be setup to direct all other messages to the log file for entry *.* , which is generally the default file /tmp/syslogd.log. Refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for configuration details.

Servers on the local system use AF_UNIX sockets for communication with syslogd. Remote servers on the network use AF_INET sockets for communication with syslogd. Refer to Figure 14-1 on page 436 for an overview of how syslogd operates in UNIX System Services.

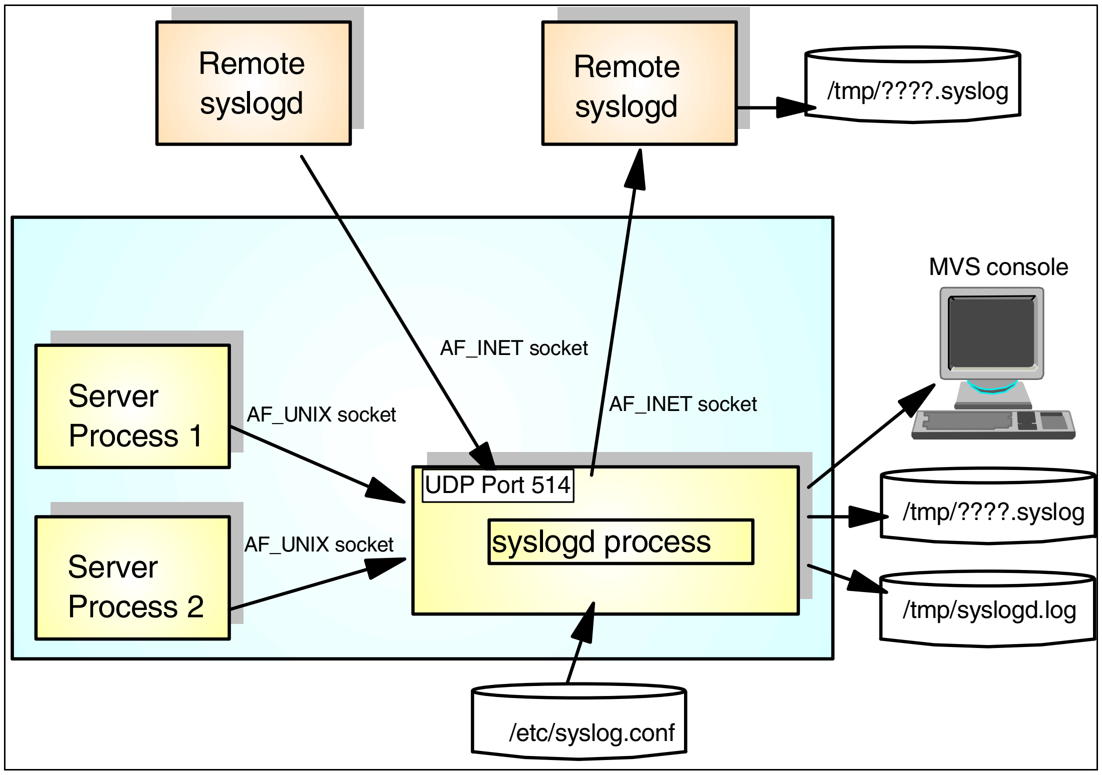


Figure 14-1 syslogd overview

14.2 syslogd features

IBM Communications Server for OS/390 V2R10 IP added additional management capabilities and usability improvements to syslogd. Managing networks and monitoring applications are key to availability requirements of systems in a distributed processing environment. CS for OS/390 V2R10 IP with its enhancements to the AF_UNIX sockets API and syslogd addressed the management and usability inadequacies of the prior versions.

In CS for OS/390 V2R10 IP, four startup options were added. Table 14-1 summarizes the start options that syslogd recognizes.

Table 14-1 syslogd startup options

Startup option	Description
-c	syslogd creates log files and directories automatically.
-d	Enable the debugging mode, in which syslogd runs in the foreground and writes a large number of trace messages to STDOUT.
-f	Specifies configuration file name.
-i	syslogd does not receive messages from the IP network.
-m	Defines number of minutes between mark messages. The default value is 20 minutes.
-p	Indicates path name of z/OS UNIX character device for the datagram socket. The default value is /dev/log.
-u	syslogd adds the user ID and job name in the record received over the AF_UNIX socket (most messages generated on the local system).

14.2.1 Management: syslogd isolation

AF_UNIX sockets provide a mechanism for determining certain information about the sender of a message. There are certain steps you must follow to enable isolation. Refer to *z/OS V1R2.0 CS: IP Migration*, GC31-8773 for further details on how to enable the isolation enhancements. The traditional record selection mechanism only supported facility name and priority parameters in the /etc/syslog.conf file.

There are two new startup parameters, -u and -i, that can be used on the command line or in a script with syslogd to enable the new parameters in /etc/syslog.conf that support isolation.

The -u parameter allows user ID and job name selection criteria. The new record selection format `{ [userid.jobname.] facility.priority }` when processed by syslogd, writes the user ID and job name with the message in the destination file specified in /etc/syslog.conf. The configuration file now allows for an easier way to manage logs because you are able to separate application program logs and user logs that use the same facility name. This is helpful when performing problem determination. The following example specifies `*.*.*.*` in the configuration file and directs all messages to be logged in a file named /etc/syslogd.logs. Figure 14-2 shows sample syslogd records when it has been started with -u option.

```
Date: Time    Host /Userid Jobname Recordid[pid]  Message
Jun 19 15:14:48 mvs03a/BTHOMPS  BTHOMPS6 FSUM1220 syslogd: restart
Jun 19 15:33:52 mvs03a/OMVSKERN INETD2   telnetd[137]: EZYTE52E
 Couldn't resolve your address into a host name.
Jun 19 15:33:52 mvs03a/OMVSKERN INETD2   telnetd[137]: IP address is
 9.24.106.127
Jun 19 19:57:57 mvs03a/TCPIP3   OMPROUTA omproute[16777227]: EZZ7827I
 Adding stack route to 172.16.233.0, mask 255.255.255.0 via 0.0.0.0,
link EZAXCF28, metric 1, type 1
Jun 19 19:57:57 mvs03a/TCPIP3   OMPROUTA omproute[16777227]: EZZ7885I
 Route not added to stack routing table - static route exists
Jun 19 19:57:57 mvs03a/TCPIP3   OMPROUTA omproute[16777227]: EZZ7882I
 Processing static route from stack, destination 172.16.233.28,
 mask 255.255.255.255, gateway 0.0.0.0
```

Figure 14-2 syslogd records with the -u option

You can tell whether or not the user ID and job name is present in the log by looking at the character after the host name. If it is a '/', the user ID and job name are present. If it is blank then the user ID and job name are not present. The absence of a "/" also gives you the indication that syslogd is running without the -u option.

If a host sends a message via UDP to the local host's syslogd server, the user ID and job name within the UDP packet is treated as user data and is not recognized as user ID and job name. So, messages received via UDP have `N/A` in the user ID and job name fields. The actual user ID and job name is part of the message, as shown below and is shifted to the right and recorded in the log file message field.

```
Date: Time    Host /Userid Jobname Recordid[pid]  Message
Jun 19 15:14:48 mvs03a/BTHOMPS  BTHOMPS6 FSUM1220 syslogd: restart
Jun 19 15:14:52 mvs28.itso.ral.ibm.com/N/A N/A BTHOMPS BTHOMPS8 FSUM1
220 syslog: restart
```

Figure 14-3 syslogd records from remote hosts

Note: The remote syslogd must be started with the -u parameter to include user ID and job name. Not all hosts support -u, but CS for z/OS IP does.

In this scenario, mvs03a.itso.ral.ibm.com running CS for OS/390 V2R10 IP with syslogd -u received a UDP packet from mvs28.itso.ral.ibm.com also running CS for OS/390 V2R10 IP with syslogd -u (`FSUM1220 syslog: restart`, message is recorded in the log when syslogd is started). Now that the user ID and job name is included with the syslogd message, verification of client credentials can be ensured before access to a log is allowed. Previous releases of CS for OS/390 IP syslogd trusted all applications.

The second syslogd startup parameter in support of isolation is the -i parameter. This parameter tells syslogd not to receive messages via UDP, which disables remote syslogd servers from logging messages on the local syslogd server. The -i parameter enforces UDP reception shut-off. It does not disable syslogd from sending its messages to remote syslogd servers; it only applies to suppressing inbound UDP packets from other syslogd daemons on the network.

Starting syslogd with the -i option and /etc/syslog.conf not containing any remote destinations to send messages prevents syslogd from getting an AF_INET socket. In order to determine the standard host name of the local host's syslogd calls, use `gethostname(char *name, size_t namelen)`. Because gethostname requires an INET socket, it will fail. Normally the value returned by the gethostname function call is the value of the HOSTNAME statement in TCPIP.DATA. If the HOSTNAME is not specified in TCPIP.DATA, the VMCF node name is returned. If all gethostname() attempts fail then "localhost" is used. Previous releases printed nothing (for a host name) in the log file. This activity does not affect normal processing of trace messages. Only AF_UNIX sockets messages will be logged since an AF_INET socket for syslogd is not available. And that's the whole point, to disable the syslogd messages from remote syslogd servers via UDP.

14.3 syslogd configuration

The processing of syslogd is controlled via a configuration file called /etc/syslog.conf by default. Another name may be chosen for the configuration file. This file can be used to define specific logging conditions and output destinations for application messages, user messages, authorization violation messages, and trace data for systemwide use.

All log files and directories without date/time format strings used by syslogd defined in /etc/syslog.conf must be created in the hierarchical file system before syslogd is started. Each statement of the configuration file has the following syntax:

```
{  [userid.jobname.]facility name.prioritycode      destination   }
```

The syslogd configuration file allows you to set up logging rules depending on:

- A logging *user ID*
- A logging *job name*
- A logging *facility name*
- A logging *priority code*

The user ID recorded in the log file is typically the same as the user ID of the logged-in user that started the application or process. It can also be the name of the parent process, such as OMVSKERN. By default, fork and spawn set the job name value to the user ID with a number (1-9) appended. However, daemons or users with the appropriate privileges can set the _BPX_JOBNAME environment variable to change the job name for forked or spawned child processes. Refer to *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800 for details about user IDs, job names, and the environment variables. Issue `D OMVS,U=userid` to display job names associated with a user ID on the MVS console. The following facility names and priority codes are predefined in the syslogd implementation:

The facility names are:

user	Messages from anyone that does not fall within any of the other categories (the default facility name).
mail	Messages from the mail system.
news	Messages from the Usenet network news system.
uucp	Messages from the UUCP system.
daemon	This facility name is generally used by server processes. The FTPD server, the RSHD server, the REXECD server, the SNMP agent, and SNMP subagent use this facility name to log trace messages.
auth	Messages about authorization.
authpriv	Same as auth.
cron	Messages from the CRON daemon or the AT command.
lpr	Messages from the line printer system.
local0-7	These names are meant for local use. The TelnetD server in the UNIX System Services uses facility name local1 for its log messages.
mark	Used for logging MARK messages.
*****	Wildcard used to represent any facility names.

CS for z/OS IP components use daemon, local1 (used by telnet), user, mail, and auth facilities names. Refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 for details about each server and facility name it uses. Each application activates and deactivates traces in a slightly different manner. For details, refer to the chapter on the individual application.

The priority codes listed in degree highest to lowest are:

emerg	Emergency messages. System is becoming unusable.
panic	Same as emerg.
alert	Immediate action is required.
crit	Critical condition. A device or a component is becoming unusable.
err(or)	Error condition.
warn(ing)	Warning condition.
notice	A condition requiring special handling
info	General information message.
debug	A message useful for debugging programs.
none	Inhibits message logging for the specified facility
*	Wildcard used to represent any priority code, except none.

The criteria for selecting messages for processing are combined with a destination, which tells syslogd what to do with selected messages.

You combine user ID, job name, facility name, and priority codes in the entries in the /etc/syslog.conf file and for each combination you specify a destination for log messages that belong to the specified combination:

```
{  [userid.jobname.]facility name.prioritycode      destination   }
```

Where the user ID and job name are optional, you may specify "*.*" or omit.

The destination can be any of the following:

- A file in the hierarchical file system

  ```
  facility_name.priority_code  /tmp/syslogd/auth.log
  ```

- One or more local users

  ```
  facility_name.priority_code  user1,user2
  ```

  Note: Be careful with this option. If the user is not logged in, all messages will be queued in UNIX System Services.

  You may send the messages to all logged-in users by specifying an asterisk as the user name:

  ```
  facility_name.priority_code  *
  ```

- A syslogd server running on another host called myaixserver

  ```
  facility_name.priority_code  @myaixserver
  ```

- Write messages to SMF record type 109

  ```
  facility_name.priority_code  $SMF
  ```

- MVS console, if syslogd is not running messages are sent here

  ```
  facility_name.priority_code  /dev/console
  ```

Priority code specifications include all higher priorities. If you specify a priority code of, for example, crit, then messages of this priority plus messages with alert, panic and emerg priorities will be logged at the specified destination. To send all messages with a priority of crit or higher to a user ID of OPER1, specify the following rule in /etc/syslog.conf:

```
*.crit     OPER1
```

A message may be logged in more destinations, depending on your rules in /etc/syslog.conf. To capture all messages from the facility name daemon into one file and all messages with a priority of crit or higher into another file, specify the following:

```
daemon.*  /tmp/syslogd/daemon.log
*.crit    /tmp/syslogd/crit.log
```

If a server sends a message to syslogd with a facility name of debug and a priority code of alert, the above rules will log the message in both the daemon.log and the crit.log files.

One of the priority codes has a special meaning; it is the *none* priority code. If you include this priority code in a rule, it means: do *not* select any messages. What is the purpose of such a rule? If you want to log all messages from facility name local1 into one file and all from daemon into another and then everything else into a third, you can use the following rule set:

```
local1.*                      /tmp/syslogd/local1.log
daemon.*                      /tmp/syslogd/daemon.log
*.*;local1.none;daemon.none   /tmp/syslogd/the_rest.log
```

```
# A '#' sign denotes a comment.
# A blank line is ignored.

*.*.*.*;bthomps.*.*.none;*.mail.none  /u/bthomps/%m%d%Y/syslogd.logs
bthomps.bthomps*.*.*                  /u/bthomps/%m%d%Y/bthomps.logs
mail.*                                /tmp/mail/%m%d%Y/mail.logs
local0.*                              /tmp/firewall/%m%d%Y/cmd_msg.log
local4.debug                          /tmp/firewall/%m%d%Y/debug.log
*.INET*.*.*                           /tmp/inet/%m%d%Y/inet.logs
```

Figure 14-4 Sample /etc/syslog.conf from mvs03a used for testing

Assuming syslogd was started on 6-21-2000, then all log messages will be recorded in /u/bthomps/06212000/syslogd.logs except messages from user ID bthomps and mail messages because none is coded. Messages from user ID and job name bthomps(1-9) will be recorded in /u/bthomps/06212000/bthomps.logs. Firewall command message results use local0 and the messages will be recorded in /tmp/firewall/06212000/cmd_msg.log. Firewall also uses local4 for info, notice, and err messages, since debug is the lowest priority all the messages will be recorded in /tmp/firewall/06212000/debug.logs. Any job name INET(1-9) will be recorded in /tmp/inet/06212000/inet.logs.

If syslogd is stopped and restarted on 06-21-2000, the logs will simply be appended. There will be a restarted message logged in /u/bthomps/06212000/syslogd.logs file.

```
# The following example stores messages with facility daemon or
# local1 in the file /directory/logfile.
#
daemon.*;local1.*   /directory/logfile
#
# Write all messages with priority crit or higher to the MVS
# console.  See the UNIX System Services Planning manual for more
# information about the /dev/console special file.
#
*.crit             /dev/console

# Write all messages from syslogd itself to the file
# /var/log/YYYY/MM/DD/syslogd.log and to the system console.
#
# Notes:
#
# a) If syslogd is invoked as a started task with job name
#    SYSLOGD, the name of the long-running syslogd job is
#    SYSLOGD1.  If syslogd is invoked from a shell script
#    (e.g., /etc/rc) with job name SYSLOGD, the name of the
#    long-running syslogd job is SYSLOGD followed by a
#    digit.
#
#    If syslogd runs with a different job name on your system, the
#    rule will have to be changed accordingly.
#
# b) During initialization, syslogd writes messages to
#    /dev/console.  These rules cover messages during steady-
#    state.
#
*.SYSLOGD*.*.*      /var/log/%Y/%m/%d/syslogd
*.SYSLOGD*.*.*      /dev/console
#
# Write all messages from otelnetd and other applications which
# specify facility "local1" when running as user SMITH to the log
# local1.smith. This could be useful if, otelnetd traces need
# to be collected for a problem which user SMITH is experiencing
# and you do not wish to collect otelnetd traces from all user IDs.
#
SMITH.*.local1.*  /var/log/%Y/%m/%d/local1.smith
#
#
# Write all messages with priority crit and higher to the syslogd on
# host 192.168.1.9.
#
# *.crit      @192.168.1.9
#
# Write all messages with priority err and higher to log file errors.
#
# THIS EXAMPLE STATEMENT IS UNCOMMENTED.
#
*.err            /var/log/%Y/%m/%d/errors
#    Write all messages from inetd to the log file inetd and to the
#    console. If inetd is invoked as a started task with job name
#    INETD, the name of the long-running inetd job is INETD1.  If
#    inetd invoked from a shell script with job name INETD, the
#    name of the job is INETD followed by a digit(1-9).
#
# *.INETD*.*.*       /var/log/%Y/%m/%d/inetd
# *.INETD*.*.*       /dev/console
```

Figure 14-5   Sample from UNIX System Services /usr/lpp/tcpip/samples/syslog.conf

14.3.1 syslogd configuration recommendations

Here are some goals for file management:

- Start over with new logs files every day in a new directory.

  Use date/time format strings to create destination files in /etc/syslog.conf that have this format for a destination: /var/log/%Y/%m/%d/logfile_name

- Delete log files that are 14 days old or older.

  Copy the sample REXX program from /usr/lpp/tcpip/samples/rmoldlogs to /usr/local/bin/rmoldlogs and change `cfg.DAYSTOKEEP` from 30 to 14, make /usr/local/bin/rmoldlogs executable with the **chmod** command, and then create a cron job:

  ```
  0 0 * * *  kill -HUP 'cat /etc/syslog.pid'
  0 0 * * * /usr/local/bin/rmoldlogs
  ```

Here are some implementation tips:

- Start syslogd with the -c option; also use -u and -i if needed.
- Copy hlq.SEZAINST(SYSLOGD) to proclib.
- Copy /usr/lpp/tcpip/samples/syslog.conf to /etc/syslog.conf, then un-comment various rules in the file to fit your environment.

Note: The ability for syslogd to substitute the current time or date file names is incompatible with the existing ability for file names to contain a percent sign (%). File names in the syslogd configuration file that contain a '%' must be changed to use two '%%' signs (/tmp/test%file.log becomes /tmp/test%%file.log).

14.3.2 syslog.h

Your own UNIX System Services application programs can use the logging facilities of the syslogd server. Your C program must include the syslog.h header file. The program can then use the following functions to open a log facility, send log messages to syslogd and close the facility again:

```
#include </usr/include/syslog.h>
int main(void) {
openlog("oec", LOG_PID, LOG_LOCAL0); 1
syslog(LOG_INFO, "Hello from oec");  2
closelog();                          3
}
```

1 Open a log facility name of local0. Prefix each line in the log file with the program name (oec) and the process ID.

2 Log an info priority message of the specified content.

3 Close the log facility name again.

The above statements resulted in the following line in the log file:

```
May 26 11:27:51 mvs18oe oec[3014660]: Hello from oec
```

For more information on the syslog function, please see *Advanced Programming in the UNIX Environment*, by Richard W. Stevens, published by Addison-Wesley, SR23-7254.

14.3.3 File syslog.pid

The syslog.pid file is created at startup by syslogd in the /etc/ directory and contains the process ID of syslogd.

```
16777472
```

14.3.4 TCPIP.PROFILE

The following statements in TCPIP.PROFILE are needed for syslogd. This port is required to accept log data from the remote syslogd servers.

```
AUTOLOG 1
    .....
    SYSLOGD JOBNAME SYSLOGD1    ; SyslogD Server
    .....
PORT
    .....
    514 UDP SYSLOGD1            ; SyslogD Server
    .....
```

14.4 Starting syslogd

Only one syslogd runs in UNIX System Services. Prior to IBM Communications Server for OS/390 V2R10 IP, starting more than one copy of syslogd caused unpredictable results. Now, syslogd detects when something is already processing its AF_UNIX socket and won't initialize a second syslogd.

You can start the syslogd process via a small shell script that allows you to set the job name and eventually also pass proper environment variables to the syslogd process; the file is named syslogd.start and located in the /etc/ directory.

```
#
# Start the syslog Daemon
#
export _BPX_JOBNAME='SYSLOGD'
/usr/sbin/syslogd -f /etc/syslog.conf -u -c &
DATE='date'
echo -- /etc/syslogd.start script executed, $DATE
```

You can execute this shell script directly from the /etc/rc file to get syslogd started at UNIX System Services initialization. In the RC file the following was added to start syslogd automatically:

```
# Start the syslog Daemon
/etc/syslogd.start
sleep 5
echo /etc/rc script executed, 'date'
```

In our implementation we used the following procedure to start the syslogd server as a POSIX(ON) program:

```
//SYSLOGD PROC PARMS='-f /etc/syslog.conf -u -c',
// MODULE=SYSLOGD
//SYSLOGD EXEC PGM=&MODULE,REGION=30M,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//            '/&PARMS')
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSOUT    DD SYSOUT=*
//SYSERR    DD SYSOUT=*
//CEEDUMP   DD SYSOUT=*
//*
```

14.5 Switching between two log files

A **kill pid -SIGHUP** command can be used to force syslogd to re-read its configuration file and activate any modified parameters without stopping syslogd.

syslogd just keeps appending log messages to the files you have specified in your /etc/syslog.conf. You need some technique to periodically delete unwanted messages or offload the current log files to another location. If you update the log files from an archive process, you need to stop syslogd while you do so. To avoid stopping syslogd during archive and cleanup, you can create two syslogd configuration files: one called /etc/syslog.conf.a and another called /etc/syslog.conf.b. The two files are equal except that they end with either an a or b.

If your current /etc/syslog.conf file was created from your syslog.conf.a file, then you can copy your syslog.conf.b file to /etc/syslog.conf and send syslogd a SIGHUP signal, which will make syslogd stop writing to the a log files and begin writing to the b log files. This will give you the opportunity to offload the a files and delete their current contents, making them ready for syslogd use again by swapping the other way from the syslog.conf.b file to the syslog.conf.a file. Following are the three configuration files in system MVS03A:

- File syslog.conf

```
 BROWSE -- /etc/syslog.conf -------------------------- Line 00000000 Col 001 067
 Command ===>                                                  Scroll ===> PAGE
********************************* Top of Data **********************************
mail.*            /tmp/mail.log.a
*.*;mail.none      /tmp/syslogd.log.a
******************************** Bottom of Data ********************************
```

- File syslog.conf.a

```
 BROWSE -- /etc/syslog.conf.a ----------------------- Line 00000000 Col 001 067
 Command ===>                                                  Scroll ===> PAGE
********************************* Top of Data **********************************
mail.*            /tmp/mail.log.a
*.*;mail.none     /tmp/syslogd.log.a
******************************** Bottom of Data ********************************
```

- File syslog.conf.b

```
 BROWSE -- /etc/syslog.conf.b------------------------ Line 00000000 Col 001 067
 Command ===>                                                  Scroll ===> PAGE
********************************* Top of Data **********************************
mail.*            /tmp/mail.log.b
*.*;mail.none     /tmp/syslogd.log.b
******************************** Bottom of Data ********************************
```

14.6 Centralized logging

As you can see in Figure 14-3 on page 438, syslogd is able to receive log data from other syslogd servers in your network as well as send log data to other syslog servers in your network. For that purpose, syslogd creates a UDP socket and binds it to port 514 during initialization. If there is no AF_INET transport provider connected to z/OS UNIX when syslogd is started, syslogd's bind() socket call fails. syslogd does not abort its initialization for this reason, but it does not retry the bind() call either, which means that syslogd in this situation will not be able to receive log data from other syslogd servers in your network. To cope with this situation, we decided to AUTOLOG syslogd. When the stack is started, the autolog function will find syslogd already running, cancel and restart it. The following log shows the syslogd being and canceled and restarted by TCP/IP when the TCP/IP stack starts.

```
S TO3ATCP
$HASP100 TO3ATCP  ON STCINRDR
IEF695I START TO3ATCP  WITH JOBNAME TO3ATCP  IS ASSIGNED TO USER TCPIP3
 , GROUP OMVSGRP
$HASP373 TO3ATCP  STARTED
IEF403I TO3ATCP - STARTED - TIME=14.04.18
IEE252I MEMBER CTIEZB01 FOUND IN SYS1.PARMLIB
EZZ7450I FFST SUBSYSTEM IS NOT INSTALLED
EZZ0300I OPENED PROFILE FILE DD: PROFILE
EZZ0309I PROFILE PROCESSING BEGINNING FOR DD: PROFILE
         "
         "
EZZ0621I AUTOLOG FORCING SYSLOGD1, REASON: TCP/IP HAS BEEN RESTARTED
CANCEL SYSLOGD1
IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=SYSLOGD1, ASID=003B.
IEE301I SYSLOGD1          CANCEL COMMAND ACCEPTED
IEF450I SYSLOGD1 STEP1 - ABEND=S222 U0000 REASON=00000000 201
        TIME=14.05.26
S SYSLOGD
$HASP100 SYSLOGD  ON STCINRDR
IEF695I START SYSLOGD  WITH JOBNAME SYSLOGD  IS ASSIGNED TO USER
OMVSKERN, GROUP OMVSGRP
$HASP373 SYSLOGD  STARTED
IEF403I SYSLOGD - STARTED - TIME=14.05.37
```

Figure 14-6 syslogd restarted by AUTOLOG

We are running three MVS systems: MVS03A, MVS28A, MVS39A. MVS28A and MVS39A send all the messages about mail to the system MVS03A according to the following definitions in MVS28A and MVS39A. All other messages of the servers go to /tmp/syslogd.log.a.

```
 BROWSE -- /etc/syslog.conf ------------------------- Line 00000088 Col
  Command ===>                                                Scroll ==
 mail.*                  @MVS03A
 *.*;mail.none           /tmp/syslogd.log.a
 ******************************* Bottom of Data ***********************
```

The three systems have syslogd running and netstat shows port 514 assigned to SYSLOGD1.

```
EZZ2585I User Id  Conn  Local Socket           Foreign Socket         State
EZZ2587I SYSLOGD1 02522 0.0.0.0..514           *..*                   UDP
```

In the remote systems MVS28A and MVS39A the byte count of the outgoing counter is increasing and the incoming counter is staying at zero, but in the system MVS03A the incoming counter is increasing and the outgoing counter is staying at zero. The system MVS03A is the system where the mail log is sent to from the systems MVS28A and MVS39A.

Figure 14-7 shows the result in the file /tmp/mail.log.a of system MVS03A:

```
Feb 8 10:50:28 MVS03A sendmail 117 0537 : gethostbyaddr(192.168.20.3) failed: 0
Feb 8 10:50:28 MVS03A sendmail 117 0537 : gethostbyaddr(192.168.100.100) failed: 0
Feb 8 10:50:28 MVS03A sendmail 117 0537 : NOQUEUE: SYSERR(OMVSKERN): dbm map  Alias0 :  unsafe map file
/etc/aliases
Feb 8 11:26:0  MVS03A sendmail 3355 63 : gethostbyaddr(192.168.20.3) failed: 0
Feb 8 11:26:0  MVS03A sendmail 3355 63 : gethostbyaddr(192.168.100.100) failed: 0
Feb 8 16:56:37 mvs39a.itso.ral.ibm.com sendmail 385875975 : EZZ7510I user  027 attempted to run daemon
Feb 8 16:58:03 mvs39a.itso.ral.ibm.com sendmail 3355 51 : EZZ751 I: sendmail starting
Feb 8 16:58:05 mvs39a.itso.ral.ibm.com sendmail 20 : starting daemon (8.8.7): SMTP
Feb 8 16:59:26 mvs28a.itso.ral.ibm.com sendmail 3355 57 : EZZ751 I: sendmail starting
Feb 8 16:59:26 mvs28a.itso.ral.ibm.com sendmail 15099 966 : starting daemon (8.8.7): SMTP
Feb 8 17:1 : 8 mvs39a.itso.ral.ibm.com sendmail 50331669 : NOQUEUE: SYSERR(OMVSKERN): dbm map  Alias0 :
unsafe map file
/etc/aliases
Feb 8 17:17:13 mvs39a.itso.ral.ibm.com sendmail 67108885 : NOQUEUE: SYSERR(OMVSKERN): dbm map  Alias0 :
unsafe map file
/etc/aliases
Feb 8 17:18: 6 mvs39a.itso.ral.ibm.com sendmail 83886101 : alias database /etc/aliases rebuilt by
PESCHKE
Feb 8 17:18: 6 mvs39a.itso.ral.ibm.com sendmail 83886101 : /etc/aliases: 13 aliases, longest 32 bytes,
289 bytes total
Feb 8 17:20:13 mvs39a.itso.ral.ibm.com sendmail 805306383 : alias database /etc/aliases rebuilt by
PESCHKE
Feb 8 17:20:13 mvs39a.itso.ral.ibm.com sendmail 805306383 : /etc/aliases: 13 aliases, longest 32 bytes,
289 bytes total
Feb 8 17:25: 1 mvs39a.itso.ral.ibm.com sendmail 536870919 : RAA536870919: collect: premature EOM:
EDC5122I Input/output error.
Feb 8 17:25: 1 mvs39a.itso.ral.ibm.com sendmail 536870919 : RAA536870919: from=PESCHKE, size=90,
class=0, pri=90, nrcpts=0,
msgid=<199902082221.RAA536870919@MVS39A.itso.ibm.com>, relay=OMVSKERN@localhost
Feb 8 17:29:32 mvs39a.itso.ral.ibm.com sendmail 251658259 : RAA251658259: from=PESCHKE, size=226,
class=0, pri=30226, nrcpts=1,
msgid=<199902082228.RAA251658259@MVS39A.itso.ibm.com>, relay=PESCHKE@localhost
Feb 8 17:29:32 mvs39a.itso.ral.ibm.com sendmail 251658259 : RAA251658259: to=PESCHKE@MVS28A,
delay=00:01:30, mailer=esmtp,
stat=queued
Feb 8 18:10:56 MVS03A sendmail 268 35 85 : EZZ751 I: sendmail starting
Feb 8 18:10:57 MVS03A sendmail 268 35 85 : gethostbyaddr(192.168.100.100) failed: 0
Feb 8 18:10:57 MVS03A sendmail 3355 336 : starting daemon (8.8.7): SMTP
```

Figure 14-7 Logging of three systems recorded at MVS03A

15

Time server (TIMED)

TIMED (RFC 868) is a server used to provide the date and time. The time service sends back to the originating source the time in seconds since midnight on January 1, 1900.

One motivation arises from the fact that not all systems have a date/time clock, and all are subject to occasional human or machine error. Network stations can obtain a clock from this server. The use of time servers makes it possible to quickly confirm or correct a system's idea of the time by making a brief poll of several independent sites on the network.

The time server is invoked using `timed` from the UNIX System Services shell environment or with the TIMED procedure in MVS.

User Datagram Protocol (UDP) is used as the protocol with `timed`. The time service works as follows:

S: Listen on port 37 (45 octal).

U: Send an empty datagram to port 37.

S: Receive the empty datagram.

S: Send a datagram containing the time as a 32-bit binary number.

U: Receive the time datagram.

The server listens for a datagram on port 37. When a datagram arrives, the server returns a datagram containing the 32-bit time value. If the server is unable to determine the time at its site, it should discard the arriving datagram and make no reply.

Note: Although the TIMED server supports the UDP protocol only, the Time service that can run over the TCP and UDP protocol is provided by the InetD super daemon internally. See 11.2, “Internet services supported internally by InetD” on page 420 for more information.

The time used by **timed** is the number of seconds since 00:00 (midnight) of January 1, 1900 GMT, such that the time 1 is 12:00:01 a.m. on January 1, 1900 GMT; this base will serve until the year 2036.

For example:

```
the time  2,208,988,800 corresponds to 00:00  1 Jan 1970 GMT,

            2,398,291,200 corresponds to 00:00  1 Jan 1976 GMT,

            2,524,521,600 corresponds to 00:00  1 Jan 1980 GMT,

            2,629,584,000 corresponds to 00:00  1 May 1983 GMT,

       and -1,297,728,000 corresponds to 00:00 17 Nov 1858 GMT.
```

16

Web server performance

This chapter describes the interaction between the HTTP Server and z/OS to optimize the server performance.

The following sections are included in this chapter:

16.1 Overview

The current design of the Web server (See Figure 16-1) has all the function running above the kernel, where kernel refers to all code residing at or below the socket API, including both UNIX Services and the TCP/IP stack.

From a performance study it appears that 85% of the processing time for a given query to the Web server is spent in the Kernel code. So to obtain better performance, changes have been made in the kernel code.

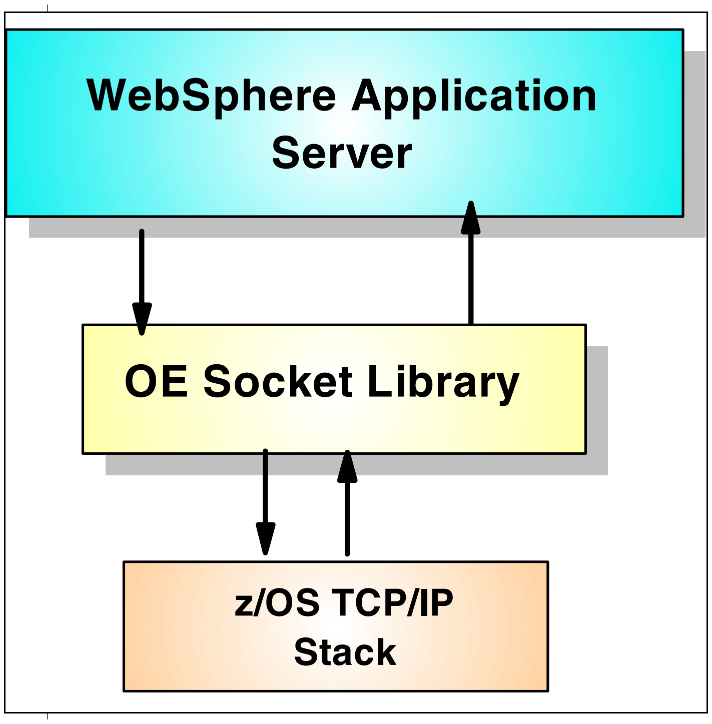


Figure 16-1 Current solution flow

16.2 Fast Response Cache Accelerator

This new solution is based upon a hybrid kernel/user Web server as shown in Figure 16-2.

Web pages are cached within the TCP/IP stack, and requests are handled without passing the entire kernel or entering the user space, but by using the z/OS Cache Accelerator exits shipped with z/OS and the WebSphere Application Server Kernel-Space exits provided by the WebSphere Application Server product.

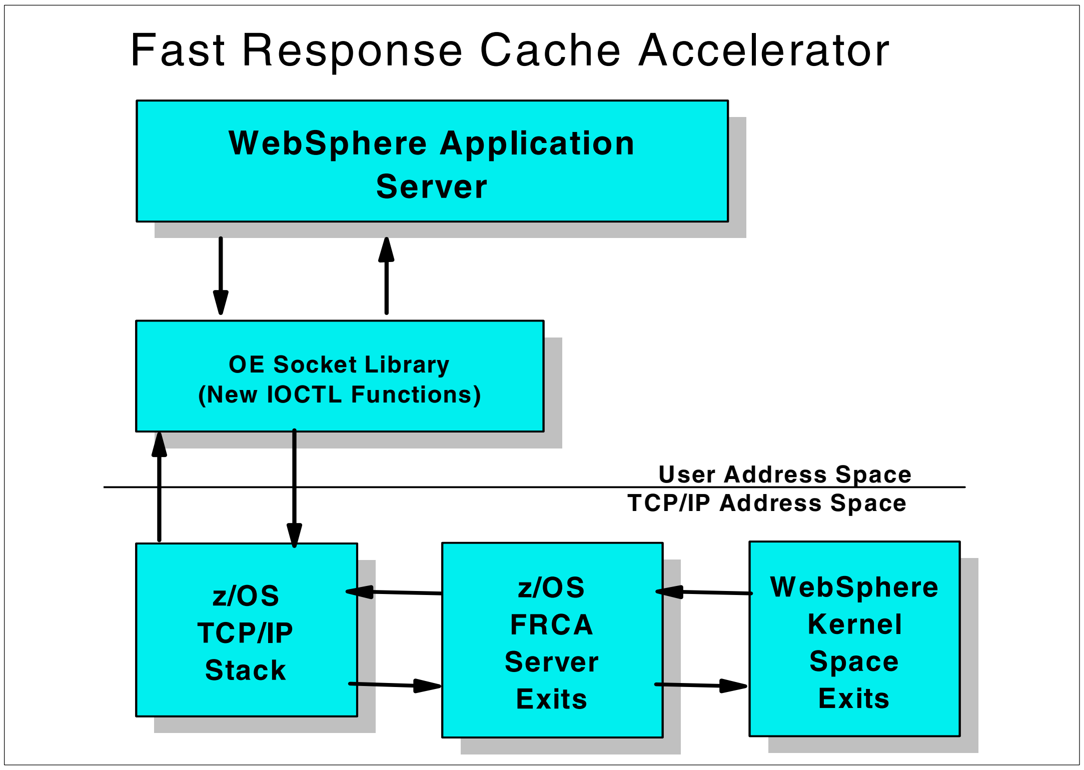

Fast Response Cache Accelerator
WebSphere Application Server
OE Socket Library (New IOCTL Functions)
User Address Space
TCP/IP Address Space
z/OS TCP/IP Stack
z/OS FRCA Server Exits
WebSphere Kernel Space Exits

Figure 16-2 Overview of the FRCA solution flow

The Cache Accelerator (if enabled) is loaded automatically during the Web server operation so you are not required to list the files to be cached in the server configuration file. In addition the server will recache the updated pages and remove the outdated pages from the cache.

The FRCA provides support for caching on multiple Web servers and on servers with multiple IP addresses. At the moment, support is not available for a proxy server.

Communications Server for z/OS IP provides an enhancement so that the cache serving responsibility can be passed back from the WebSphere application to the TCP/IP stack Fast Response Cache Accelerator.

This results in faster response time for the Web client if the Web content includes dynamic Web pages.

16.3 Starting the HTTP server

Before you can start the HTTP Server for OS/390 V5.3 you have to do several system and RACF definitions. For detailed information about these please refer to *z/OS V1R1.0-V1R3.0 IBM HTTP Server Planning, Installing, and Using V5.3*, SC34-4826.

The following procedure is used to start the server:

```
//IMWPROC PROC  LEPARM=,
//*IMWPROC PROC  LEPARM=,ICSPARM=
//************************************************************
//*
//* LEPARM  ==> LE runtime opts
//* LEPARM='ENVAR("_CEE_ENVFILE=/etc/httpd.envvars")'
```

```
//*
//* ICSPARM ==> Internet Connection Server parameters
//* # Standalone HTTPD
//* ICSPARM='-p 8080 -r /etc2/httpd.conf'
//* ICSPARM='-p 80 -vv -r /etc/httpd.conf'
//*              1       2        3        4
//  ICSPARM='-p 80 -SN FRCAHTTP -vv -r /etc/httpd.conf'
//* # WLM Queue Manager
//* ICSPARM='-SN WEBSN1 -p 8080 -r /etc/httpd.conf'
//* # WLM ApplEnv Queue Server
//* ICSPARM='-SN WEBSN1 -AE WEBHTML'
//*
//*  Internet Connection Server Parameters:
//*   -SN                         # WLM - subsystem name
//*   -AE                         # WLM - Application Environment
//*
//*   -fscp      nnn              # File system codepage - EBCDIC
//*   -netcp     nnn              # net code page        - ASCII
//*
//*   -gc_only                    # clean cache & exit (garbage collect)
//*
//*   -normalmode
//*   -p         nnnn             # use port nnn (default 80)
//*   -sslmode
//*   -sslport   nnnn             # use port nnn (default 443)
//*   -nosec                      # no security
//*
//*   -nosmf                      # no smf processing on
//*   -smf                        # smf processing on
//*
//*   -r         /etc/httpd.conf  # use rule file xxxx
//*   -restart
//*   -v                          # trace to stderr
//*   -vv                         # trace all to stderr
//*   -vc                         # cache trace to stderr
//*
//*   -version                    # show version and exit
//*
//*
//*   xxxxxxx                     # ServerRoot xxxxxxx; Pass /*
//*
//*****************************************************************
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=OK,TIME=NOLIMIT,
//   PARM=('&LEPARM/&ICSPARM')
//*STEPLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//*****************************************************************
//SYSIN    DD DUMMY
//OUTDSC   OUTPUT DEST=HOLD
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
```

In this PROC:

1 This is the HTTP Server port number (default).

2 This is the WLM subsystem name and must match the value specified in the configuration file (See Figure 242 on page 405).

3 A trace of the HTTP Server component will be taken and directed to STDOUT.

4 This is the name of the configuration file to be used (default).

16.4 Configuring the HTTP server

The Web administrator can use two different methods to configure the HTTP Server:

- Using the configuration and administration forms:

  The server comes with configuration and administration forms. These forms are a combination of CGI programs and HTML forms that provide an easy way for you to configure or to view the current configuration of your server.

- Manually editing the configuration file:

  The configuration file (by default, /etc/httpd.conf) is made up of statements called directives. You change your configuration by editing this file, updating the directives and saving your changes.

In our implementation we used the first method, and we strongly suggest all beginner Web administrators use it. Each form provides instructions to assist you in deciding what changes to make. For further information you can click the question mark (**?**) in the upper right-hand corner of the page and choose from three kinds of help:

- **Field description** for information about each field on the form
- **How do I?** for information about configuration tasks
- **Index** for an index of all help topics in alphabetical order

After you fill in the form, click **Submit** to update the server configuration (/etc/httpd.conf will be updated) with the changes you made.

A status message on the top of the page will tell you if the operation completed successfully or not. After a successful completion click the **Restart** button (|) in the upper right-hand corner of the page.

Note: If you change some basic values, you have to stop and restart your server in order for your changes to take effect.

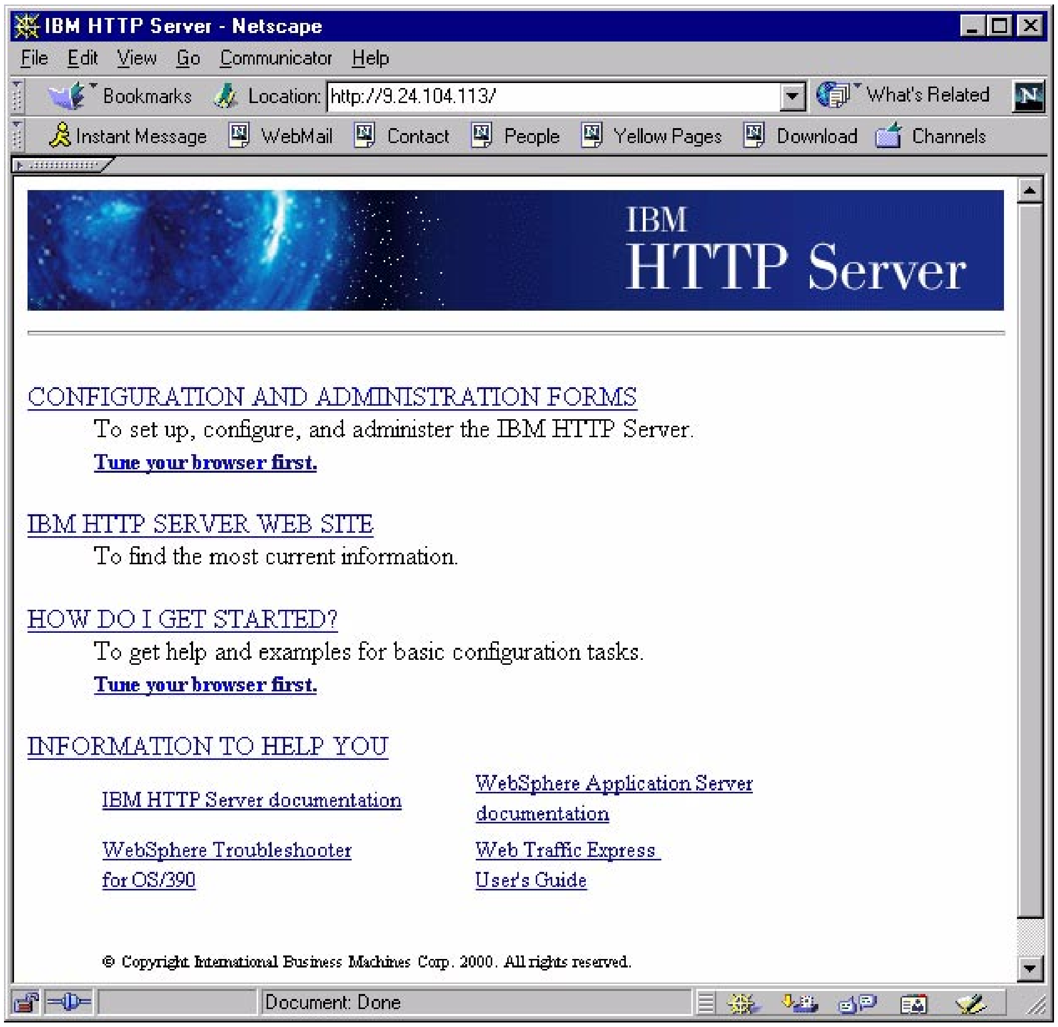
16.4.1 Accessing the configuration and administration forms

When the HTTP Server is started for the first time, go to the server's Front Page by typing the following URL:

```
http://your.server.name
```

where `your.server.name` is the fully qualified name or address of your host.

Then you will see the following window:


IBM HTTP Server - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://9.24.104.113/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
IBM HTTP Server

CONFIGURATION AND ADMINISTRATION FORMS

To set up, configure, and administer the IBM HTTP Server.

Tune your browser first.

IBM HTTP SERVER WEB SITE

To find the most current information.

HOW DO I GET STARTED?

To get help and examples for basic configuration tasks.

Tune your browser first.

INFORMATION TO HELP YOU

IBM HTTP Server documentation | WebSphere Application Server documentation

WebSphere Troubleshooter for OS/390 | Web Traffic Express User's Guide

© Copyright International Business Machines Corp. 2000. All rights reserved.

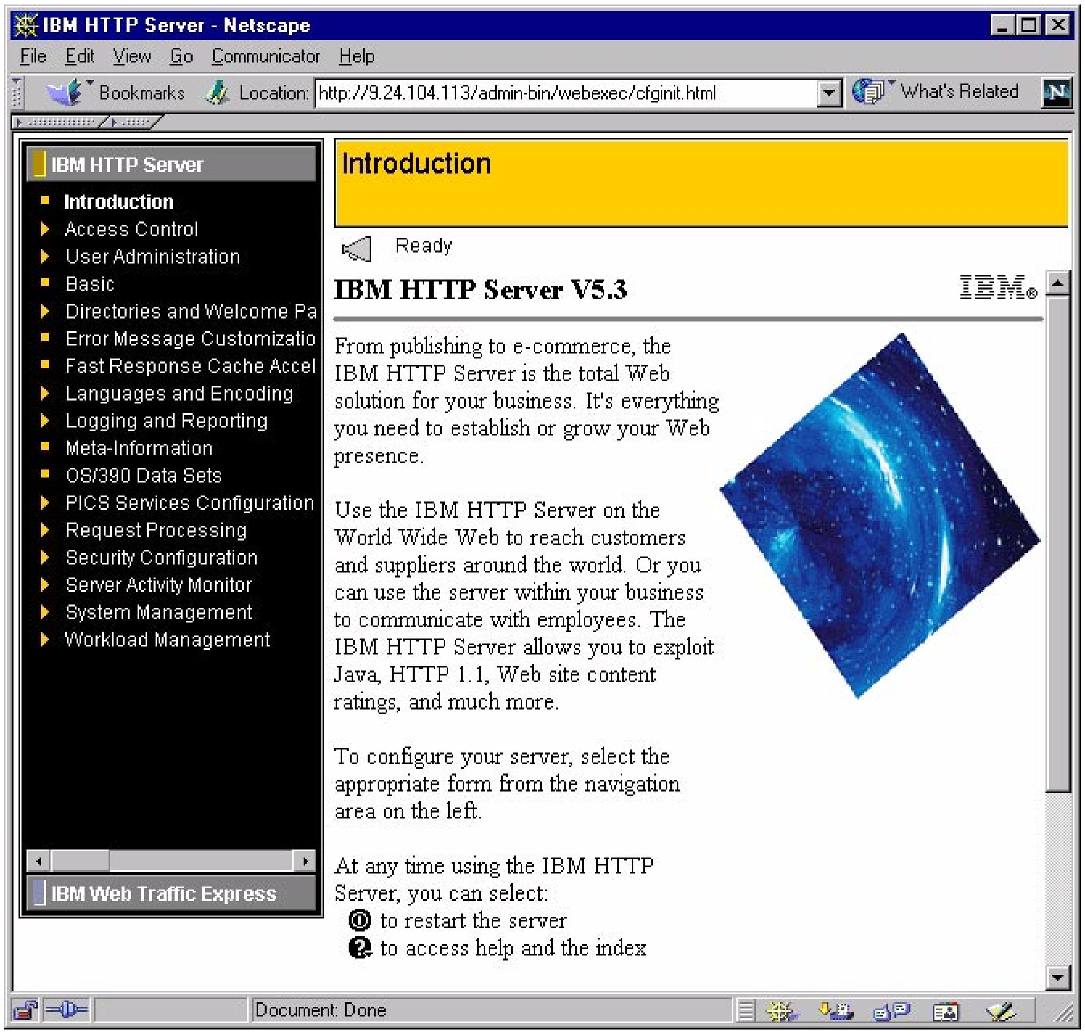
Document: Done

Figure 16-3 Web server default welcome page

Note: You will be prompted for a user name and password the first time you use the forms.

Click **Configuration and Administration Forms**.

The introduction page is shown in Figure 16-4.



IBM HTTP Server V5.3

From publishing to e-commerce, the IBM HTTP Server is the total Web solution for your business. It's everything you need to establish or grow your Web presence.

Use the IBM HTTP Server on the World Wide Web to reach customers and suppliers around the world. Or you can use the server within your business to communicate with employees. The IBM HTTP Server allows you to exploit Java, HTTP 1.1, Web site content ratings, and much more.

To configure your server, select the appropriate form from the navigation area on the left.

At any time using the IBM HTTP Server, you can select:
- to restart the server
- to access help and the index

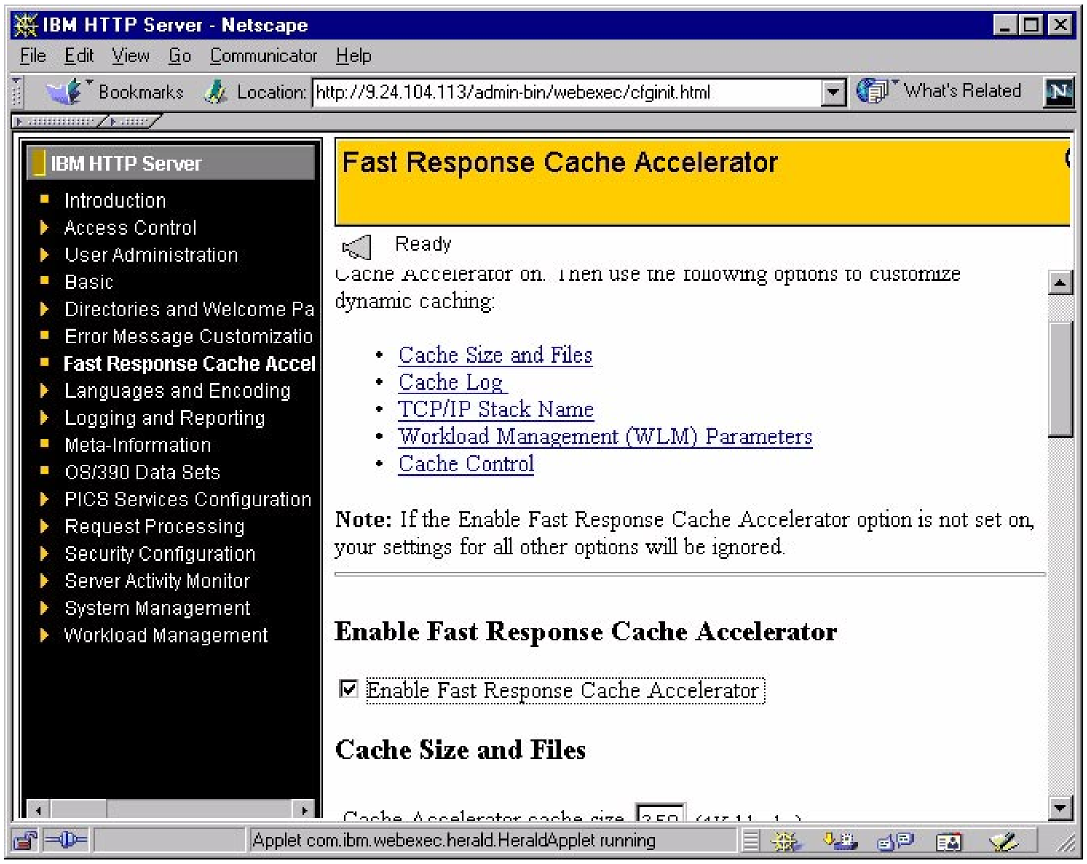
Figure 16-4 Introduction HTTP Server page

16.4.2 Configuring the Fast Response Cache Accelerator

Here we will describe the necessary steps to take to configure correctly the Fast Response Cache Accelerator option. For more information about any parameter value please consult the online guide or *z/OS V1R1.0-V1R3.0 IBM HTTP Server Planning, Installing, and Using V5.3*, SC34-4826.

To access the FRCA forms you have to select, on the left frame of the introduction page, **Fast Response Cache Accelerator**.

You will see the following window:


IBM HTTP Server - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://9.24.104.113/admin-bin/webexec/cfginit.html
What's Related
IBM HTTP Server
Introduction
Access Control
User Administration
Basic
Directories and Welcome Pa
Error Message Customizatio
Fast Response Cache Accel
Languages and Encoding
Logging and Reporting
Meta-Information
OS/390 Data Sets
PICS Services Configuration
Request Processing
Security Configuration
Server Activity Monitor
System Management
Workload Management
Ready
Applet com.ibm.webexec.herald.HeraldApplet running

Fast Response Cache Accelerator

Cache Accelerator on. Then use the following options to customize dynamic caching:

- Cache Size and Files
- Cache Log
- TCP/IP Stack Name
- Workload Management (WLM) Parameters
- Cache Control

Note: If the Enable Fast Response Cache Accelerator option is not set on, your settings for all other options will be ignored.

Enable Fast Response Cache Accelerator

[✓] Enable Fast Response Cache Accelerator

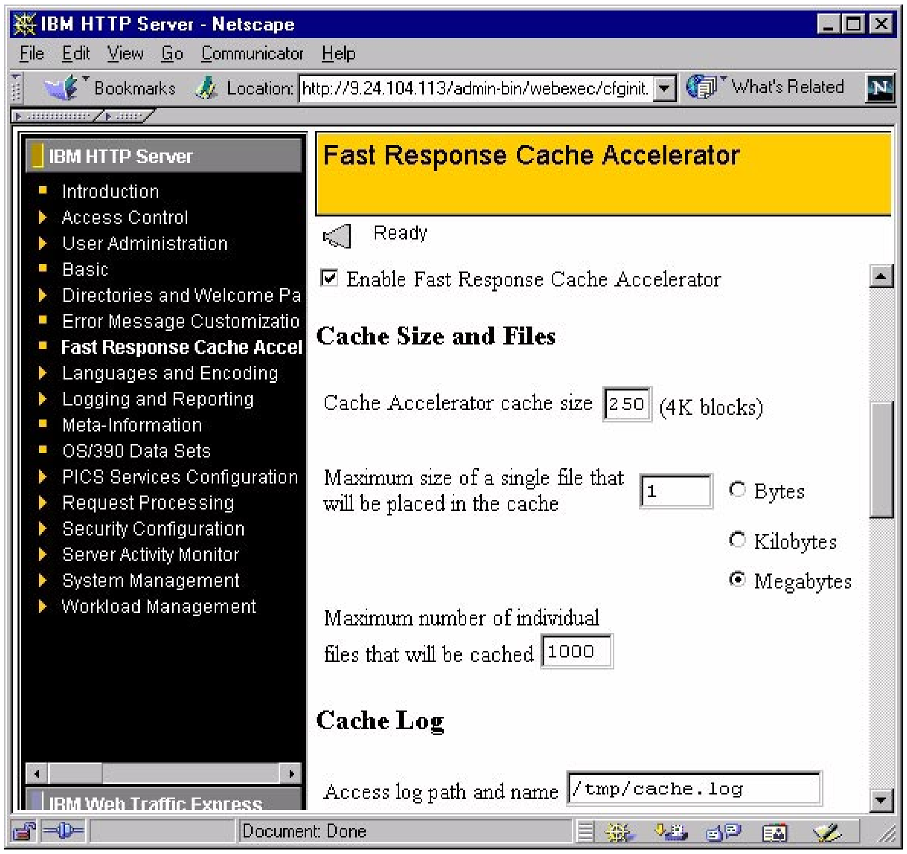
Cache Size and Files

Cache Accelerator cache size

Figure 16-5 Fast Response Cache Accelerator form (part 1)

You must check **Enable Fast Response Cache Accelerator**. If you don't set this value, any other parameter entered will be ignored.

Next, you can set the cache size (the default values are acceptable) and the location of the cache log (if you don't set the value the entries will be recorded in the common log file).



Fast Response Cache Accelerator

Ready

[✓] Enable Fast Response Cache Accelerator

Cache Size and Files

Cache Accelerator cache size 250 (4K blocks)

Maximum size of a single file that will be placed in the cache 1 ( ) Bytes ( ) Kilobytes [✓ Megabytes]

Maximum number of individual files that will be cached 1000

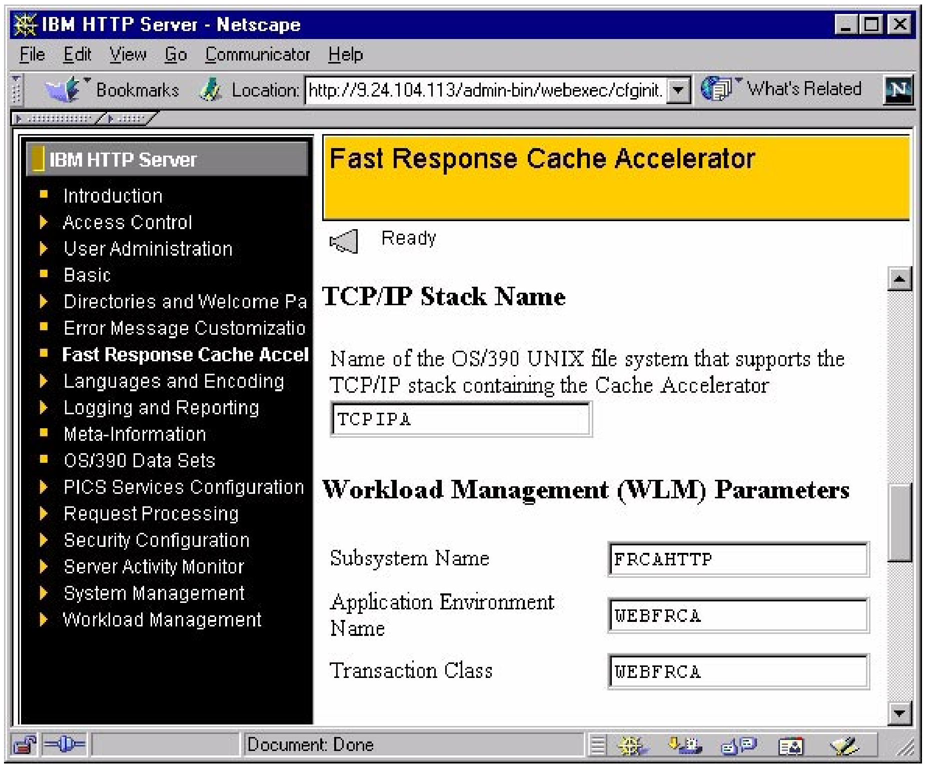
Cache Log

Access log path and name /tmp/cache.log

Figure 16-6 Fast Response Cache Accelerator form (part 2)

If you use the CINET function with multiple TCP/IP stacks, specify the name of the stack used by the Fast Response Cache Accelerator. This name must match the name on the SUBFILESYSTYPE statement in the UNIX BPXRMxx parmlib member.

To classify the work performed by the Fast Response Cache Accelerator under the Workload Manager (WLM), the only requested parameter is the Subsystem Name, which must match the SN parameter in the HTTP Server startup procedure (See 16.3, “Starting the HTTP server” on page 453).



Fast Response Cache Accelerator

TCP/IP Stack Name

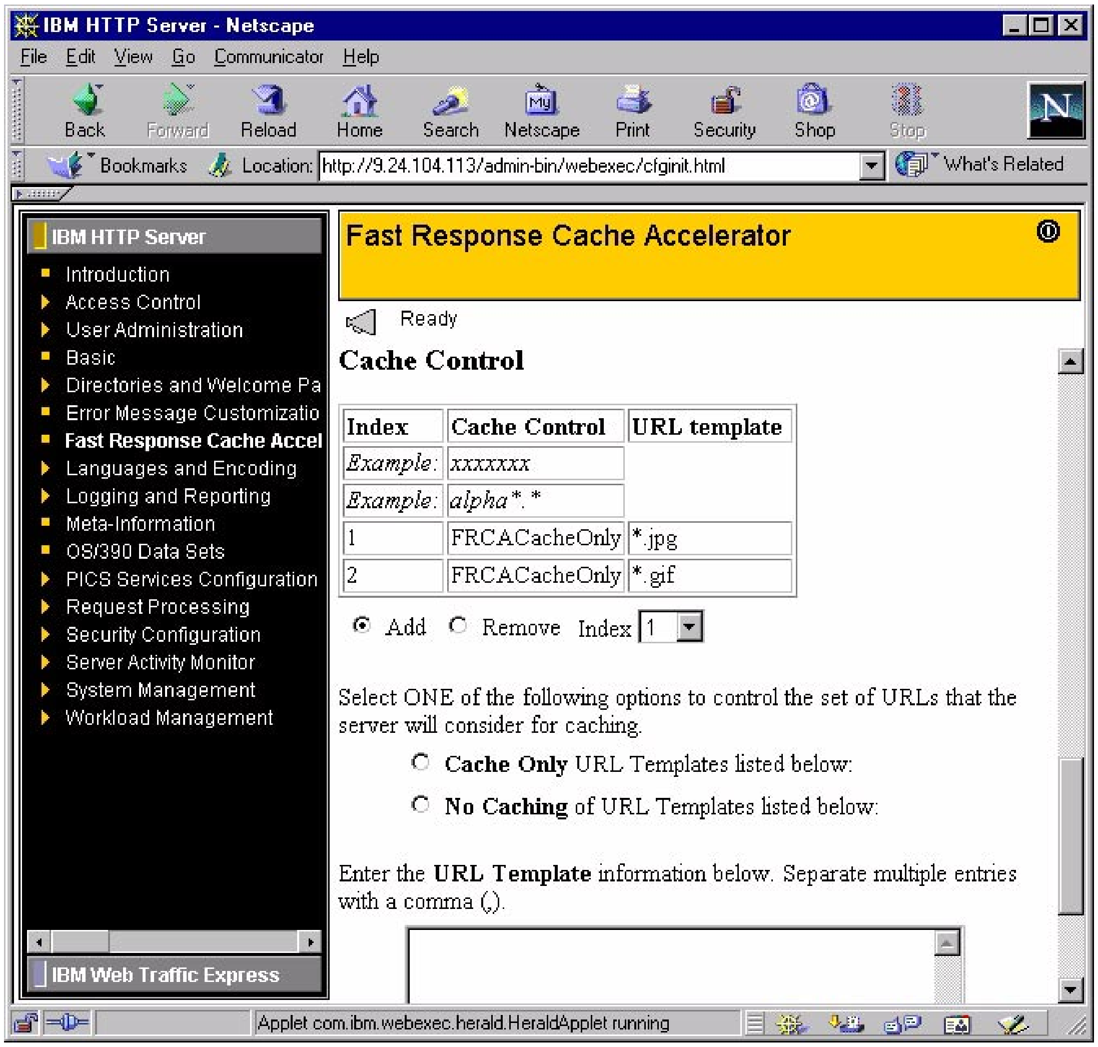
Name of the OS/390 UNIX file system that supports the TCP/IP stack containing the Cache Accelerator: TCPIPA

Workload Management (WLM) Parameters

Subsystem Name	FRCAHTTP
Application Environment Name	WEBFRCA
Transaction Class	WEBFRCA

Figure 16-7 Fast Response Cache Accelerator form (part 3)

If you want to control the set of URLs that the FRCA will consider for cache (it means you want to limit the cache utilization) you can specify the appropriate value in the following form shown in Figure 16-8 on page 461. Wildcards are supported in those fields.


IBM HTTP Server - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://9.24.104.113/admin-bin/webexec/cfginit.html
What's Related
IBM HTTP Server
Introduction
Access Control
User Administration
Basic
Directories and Welcome Pa
Error Message Customizatio
Fast Response Cache Accel
Languages and Encoding
Logging and Reporting
Meta-Information
OS/390 Data Sets
PICS Services Configuration
Request Processing
Security Configuration
Server Activity Monitor
System Management
Workload Management
IBM Web Traffic Express
Fast Response Cache Accelerator
Ready
Applet com.ibm.webexec.herald.HeraldApplet running

Cache Control

Index	Cache Control	URL template
Example:	*xxxxxxx*	
Example:	*alpha**.*	
1	FRCACacheOnly	*.jpg
2	FRCACacheOnly	*.gif

[✓ Add] Remove Index 1

Select ONE of the following options to control the set of URLs that the server will consider for caching.

- **Cache Only** URL Templates listed below:
- **No Caching** of URL Templates listed below:

Enter the **URL Template** information below. Separate multiple entries with a comma (,).

Figure 16-8 Fast Response Cache Accelerator form (part 4)

At this point the FRCA configuration has done. Then you have to click **Submit**, and after a successful response, restart server by selecting the **to restart the server** button (see page Figure 16-4 on page 457).

16.4.3 Monitoring the Fast Response Cache Accelerator

A new **DISPLAY TCPIP** command has been added to show the statistics of the cache accelerator utilization. The syntax is the following:

```
DISPLAY TCPIP,FRCA_Stack_Name,NET,CACH
```

where `FRCA_Stack_Name` is the TCP/IP stack name used by FRCA (See Figure 16-7 on page 460).

This is a sample output obtained:

```
D TCPIP,TCPIPA,N,CACH
EZZ2500I NETSTAT CS V2R10 TCPIPA 367
CLIENT: WEBSRV         LISTENING SOCKET:  0.0.0.0..80
  MAXCACHESIZE:       0000025000 1  CURRCACHESIZE:      0000000010  8
  MAXNUMOBJECTS:      0000001000 2  CURRNUMOBJECTS:     0000000005  9
  NUMCONNS:           0000000010 3  CONNSPROCESSED:     0000000000  10
  CONNSDEFERRED:      0000000010 4  CONNSTIMEDOUT:      0000000000  11
  REQUESTSPROCESSED:  0000000000 5  INCOMPLETEREQUESTS: 0000000000  12
  NUMCACHEHITS:       0000000000 6  NUMCACHEMISSES:     0000000018  13
  NUMUNPRODCACHEHITS: 0000000000 7
1 OF 1 RECORDS DISPLAYED
```

Figure 16-9 Sample report from the NETSTAT CACHE command

1 The maximum number of 4K (4096-byte) blocks of memory that may be used for storing cache objects by the FRCA (See Figure 16-6 on page 459).

2 The maximum number of cache objects that may be stored by the FRCA (See Figure 16-6 on page 459).

3 The number of connections established.

4 The number of connections that are processed by the Web server application.

5 The number of times a response is sent to a client.

6 The number of cache objects that were successfully located and transmitted to clients.

7 The number of cache objects which were successfully located but not transmitted to clients.

8 The current number of 4KB pages used to cache objects.

9 The current number of cached objects.

10 The number of connections that have successfully completed an in-kernel transaction, resulting in a response sent to the client.

11 The number of connections for which timeout is expired.

12 The number of times a request is received by a client but additional data is required to process the request.

13 The number of cache objects that were not successfully located and transmitted to clients.

Note: In our test, we were not able to heavily load the HTTP Server, so the statistics showed that we just issued a few get requests for some Web page to the Web server, which might not be enough for the application to cache the object. Since there was no cached entry in FRCA, every connection was deferred to the Web server application.

To diagnose a problem in the HTTP Server for OS/390 V5.3 it is possible to activate a trace in the startup procedure (See 16.3, "Starting the HTTP server" on page 453).

The following is a sample log output obtained with an active trace. We considered only the entries regarding the FRCA process.

```
 :
Initializing FRCA support
FRCA log size is 184320                                        1
FRCA Timeout value is 5000
Generated server name to be used by FRCA:  IBM HTTP Server/V5R3M0
FRCA Cache objects per connection is 5.
FRCA Maximum Cache Size is 25000 4K blocks
FRCA maximum number of Cache entries is 1000
FRCA TCPIP stackname is TCPIPA
FRCA version is 1
FRCA LoadMod Name = "IMWUWDX "
SiocAfpaConfigure was successful, FRCA support initialized
signal...... 23 could not be set!
Timer....... not being created for a type-12 thread.
Log......... "/tmp/cache.log.Jul112000" opened
Logging..... FRCA updating SNMP/PerfMon counters.
                       :
Setting __SERVER_CLASSIFY_TRANSACTION_CLASS=WEBFRCA worked       2
                       :
etting __SERVER_CLASSIFY_TRANSACTION_NAME=GET worked
                       :
etting __SERVER_CLASSIFY_USERID=WEBSRV worked
FRCA enclave created successfully
                       :
  FRCA: Fast Response Cache Accelarator is enabled.
  Local....... filename is "/usr/lpp/internet/server_root/icons/child.jpg
  FRCA: Fast Response Cache Accelerator values being checked.
  HTStat...... on file "/usr/lpp/internet/server_root/icons/child.jpg" --
 HTAccess.... Õ/icons/child.jpg' has been accessed.                    3
  FRCA: Fast Response Cache Accelarator is enabled.
  FRCA: Fast Response Cache Accelerator values being checked.
  HTServer.c: Entry to RequestCacheable req = 1574e134
  Searching for URL /icons/child.jpg in FRCA cacheable list
 ----------  We matched|
```

Figure 16-10 Sample HTTP Server trace

1 The configuration values are processed

2 The WLM environment is established

3 A request from a client to obtain an object (child.jpg) is processed and according to the mask we specified (See Figure 16-8 on page 461) it has been found eligible to be cached by FRCA.

Part 6

Appendixes

A

BIND DNS sample configuration

In Chapter 10, “BIND Domain Name System (DNS)” on page 315, we described the implementation of several DNS scenarios, including both BIND 4.9.3-based DNS and BIND 9-based DNS. This appendix contains all the files created to build those scenarios.

A.1 BIND 4.9.3-based DNS implementation

This section includes sample configurations for our BIND 4.9.3 implementation.

A.1.1 Basic scenario (no WLM)

Please refer to Figure A-1 for the configuration used in this section.

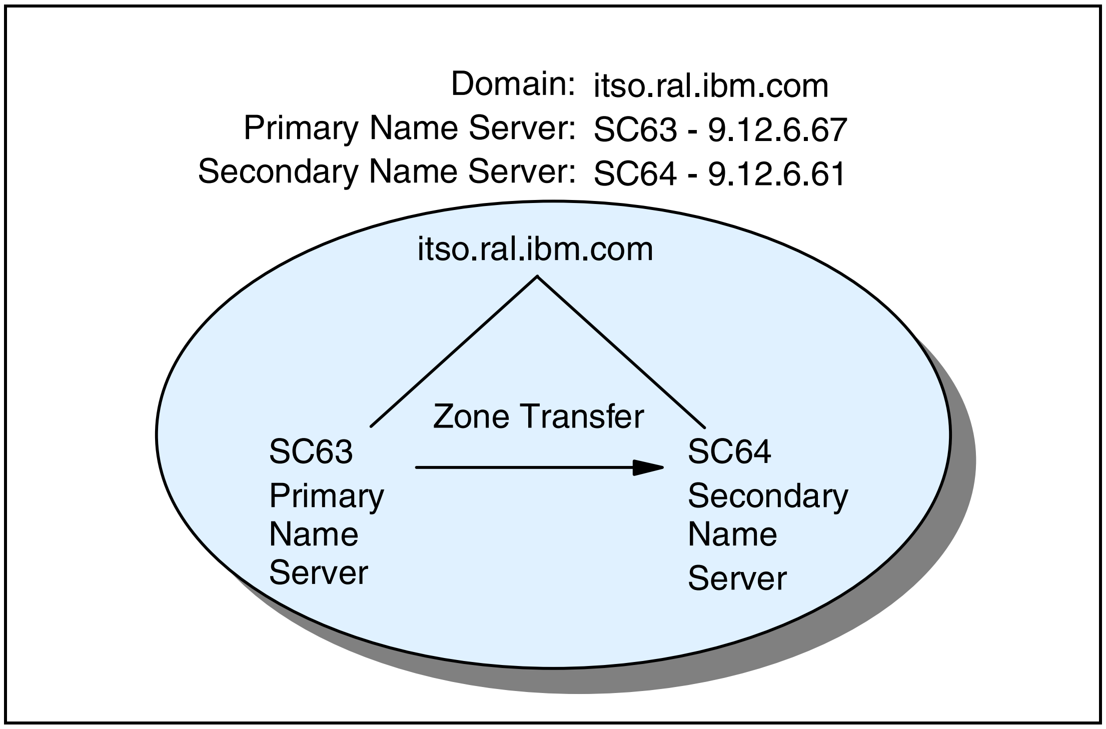

Domain: itso.ral.ibm.com
Primary Name Server: SC63 - 9.12.6.67
Secondary Name Server: SC64 - 9.12.6.61
itso.ral.ibm.com
Zone Transfer
SC63 Primary Name Server
SC64 Secondary Name Server

Figure A-1 SC63, SC64 basic name server implementation

A.1.1.1 SC63 definitions (primary name server)

SC63 boot file (named.boot)

```
********************************* Top of Data ************************
;      boot file for BIND 4.9.3-Based Name Server
;
;type      domain                  source file or host
;
directory  /etc/octavio/dnsdata
primary    itso.ral.ibm.com          zos12.for.v4
primary    6.12.9.in-addr.arpa       zos12.rev.v4
primary    0.0.127.in-addr.arpa      zos12.lbk.v4
cache      .                         zos12.ca.v4
forwarders 9.12.6.68
options    forward-only
options    query-log
******************************** Bottom of Data **********************
```

SC63 BIND 4.9.3 forward domain file (itso.for.v4)

```
********************************* Top of Data ***************************
;  /etc/octavio/dnsdata/itso.for.v4
;  this file is the forward file for the Name Server SC63
$ORIGIN com.
```

```
; this record defines the authoritative server for itso subdomain
itso    IN   SOA   dns63.itso admin.itso (
             1               ; Serial (incremented when database is changed
             10800           ; Refresh (slave will check every 3 hours
             3600            ; Retry  (retry every hour after refresh failure
             604800          ; Expire (slave gives up retry after one week
             86400 )         ; Time to Live (data cached in other servers 1 day
$ORIGIN itso.com.
; define domain nameservers
                   IN      NS      dns63
                   IN      NS      dns64
;the following records specify the location of a specific service
_http._tcp              SRV  0  0 80   www.itso.com
                        SRV  10 0 8000 www2.itso.com
;the following records map the host names to their related ip addresses
localhost          IN   A    127.0.0.1
dns63              IN   A    9.12.6.68
dns64              IN   A    9.12.6.62
sc63               IN   A    9.12.6.68
sc64               IN   A    9.12.6.62
host1              IN   A    9.12.6.67
host2              IN   A    9.12.6.61
www2               IN   A    9.179.147.237
www                IN   A    9.179.147.237
gateway            IN   A    9.12.6.68
                   IN   A    9.12.6.67
;these records define an alias for hosts host1 and host2
mail               IN   CNAME host1
ftp                IN   CNAME host2
******************************** Bottom of Data **********************
```

SC63 BIND 4.9.3 reverse files - in-addr.arpa (itso.rev.v4)

```
********************************* Top of Data ***************************
;
;   /etc/octavio/dnsdata/itso.rev.v4
;
$ORIGIN 12.9.in-addr.arpa.

6  IN SOA dns63.itso.com. admin.itso.com. (
             1  10800  3600 604800 86400 )
              IN      NS    dns63.itso.com.
              IN      NS    dns64.itso.com.
$ORIGIN 6.12.9.in-addr.arpa.
68           IN      PTR    dns63.itso.com.
62           IN      PTR    dns64.itso.com.

67           IN      PTR    host1.itso.com.
61           IN      PTR    host2.itso.com.
68           IN      PTR    www2.itso.com.
62           IN      PTR    www.itso.com.
20           IN      PTR    printserver.itso.com.
******************************** Bottom of Data **********************
```

SC63 BIND 4.9.3 loopback file (itso.lbk.v4)

```
********************************* Top of Data ***************************
;   /etc/octavio/dnsdata/itso.lbk.v4
0.0.127.in-addr.arpa. IN SOA  dns63.itso.com. admin.itso.com (
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  dns63.itso.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.itso.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
******************************** Bottom of Data *************************
```

SC63 BIND 4.9.3 hints file (cache file) (itso.ca.v4)

The file replicated here is found on the Web at:

ftp://ftp.rs.internic.net/domain/named.root

```
********************************* Top of Data ***************************
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
; formerly NS.INTERNIC.NET
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
; formerly NS1.ISI.EDU
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
; formerly C.PSI.NET
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
; formerly TERP.UMD.EDU
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
; formerly NS.NASA.GOV
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
; formerly NS.ISC.ORG
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
; formerly NS.NIC.DDN.MIL
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
; formerly AOS.ARL.ARMY.MIL
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
; formerly NIC.NORDU.NET
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
; temporarily housed at NSI (InterNIC)
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
; housed in LINX, operated by RIPE NCC
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
; temporarily housed at ISI (IANA)
```

```
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
; housed in Japan, operated by WIDE
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
******************************** Bottom of Data *************************
```

A.1.1.2 SC64 BIND 4.9.3-based definitions (secondary name server)

SC63 BIND 4.9.3 boot file (named.boot)

```
********************************* Top of Data ***************************
;  /SC64/etc/named.boot
;       boot file for name server
;type    domain                  source file or host
directory   /etc/octavio/dnsdata
secondary   itso.com           9.12.6.68       itso.bk.for.v4
secondary   6.12.9.in-addr.arpa 9.12.6.68       itso.bk.rev.v4
primary     0.0.127.in-addr.arpa                itso.lbk.v4
cache       .                                   itso.ca.v4
options query-log
******************************** Bottom of Data *************************
```

SC64 BIND 4.9.3 loopback file (itso.lbk.v4)

```
********************************* Top of Data ***************************
;   /SC64/etc/octavio/dnsdata/itso.lbk.v4
0.0.127.in-addr.arpa. IN SOA  dns64.itso.com. admin.itso.com (
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  dns63.itso.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.itso.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
******************************** Bottom of Data *************************
```

SC64 BIND 4.9.3 hints file (cache file) (itso.ca.v4)

The file is not replicated here; it is the same file shown at “SC63 BIND 4.9.3 hints file (cache file) (itso.ca.v4)” on page 470 of this appendix.

A.1.2 BIND 4-based DNS/WLM scenario

Please refer to Figure A-2 for the configuration used in the following discussion.

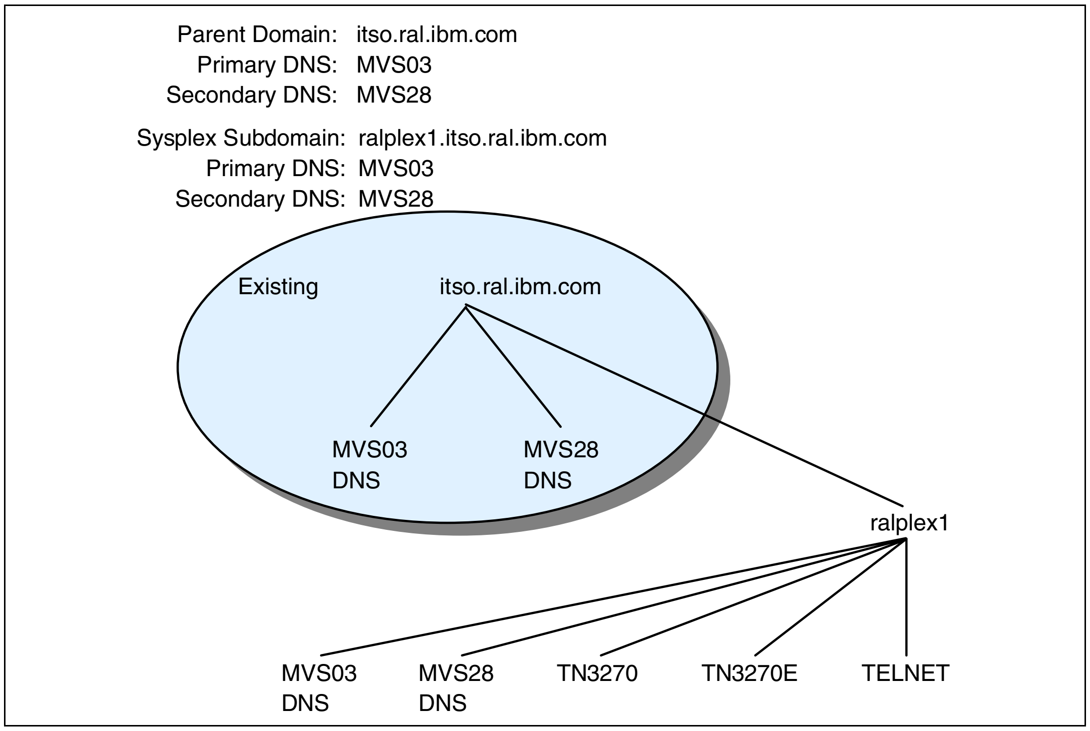


Figure A-2 MVS03 + MVS28: DNS server for parent and SYSPLEX domains

A.1.2.1 MVS03 definitions

MVS03 is the primary DNS server for both the parent domain itso.ral.ibm.com. and the sysplex domain ralplex1.itso.ral.ibm.com.

MVS03 boot file

```
 ***************************** Top of Data ******************************
;    /etc/named.boot.wlm2     based on    /etc/named.boot.wlm1
;
;  TYPE        DOMAIN                        FILE OR HOST
directory      /etc/dnsdata
;
; (Following for sysplex name server as both parent + sysplex DNS)
primary      ralplex1.itso.ral.ibm.com       named.for.wlm1  cluster
primary      itso.ral.ibm.com                named.for2
primary      251.168.192.in-addr.arpa        named.wlm2192.168.251
primary      252.168.192.in-addr.arpa        named.wlm2192.168.252
primary      236.168.192.in-addr.arpa        named.wlm2192.168.236
primary      235.168.192.in-addr.arpa        named.wlm2192.168.235
primary      105.24.9.in-addr.arpa           named.wlm29.24.105
primary      104.24.9.in-addr.arpa           named.wlm29.24.104
primary      221.168.192.in-addr.arpa        named.wlm2192.168.221
primary      109.168.192.in-addr.arpa        named.wlm2192.168.109
primary      0.0.127.in-addr.arpa            named.lbk.wlm2
cache         .                              named.ca
forwarders   9.24.104.108
options query-log
**************************** Bottom of Data ****************************
```

MVS03 domain file named.for.wlm1 (ralplex1.itso.ral.ibm.com.)

```
 ***************************** Top of Data ******************************
 ;
 ;  /etc/dnsdata/named.for.wlm1 for MVS03 and SYSPLEX
 ;
 $ORIGIN itso.ral.ibm.com.
 ralplex1 IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. gdente@ralplex1.i
                       5 10800   3600   604800   86400 )
 $ORIGIN ralplex1.itso.ral.ibm.com.
                  IN     NS    mvs03
 ;
                  IN     NS    mvs28
 ;
 ; OWNER    CLASS   TYPE    RECORD DATA
 localhost    IN      A       127.0.0.1
 mvs03        IN      A       9.24.105.126      ;  Ethernet LCS
              IN      A       192.168.251.1     ;  Vipa address
              IN      A       192.168.252.1     ;  SAMEHOST to MVS03C
              IN      A       192.168.236.1     ;  RAS XCF to 28
              IN      A       192.168.235.3     ;  MPC to 25
              IN      A       192.168.221.20    ;  LICP03 Connection
              IN      A       192.168.109.3     ;  LICCP25 Connection
 mvs03c       IN      A       192.168.252.2     ;  SAMEHOST at MVS03C
 mvs28        IN      A       9.24.105.75       ;  Ethernet LCS at MVS28
              IN      A       192.168.236.2     ;  RAS XCF to 03
              IN      A       192.168.221.24    ;  LICP03 Connection
              IN      A       192.168.109.1     ;  LICCP25 Connection
**************************** Bottom of Data ****************************
```

MVS03 domain file named.for2 (itso.ral.ibm.com.)

```
 ***************************** Top of Data ******************************
;  /etc/dnsdata/named.for2 for MVS03
;
$ORIGIN ral.ibm.com.
itso IN     SOA   mvs03.itso.ral.ibm.com. gdente@itso.ral.ibm.com. (
                      5 10800   3600   604800   86400 )
$ORIGIN itso.ral.ibm.com.
     IN     NS    mvs03
;
     IN     NS    mvs28
;
; OWNER    CLASS   TYPE    RECORD DATA
localhost    IN      A       127.0.0.1
mvs03        IN      A       9.24.105.126      ;  Ethernet LCS
             IN      A       192.168.251.1     ;  Vipa address
             IN      A       192.168.252.1     ;  SAMEHOST to MVS03C
             IN      A       192.168.236.1     ;  RAS XCF to 28
             IN      A       192.168.235.3     ;  MPC to 25
             IN      A       192.168.221.20    ;  LICP03 Connection
             IN      A       192.168.109.3     ;  LICCP25 Connection
mvs03c       IN      A       192.168.252.2     ;  SAMEHOST at MVS03C
;
mvs28        IN      A       9.24.105.75       ;  Ethernet LCS at MVS28
             IN      A       192.168.236.2     ;  RAS XCF to 03
             IN      A       192.168.221.24    ;  LICP03 Connection
             IN      A       192.168.109.1     ;  LICCP25 Connection
;
rsserver     IN      A       9.24.104.108      ;  DNS (6611) at ITSO
gwen         IN      A       9.24.104.35
henk         IN      A       9.24.104.201
```

```
kakky        IN      A       9.24.104.47
silvia       IN      A       9.24.104.213
;
ourmvs       IN      CNAME   mvs03
othermvs     IN      CNAME   mvs28
dns2         IN      CNAME   rsserver
wtr05199     IN      CNAME   gwen
wtr05101     IN      CNAME   henk
wtr05118     IN      CNAME   kakky
wtr05119     IN      CNAME   silvia
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.109

```
 ***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.109 for MVS03-iccp 192.168.109.3
;
;$ORIGIN 109.168.192.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                      5 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
     IN     NS    rsserver.itso.ral.ibm.com.
;
;   MVS03 ICCP  entry
3    IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
3    IN     PTR   mvs03.itso.ral.ibm.com.
;   MVS28 ICCP  entry
1    IN     PTR   mvs28.ralplex1.itso.ral.ibm.com.
1    IN     PTR   mvs28.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.221

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.221 for MVS03-icp 192.168.221.20
;
;$ORIGIN 221.168.192.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;
;   MVS03 ICP  entry
20   IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
20   IN     PTR   mvs03.itso.ral.ibm.com.
;   MVS28 ICP  entry
24   IN     PTR   mvs28.ralplex1.itso.ral.ibm.com.
24   IN     PTR   mvs28.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.235

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.235  MVS03-MPC 192.168.235.3 (> MVS25)
;
;$ORIGIN 235.168.192.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;
```

```
;   MVS03 MPC  entry to 25
3    IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
3    IN     PTR   mvs03.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.236

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.236 MVS03-XCF to MVS28 192.168.236.1
;
;$ORIGIN 236.168.192.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;
;   MVS03 XCF  entry to 28
1    IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
1    IN     PTR   mvs03.itso.ral.ibm.com.
;   MVS28 XCF  entry to 03
2    IN     PTR   mvs28.ralplex1.itso.ral.ibm.com.
2    IN     PTR   mvs28.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.251

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.251 - mvs-3-vipa for MVS03
;
;ORIGIN 168.192.in-addr.arpa.
;   MVS03 VIPA entry
;251  IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.
;$ORIGIN 251.168.192.in-addr.arpa.
;tso.ral.ibm.com. IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ra
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
     ;
1    IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
1    IN     PTR   mvs03.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm2192.168.252

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm2192.168.252 for MVS03 Samehost to MVS03C
;
;ORIGIN 252.168.192.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;
1    IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
1    IN     PTR   mvs03.itso.ral.ibm.com.
2    IN     PTR   mvs03c.ralplex1.itso.ral.ibm.com.
2    IN     PTR   mvs03c.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm29.24.104

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm29.24.104 for team workstations
;
$ORIGIN 24.9.in-addr.arpa.
104  IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
$ORIGIN 104.24.9.in-addr.arpa.
;   Other DNS at ITSO
108  IN     PTR   rsserver.itso.ral.ibm.com.
;   Gwen's workstation
35   IN     PTR   wtr05199.itso.ral.ibm.com.
;   Henk's workstation
201  IN     PTR   wtr05101.itso.ral.ibm.com.
;   Kakky's workstation
47   IN     PTR   wtr05118.itso.ral.ibm.com.
;   Silvia's workstation
213  IN     PTR   wtr05119.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 reverse file (in-addr.arpa file) named.wlm29.24.105

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.wlm29.24.105 for MVS03-en1 9.24.105.126
;
;$ORIGIN 105.24.9.in-addr.arpa.
@    IN     SOA   mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralplex1.itso.r
                     2 10800   3600   604800   86400 )
     IN     NS    mvs03.ralplex1.itso.ral.ibm.com.
     IN     NS    mvs28.ralplex1.itso.ral.ibm.com.
;   Address of other dns
209  IN     PTR   rsserver.itso.ral.ibm.com.
;     LCS entry at MVS03
126  IN     PTR   mvs03.ralplex1.itso.ral.ibm.com.
126  IN     PTR   mvs03.itso.ral.ibm.com.
;     LCS entry at MVS28
75   IN     PTR   mvs28.ralplex1.itso.ral.ibm.com.
75   IN     PTR   mvs28.itso.ral.ibm.com.
**************************** Bottom of Data ****************************
```

MVS03 loopback file

```
***************************** Top of Data ******************************
;  /etc/dnsdata/named.lbk.wlm2
;
0.0.127.in-addr.arpa. IN SOA mvs03.ralplex1.itso.ral.ibm.com. mvs03.ralp
                                    5
                                    10800
                                    3600
                                    604800
                                    86400     )
0.0.127.in-addr.arpa.   IN    NS    mvs03.ralplex1.itso.ral.ibm.com.
0.0.127.in-addr.arpa.   IN    NS    mvs03.itso.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN    PTR   localhost.
**************************** Bottom of Data ****************************
```

A.1.2.2 MVS28 definitions (SYSPLEX and parent DNS server)

MVS28 boot file (SYSPLEX and parent DNS server)

```
***************************** Top of Data ******************************
;    /etc/named.boot.wlm2     based on    /etc/named.boot
;
;  TYPE        DOMAIN                       FILE OR HOST
directory      /etc/dnsdata
;
; (Following for sysplex name server as both parent + sysplex DNS)
secondary    ralplex1.itso.ral.ibm.com  192.168.236.1   named.for.wlm1  cluster
secondary    itso.ral.ibm.com           192.168.236.1   named.for2
secondary    251.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.251
secondary    252.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.252
secondary    236.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.236
secondary    235.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.235
secondary    105.24.9.in-addr.arpa      192.168.236.1   named.wlm29.24.105
secondary    104.24.9.in-addr.arpa      192.168.236.1   named.wlm29.24.104
secondary    221.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.221
secondary    109.168.192.in-addr.arpa   192.168.236.1   named.wlm2192.168.109
primary      0.0.127.in-addr.arpa       named.lbk.wlm2
cache         .                         named.ca
forwarders   9.24.104.108
options query-log
**************************** Bottom of Data ****************************
```

MVS28 loopback file

```
 ***************************** Top of Data ******************************
;  /etc/dnsdata/named.lbk.wlm2
;
0.0.127.in-addr.arpa. IN SOA mvs28.ralplex1.itso.ral.ibm.com. mvs28.ralp
                                   5
                                   10800
                                   3600
                                   604800
                                   86400     )
0.0.127.in-addr.arpa.   IN    NS    mvs28.ralplex1.itso.ral.ibm.com.
0.0.127.in-addr.arpa.   IN    NS    mvs28.itso.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN    PTR   localhost.
**************************** Bottom of Data ****************************
```

MVS28 hints file (cache file)

The file is not replicated here; it is the same file shown in “SC63 BIND 4.9.3 hints file (cache file) (itso.ca.v4)” on page 470 of this appendix.

A.2 DHCP + DDNS on MVS03

This section shows the configuration files we used in our DDNS scenario.

A.2.1 MVS03 DHCP configuration file

The sample file on which this is based can be found in Appendix C, “Sample DHCP configuration file” on page 503.

```
******************************** Top of Data **************************
#
#   dhcpsd.cfg -- DHCP Server Configuration File
#
#   SMP/E Distribution name:  EZATDDSD
#
numLogFiles     4
logFileSize     400
logFileName     /tmp/dhcpsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE

#
leaseTimeDefault        7 minute
leaseExpireInterval     20 seconds
# supportBOOTP            yes
supportBOOTP            no
supportUnlistedClients  yes

vendor ibm
{
        option 42 hex"ab dc"
}

vendor sun hex"ef 12 34 56 78"

subnet 192.168.100.0 255.255.255.0  192.168.100.101-192.168.100.120
{
option 1    255.255.255.0
option 3    192.168.100.100
option 4    192.168.100.100
option 5    192.168.100.100
option 6    192.168.100.100
option 15   small.isp.com                # domain name
option 51   10800
}
#
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;3110400;q" -q"
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q" -q"
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q" -q"

class fruit
{
        option 48 6.5.4.3
        option 48 8.8.8.8
}
class veggie
{
        option 49 1.2.3.4
        option 48 6.6.6.6
}
```

```
#
# end of dhcpsd.cfg
#
```

A.2.2 MVS03 DDNS definitions

Please examine the DDNS Boot File in A.2.2.1, “MVS03 DDNS boot file” on page 479. You will notice that the majority of the domain (forward) files, reverse files, hints file, and loopback file have already been made available in the section on Scenario 1 (A.1, “BIND 4.9.3-based DNS implementation” on page 468). Therefore, we show here only the files that are different.

A.2.2.1 MVS03 DDNS boot file

```
;
;    /etc/named.boot.dyn (modeled after named.boot)
;     requires new boot file, new for.dyn file, and new rev.dyn file
;     optional=slave
;  TYPE        DOMAIN                    FILE OR HOST
directory      /etc/dnsdata
;
primary      itso.ral.ibm.com            named.for
primary      small.isp.com               named.for.dyn  dynamic secured
primary      100.168.192.in-addr.arpa    named.rev192.168.100.dyn dynamic secured
primary      251.168.192.in-addr.arpa    named.rev192.168.251
primary      252.168.192.in-addr.arpa    named.rev192.168.252
primary      236.168.192.in-addr.arpa    named.rev192.168.236
primary      235.168.192.in-addr.arpa    named.rev192.168.235
primary      105.24.9.in-addr.arpa       named.rev9.24.105
primary      104.24.9.in-addr.arpa       named.rev9.24.104
primary      221.168.192.in-addr.arpa    named.rev192.168.221
primary      109.168.192.in-addr.arpa    named.rev192.168.109
primary      0.0.127.in-addr.arpa        named.lbk
cache         .                          named.ca
forwarders   9.24.104.108
slave
options query-log
```

A.2.2.2 MVS03 DDNS forward domain file

```
;
;  /etc/dnsdata/named.for.dyn for MVS03 as DHCP and DDNS
;
$ORIGIN small.isp.com.
@   .IN KEY 80 0 1 AQPYZeXmV/uIJXttTwIlvLcvtDfH5RNE+7GeYix01+JRWqsFluxiSU
@   .IN SOA        mvs03.itso.ral.ibm.com. gdente@itso.ral.ibm.com. (
                      6 10800   3600   604800   86400 )
     IN     NS    mvs03.itso.ral.ibm.com.
;
; OWNER    CLASS   TYPE    RECORD DATA
localhost    IN      A       127.0.0.1
murli750     IN      A       192.168.100.254   ;  DNS on small.is

ns-updates.small.isp.com.   IN  A       192.168.100.100   ;
```

A.2.2.3 MVS03 DDNS reverse file

```
;
;   /etc/dnsdata/named.rev192.168.100.dyn
;
;$ORIGIN 100.168.192.in-addr.arpa.
@ IN   KEY 80 0 1 AQO/dB7EGk1IPpp19eSSbnazFpvtcxkFoWor7JpIMM/om6CrZrsMUO
@    IN     SOA   mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
                      6 10800   3600   604800   86400 )
     IN     NS    mvs03.itso.ral.ibm.com.
;
100  IN     PTR   ns-updates.small.isp.com.
254  IN     PTR   murli750.small.isp.com.
```

A.3 BIND 9-based DNS implementation

The following section includes the implementation configuration for a BIND 9-based environment.

A.3.1 BIND 9 basic scenario

This section describes a basic BIND 9 scenario. Please refer to Figure A-3 for a description.

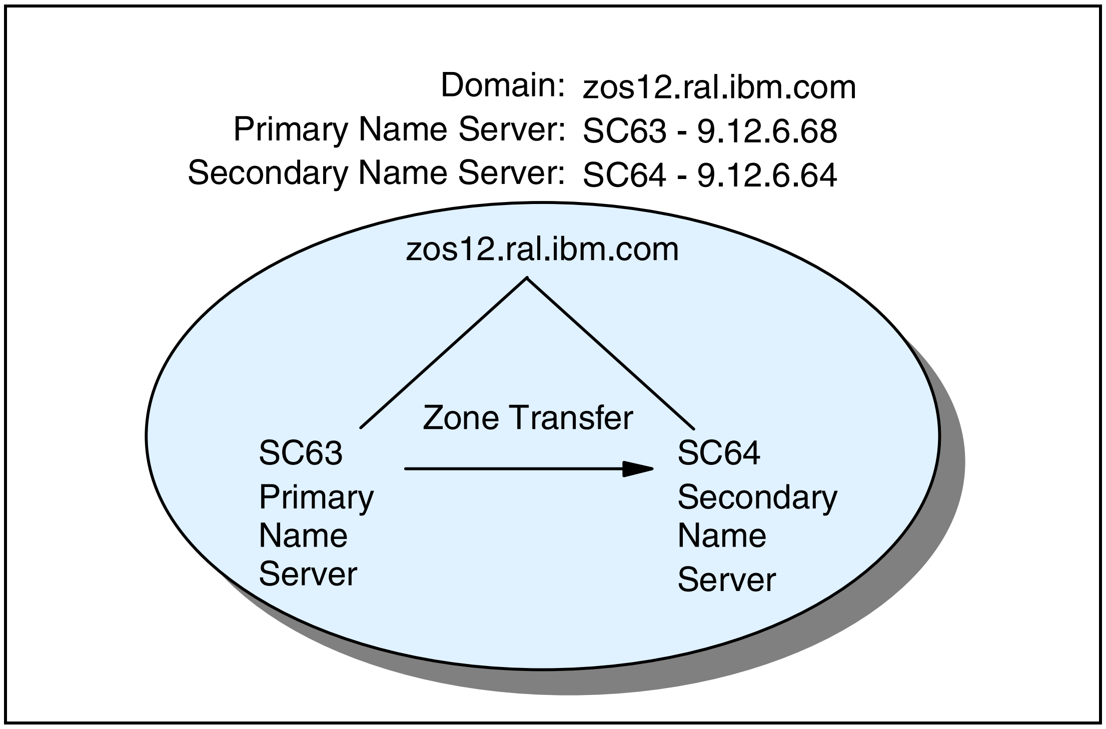

Domain: zos12.ral.ibm.com
Primary Name Server: SC63 - 9.12.6.68
Secondary Name Server: SC64 - 9.12.6.64
zos12.ral.ibm.com
Zone Transfer
SC63 Primary Name Server
SC64 Secondary Name Server

Figure A-3 SC63, SC64 basic name server implementation

A.3.1.1 SC63 definitions (primary name server)

SC63 bind 9 conf file (named.conf)

```
******************************** Top of Data ***********************
options {
   pid-file "/etc/named.pid" ;
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.67;9.12.6.68; };
};
logging {
   channel "default_debug" {
   file "/etc/octavio/named.run" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel main_log {
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel security_log {
   file "/etc/octavio/named_security.log" versions 2 size 1M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel query_log {
   file "/etc/octavio/named_query.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                   };
   channel transfer_log {
   file "/etc/octavio/named_transfer.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                  };
   category client   { main_log; };
   category config   { main_log; };
   category "database" { main_log; };
   category dispatch { main_log; };
   category dnssec { security_log; main_log; };
   category general  { main_log; };
   category network  { main_log; };
   category "notify" { main_log; };
   category resolver { main_log; };
   category security { security_log; main_log; };
   category update   { main_log; };
   category queries  { query_log;};
   category lame-servers { query_log; main_log; };
   category xfer-in  { "transfer_log"; };
   category xfer-out { "transfer_log"; };
   category default  { main_log; };
   };
zone "zos12.ral.ibm.com" in {
   type master;
   file "zos12.for.v9";
};
zone "6.12.9.in-addr.arpa" in {
   type master;
   file "zos12.rev.v9";
};
zone "0.0.127.in-addr.arpa" in {
   type master;
   file "zos12.lbk.v9";
};

zone "." in {
   type hint;
   file "zos12.ca.v9";
};
******************************** Bottom of Data **********************
```

SC63 bind 9 forward domain file (zos12.for.v9)

```
******************************** Top of Data ***************************
;  /etc/octavio/dnsdata/zos12.for.v9
;        name server zone data
;
; Default TTL value
$TTL 86400
$ORIGIN ral.ibm.com.
zos12             IN      SOA   dns63.zos12 admin.zos12 (
             1               ; Serial (incremented when database is changed
             10800           ; Refresh (slave will check every 3 hours
             3600            ; Retry  (retry every hour after refresh failure
             604800          ; Expire (slave gives up retry after 1 week
             86400 )         ; Neg. Cache (cache NXDOMAIN/RRSET responses 1 day
;
$ORIGIN zos12.ral.ibm.com.
; define domain nameservers
                   IN      NS      dns63
                   IN      NS      dns64
; define the host locations of specific services
_http._tcp              SRV  0  0 80   www.zos12.ral.ibm.com.
; define hosts-to-address
localhost          IN   A    127.0.0.1
dns63              IN   A    9.12.6.68
dns64              IN   A    9.12.6.64
host1              IN   A    9.12.6.67
host2              IN   A    9.12.6.63
www                IN   A    9.179.147.237
;define multiple address to a given host
sc63               IN   A    9.12.6.67
                   IN   A    9.12.6.68

;IPv6 addresses samples
;www                IN   AAAA 3ffe:8050:201:1860:42::1
;www                IN   A6   0 3ffe:8050:201:1860:42::1
;alias definitions
mail               IN   CNAME host1
ftp                IN   CNAME host2
******************************** Bottom of Data **********************
```

SC63 bind 9 reverse files - in-addr.arpa (zos12.rev.v9)

```
********************************* Top of Data ***************************
;   /etc/octavio/dnsdata/zos12.rev.v9
$TTL 86400
$ORIGIN 12.9.in-addr.arpa.
6  IN SOA dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
              1  10800  3600 604800 86400 )
              IN      NS    dns63.zos12.ral.ibm.com.
              IN      NS    dns64.zos12.ral.ibm.com.
$ORIGIN 6.12.9.in-addr.arpa.
68            IN      PTR    dns63.zos12.ral.ibm.com.
62            IN      PTR    dns64.zos12.ral.ibm.com.
$GENERATE 3-6 $ PTR host$.zos12.ral.ibm.com.
; The following records are generated by the above $GENERATE directive.
;1            IN      PTR    host1.zos12.ral.ibm.com.
;2            IN      PTR    host2.zos12.ral.ibm.com.
;3            IN      PTR    host3.zos12.ral.ibm.com.
;4            IN      PTR    host4.zos12.ral.ibm.com.
67            IN      PTR    host1.zos12.ral.ibm.com.
61            IN      PTR    host2.zos12.ral.ibm.com.
******************************** Bottom of Data **********************
```

SC63 bind 9 loopback file (zos12.lbk.v9)

```
********************************* Top of Data ***************************
;   /etc/octavio/dnsdata/zos12.lbk.v9
$TTL 86400
0.0.127.in-addr.arpa. IN SOA  dns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com (
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  dns63.zos12.ral.ibm.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.zos12.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
******************************** Bottom of Data *************************
```

SC63 BIND 9 hints file (zos12.ca.v9)

The file replicated here is found on the Web at:

```
ftp://ftp.rs.internic.net/domain/named.root
********************************* Top of Data ***************************
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
; formerly NS.INTERNIC.NET
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
; formerly NS1.ISI.EDU
```

```
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
; formerly C.PSI.NET
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
; formerly TERP.UMD.EDU
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
; formerly NS.NASA.GOV
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
; formerly NS.ISC.ORG
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
; formerly NS.NIC.DDN.MIL
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
; formerly AOS.ARL.ARMY.MIL
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
; formerly NIC.NORDU.NET
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
; temporarily housed at NSI (InterNIC)
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
; housed in LINX, operated by RIPE NCC
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
; temporarily housed at ISI (IANA)
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
; housed in Japan, operated by WIDE
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
******************************** Bottom of Data *************************
```

A.3.1.2 SC64 definitions (secondary name server)

SC63 BIND 9 conf file (named.conf)

```
********************************* Top of Data ***************************
options {
   pid-file "/etc/named.pid" ;
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.63;9.12.6.64; };
};
logging {
   channel "default_debug" {
   file "/etc/octavio/named.run.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel main_log {
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
```

```
                 };
   channel security_log {
   file "/etc/octavio/named_security.log" versions 2 size 1M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                 };
   channel query_log {
   file "/etc/octavio/named_query.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                };
   channel transfer_log {
   file "/etc/octavio/named_transfer.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
               };
   category client   { main_log; };
   category config   { main_log; };
   category "database" { main_log; };
   category dispatch { main_log; };
   category dnssec { security_log; main_log; };
   category general  { main_log; };
   category network  { main_log; };
   category "notify" { main_log; };
   category resolver { main_log; };
   category security { security_log; main_log; };
   category update   { main_log; };
   category queries  { query_log;};
   category lame-servers { query_log; main_log; };
   category xfer-in  { "transfer_log"; };
   category xfer-out { "transfer_log"; };
   category default  { main_log; };
   };
zone "zos12.ral.ibm.com" in {
   type slave;
   file "zos12.for.bak.v9";
   masters { 9.12.6.68; };
};
zone "6.12.9.in-addr.arpa" in {
   type slave;
   file "zos12.rev.bak.v9";
   masters { 9.12.6.68; };
};
zone "0.0.127.in-addr.arpa" in {
   type master;
   file "zos12.lbk.v9";
};

zone "." in {
   type hint;
   file "zos12.ca.v9";
};
******************************** Bottom of Data *************************
```

SC64 BIND 9 loopback file (zos12.lbk.v9)

```
********************************* Top of Data ***************************
;   /etc/octavio/dnsdata/zos12.ral.ibm.lbk.v9
$TTL 86400
0.0.127.in-addr.arpa. IN SOA  dns64.zos12.ral.ibm.com. admin.zos12.ral.ibm.com (
     1
     10800
     3600
     604800
     86400   )
0.0.127.in-addr.arpa.   IN   NS  dns63.zos12.ral.ibm.com.
0.0.127.in-addr.arpa.   IN   NS  dns64.zos12.ral.ibm.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
******************************** Bottom of Data *************************
```

SC64 BIND 9 hints file (zos12.ca.v9)

The file is not replicated here; it is the same file shown in “SC63 BIND 9 hints file (zos12.ca.v9)” on page 483 of this appendix.

A.3.2 Transaction Signature (TSiG) - key and configuration files

The following files were created using the BIND 9 tool **dnssec-keygen**, which generates the key to be inserted in the config files on both the primary and secondary Domain Name Servers.

A.3.2.1 Ksc63-sc64.+157+35544.key file

```
***************************** Top of Data *****************************
sc63-sc64. IN KEY 512 3 157 8kOIjL3m4sSO13DBNs6S6w==
**************************** Bottom of Data ***************************
```

A.3.2.2 Ksc63-sc64.+157+35544.private file

```
***************************** Top of Data *****************************
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: 8kOIjL3m4sSO13DBNs6S6w==
**************************** Bottom of Data ***************************
```

A.3.2.3 SC63 BIND 9-based named.conf with Transaction Signature

```
***************************** Top of Data *****************************
controls {
     inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};
key rndc_key {
algorithm "hmac-md5";
secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnVOIG1hZGUgZm9yIGEgd29tYW4K";
};
key sc63-sc64 {
algorithm "hmac-md5";
secret "8kOIjL3m4sSO13DBNs6S6w==";
};
options {
   pid-file "/etc/named.v9.pid";
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.68; };
   forwarders { 9.12.6.67; };
};
logging {
   channel "default_debug" {
   file "/etc/octavio/named.run" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel main_log {
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel security_log {
   file "/etc/octavio/named_security.log" versions 2 size 1M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel query_log {
   file "/etc/octavio/named_query.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                   };
   channel transfer_log {
   file "/etc/octavio/named_transfer.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                  };
   category client   { main_log; };
   category config   { main_log; };
   category "database" { main_log; };
   category dispatch { main_log; };
   category dnssec { security_log; main_log; };
   category general  { main_log; };
   category network  { main_log; };
   category "notify" { main_log; };
   category resolver { main_log; };
   category security { security_log; main_log; };
   category update   { main_log; };
   category queries  { query_log;};
   category lame-servers { query_log; main_log; };
   category xfer-in  { "transfer_log"; };
   category xfer-out { "transfer_log"; };
   category default  { main_log; };
   };
zone "zos12.ral.ibm.com" in {
   type master;
   file "zos12.for.v9";
   allow-transfer { key sc63-sc64; };
};
zone "6.12.9.in-addr.arpa" in {
   type master;
   file "zos12.rev.v9";
   allow-transfer { key sc63-sc64; };
};
zone "0.0.127.in-addr.arpa" in {
   type master;
   file "zos12.lbk.v9";
   allow-transfer { key sc63-sc64; };
};
zone "." in {
   type hint;
   file "zos12.ca.v9";
};
**************************** Bottom of Data ***************************
```

A.3.2.4 SC64 BIND 9-based named.conf with Transaction Signature

```
**************************** Top of Data ***************************
controls {
     inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};
key rndc_key {
algorithm "hmac-md5";
secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};
key sc63-sc64 {
algorithm "hmac-md5";
secret "8kOIjL3m4sSO13DBNs6S6w==";
};
server 9.12.6.68 {
keys { sc63-sc64; };
};
options {
   pid-file "/etc/named.pid" ;
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.63; };
};
logging {
   channel "default_debug" {
   file "/etc/octavio/named.run.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                 };
   channel main_log {
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                 };
   channel security_log {
   file "/etc/octavio/named_security.log" versions 2 size 1M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                 };
   channel query_log {
   file "/etc/octavio/named_query.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                   };
   channel transfer_log {
   file "/etc/octavio/named_transfer.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                  };
   category client   { main_log; };
   category config   { main_log; };
   category "database" { main_log; };
   category dispatch { main_log; };
   category dnssec { security_log; main_log; };
   category general  { main_log; };
   category network  { main_log; };
   category "notify" { main_log; };
   category resolver { main_log; };
   category security { security_log; main_log; };
   category update   { main_log; };
   category queries  { query_log;};
   category lame-servers { query_log; main_log; };
   category xfer-in  { "transfer_log"; };
   category xfer-out { "transfer_log"; };
   category default  { main_log; };
   };
zone "zos12.ral.ibm.com" in {
   type slave;
   file "zos12.for.bak.v9";
   masters { 9.12.6.68; };
   allow-update-forwarding { any; };
};
zone "6.12.9.in-addr.arpa" in {
   type slave;
   file "zos12.rev.bak.v9";
   masters { 9.12.6.68; };
   allow-update-forwarding { any; };
};
zone "0.0.127.in-addr.arpa" in {
   type master;
   file "zos12.lbk.v9";
};
zone "." in {
   type hint;
   file "zos12.ca.v9";
};
**************************** Bottom of Data ****************************
```

A.3.3 BIND 9-based DNSSEC - primary DNS related files

Here we show some the files we generated in our scenarios.

A.3.3.1 SC63 zone generated key files (dnssec-keygen tool)

Kzos12.ral.ibm.com.+003+09520.key

```
**************************** top of Data ******************************
zos12.ral.ibm.com. IN KEY 256 3 3
AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbEERPh9rWyMlQbHuUBXTHd
U/wsIvBr7omnRUXvLQxXEF4mLUnILL7jG5yncdRBo1KR7XTDkzGI9CW4
qU4UGpH98QnbnZupGXDwnrHs8pK7stgssfjniShyUAbzVVnSxCtCHC7E
QbnvIKdfjI5pydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ/+lhTjUK
TZGYFDc57i6yMPNErsP8DI3GwQ41ONaP2OtCGMTJYtIOhb9oNkUOtS9f gtIP
**************************** Bottom of Data ***************************
```

Kzos12.ral.ibm.com.+003+09520.private

```
**************************** top of Data ******************************
Private-key-format: v1.2
Algorithm: 3 (DSA)
Prime(p):
zwmtkRbEERPh9rWyMlQbHuUBXTHdU/wsIvBr7omnRUXvLQxXEF4mLUnILL7jG5yncdRBo1KR7XTDkzGI9CW4qQ==
Subprime(q): gts7nIgYVkCwYJ6HrsM1WHtxjKO=
Base(g):
ThQakf3xCdudm6kZcPCesezykruy2Cyx+OeJKHJQBvNVWdLEKOIcLsRBue8gp1+MjmnJ1goV6ihcFO7aQY8qZA==
Private_value(x): RjJE/yvLNeJYSpcwV/DzdI9RnVA=
Public_value(y):
Ip2RVXUTiqrLQ+YxWf/pYU41Ck2RmBQ3Oe4usjDzRK7D/AyNxsEONdDWj9tLQhjEyWLSNIW/aDZFDrUvX4LSDw==
**************************** Bottom of Data ***************************
```

A.3.3.2 Parent zone generated key files (dnssec-keygen tool)

Kral.ibm.com.+003+64185.key

```
**************************** top of Data ******************************
ral.ibm.com. IN KEY 256 3 3 AJzuoUPYucqE+XdnZ/hZ3fhXRYYf+2cxYtB73IDDqMRkvDiXdhvkztt8
hHVLymIuuNpGhNVQSBR+/pffL4PCAeUls2BE+ovCr8MhC8t/F7gDGBhD
8afYXVx8dyGndSCNNaOpA5G3PGJhDoG7EdKjWiu9tFcW8g15zsOe8uP/
17UpYjMlPsG7lbOyzmIFabNouXhO5u9bJKrXrxeml3p2JAvoyX+tO+EX
IbunH7Itrh7ADLEbpav5doYBzJ4zjD/ojTQK4og7tNaIeT6qwms9EQ2B IWzt
**************************** Bottom of Data ***************************
```

Kral.ibm.com.+003+64185.private

```
**************************** top of Data ******************************
Private-key-format: v1.2
Algorithm: 3 (DSA)
Prime(p):
+2cxYtB73IDDqMRkvDiXdhvkztt8hHVLymIuuNpGhNVQSBR+/pffL4PCAeUls2BE+ovCr8MhC8t/F7gDGBhD8Q==
Subprime(q): nO6hQ9i5yoT5d2dn+Fnd+FdFhh8=
Base(g):
p9hdXHx3Iad1IIO1rSkDkbc8YmEOgbsROqNaK72OVxbyDXnOw57y4//XtSliMyU+wbuVs7LOYgVps2i5eE7m7w==
Private_value(x): Jjg2jsnZoSNYNcvOSD15dOGYwFo=
Public_value(y):
WySq168Xppd6diQL6Ml/rTvhFyG7px+yLa4ewAyxG6Wr+XaGAcyeM4w/6IOOCuKIO7TWiHk+qsJrPRENgSFs7Q==
**************************** Bottom of Data ***************************
```

A.3.3.3 SC63 zone keyset file (dnssec-makekeyset tool)

keyset-zos12.ral.ibm.com

```
**************************** top of Data ******************************
$ORIGIN .
$TTL 172800  ; 2 days
zos12.ral.ibm.com IN   KEY  256 3 3 (
                       AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbEERPh9rWy
                       MlQbHuUBXTHdU/wsIvBr7omnRUXvLQxXEF4mLUnILL7j
                       G5yncdRBo1KR7XTDkzGI9CW4qU4UGpH98QnbnZupGXDw
                       nrHs8pK7stgssfjniShyUAbzVVnSxCtCHC7EQbnvIKdf
                       jI5pydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ/+lh
                       TjUKTZGYFDc57i6yMPNErsP8DI3GwQ41ONaP2OtCGMTJ
                       YtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
                  SIG  KEY 3 4 172800 20020626173942 (
                       20020527173942 9520 zos12.ral.ibm.com.
                       AHUWn73ryAc3TInk8OlCso/1Mb3QFO2026Xop+8nibcK
                       WrtY24+DJsE= )
**************************** Bottom of Data ***************************
```

A.3.3.4 parent zone signed key file (dnssec-signkey tool)

signedkey-zos12.ral.ibm.com

```
**************************** top of Data ******************************
$ORIGIN .
$TTL 172800 ; 2 days
zos12.ral.ibm.com IN    KEY  256 3 3 (
                        AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbEERPh9rWy
                        MlQbHuUBXTHdU/wsIvBr7omnRUXvLQxXEF4mLUnILL7j
                        G5yncdRBo1KR7XTDkzGI9CW4qU4UGpH98QnbnZupGXDw
                        nrHs8pK7stgssfjniShyUAbzVVnSxCtCHC7EQbnvIKdf
                        jI5pydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ/+lh
                        TjUKTZGYFDc57i6yMPNErsP8DI3GwQ41ONaP2OtCGMTJ
                        YtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
                  SIG   KEY 3 4 172800 20020626173942 (
                        20020527173942 64185 ral.ibm.com.
                        AEvxWeXonAFwpaQp9nYhI5GDqafcHSTblwL1oAh+841m
                        4FZzImMpJF4= )
**************************** Bottom of Data ***************************
```

A.3.3.5 SC63 signed forward zone file

zos12.for.v9.signed:

```
**************************** top of Data ******************************
; File written on Mon May 27 14:26:46 2002
; dnssec_signzone version 9.1.1
zos12.ral.ibm.com.86400IN SOAdns63.zos12.ral.ibm.com. admin.zos12.ral.ibm.com. (
                4          ; serial
                10800      ; refresh (3 hours)
                3600       ; retry (1 hour)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
          86400SIGSOA 3 4 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
                AF2UFK1AP5eTSlpvgayWUN84rvREMzM3paa3
                /KJwdJ/EZOvnAuIAuc4= )
          86400NSdns63.zos12.ral.ibm.com.
          86400NSdns64.zos12.ral.ibm.com.
          86400SIGNS 3 4 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
```

```
               AH1qWnUz5L9xVKBEOc6twGTXyp6BKLPtNRJo
               PEhuc2GXBvgHhYwssj4= )
         86400KEY256 3 3 (
               AILbO5yIGFZAsGCeh67DNVh7cYytzwmtkRbE
               ERPh9rWyMlQbHuUBXTHdU/wsIvBr7omnRUXv
               LQxXEF4mLUnILL7jG5yncdRBo1KR7XTDkzGI
               9CW4qU4UGpH98QnbnZupGXDwnrHs8pK7stgs
               sfjniShyUAbzVVnSxCtCHC7EQbnvIKdfjI5p
               ydYKFeooXBdO2kGPKmQinZFVdROKqstD5jFZ
               /+lhTjUKTZGYFDc57i6yMPNErsP8DI3GwQ41
               ONaP2OtCGMTJYtIOhb9oNkUOtS9fgtIP ) ; key id = 10547
         172800SIGKEY 3 4 172800 20020626173942 (
               20020527173942 64185 ral.ibm.com.
               AEvxWeXonAFwpaQp9nYhI5GDqafcHSTblwL1
               oAh+841m4FZzImMpJF4= )
         86400NXT_http._tcp.zos12.ral.ibm.com. NS SOA SIG KEY NXT
         86400SIGNXT 3 4 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AGABIFS/2SuBeeTgRy9EgXR+qeM7TXIkjS+q
               sOPBKJg4jsPIVXb5Y6c= )
_http._tcp.zos12.ral.ibm.com. 86400 IN NXT dns63.zos12.ral.ibm.com. SIG NXT SRV
         86400SIGNXT 3 6 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AF/TB3xPnlGo3v7LimsystaDM6UcJfIsL/EC
               x54bn6AixO2Rj/mOH4w= )
         86400SRV0 0 80 www.zos12.ral.ibm.com.
         86400SIGSRV 3 6 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AC7G5SMMRTURT1pZaLWh9JVoIDOqHJmFO4S3
               vM4ntWDnWdgleyAU7pM= )
dns63.zos12.ral.ibm.com. 86400IN A9.12.6.68
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AAPerGMTlMZE7jMq+biqWzIQUnlqOacC+GWV
               qNHXDzjLzs1cx8zjSMs= )
         86400NXTdns64.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AB2XfwVfn5mSvOR24WfcgFhbrj+tgLEO2zIK
               ilUO++7IN8DCiGe721M= )
dns64.zos12.ral.ibm.com. 86400IN A9.12.6.63
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AHNttD6WLIiXOQDBhC9yOHAQzhWKPsmrDakJ
               Q6MronRtIfsrLrMpgTA= )
         86400NXTftp.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AFE9KYNoCGK2iipNvgR2CcRnY61vZEwwTSUc
               NNRrGOSteO/NkHMhgiQ= )
ftp.zos12.ral.ibm.com.86400IN CNAME host2.zos12.ral.ibm.com.
         86400SIGCNAME 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AEewWOzl4n6wJkYQfQgOtGlt5PhsYqTx12nk
               dkCrM/5/sVSyU1R+S3c= )
         86400NXThost1.zos12.ral.ibm.com. CNAME SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               ADuSM92ypsdSHARzh5f6zde2U9DlB6UUsbnx
               +DWN2dYGkbFq8YImvAc= )
```

```
host1.zos12.ral.ibm.com. 86400IN A9.12.6.67
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AC6kBQXVqfvMR3PYnos4gu3/mCffCO4lvSSW
               3j8Iov8Zv58toZKpo+w= )
         86400NXThost2.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               ABXbLOsxkuSApfUYYbDxIXzlOPz6UjkmoGKt
               quyjmMOIcmra3POa6lO= )
host2.zos12.ral.ibm.com. 86400IN A9.12.6.63
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AGnASy/SGCEHgmXseIxGw+wZwVXhdXO3fhvC
               Qk/Slb8vfFswLQ2CKRY= )
         86400NXTkerbftp3.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AGJTpDTpbEtwGjNfqizSbJ5sDdnYLEA1wPNK
               3yVm91qjKaYXBA3PNfO= )
kerbftp3.zos12.ral.ibm.com. 86400 IN A9.12.6.61
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AA8OpereY3NNWmHnevLwWWdBFs38crirugfW
               lBN4NAMtT4yA2Bhg6xs= )
         86400NXTlocalhost.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               ACvLtZtzpxh5oN9F+u1u5c2xx+gARp45ALkI
               gNwyloitdKsbkVOEiU8= )
localhost.zos12.ral.ibm.com. 86400 IN A127.0.0.1
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AHnM+FPE5STbT1nD9F4xYy1+e3oZYU1CLqMy
               XZy7RsKwOdQBoSShfQs= )
         86400NXTmail.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AGQR3hmzrYiUxkYEVuxLPLzzJksgIU3ZZ3U6
               5IzmmIhWVejqhU6MQ2I= )
mail.zos12.ral.ibm.com.86400IN CNAME host1.zos12.ral.ibm.com.
         86400SIGCNAME 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AFA7Fmoej3rbrZx1RhqanT7jI1hiel/5FiiA
               t6sOT8Z752NZyxJL1bc= )
         86400NXTsc63.zos12.ral.ibm.com. CNAME SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AAXOOFN3OLlbNkiEQChUokhhhTnSEHIWh2CO
               X+ViMfODKKkIWG6RscO= )
sc63.zos12.ral.ibm.com.86400IN A9.12.6.67
         86400IN A9.12.6.68
         86400SIGA 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AHSLjBeMfbxqZcAJqUBbLFwkydRyXpH/9irq
               5O1hmCP9ayM2/+QzMNA= )
         86400NXTwtsc63c.zos12.ral.ibm.com. A SIG NXT
         86400SIGNXT 3 5 86400 20020626182646 (
               20020527182646 9520 zos12.ral.ibm.com.
               AB2DmCx4ofy9pbNKvWBFESByEa1rYwoYNdoK
```

```
                G3RC+q9cEHPHu7pOZDg= )
wtsc63c.zos12.ral.ibm.com. 86400 IN A9.12.6.61
          86400SIGA 3 5 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
                AAlwKl2xSsq3DsoXKW/u2gpTzpmHCSlOC2ko
                CXLHpsjsbS+qKOwCVek= )
          86400NXTwww.zos12.ral.ibm.com. A SIG NXT
          86400SIGNXT 3 5 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
                ABjzPdVO14PXvdpfzuG+kzD7c85vXOzDiqTg
                rfIvo4GBToqLBOsqrLk= )
www.zos12.ral.ibm.com.86400IN A9.179.147.237
          86400SIGA 3 5 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
                ABTEo9fOBqqNWlYEOvRBMjNbykBrIfPiNQyC
                IgpxCc4cKFH+PLVlIic= )
          86400NXTzos12.ral.ibm.com. A SIG NXT
          86400SIGNXT 3 5 86400 20020626182646 (
                20020527182646 9520 zos12.ral.ibm.com.
                AG2t2kiSm9yEWpuiKkRp4Av223RRFP7I/WzX
                FoNdtKcaUdCYQtUfiWg= )
**************************** bottom of Data ******************************
```

A.3.3.6 SC63 BIND 9-based named.conf altered file

```
**************************** top of Data ******************************
controls {
     inet 127.0.0.1 allow { localhost; } keys { rndc-dns63;
};};
key rndc-dns63 {
algorithm "hmac-md5";
secret "iRIgqf6MhdHFCKl2FzqkgA==";
};
acl "zos12-net" { 9.12.6/24; };
options {
   pid-file "/etc/named.pid";
   directory "/etc/octavio/dnsdata";
   listen-on { 9.12.6.67;9.12.6.68; };
};
logging {
   channel "default_debug" {
   file "/etc/octavio/named.run" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel main_log {
   file "/etc/octavio/named_main.log" versions 2 size 20M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel security_log {
   file "/etc/octavio/named_security.log" versions 2 size 1M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                    };
   channel query_log {
   file "/etc/octavio/named_query.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                   };
   channel transfer_log {
   file "/etc/octavio/named_transfer.log" versions 2 size 10M;
   print-time yes;
   print-category yes;
   print-severity yes;
   severity dynamic;
                  };
   category client   { main_log; };
   category config   { main_log; };
   category "database" { main_log; };
   category dispatch { main_log; };
   category dnssec { security_log; main_log; };
   category general  { main_log; };
   category network  { main_log; };
   category "notify" { main_log; };
   category resolver { main_log; };
   category security { security_log; main_log; };
   category update   { main_log; };
   category queries  { query_log;};
   category lame-servers { query_log; main_log; };
   category xfer-in  { "transfer_log"; };
   category xfer-out { "transfer_log"; };
   category default  { main_log; };
   };
zone "zos12.ral.ibm.com" in {
   type master;
   file "zos12.for.v9.signed";
   allow-query { "zos12-net";};
   allow-update { 127.0.0.1;9.12.6.68; };
};
zone "6.12.9.in-addr.arpa" in {
   type master;
   file "zos12.rev.v9";
   allow-query { "zos12-net";};
   allow-update { 127.0.0.1;9.12.6.68; };
};
zone "0.0.127.in-addr.arpa" in {
   type master;
   file "zos12.lbk.v9";
   allow-query { "zos12-net";};
   allow-update { 127.0.0.1;9.12.6.68; };
};

zone "." in {
   type hint;
   file "zos12.ca.v9";
};

**************************** Bottom of Data ***************************
```

B

Dump of BIND DNS table (SIGINT)

A discussion of this table can be found in Chapter 10, “BIND Domain Name System (DNS)” on page 315.

/tmp/named_dump.db (from a SIGINT to a DNS process)

```
; Dumped at Wed Feb 25 01: 0:31 1998
;; ++zone table++
; itso.ral.ibm.com (type 1, class 1, source named.for)
;.time=888379335, lastupdate=888330120, serial=5,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=888330120, xaddr=[0.0.0.0], state=00 1, pid=0
; 251.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.251)
;.time=888379232, lastupdate=887 07201, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07201, xaddr=[0.0.0.0], state=00 1, pid=0
; 252.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.252)
;.time=888375013, lastupdate=887 07239, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07239, xaddr=[0.0.0.0], state=00 1, pid=0
; 236.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.236)
;.time=8883777 , lastupdate=887 07103, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07103, xaddr=[0.0.0.0], state=00 1, pid=0
; 235.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.235)
;.time=88837 936, lastupdate=887 07162, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07162, xaddr=[0.0.0.0], state=00 1, pid=0
; 105.2 .9.in-addr.arpa (type 1, class 1, source named.rev9.2 .105)
;.time=888378996, lastupdate=887 07270, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07270, xaddr=[0.0.0.0], state=00 1, pid=0
; 10 .2 .9.in-addr.arpa (type 1, class 1, source named.rev9.2 .10 )
;.time=88837922 , lastupdate=887 0725 , serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
```

```
;.ftime=887 0725 , xaddr=[0.0.0.0], state=00 1, pid=0
; 221.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.221)
;.time=888375568, lastupdate=887 071 8, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 071 8, xaddr=[0.0.0.0], state=00 1, pid=0
; 109.168.192.in-addr.arpa (type 1, class 1, source named.rev192.168.109)
;.time=888375763, lastupdate=887 07132, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07132, xaddr=[0.0.0.0], state=00 1, pid=0
; 0.0.127.in-addr.arpa (type 1, class 1, source named.lb )
;.time=888375759, lastupdate=887 07058, serial=2,
;.refresh=10800, retry=3600, expire=60 800, minimum=86 00
;.ftime=887 07058, xaddr=[0.0.0.0], state=00 1, pid=0
;; --zone table--
; Note: Cr=(auth,answer,addtnl,cache) tag only shown for non-auth RR's
; Note: NT=milliseconds for any A RR which we've used as a nameserver
; --- Cache & Data ---
$ORIGIN .
..360062.IN.NS.B.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.C.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.D.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.E.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.I.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.F.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.G.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.J.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.K.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.L.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.M.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.A.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
.360062.IN.NS.H.ROOT-SERVERS.NET..;Cr=answer [9.2 .10 .108]
$ORIGIN ral.ibm.com.
itso..IN.SOA.mvs03.itso.ral.ibm.com. gdente@itso.ral.ibm.com. (
..5 10800 3600 60 800 86 00 ).;Cl=
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=
$ORIGIN itso.ral.ibm.com.
localhost.IN.A.127.0.0.1.;Cl=
mvs03..IN.A.9.2 .105.126.;Cl=
..IN.A.192.168.251.1.;Cl=
..IN.A.192.168.252.1.;Cl=
..IN.A.192.168.236.1.;Cl=
..IN.A.192.168.235.3.;Cl=
..IN.A.192.168.221.20.;Cl=
..IN.A.192.168.109.3.;Cl=
wtr05199.IN.CNAME.gwen.itso.ral.ibm.com..;Cl=
mvs03-mpc.IN.A.192.168.235.3.;Cl=
mvs28..IN.A.9.2 .105.75.;Cl=
..IN.A.192.168.236.2.;Cl=
..IN.A.192.168.221.2 .;Cl=
..IN.A.192.168.109.1.;Cl=
rsserver.IN.A.9.2 .10 .108.;Cl=
mvs03-vipa.IN.A.192.168.251.1.;Cl=
dns2..IN.CNAME.rsserver.itso.ral.ibm.com..;Cl=
wtr05101.IN.CNAME.hen .itso.ral.ibm.com..;Cl=
mvs03c..IN.A.192.168.252.2.;Cl=
thismvs..IN.A.192.168.235.3.;Cl=
mvs28-iccp.IN.A.192.168.109.1.;Cl=
mvs03-xcf.IN.A.192.168.236.1.;Cl=
wtr05118.IN.CNAME. a y.itso.ral.ibm.com..;Cl=
wtr05119.IN.CNAME.silvia.itso.ral.ibm.com..;Cl=
a y..IN.A.9.2 .10 . 7.;Cl=
ourmvs..IN.CNAME.mvs03.itso.ral.ibm.com..;Cl=
gwen..IN.A.9.2 .10 .35.;Cl=
mvs03-en1.IN.A.9.2 .105.126.;Cl=4
othermvs.IN.CNAME.mvs28.itso.ral.ibm.com..;Cl=
hen ..IN.A.9.2 .10 .201.;Cl=
mvs03-icp.IN.A.192.168.221.20.;Cl=
silvia..IN.A.9.2 .10 .213.;Cl=
mvs03-iccp.IN.A.192.168.109.3.;Cl=
mvs28-ra3.IN.A.192.168.236.2.;Cl=
mvs03-sh.IN.A.192.168.252.1.;Cl=
mvs28-en1.IN.A.9.2 .105.75.;Cl=
mvs28-icp.IN.A.192.168.221.2 .;Cl=
$ORIGIN ROOT-SERVERS.NET.
A.600 99.IN.A.198. 1.0. .;Cr=answer [9.2 .10 .108]
B.600 99.IN.A.128.9.0.107.;Cr=answer [9.2 .10 .108]
C.600 99.IN.A.192.33. .12.;Cr=answer [9.2 .10 .108]
D.600 99.IN.A.128.8.10.90.;Cr=answer [9.2 .10 .108]
E.600 99.IN.A.192.203.230.10.;Cr=answer [9.2 .10 .108]
J.600 98.IN.A.198. 1.0.10.;Cr=answer [9.2 .10 .108]
F.600 99.IN.A.192.5.5.2 1.;Cr=answer [9.2 .10 .108]
K.600 98.IN.A.193.0.1 .129.;Cr=answer [9.2 .10 .108]
G.600 99.IN.A.192.112.36. .;Cr=answer [9.2 .10 .108]
H.600 99.IN.A.128.63.2.53.;Cr=answer [9.2 .10 .108]
L.600 98.IN.A.198.32.6 .12.;Cr=answer [9.2 .10 .108]
M.600 98.IN.A.202.12.27.33.;Cr=answer [9.2 .10 .108]
I.600 99.IN.A.192.36.1 8.17.;Cr=answer [9.2 .10 .108]
$ORIGIN 168.192.in-addr.arpa.
221..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
235..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5


251..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
236..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
252..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
109..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
```

```
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5


$ORIGIN 221.168.192.in-addr.arpa.
2 ..IN.PTR.mvs28.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs28-icp.itso.ral.ibm.com..;Cl=5
20..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03-icp.itso.ral.ibm.com..;Cl=5
$ORIGIN 235.168.192.in-addr.arpa.
3..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
$ORIGIN 251.168.192.in-addr.arpa.
1..IN.PTR.mvs03-vipa.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
$ORIGIN 236.168.192.in-addr.arpa.
2..IN.PTR.mvs28.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs28-xcf.itso.ral.ibm.com..;Cl=5
1..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03-xcf.itso.ral.ibm.com..;Cl=5
$ORIGIN 252.168.192.in-addr.arpa.
2..IN.PTR.mvs03c.itso.ral.ibm.com..;Cl=5
1..IN.PTR.mvs03-sh.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
$ORIGIN 109.168.192.in-addr.arpa.
1..IN.PTR.mvs28.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs28-iccp.itso.ral.ibm.com..;Cl=5
3..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03-iccp.itso.ral.ibm.com..;Cl=5
$ORIGIN 0.127.in-addr.arpa.
0..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5


$ORIGIN 0.0.127.in-addr.arpa.
1..IN.PTR.localhost..;Cl=5
$ORIGIN 2 .9.in-addr.arpa.
10 ..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
105..IN.SOA.mvs03.itso.ral.ibm.com. mvs03.itso.ral.ibm.com. (
..2 10800 3600 60 800 86 00 ).;Cl=5
..IN.NS.mvs03.itso.ral.ibm.com..;Cl=5
..IN.NS.mvs28.itso.ral.ibm.com..;Cl=5
..IN.NS.rsserver.itso.ral.ibm.com..;Cl=5
$ORIGIN 10 .2 .9.in-addr.arpa.
7..IN.PTR.wtr05118.itso.ral.ibm.com..;Cl=5
35..IN.PTR.wtr05199.itso.ral.ibm.com..;Cl=5
108..IN.PTR.rsserver.itso.ral.ibm.com..;Cl=5
..IN.PTR.dns2.itso.ral.ibm.com..;Cl=5
213..IN.PTR.wtr05119.itso.ral.ibm.com..;Cl=5
201..IN.PTR.wtr05101.itso.ral.ibm.com..;Cl=5
$ORIGIN 105.2 .9.in-addr.arpa.
126..IN.PTR.mvs03-en1.itso.ral.ibm.com..;Cl=5
..IN.PTR.mvs03.itso.ral.ibm.com..;Cl=5
75..IN.PTR.mvs28.itso.ral.ibm.com..;Cl=5
```

```
..IN.PTR.mvs28-en1.itso.ral.ibm.com..;Cl=5
209..IN.PTR.rsserver.itso.ral.ibm.com..;Cl=5


; --- Hints ---
$ORIGIN .
..3600.IN.NS.A.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.B.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.C.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.D.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.E.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.F.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.G.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.H.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.I.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.J.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.K.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.L.ROOT-SERVERS.NET..;Cl=0
.3600.IN.NS.M.ROOT-SERVERS.NET..;Cl=0
$ORIGIN ROOT-SERVERS.NET.
A.3600.IN.A.198. 1.0. .;Cl=0
B.3600.IN.A.128.9.0.107.;Cl=0
C.3600.IN.A.192.33. .12.;Cl=0
D.3600.IN.A.128.8.10.90.;Cl=0
E.3600.IN.A.192.203.230.10.;Cl=0
J.3600.IN.A.198. 1.0.10.;Cl=0
F.3600.IN.A.192.5.5.2 1.;Cl=0
K.3600.IN.A.193.0.1 .129.;Cl=0
G.3600.IN.A.192.112.36. .;Cl=0
L.3600.IN.A.198.32.6 .12.;Cl=0
H.3600.IN.A.128.63.2.53.;Cl=0
M.3600.IN.A.202.12.27.33.;Cl=0
I.3600.IN.A.192.36.1 8.17.;Cl=0
```

C

Sample DHCP configuration file

This sample DHCP configuration file is from /usr/lpp/tcpip/samples/dhcpsd.cfg.

```
##########################################################################
#   IBM Communications Server for OS/390
#   SMP/E distribution name: EZATDDSD
#
#   5647-A01 (C) Copyright IBM Corp. 1998.
#   Licensed Materials - Property of IBM
##########################################################################
#
#   dhcpsd.cfg -- DHCP/PXE Redirection Server Configuration File
#
#   This file contains most of the directives that can be specified by the
#   server's administrator to configure the server and enforce
#   policies.  Please reference the official DHCP Server documentation for
#   a complete listing of server directives.  This file is only a sample.
#   The finished file must be placed in the directory specified by the ETC
#   environment variable.
#
#   Do not put any long line without spaces in this file.
#
#   A line starting with a '#' character is a comment and is ignored.
#   A '#' on a line which is not part of a quoted string indicates
#   that anything to the right of this character is a comment and should
#   be ignored.
#
#   A continuation character of '\' is supported.  It must be
#       the last non-whitespace character on the line prior to
#       any comments.  It should not be used in the updateDNS line.
#
#   The directives are specified in the form of
#   <keyword> <value1> ... <valueN>.
#
#   Here is a partial list of keywords whose value can be specified
#   in this file:
#
#   Keyword          Effect
#   -------------    ---------------------------------------------------
#   servertype       Specifies the mode the server is running in.
#   imageserver      Specifies the address of the image server.
#   numLogFiles      The number of log files desired.
#   logFileSize      The size of log files in kilobytes.
#   logFileName      The name of the most recent log file.
#   logItem          An item to be logged.
#
#   subnet           Address of one subnet within a network.
#   option           A configuration option value to pass to clients.
#
#   leaseTimeDefault
#                    The default duration of leases issued by this server.
#
#   leaseExpireInterval
#                    The time interval at which the expiration condition
#                    of the leases currently running is examined.
#
#   supportBOOTP     Whether or not to support BOOTP clients.
#   supportUnlisted Clients  Whether or not to support clients that are
#                            not listed specifically with individual
#                            client statements. (see "client" below.)
#   updateDNSP       String defining command to use to update the DNS PTR
#                    IP address to name mappings for IP addresses assigned
#                    by this server.
#   updateDNSA       String defining command to use to update the name to DNS PTR
#                    IP address mappings for IP addresses assigned
#                    by this server.
#   proxyArec        Administrator policy to perform proxy A record updates on beha
#                    of a client.
#   class            Definition of a set of options for a specific class
#                    of clients.
#   client           Definition of a set of options for a specific client
#                    or a definition of a client not to be serviced
#                    or a definition of an address not to be used.
#   vendor           Definition of a vendor class option (43)
#
#
#   The scope of a keyword is limited by a pair of curly brackets ({, })
#   within which the keyword is located.  If a keyword is located outside
#   of any pair of curly brackets, its scope is applicable to all the
#   clients served by this server.  The curly brackets must appear alone
#   on a line.
#
#  Log files.  This set of parameters specifies the log files that will be
#  maintained by this server.  Each parameter is identified by a keyword
#  and followed by its value.
#
#  Keyword      Value           Definition
#  --------     ------------    -----------------------------------------
#  numLogFiles  0 to 99         number of log files.  If 0 is specified,
#                               no log file will be maintained and no log
#                               message is displayed anywhere.  When a log
#                               file reaches maximum size, a new log file
#                               is created, until the maximum number of
#                               log files have been created.  Only the most
#                               recent n log files are kept/
#
#  logFileSize  in K bytes      maximum size of a log file.  When the size
#                               of the most recent log file reaches this
#                               value, it is renamed and a new log file is
#                               created.
#
#  logFileName  file path       name of the most recent log file.  Less
#                               recent log files have the number 1 to
#                               n-1 appended to their names; the larger
#                               the number, the less recent the file.
#
#  logItem                      An item that will be logged.
#               SYSERR          System error, at the interface to the platform.
#               OBJERR          Object error, in between objects in the process.
#               PROTERR         Protocol error, between client and server.
#               WARNING         Warning, worth of attention from the user.
#               EVENT           Event occurred to the process.
#               ACTION          Action taken by the process.
#               INFO            Information that might be useful.
#               ACNTING         Accounting information on clients served.
#               TRACE           Code flow, for debugging.
#               STAT            Statistical information.
#
#


#
#  Subnet.   Statements with this keyword specify the addresses of the
#  subnets and pools of addresses to use for clients within the subnet.
#
#  Here is a list of the parameters in this set and their definitions:
#
#  Keyword   Value              Definition
#  --------  ---------------    ------------------------------------------
#
#  subnet    <Subnet address> [<Subnet Mask>] <range> [(alias=name
#            DDNSServer=ip_address]
#
#                               Parameters to the right of a left parenthesis
#                               are used only by the DHCP Server Configuration
#                               program.  A space must precede the left
#                               parenthesis.  The DHCP server parses statements
#                               to the right of a left parenthesis as comments.
#
#                               subnet statement.  One or more subnet
#                               statements are allowed in a configuration file.
#
#                               "Subnet address" is the address of this subnet.
#                               This address is specified in dotted decimal
#                               notation (e.g., 9.17.32.0 or 128.81.22.0).
#                               The subnet must be within the subnet mask, and
#                               the address can be no longer (in bits)
#                               than the subnet mask.  For example,
#                               if the subnet mask is 255.255.255.0, the
#                               address 9.67.10.128 is too long.
#
#                               The subnet address may optionally be followed
#                               by the subnet mask or a range.
#
#                               The mask for the subnet in dotted decimal
#                               notation or in integer format.  A subnet mask
#                               divides the subnet address into a subnet
#                               portion and a host portion.  If no value is
#                               entered for the subnet mask, the default is the
#                               class mask appropriate for an A, B, or C class
#                               network.
#
#                               The subnet mask may be specified either in the
#                               dotted notation (e.g., 255.255.255.128) or as
#                               a number indicating the number of 1 bits in the
#                               mask (e.g., 25, which is equivalent to
#                               255.255.255.128).
#
#                               Subnet address may be followed by a range.
#                               If a range is specified, it describes the
#                               pool of addresses this server will administer
#                               for this subnet. A
#                               range is specified by host addresse in
#                               dotted decimal notation
#                               separated by a hyphen with no intervening
#                               spaces (e.g., 192.81.20.1-129.81.20.128).
#
#                               If no range is specified,
#                               all host addresses in the subnet are
#                               administered by this server.
#
#                               The address pools administered by different
#                               DHCP servers must not overlap. Otherwise
#                               two hosts may be assigned the same address.
#
#                               Parameters to the right of a left parenthesis
#                               are used only by the DHCP Server Configuration
#                               program.  A space must precede the left
#                               parenthesis.  The DHCP server parses statements
#                               to the right of a left parenthesis as comments.
#
#                               The parameter alias=name immediately after a
#                               left parenthesis contains the symbolic name,
#                               which appears in the DHCP Server Configuration
#                               program graphic display of the server
#                               configuration.  If no name is entered, the
#                               subnet IP address is used to identify the
#                               subnet in the DHCP Server Configuration
#                               program display.
#
#                               A subnet statement may be immediately followed
#                               by a pair of curly brackets, in which parameters
#                               (e.g., options) particular to this subnet can
#                               be specified.
#
#  subnet    <subnet_address> [<subnet_mask>] <range> [ <label:value[/priority]]
#                               To define a subnet group, use
#                               label:value[/priority]] in the Subnet statement.
#
#                               The subnet_address, subnet_mask, and range
#                               parameters are described in Defining Subnets.
#
#                               label identifies subnets grouped together on
#                               the same wire.
#
#                               value[/priority] is a string of 1 to 64
#                               alphanumeric characters that identifies the
#                               subnet, followed by the priority in which this
#                               subnet's address pool is used.  No spaces are
#                               allowed in labels.  Priority is a positive
#                               integer.
#
#  inOrder:<labellist>
#  balance:<labellist>
#                               To specify the policy by which IP addresss
#                               are served from multiple subnets.
#
#                               The labelslist is a list of labels in which
#                               each label indentifies a subnet group.
#
#  class     <class_name>   [<range>]
#                              Definition of a class.  The class name is a
#                               simple ascii string.  Enclose the class name
#                               in double quotes if the name contains spaces.
#                               A class's scope is determined by the curly
#                               brackets in which it is enclosed.  If it is
#                               outside all curly brackets, then its scope is
#                               the entire file.
#
#                               A class name may be followed by a range.
#                               If a range of addresses is
#                               specified, then addresses in that range
#                               will be assigned to clients who request this
#                               class, and only to those clients.  If a
#                               range is specified on a class statement,
#                               that statement must be within the scope
#                               of a subnet statement which
#                               also specifies a range, and the range of
#                               the class must be within the range of the
#                               subnet.
#
#                               Note that if a client requests this class,
#                               but it is not within the subnet,
#                               it will not be assigned an address from
#                               the subnet.
#
#                               If an address range is not specified, then
#                               addresses will be given to clients using
#                               the usual rules of assignment.
#
#                               The class statement may be immediately followed
#                               by a pair of curly brackets, in which
#                               the options particular to this class can be
#                               specified.  A class may be defined within
#                               the curly brackets of a subnet, but a subnet
#                               may not be defined within the curly brackets
#                               of a class.
#
#                               Options set up in the subnet
#                               containing a class definition will also
#                               apply to the class.
#
#  vendor    <vendor_name>   [hex "<value>"]
#
#                              Definition of a vendor option.  The vendor name
#                              is a simple ascii string.  Enclose the class name
#                              in double quotes if the name contains spaces.
#                              The scope is that of the entire file.  Vendor
#                              statements within subnet, class, or client scopes
#                              are ignored.
#
#                               A vendor name may be followed by a hex ascii
#                               string which represents the hex value of the
#                               data portion of the option. A client may
#                               request a vendor option by specifying
#                               the vendor name in option 43 of the Bootp
#                               or DHCP request.  The server returns the
#                               specified value as the data in option 43.
#                               The data to be returned is specified in
#                               hexadecimal, e.g.:
#
#                                 hex "01 a0 23"
#
#                               The vendor statement may be immediately followed
#                               by a pair of curly brackets, in which
#                               the options particular to this vendor can be
#                               specified.  Within these curly braces, the
#                               usual option value encoding/decoding rules
#                               do not apply.  This means that the value for
#                               each option must be specified either as an
#                               ascii string, or as hex in the hex ascii
#                               string construct:
#                                       hex "<value>"
#                               where <value> is a string of hex values
#                               each byte separated by a space.  I.e.
#                                       hex "01 02 03"
#
#  client   <id_type>  <id_value>  <address>
#                               Definition of a client record.
#
#                               <id_type> is one of the hardware types defined
#                               in RFC 1340 (e.g. 1 for 10 megabit Ethernet,
#                               6 for 802.5 Token Ring.)  The type may be
#                               0, in which case the hardware type is not
#                               specified and the
#                               id_value may be a string of any format.
#
#                               <id_value> is a character string if the type
#                               is 0. Typically, this would be a domain name.
#                               For a non-zero <id_type>, the <id_value> is
#                               a hexadecimal string representing
#                               the hardware address of the client.
#
#                               Note: An <id_type> of 0 and an <id_value> of
#                               '0' indicates that the <address> specified
#                               should not be administered by this server.
#                               A client record of this format is used to
#                               reserve addresses within a subnet range;
#                               such addresses are not offered to clients.
#
#                               The <address> can be the string "none" to
#                               indicate that the client matching
#                               <id_type> and <id_value> should
#                               not be serviced by this server.
#                               The <address> can be the string "any" to
#                               indicate that the server should choose
#                               an appropriate address for this client.
#                               The <address> can be an internet address
#                              in dotted notation (eg. 9.2.15.82).  This
#                              will be the address given to the
#                              particular client specified by <id_type>
#                              and <id_value>.  As mentioned above, an
#                              <id_type> of 0 and an <id_value> of '0'
#                              indicates that the <address> specified
#                              should not be distributed by this server.
#
#       The client statement may be immediately followed
#       by a pair of curly brackets, in which the options
#       particular to this client can be specified.
#
#       Note: All clients inherit all globally defined options.
#               A client defined in a subnet scope inherits
#               options defined for that subnet and emcompassing network.
#
#       A class definition inside a client scope is not allowed.
#
#       The client statement may be used to configure bootp clients.
#       To do this, specify all the bootp options using the option
#       syntax defined below.
#
#
#  Option.  This keyword identifies an option statement.  The BOOTP/DHCP
#  options are defined in  "DHCP Options and BOOTP Vendor Extensions"
#  (RFC 1533).
#
#  An option is specified by the "option" keyword followed by the option code
#  of this option and its data field, in a single line.  One or more options
#  may be specified.
#
#  The scope within which an option applies is delimited by a pair of curly
#  brackets ({, }) surrounding the option statement.
#
#  If two or more options with the same option code are specified, the
#  one with the most specific scope is used.  This allows, for example,
#  an option specified at the subnet scope to override that same option
#  specified at the network or global scope.  If two or more options
#  with the same option code are specified within the same scope,
#  the first one read by the server will be the one used, (subject to
#  its being overridden by the same option in a more specific scope).
#
#  Some options defined in the RFC may not be configured on an options
#  statement, but are used automatically by the server and/or client.
#  These are:
#
#       0       Pad Option
#       255     End Option
#
#  and the following DHCP extensions:
#
#       52      Option Overload
#       53      DHCP Message Type
#       54      Server Identifier
#       55      Parameter Request List
#       57      Maximum DHCP Message Size
#       60      Class-identifier of client
#       61      Client-identifier.
#
#  All other options may be specified by the option statements.
```

```
#
#  When specifying an option, its data field takes one of the following
#  formats:
#
#       IP Address         :   a single IP address.
#       IP Addresses       :   One or more IP addresses separated by
#                               spaces.
#       IP Address Pair    :   two IP addresses separated by a single colon.
#       IP Address Pairs   :   One or more IP address pairs separated by
#                               spaces.
#       Boolean            :   [0, 1]
#       Byte               :   [-128, 127]
#       Unsigned Byte      :   [0, 255]
#       Unsigned Bytes     :   space delimited list of values in range [0, 255]
#       Short              :   [-32768, 32767]
#       Unsigned Short     :   [0, 65535]
#       Unsigned Shorts    :   space delimited list of values in range [0, 65535]
#       Long               :   [-2147483648, 2147483647]
#       Unsigned Long      :   [0, 4294967295]
#       String             :   string of characters possible enclosed between two
#                               double quotes.
#
#       Note: All IP addresses are specified in dotted-decimal form.
#
#
#  Each of the defined options is listed below by its code and name, followed
#  by the format of its data field.
#
#  Code Name                          Data field format/Notes
#  ---- -----------------------       -----------------------------------
#
#  RFC 1497 Vendor Extensions
#
#  0    Pad Option                    No need to specify
#  255  End Option                    No need to specify
#  1    Subnet Mask                   Unsigned Long
#  2    Time Offset                   Long
#  3    Router Option                 IP Addresses
#  4    Timer Server Option           IP Addresses
#  5    Name Server Option            IP Addresses
#  6    Domain Name Server Option     IP Addresses
#  7    Log Server Option             IP Addresses
#  8    Cookie Server Option          IP Addresses
#  9    LPR Server Option             IP Addresses
#  10   Impress Server Option         IP Addresses
#  11   Resource Location Server Option IP Addresses
#  12   Host Name Option              String
#     Note: Option 12 should only be specified within the scope of client
#           statements.
#  13   Boot File Size Option         Unsigned Short
#  14   Merit Dump File               String
#  15   Domain Name                   String
#  16   Swap Server                   IP Address
#  17   Root Path                     String
#  18   Extensions Path               String
#
#  IP Layer Parameters per Host
#
#  19   IP Forwarding Enable/Disable
#           Option                    Boolean, or client class profile file
#  20   Non-local Source Routing
#           Enable/Disable Option       Boolean, or client class profile file
#  21   Policy Filter Option            IP Address Pairs
#  22   Maximum Datagram Reassembly Size
#                                       Unsigned Short
#  23   Default IP Time-to-live         Unsigned Byte
#  24   Path MTU Aging Timeout Option   Unsigned Long
#  25   Path MTU Plateau Table          One or more Unsigned Short separated by
#                                        spaces
#
#  IP Layer Parameters per Interface
#
#  26   Interface MTU Option            Unsigned Short
#  27   All Subnets are Local Option    Boolean
#  28   Broadcast Address Option        IP address
#  29   Perform Mask Discovery Option   Boolean
#  30   Mask Supplier Option            Boolean
#  31   Perform Router Discovery Option Boolean
#  32   Router Solicitation Address
#           Option                      IP Address
#  33   Static Route Option             IP Address Pairs
#
#  Link Layer Parameters per Interface
#
#  34   Trailer Encapsulation Option    Boolean
#  35   ARP Cache Timeout Option        Unsigned Long
#  36   Ethernet Encapsulation Option   Boolean
#
#  TCP Parameters
#
#  37   TCP Default TTL Option          Unsigned Byte
#  38   TCP Keepalive Interval Option   Unsigned Long
#  39   TCP Keepalive Garbage Option    Boolean
#
#  Application and Service Parameters
#
#  40   NIS Domain Option               String
#  41   NIS Option                      IP Addresses
#  42   Network Time Protocol Servers
#           Option                      IP Addresses
#  43   Vendor Specific Information     MUST use "vendor" keyword syntax.
#  44   NetBIOS over TCP/IP Name Server
#           Option                      IP Addresses
#  45   NetBIOS over TCP/IP Datagram
#           Distribution Server         IP Addresses
#  46   NetBIOS over TCP/IP Node
#           Type Option                 Unsigned Byte
#  47   NetBIOS over TCP/IP Scope
#           Option                      Unsigned Bytes
#  48   X Window System Font Server
#           Option                      IP Addresses
#  49   X Window System Display
#           Manager Option              IP Addresses
#
#  DHCP Extensions
#
#  51   IP Address Lease Time           Unsigned Long
#           May be specified in a network, subnet, class or client definition
#           to indicate the lease time to be be used in that scope.
#           Use 0xffffffff to indicate an infinite/permanent lease.
#  56   Message                         String
#  58   Renewal (T1) Time Value
#  59   Rebinding (T2) Time Value
#
#



#  Servertype.  This parameter specifies the running mode of the server.
#  If the value is not specified, standard DHCP operation will take place.
#  No pxe extensions is recognized.
#
#  Keyword             Value                 Definition
#  -----------   -----------------------    --------------------------
#  servertype    [dhcp|pxeproxy|pxedhcp]    If "dhcp" regular non pxe
#                                              extension server
#
#                                           If "pxeproxy" only pxe-extensions
#                                             is recognized. Pure redirection
#                                             server.
#
#                                           If "pxedhcp" combined server.  Performs
#                                             both standard dhcp and pxe functions.
#
#  Imageserver.  The address of the BINL server.  Only used for pxeclients and only
#  at the global level.
#
#  Keyword           Value                     Definition
#  -----------     -------------------        -------------------------------------
#  imageserver    [ipaddress|hostname]        If "ipaddress" actual address of imag
#
#                                             If "hostname" dns lookup is done to r
#                                               ipaddress of image server.
#
#  Default Lease Time.  This parameter specifies the default lease
#  duration for the leases issued by this server.  In the absence
#  of any more specific lease duration (e.g., lease duration for
#  specific client(s) or class of clients,) the lease duration
#  specified by this parameter takes effect.  The keyword for this
#  parameter and its values are as follows:
#
#  Keyword              Value
#  -----------          -----------------------
#  leaseTimeDefault     <amount> [<unit>]
#
#  The amount is specified by a decimal number.  The unit is one
#  of the following (plural is accepted):
#
#              year
#              month
#              week
#              day
#              hour
#              minute  (default if no unit is specified)
#              second
#
#  There is at lease one space between the amount and the unit.
#  Only the first amount following the keyword has any effect.
#
#  If this parameter is not specified, the default lease duration is
#  24 hours.
```

```
#
#  This statement should appear outside of any pair of curly brackets,
#  and it applies to all leases issued by this server.
#
#  NOTE: This value may be overridden for specific subnets, networks,
#  or classes, by specifying option 51 within the scope of the network,
#  subnet, or class.
#
#  Lease Expiration Time Interval.  This parameter specifies the time
#  interval at which the lease expirations are checked. If a lease
#  has expired, it is returned to the free pool.
#  The keyword for this statement and its values are as follows:
#
#  Keyword              Value
#  -----------          -----------------------
#  leaseExpireInterval  <amount> [<unit>]
#
#  The amount is specified by a decimal number.  The unit is one
#  of the following (plurals are accepted):
#
#              year
#              month
#              week
#              day
#              hour
#              minute  (default if no unit is specified)
#              second
#
#  There is at least one space between the amount and the unit.
#  Only the first amount following the keyword has any effect.
#
#  If this parameter is not specified, the default interval is
#  one minute.
#
#  This statement should appear outside of any pair of curly brackets,
#  and it applies to all leases issued by this server.
#
#  The value of this parameter should be some fraction of the value
#  specified for leaseTimeDefault so that lease expirations are
#  recognized on time.
#
#
#  BOOTP Support.  This parameter indicates to the server whether or
#  not to support requests from BOOTP clients.  The keyword for this
#  parameter and its values are as follows:
#
#  Keyword              Value           Definition
#  -------------        ----------      ------------------------------
#  supportBOOTP         [yes | no]
#                                       If "yes" is specified, the
#                                       server will support BOOTP
#                                       clients.
#
#                                       If the value field is not
#                                       "yes", or the keyword is omitted,
#                                       the server will not support
#                                       BOOTP clients.
#
#  The scope of this parameter covers all subnets administered by this server.
#
```

```
#  If the server previously supported BOOTP clients and has been
#  reconfigured not to support BOOTP clients, the address binding
#  for a BOOTP client established before the reconfiguration, if any,
#  will still be maintained until the time when that BOOTP client sends
#  a request again (when it is rebooting.)  At that time, the server
#  will not respond, and the binding will be removed.
#
#
#
#  Support for unlisted clients.  This parameter indicates to the server whether
#  or not to support requests from clients that are not specifically configured
#  with their own individual client statements in the server.  The keyword for
#  this parameter and its values are as follows:
#
#  Keyword                   Value             Definition
#  -------------             ---------- ------------------------------
#  supportunlistedClients    [yes | dhcp | bootp | both | no]
#                                       If "yes" or "both" is specified, the
#                                       server will support unlisted
#                                       clients.
#
#                                       If "dhcp" is specified, the
#                                       server will support unlisted
#                                       dhcp clients but not bootp clients.
#
#                                       If "bootp" is specified, the
#                                       server will support unlisted
#                                       bootp clients but not dhcp clients.
#
#                                       If the value field is anything other
#                                       than "yes", the server will not support
#                                       unlisted clients.
#
#  If this keyword is not found in the file, the server WILL support
#  clients not specifically configured with a client statement.
#
#  Keyword          Value               Definition
#  -------------    ----------  ------------------------------
#  updateDNSP        string      A string enclosed in quotes. It includes
#                               the name of a program to execute to update
#                               the DNS server with the new hostname
#                               of the assigned IP address, and the parameters
#                               to pass to the program.
#
#              Ip Address - This is the IP address leased to this client
#                      by the server.   The string is supplied in dotted
#                      notation, ie 9.2.23.43.
#
#              hostname.domainname
#                 hostname - the value of option 12.
#                         This is the value the server would return for option
#                         12 for the client's DHCP request, if such a value
#                         is configured.  If not, then if the client
#                         specifies option 12 in its DHCP request, that value
#                         is used.  Otherwise, the statement is not
#                         executed.
#
#                 domainname - This is the value of option 15.
#                         This is the value the server would return for option
#                         15 for the client's DHCP request, if such a value
#                           is configured.  If not, then if the client
#                           specifies option 15 in its DHCP request, that value
#                           is used.  Otherwise, a null string "" is passed
#                           on the name server update command.
#
#               leasetime - This is the lease time granted by the server.
#                        This string is a decimal number representing the
#                        number of seconds of the lease.
#
#       These values are output by dhcp in this order:
#               Ip Address hostname.domainname leasetime
#
#       An example updateDNSP string might be:
#               updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
#
#       Only one updateDNSP string is allowed in this file.  If multiple
#               instances occur, the last one found in the file will be used.
#
#
#
#  Keyword          Value               Definition
#  -------------    ----------  ------------------------------
#  updateDNSA        string      A string enclosed in quotes. It includes
#                               the name of a program to execute to update
#                               the DNS server with the new hostname
#                               of the assigned IP address, and the parameters
#                               to pass to the program.
#
#               hostname.domainname
#                  hostname - the value of option 12.
#                          This is the value the server would return for option
#                          12 for the client's DHCP request, if such a value
#                          is configured.  If not, then if the client
#                          specifies option 12 in its DHCP request, that value
#                          is used.  Otherwise, the statement is not
#                          executed.
#
#                  domainname - This is the value of option 15.
#                          This is the value the server would return for option
#                          15 for the client's DHCP request, if such a value
#                          is configured.  If not, then if the client
#                          specifies option 15 in its DHCP request, that value
#                          is used.  Otherwise, a null string "" is passed
#                          on the name server update command.
#
#               Ip Address - This is the IP address leased to this client
#                       by the server.   The string is supplied in dotted
#                       notation, ie 9.2.23.43.
#
#               leasetime - This is the lease time granted by the server.
#                       This string is a decimal number representing the
#                       number of seconds of the lease.
#
#       These values are output by dhcp in this order:
#               hostname.domainname Ip Address leasetime
#
#
#       An example updateDNSA string might be:
#        updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;0;q" -q"
#
```

```
#
#  Keyword       Value                       Definition
#  ----------   ------------------------    ------------------------------
#  proxyarec    [standard|protected|no]     Allows the administrator to have
#                                           a server policy to perform proxy
#                                           A record updates on behalf of a
#                                           client.
#
#                                           <standard>. No verification is made
#                                           to ensure the client id match before
#                                           an update is performed.
#
#                                           <protected>. A check is made to ensure
#                                           that the client id matches on each upda
#
#                                           <no>.  No proxy A records are performed
#                                           at the current scope.
#
#
##################################################################################

# The remaining portion of this file is an sample configuration file.
# Comments are added to assist in understanding the configuration file.
# Further information and detail is found in the online user documentation.

# Setup of the log file information.  This includes the size and name of the
# logfile along with number of logfiles maintained and type of information that
# will be logged.
numLogFiles     10
logFileSize     1000
logFileName     dhcpsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE
logItem         STAT

# Server mode setup
servertype pxedhcp

imageserver   5.4.3.2
# Configuration of server parameters.
leaseTimeDefault       120                 # 120 minutes
leaseExpireInterval    20 seconds
supportBOOTP           yes
supportUnlistedClients  yes

# Configuration of DNS parameters.
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;0;q" -q"

# This administrator will attempt to do proxy A record updates for all
# clients unless overridden at a lower scope.
proxyarec   standard
```

```
# Server Address for PXE clients to obtain image information.
imageserver   5.4.3.2

# Global options.  Passed to every client unless overridden at a lower scope.
option 15       "raleigh.ibm.com"                # domain name
option 6        9.67.1.5                         # dns server

class manager
{
  option 48   6.5.4.3
  option 9    9.37.35.146
  option 210  "manager_authority"   # site specific option giving to all managers

}

# Setup to send options to the pxeclient.  Sent in the encapsulated option 43.
vendor PXEClient
 {
   option 1 1.1.4.5
   option 2 2
   option 3 3
   option 4 4
   option 5 5
}

# This subnet specifies it subnetmask as number of bits instead of dotted decimal
# notation.  24 bits corresponds to 255.255.255.0
subnet  9.2.23.0    24        9.2.23.120-9.2.23.126
{
        option 28       9.2.23.127         # broadcast address
        option 9   5.6.7.8
        option 51  200

        # class manager defined at the subnet scope.  Option 9 here will override
        # the option 9 specified in the global manager class.
        class manager
        {
                option 9   9.2.23.98
        }

        # Programmers have their own subnet range.
        class developers  9.2.23.125-9.2.23.126
        {
                option 51       -1               # infinite lease.
                option 9        9.37.35.1        # printer used by the developers
        }
}

##
##  The next two subnets are an example of how to create gaps in a subnet range.
##
subnet 129.42.1.0 255.255.255.0       129.42.1.1-129.42.1.50
{
        option 51       650             # lease time
        option 9        6.7.8.9         # printer for all
        option 15 "mobile.watson.ibm.com"       # domain
}

subnet 129.42.1.0  255.255.255.0      129.42.1.200-129.42.1.254
{
        option 9        1.2.3.4                 # printer for all
        option 15 "mobile.watson.ibm.com"       # domain
        option 51       600                     # lease time
}

# Example of a client that will accept any address but will have its own set of
# options.
client 6        0x10005aa4b9ab  ANY
{
        option 51 999
        option 1 255.255.255.0
}

# Exclude an address from service.
client 0       0               9.2.23.121

#
# end of dhcpsd.cfg
#
```

D

The DHCP log data set

This appendix contains the dhcpsd.log data set after starting the IBM Network Station with the configuration file we used in Figure 9-1 on page 286.

In the log file, we discover the following:

1 is the start of the DHCP server.

2 starts the adaption of the address range 185 - 220.

3 shows the selection of the default port 67 and 4 starts listening to the port.

5 receives a package from the IBM Network Station for a boot request.

6 is the DHCPDISCOVER request.

7 here the client's MAC address is not mapped to a specific IP address, but is found in the clientele list 8.

9 here the client gets a mapping to the first free address of the pool 10, reserves it, and checks that the subnet definitions are matching 11.

12 when this is done, it sends an offer (DHCPOFFER).

```
09/30 14:41:11 : INFO:   ..log_initialize: ***************************
09/30 14:41:11 : INFO:   ..log_initialize: *     NEW LOG FOLLOWS     *
09/30 14:41:11 : INFO:   ..log_initialize: * | | | | | | | | | | | | *
09/30 14:41:11 : INFO:   ..log_initialize: * V V V V V V V V V V V V *
09/30 14:41:11 : INFO:   ..log_initialize: ***************************
09/30 14:41:11 : SYSERR: ..log_initialize: Logging ENABLED
09/30 14:41:11 : OBJERR: ..log_initialize: Logging ENABLED
09/30 14:41:11 : PROTERR:..log_initialize: Logging ENABLED
09/30 14:41:11 : WARNING:..log_initialize: Logging ENABLED
09/30 14:41:11 : EVENT:  ..log_initialize: Logging ENABLED
09/30 14:41:11 : ACTION: ..log_initialize: Logging ENABLED
09/30 14:41:11 : INFO:   ..log_initialize: Logging ENABLED
09/30 14:41:11 : ACNTING:..log_initialize: Logging ENABLED
09/30 14:41:11 : TRACE:  ..log_initialize: Logging ENABLED
09/30 14:41:11 : TRACE:  ..server_initialize: function entered
```

```
09/30 14:41:11 : TRACE:  ....initialize_statistic_list: function Entered
1
09/30 14:41:11 : TRACE:  ....profile_repository_initialize: function entered
09/30 14:41:11 : TRACE:  ......pr_mismatch_quote_check: function entered
09/30 14:41:11 : TRACE:  ......freenetprofile: function entered
09/30 14:41:11 : TRACE:  ......pr_initialize_server_parms: function entered
09/30 14:41:11 : TRACE:  ........pr_set_time_interval: function entered
09/30 14:41:11 : TRACE:  ........pr_set_time_interval: function entered
09/30 14:41:11 : TRACE:  ......pr_initialize_boot_canon_append: function entered
09/30 14:41:11 : TRACE:  ......pr_initialize_all_subnets: function entered
09/30 14:41:11 : TRACE:  ........pr_initialize_one_subnet: function entered
09/30 14:41:11 : TRACE:  ..........pr_initialize_boot_canon_append: function entered
09/30 14:41:12 : TRACE:  ..........pr_initialize_options: function entered
09/30 14:41:12 : TRACE:  ..........pr_initialize_clients: function entered
09/30 14:41:12 : TRACE:  ............pr_initialize_one_client: function entered
09/30 14:41:12 : TRACE:  ..............pr_initialize_boot_canon_append: function entered
09/30 14:41:12 : TRACE:  ..............pr_initialize_options: function entered
09/30 14:41:12 : TRACE:  ......pr_initialize_classes: function entered
09/30 14:41:12 : TRACE:  ......pr_initialize_classes:  Class IBM Network Station defined
09/30 14:41:12 : TRACE:  ........pr_initialize_one_class: function entered
09/30 14:41:12 : TRACE:  ..........pr_initialize_boot_canon_append: function entered
09/30 14:41:12 : TRACE:  ..........pr_initialize_options: function entered
09/30 14:41:12 : TRACE:  ....am_initialize: Function entered
09/30 14:41:12 : TRACE:  ......am_initMapper: function Entered
09/30 14:41:12 : TRACE:  ........am_initPool: function entered
09/30 14:41:12 : TRACE:  ..........pr_numAddresses: function entered
09/30 14:41:12 : TRACE:  ............countAddresses: function entered
09/30 14:41:12 : TRACE:  ..........pr_numClientAddresses: function entered
09/30 14:41:12 : TRACE:  ............countClients: function entered
09/30 14:41:12 : TRACE:  ..........pr_fillAddressRecords: function entered
09/30 14:41:12 : TRACE:  ..........pr_classifyAddressRecords: function entered
09/30 14:41:12 : INFO:   ......am_initMapper:  Previous map files not removed; try to
accomodate within new configuration
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.185 has been
adopted         2
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.186 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.187 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.188 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.189 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.190 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
>>>>>>>>>>>>>>>>>>>>>>>>> same for the addresses in between <<<<<<<<<<<
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.216 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.217 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
```

```
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.218 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.219 has been
adopted
09/30 14:41:12 : TRACE:  ........locateAddressRecord: function Entered
09/30 14:41:12 : TRACE:  ......am_initMapper:  Previous address 9.24.104.220 has been
adopted
09/30 14:41:12 : TRACE:  ..main:  Local Network Address found
09/30 14:41:12 : TRACE:  ..main:  IPC INIT Server done
09/30 14:41:13 : INFO:  ......getPortNum: dhcps/udp unknown service, assuming port 67
3
09/30 14:41:13 : TRACE:  ....initUserComm:  Opening usercomm socket.
09/30 14:41:13 : TRACE:  ....initUserComm: Binding usercomm socket, addr=0x00000000
port=0x03ae
09/30 14:41:13 : TRACE:  ....initUserComm:  Listening on usercomm port
09/30 14:41:13 : TRACE:  ..main:  Mailbox created

09/30 14:41:13 : TRACE:  ..main:  Garbage Collection started
: INFO:  EZZ7277 DHCP Server Initialized at Tue Sep 30 14:41:13 1997
4
09/30 14:43:06 : TRACE:  ....receiveMailbox:  select completed:
09/30 14:43:06 : SYSERR: ....receiveMailbox:  recvmsg got 548 bytes.
09/30 14:43:06 : TRACE:  ....receiveMailbox: SELECT_SEMAPHORE
09/30 14:43:06 : TRACE:  ..main:  Size of incoming packet is: 548
5
09/30 14:43:06 : TRACE:  ....process_bootrequest: function entered
09/30 14:43:06 : TRACE:  ....process_bootrequest: received packet xid = 1e4
09/30 14:43:06 : INFO:   ......primeOptions:  Option: 53, length:1
09/30 14:43:06 : INFO:   ......primeOptions:  Option: 57, length:2
09/30 14:43:06 : INFO:   ......primeOptions:  Option: 77, length:12
09/30 14:43:06 : INFO:   ......primeOptions:  Option: 60, length:19
09/30 14:43:06 : TRACE:  ......identifiableClient: function entered
09/30 14:43:06 : TRACE:  ......identifiableClient:  Using htype, hlen and chaddr to id
client
09/30 14:43:06 : TRACE:  ......legibleRequest: function entered
09/30 14:43:06 : TRACE:  ......legibleRequest:  DHCP msg type DHCPDISCOVER
6
09/30 14:43:06 : TRACE:  ....process_bootrequest:  Request is self-consistent
09/30 14:43:06 : TRACE:  ....reply_generator: function entered
09/30 14:43:06 : TRACE:  ......processDISCOVER: function entered
09/30 14:43:06 : TRACE:  ........locateExchange: function entered
09/30 14:43:06 : TRACE:  ........newExchangeBlock: function entered
09/30 14:43:06 : TRACE:  ........locateConfiguredClient: function entered
09/30 14:43:06 : TRACE:  ........addressManager: Function entered
09/30 14:43:06 : TRACE:  ..........am_queryClient: Function entered
09/30 14:43:06 : TRACE:  ............am_queryMapper: function Entered
09/30 14:43:06 : TRACE:  ..............locateClientRecord: function Entered
7
09/30 14:43:06 : TRACE:  ............am_queryMapper:  Cannot find client
6-0x0000e568afdf in client records
09/30 14:43:06 : TRACE:  ..........am_queryClient:  Client 6-0x0000e568afdf is not known
to address mapper, ask clientele
09/30 14:43:06 : TRACE:  ............cl_queryClientele: function entered
09/30 14:43:06 : TRACE:  ..............checkclient: function entered
8
09/30 14:43:06 : TRACE:  ............cl_queryClientele:  Client 6-0x0000e568afdf
authenticated by clientele list
09/30 14:43:06 : TRACE:  ......processDISCOVER: binder.subnet [0x00000000]
09/30 14:43:06 : TRACE:  ......processDISCOVER: AM_STATUS_AUTHENTIC
```

```
09/30 14:43:06 : TRACE:  ........addressManager: Function entered
09/30 14:43:06 : TRACE:  ..........am_reserve: Function entered
09/30 14:43:06 : TRACE:  ............am_addressClient: function Entered
09/30 14:43:06 : TRACE:  ............am_addressClient: pbinder->ipaddress [0x00000000]
09/30 14:43:06 : TRACE:  ............am_addressClient: pbinder->subnet [0x0918687d]
09/30 14:43:06 : TRACE:  ..............locateClientRecord: function Entered
09/30 14:43:06 : TRACE:  ..............newClientRecord: function Entered
09/30 14:43:06 : TRACE:  ............am_addressClient:  Garbage Collection started
09/30 14:43:06 : TRACE:  ............am_addressClient: pbinder->address [0x00000000]
09/30 14:43:06 : TRACE:  ............am_addressClient: pbinder->subnet [0x0918687d]
09/30 14:43:06 : TRACE:  ............am_addressClient: pcr->mssw 1082238115
09/30 14:43:06 : TRACE:  ..............indexAddressRecord: function Entered
09/30 14:43:06 : OBJERR: ..............indexAddressRecord: indexAddressRecord: Zero
Index
09/30 14:43:06 : TRACE:  ............am_addressClient: client address index 0
09/30 14:43:06 : TRACE:  ............am_addressClient: addrFolio.netClue = 0x0918687d
09/30 14:43:06 : TRACE:  ..............pr_queryAddr: function entered
09/30 14:43:06 : TRACE:  ..............pr_queryAddr: clue = [0x0918687d], 152594557
09/30 14:43:06 : TRACE:  ..............pr_queryAddr: netaddr = 9.0.0.0
09/30 14:43:06 : TRACE:  ..............pr_queryAddr: hostaddr = 0.24.104.125
09/30 14:43:06 : TRACE:  ..............pr_queryAddr:  Garbage Collection started
09/30 14:43:06 : INFO:   ............am_addressClient:  Client 6-0x0000e568afdf had no
previous mapping, getting one
09/30 14:43:06 : TRACE:  ..............newAddressRecord: function Entered
9
09/30 14:43:06 : TRACE:  ................isAddressRecordUsed: function Entered
09/30 14:43:06 : TRACE:  ..................isAddressInUse: Function Entered
09/30 14:43:09 : TRACE:  ..................isAddressInUse:  IP address 9.24.104.185, not
in use. rc=146692
09/30 14:43:09 : TRACE:  ..............indexAddressRecord: function Entered
10
09/30 14:43:10 : TRACE:  ..............nonvolatilizeAR: function Entered
09/30 14:43:10 : TRACE:  ..............nonvolatilizeCR: function Entered
09/30 14:43:10 : ACTION: ........addressManager:  Address 9.24.104.185 has been reserved
09/30 14:43:10 : TRACE:  ..........pr_new_menu : Function entered
09/30 14:43:10 : TRACE:  ..........pr_fill_menu_class: function entered
09/30 14:43:10 : TRACE:  ............pr_queryAddr: function entered
09/30 14:43:10 : TRACE:  ............pr_queryAddr: clue = [0x091868b9], 152594617
09/30 14:43:10 : TRACE:  ............pr_queryAddr: netaddr = 9.0.0.0
09/30 14:43:10 : TRACE:  ............pr_queryAddr: hostaddr = 0.24.104.185
09/30 14:43:10 : TRACE:  ............pr_queryAddr:  Garbage Collection started
09/30 14:43:10 : TRACE:  ............locateAddressRecord: function Entered
09/30 14:43:10 : TRACE:  ..........pr_fill_menu_net: function entered
09/30 14:43:10 : TRACE:  ............pr_queryAddr: function entered
09/30 14:43:10 : TRACE:  ............pr_queryAddr: clue = [0x091868b9], 152594617
09/30 14:43:10 : TRACE:  ............pr_queryAddr: netaddr = 9.0.0.0
09/30 14:43:10 : TRACE:  ............pr_queryAddr: hostaddr = 0.24.104.185
09/30 14:43:10 : TRACE:  ............pr_queryAddr:  Garbage Collection started
09/30 14:43:10 : TRACE:  ............locateAddressRecord: function Entered
09/30 14:43:10 : TRACE:  ..........pr_fill_menu_house: function entered
09/30 14:43:10 : TRACE:  ..........pr_fill_dish: function entered
09/30 14:43:10 : TRACE:  ............pr_new_menu : Function entered
09/30 14:43:10 : TRACE:  ............locateConfiguredClient: function entered
09/30 14:43:10 : TRACE:  ..............pr_queryAddr: function entered
09/30 14:43:10 : TRACE:  ..............pr_queryAddr: clue = [0x091868b9], 152594617
09/30 14:43:10 : TRACE:  ..............pr_queryAddr: netaddr = 9.0.0.0
09/30 14:43:10 : TRACE:  ..............pr_queryAddr: hostaddr = 0.24.104.185
09/30 14:43:10 : TRACE:  ..............pr_queryAddr:  Garbage Collection started
09/30 14:43:10 : TRACE:  ..............locateAddressRecord: function Entered
```

```
09/30 14:43:10 : TRACE:  ............locateConfiguredClient: first, check for this
client in the subnet scope
09/30 14:43:10 : TRACE:  ............locateConfiguredClient: look for client match in
this subnet    11
09/30 14:43:10 : TRACE:  ............locateConfiguredClient: found match !
09/30 14:43:10 : TRACE:  ............pr_queryAddr: function entered
09/30 14:43:10 : TRACE:  ............pr_queryAddr: clue = [0x091868b9], 152594617
09/30 14:43:10 : TRACE:  ............pr_queryAddr: netaddr = 9.0.0.0
09/30 14:43:10 : TRACE:  ............pr_queryAddr: hostaddr = 0.24.104.185
09/30 14:43:10 : TRACE:  ............pr_queryAddr:  Garbage Collection started
09/30 14:43:10 : TRACE:  ............locateAddressRecord: function Entered
09/30 14:43:10 : INFO:   ..........getPortNum:  dhcpc/udp unknown service, assuming port
68
09/30 14:43:10 : TRACE:  ..........newReplyPacket: function entered
09/30 14:43:10 : TRACE:  ..........enqueueExchange: function entered
09/30 14:43:10 : TRACE:  ......generate_bootreply: function entered
09/30 14:43:10 : INFO:  ......generate_bootreply: Generating a DHCPOFFER reply
12
09/30 14:43:10 : TRACE:  ......transmitMailbox:  transmitting to (255.255.255.255 #68)
09/30 14:43:15 : TRACE:  ......receiveMailbox:  select completed:
09/30 14:43:15 : SYSERR: ......receiveMailbox:  recvmsg got 548 bytes.
09/30 14:43:15 : TRACE:  ......receiveMailbox: SELECT_SEMAPHORE
09/30 14:43:15 : TRACE:  ....reply_generator:  Size of incoming packet is: 548
09/30 14:47:13 : TRACE:  ....reply_generator:  Garbage Collecting...
09/30 14:47:13 : TRACE:  ......event_timeout: function entered
09/30 14:47:13 : TRACE:  ......event_timeout:  Garbage collection (every 360 seconds).
09/30 14:47:13 : TRACE:  ........removeExpiredLeases: function Entered
09/30 14:47:13 : TRACE:  ........update_statistic_list: function Entered
09/30 14:47:13 : TRACE:  ....reply_generator:  Done.
```

E

FTP user exits and sample code

This appendix contains the source code of the sample FTP user exits, which were developed and tested at ITSO-Raleigh during installation and testing of the Communications Server for z/OS IP.

This appendix also includes two sample assembler macros used for various exit routines in the ITSO Raleigh environment, including the FTP security exits. The macros are called INIT and TERM and are used to generate assembler entry and exit code. See E.6, “assembler entry code (INIT MACRO)” on page 541 and E.7, “assembler exit code (INIT MACRO)” on page 544.

E.1 FTCHKIP security exit

```
*********************************************************************** 
*                                                                     * 
* Name:       FTCHKIP                                                 * 
*                                                                     * 
* Function:   Check if an FTP Client host may connect to this         * 
*             MVS FTP Server                                          * 
*                                                                     * 
* Interface:  R1 -> parameterlist with 6 pointers:                    * 
*             +0  -> fullword returncode (Out)                        * 
*             +4  -> fullword with value of 4 (In)                    * 
*             +8  -> fullword client IP address (In)                  * 
*             +12 -> fullword client port number (In)                 * 
*             +16 -> fullword local IP address (In)                   * 
*             +20 -> fullword local port number (In)                  * 
*                                                                     * 
* Logic:      If the foreign IP host belongs to our subnets,          * 
*             the connection will be allowed.                         * 
*             If not, this exit will reject it                        * 
*                                                                     * 
* Abends:     - none -                                                * 
*                                                                     * 
* Returncode: RC = 0: Accept the connection                           * 
*             RC = non zero: Reject the connection                    * 
*                                                                      *
* Written:    July 16'th 1992 at ITSC Raleigh                          *
*                                                                      *
* Changes:    May 11'th 1994 at ITSO Raleigh                           *
*             TCP/IP V3R1: No modifications required.                  *
*             Subnet and IP address values are changed.                *
*             Feb. 14'th 1998 at ITSO Raleigh                          *
*             CS for OS/390 V2R5: No modifications required.           *
*             Subnet and IP address values are changed.                *
*                                                                      *
************************************************************************
PARMS    DSECT
PTRRC    DC    F'0'
         DC    F'0'
PTRFIP   DC    F'0'
PTRFPORT DC    F'0'
PTRLIP   DC    F'0'
PTRLPORT DC    F'0'
*
FTCHKIP  INIT  'FTP Check IP adress of foreign host',                  C
               RENT=YES
*
* ---------------------------------------------------------------------
*
* The subnet mask is 255.255.255.0
* Allowed subnets are in table in this program.
*
* ---------------------------------------------------------------------
*
         LR    R2,R1                   *Parm pointer
         USING PARMS,R2
         L     R4,PTRRC                *-> Return code field
         L     R3,PTRFIP               *-> Foreign IP address
*****
*        B     ACCEPT                  ******Disbale this exit******
*****
         L     R3,0(R3)                *Foreign IP Address
         SRL   R3,8                    *Get rid of the 8 loworders
         SLL   R3,8                    *Back into line again
         LM    R5,R7,OURBXLE           *Adresses for net loop
NETLOOP  EQU   *
         C     R3,0(R5)                *One of our subnets ?
         BE    ACCEPT                  *- Yes, accept connection
         BXLE  R5,R6,NETLOOP           *Loop through all subnets
         SRL   R3,24                   *only first byte
         C     R3,=A(10)               *All 10 network addresses are OK
         BE    ACCEPT
         LA    R15,16                  *Not allowed to connect
         B     DONE
ACCEPT   EQU   *
         SR    R15,R15                 *Zero RC means accept it
DONE     EQU   *
         ST    R15,0(R4)               *Return the RC
         TERM
         LTORG
OURBXLE  DC    A(OURSUB,4,OURSUBSL-4)
OURSUB   EQU   *
         DC    AL1(9,67,38,0)                 *9.67.38.0 (TR)
         DC    AL1(9,67,32,0)                 *9.67.38.0 (Ethernet)
         DC    AL1(9,24,104,0)                *9.24.104.0 (Production)
         DC    AL1(9,24,103,0)                *9.24.103.0 MPN dialin    00181100
         DC    AL1(9,67,41,0)                 *9.24.41.0 (MVS03 link)   00181200
         DC    AL1(9,67,46,0)                 *9.24.46.0 WAN FR link    00181300
         DC    AL1(9,67,51,0)                 *9.24.51.0 3174 private   00181400
         DC    AL1(9,12,11,0)                 *9.12.11.0 (new POK link) 00181500
OURSUBSL EQU   *                                                        00182000
         END                                                            00190000
```

E.2 FTCHKPWD security exit

```
***********************************************************************
*                                                                     *
* Name:       FTCHKPWD                                                *
*                                                                     *
* Function:   Check if we will allow a given user to log on to the    *
*             MVS FTP Server                                          *
*                                                                     *
* Interface:  R1 -> parameterlist with 4 pointers:                    *
*             +0  -> fullword returncode (Out)                        *
*             +4  -> fullword with value of 2 (In)                    *
*             +8  -> 8 bytes user id (In)                             *
*             +12 -> 8 bytes password (In)                            *
*                                                                     *
* Logic:      If the user is in our hardcoded include list, the log   *
*             on will be accepted.                                    *
*             If not, this exit will reject it                        *
*                                                                     *
* Abends:     - none -                                                *
*                                                                     *
* Returncode: RC = 0: Accept the log on                               *
*             RC = non zero: Reject the log on                        *
*                                                                     *
* Written:    July 16'th 1992 at ITSC Raleigh                         *
*                                                                     *
* Changed:    May 11'th 1994 at ITSO Raleigh                          *
*             TCP/IP V3R1: No modifications required.                 *
*             Allowed userIDs updated.                                *
*             Feb. 14'th 1998 at ITSO Raleigh                         *
*             CS for OS/390 V2R5: No modifications required.          *
*             Allowed userIDs updated.                                *
*                                                                     *
***********************************************************************
PARMS    DSECT
PTRRC    DC    F'0'
         DC    F'0'
PTRUSER  DC    F'0'
PTRPW    DC    F'0'
*
USERID   DSECT
USER     DC    CL8' '
*
FTCHKPWD INIT  'FTP Check if user is acceptable',                      C
               RENT=YES
*
         LR    R2,R1                   *Parm pointer
         USING PARMS,R2
         L     R4,PTRRC                *-> Return code field
         L     R3,PTRUSER              *-> 8 byte user id
*
```

```
* Accept all
*
*        B     ACCEPT                 *****Disable this exit********
*
         LM    R5,R7,USERBXLE         *BXLE for user lookup
         USING USERID,R5
LOOKALL  EQU   *
         CLC   USER,0(R3)             *Is user in our positive list?
         BE    ACCEPT                 *- Yes, accept the log on
         BXLE  R5,R6,LOOKALL          *Search all entries
DROPIT   EQU   *                      *If not, reject the log on
         LA    R15,16                 *Any value which is not zero
         B     DONE
ACCEPT   EQU   *
         SR    R15,R15                *Zero RC means accept it
DONE     EQU   *
         ST    R15,0(R4)              *Return the RC
         TERM
         LTORG
*
USERBXLE DC    A(USERSTRT,L'USER,USEREND-L'USER)
USERSTRT EQU   *
         DC    CL8'WOZA'
         DC    CL8'GDENTED'
         DC    CL8'SILVIAR'
         DC    CL8'EIKENS'
         DC    CL8'KAKKY'
         DC    CL8'FTPANOM'           *Anonymous userID
USEREND  EQU   *
*
         END
```

E.3 FTCHKCMD security exit

```
*********************************************************************
*                                                                   *
* Name:         FTCHKCMD                                            *
*                                                                   *
* Function:     This exit will reject any FTP Delete command.       *
*               It will further reject an attempt to change password *
*               from the anonymous userID (FTPANOM).                *
*                                                                   *
* Interface:    R1 -> parameterlist with 5 pointers:                *
*               +0  -> fullword returncode (Out)                    *
*               +4  -> fullword with value of 3 (In)                *
*               +8  -> 8 bytes user id (In)                         *
*               +12 -> 8 bytes command name (In)                    *
*               +16 -> buffer with argument string, 2 first bytes   *
*                      holds length of remaining buffer.            *
*                                                                   *
* Logic:        If the command is DELE, the request will            *
*               be rejected no matter who the user is,              *
*               If the command is a PASS command from user FTPANOM, the *
*               command buffer is searched for an '/' - if one is found *
*               the command is rejected.                            *
*                                                                   *
* Abends:       - none -                                            *
*                                                                   *
* Returncode: RC = 0: Accept the command                            *
*             RC = non zero: Reject the command                     *
*                                                                   *
* Written:    July 16'th 1992 at ITSC Raleigh                       *
*                                                                   *
* Changed:    May 11'th 1994 at ITSO Raleigh                        *
*             TCP/IP V3R1: No modifications required.               *
*                                                                   *
*             Oct 17'th 1994 at ITSO Raleigh                        *
*             Check for FTPANOM PASS command implemented.           *
*                                                                   *
*********************************************************************
PARMS    DSECT
PTRRC    DC    F'0'
         DC    F'0'
PTRUSER  DC    F'0'
PTRCMD   DC    F'0'
PTRBUF   DC    F'0'
*
BUFFER   DSECT
BUFLEN   DC    AL2(0)
BUFDATA  DS    0C
*
CMDENTRY DSECT
CMDNAME  DC    CL8' '                  *Command name
CMDSTAT  DC    AL1(0)                  *Command status
CMDENTRL EQU   *-CMDNAME               *L'Command table entry
*
WORKAREA DSECT
         DC    18F'0'                  *Save area
LCCMD    DC    CL8' '                  *Command name in lower case
*
FTCHKCMD INIT  'FTP Reject any DELETE command',                     C
               RENT=YES,WORKLEN=512
*
         USING WORKAREA,R13            *Adressability work extension
         LR    R2,R1                   *Parm pointer
         USING PARMS,R2
         L     R4,PTRRC                *-> Return code field
         L     R3,PTRCMD               *-> 8 byte command name
         MVC   LCCMD,0(R3)             *It may be lower case (QUOTE)
         OC    LCCMD,=8B'01000000'     *Be sure it is upper-case
*
* -------------------------------------------------------------------
*
* If command is PASS and user id FTPANOM, we parse the command buffer
* to see if any '/'s are present.  If an anonymous user tries to change
* the password of the FTPANOM user, we will reject the command.
*
* -------------------------------------------------------------------
*
         CLC   LCCMD,=CL8'PASS'        *Password ?
         BNE   LOOKCMD                 *- No, normal process
         L     R1,PTRUSER              *-> UserID
         CLC   0(8,R1),=CL8'FTPANOM'   *Anonymous FTP User ?
         BNE   LOOKCMD                 *- No, normal process
         L     R7,PTRBUF               *-> Command buffer
         USING BUFFER,R7
         LH    R9,BUFLEN               *L'Buffer data
         LA    R7,BUFDATA              *Start of buffer data
         DROP  R7
         LA    R8,1                    *Scan one char at a time
         AR    R9,R7                   *First byte after buffer
         BCTR  R9,0                    *Last byte of buffer
LOOKNPW  EQU   *
         CLI   0(R7),C'/'              */ means he/she tries new pw
         BE    DROPIT                  *Not allowed for FTPANOM
         BXLE  R7,R8,LOOKNPW           *Scan whole buffer
*
* ---------------------------------------------------------------------
*
* The passed command is checked against the command table.  If the
* table disables the command, it is rejected - if not, it is
* accepted.
*
* ---------------------------------------------------------------------
*
LOOKCMD  EQU   *
         LM    R5,R7,CMDSBXLE          *BXLE for command lookup
         USING CMDENTRY,R5
LOOKALL  EQU   *
         CLC   CMDNAME,LCCMD           *Is this our command ?
         BE    LOOKSTAT                *- Yes, look at status
         BXLE  R5,R6,LOOKALL           *Search all entries
         B     DROPIT                  *If not found, we drop it.
LOOKSTAT EQU   *
         CLI   CMDSTAT,1               *Is command enabled?
         BE    ACCEPT                  *- Yes, Accept it
DROPIT   EQU   *                       *- No, reject the command
         LA    R15,16                  *Any value which is not zero
         B     DONE
ACCEPT   EQU   *
         SR    R15,R15                 *Zero RC means accept it
DONE     EQU   *
         ST    R15,0(R4)               *Return the RC
         TERM
         LTORG
*
* ---------------------------------------------------------------------
*
* This table includes all possible FTP commands (as listed in
* RFC765) - they may not all be implemented in the MVS FTP Server.
*
* Each entry in the tables consists of
*
* - 8 bytes command name
* - 1 byte status code of 0: Not allowed or 1: Allowed
*
* At the ITSC Raleigh site, we have disabled DELETE and RENAME
* functions.
*
* ---------------------------------------------------------------------
*
CMDSBXLE DC    A(CMDSSTRT,CMDENTRL,CMDSEND-CMDENTRL)
CMDSSTRT EQU   *
*                                      *Access Control Commands
         DC    CL8'USER',AL1(1)        *User name
         DC    CL8'PASS',AL1(1)        *Password
         DC    CL8'ACCT',AL1(1)        *Account
         DC    CL8'REIN',AL1(1)        *Reinitialize
         DC    CL8'QUIT',AL1(1)        *Log out
*                                      *Transfer parameter commands
         DC    CL8'PORT',AL1(1)        *Data Port
         DC    CL8'PASV',AL1(1)        *Passive
         DC    CL8'TYPE',AL1(1)        *Type
         DC    CL8'STRU',AL1(1)        *File Structure
         DC    CL8'MODE',AL1(1)        *Transfer mode
*                                      *FTP service commands
         DC    CL8'MKD ',AL1(1)        *Make Directory (*)
         DC    CL8'XMKD',AL1(1)        *Make Directory (*)
         DC    CL8'RETR',AL1(1)        *Retrieve
         DC    CL8'STOR',AL1(1)        *Store
         DC    CL8'APPE',AL1(1)        *Append (with create)
         DC    CL8'MLFL',AL1(1)        *Mail File
         DC    CL8'MAIL',AL1(1)        *Mail
         DC    CL8'MSND',AL1(1)        *Mail send to term
         DC    CL8'MSOM',AL1(1)        *Mail send to term or mailbox
         DC    CL8'MSAM',AL1(1)        *Mail send to term and mailbox
         DC    CL8'MRSQ',AL1(1)        *Mail recipient scheme question
         DC    CL8'MRCP',AL1(1)        *Mail recipient
         DC    CL8'ALLO',AL1(1)        *Allocate
         DC    CL8'REST',AL1(1)        *Restart
         DC    CL8'RNFR',AL1(1)        *Rename from
         DC    CL8'RNTO',AL1(1)        *Rename to
         DC    CL8'ABOR',AL1(1)        *Abort
         DC    CL8'DELE',AL1(0) DISABL *Delete
         DC    CL8'PWD ',AL1(1)        *Print working directory
         DC    CL8'XPWD',AL1(1)        *Print working directory
         DC    CL8'CWD ',AL1(1)        *Change working directory
         DC    CL8'XCWD',AL1(1)        *Change working directory
         DC    CL8'LIST',AL1(1)        *List
         DC    CL8'NLST',AL1(1)        *Name-list
         DC    CL8'SITE',AL1(1)        *Site parameters
         DC    CL8'SYST',AL1(1)        *System information
         DC    CL8'STAT',AL1(1)        *Status
         DC    CL8'HELP',AL1(1)        *Help
         DC    CL8'NOOP',AL1(1)        *No Operation
         DC    CL8'STOU',AL1(1)        *Store Unique
CMDSEND  EQU   *
         END
```

E.4 FTCHKJES security exit

```
***********************************************************************
*                                                                     *
* Name:       FTCHKJES                                                *
*                                                                     *
* Function:   We will only allow users to submit jobs with their      *
*             own user id.                                            *
*                                                                     *
* Interface:  R1 -> parameterlist with 9 pointers:                    *
*             +0  -> fullword returncode (Out)                        *
*             +4  -> fullword with value of 7 (In)                    *
*             +8  -> 8 bytes user id (In)                             *
*             +12 -> buffer with one JCL statement (In)               *
*             +16 -> fullword with length of buffer (In)              *
*             +20 -> fullword with client JESLrecl (In)               *
*             +24 -> fullword with buffer number (In)                 *
*            +28 -> fullword with unique client id (In)            *
*            +32 -> fullword with JESRecfm (In)                    *
*                   JESRecfm = 0: Fixed length                     *
*                   JESRecfm = 1: Variable length                  *
*            +34 -> 4 bytes client specific work area              *
*                                                                  *
* Logic:     This exit will be called once for every logical       *
*            line (input record) with JCL data.                    *
*            If recordformat is fixed, the buffersize equals       *
*            JESLrecl. The JCL data is padded with spaces.         *
*            If recordformat is variable, the format of the        *
*            buffer is 4 bytes rdw (2 bytes length, 2 bytes 0)     *
*            followed by data (LLzzdata.....). Length includes     *
*            the 4 byte rdw.                                       *
*                                                                  *
*            Scan max 20 buffers (JOB statement assumed to be      *
*            within the first 20 records). Look for USER=          *
*                                                                  *
*            Compare the USER= value with the FTP Client user id   *
*            passed on the call to this exit routine.              *
*                                                                  *
*            Characters have been translated to EBCDIC.            *
*            At the time, the exit is driven, no userid or password *
*            has been put into the buffer - unless they were       *
*            supplied by the client. The exit must be              *
*            able to distinguish between the following situations: *
*                                                                  *
*            * JOB statement does not include any USER=  keyword - *
*              the client user id and password will be             *
*              propagated at submission time, which will be OK.    *
*                                                                  *
*            * The JOB  statement does include USER= keyword and   *
*              contents of this must be checked against the client's *
*              log on user id.                                     *
*                                                                  *
* Restric-   This exit only checks the first JOB stmt. in the file *
* tions:     that is about to be submitted to JESx.                *
*                                                                  *
* Abends:    - none -                                              *
*                                                                  *
* Returncode: RC = 0: Accept the job submission                    *
*            RC = non zero: Reject submission of the job           *
*                                                                  *
* Written:   July 16'th 1992 at ITSC Raleigh                       *
*                                                                  *
* Changed:   May 11'th 1994 at ITSO Raleigh                        *
*            TCP/IP V3R1: Interface changed. Code rewritten to use *
*                         the provided 4-bytes work area.          *
*                                                                  *
*            Feb 2'nd 1995 at ITSO Relaigh                         *
*                         Added logic to replace first comment     *
*                         stmt. with another (test) comment        *
*                         stmt.  Works only for JESRecfm fixed.    *
*                                                                  *
*            Sep 4'th 1996 at ITSO Raleigh                         *
*                         Changed the option from Feb 2, 1995 to be *
*                         optional.  Default is that the exit does *
*                         not modify any comment statements.  By   *
*                         removing the identified comment character *
*                         for an MVC statement, the exit will work *
*                          as described above.                      *
*                                                                   *
*             Jan 28 1997 at ITSO Raleigh                           *
*                          Added a WTO to inform on MVS joblog when   *
*                          this exit rejects a job because of misuse  *
*                          of USER= keyword in JOB card.              *
*                                                                   *
*********************************************************************
PARMS    DSECT
PTRRC    DC    F'0'                   *-> Returncode fullword
         DC    F'0'
PTRUSER  DC    F'0'                   *-> user id
PTRBUF   DC    F'0'                   *-> buffer
PTRBUFLN DC    F'0'                   *-> word with length of buffer
PTRJESLR DC    F'0'                   *-> word with JESLrecl
PTRBUFNO DC    F'0'                   *-> word with buffer number
PTRCLNID DC    F'0'                   *-> word with client id
PTRJESRF DC    F'0'                   *-> word with JESRecfm
PTRJWORK DC    F'0'                   *-> work word
*
USERID   DSECT
USER     DC    CL8' '
*
WORKAREA DSECT
         DC    18F'0'                 *Save areas
JOBUSER  DC    CL8' '                 *USER value from JOB card
WTOWORK  DS    0F                     *WTO Message build area
         DC    XL4'00',C'FTCHKJES - '  *Placeholder
         DC    C'Job from FTP userID=' *Placeholder
WTOUSER  DC    CL8' '                 *FTP User ID
         DC    C' cancelled by exit'   *Placeholder
         DC    XL4'00'                *Placeholder
*
EXITWORD DSECT
LASTNONB DC    CL1' '                 *Last non-blank
STATBYTE DC    X'00'                  *Scan status byte
STATJOB  EQU   BIT0                   *JOB card in process
STATCONT EQU   BIT1                   *JOB Card is continuing
STATUSER EQU   BIT2                   *USER keyword has been found
STATSTOP EQU   BIT3                   *Stop further SCAN
APOST    EQU   BIT4                   *Within apostrophies
MODDONE  EQU   BIT5                   *//* Modification is done
*
FTCHKJES INIT  'FTP Check if user id match JOB statement',             C
               RENT=YES,WORKLEN=512
*
         USING WORKAREA,R13
         LR    R2,R1                  *Parm pointer
         USING PARMS,R2
         L     R11,PTRJWORK           *-> 4 bytes work area
         USING EXITWORD,R11
         L     R4,PTRRC               *-> Return code field
         L     R3,PTRUSER             *-> 8 byte user id
         L     R5,PTRBUFNO            *-> Buffer number
         L     R5,0(R5)               *Buffer number
         CH    R5,=AL2(1)             *First buffer ?
         BH    NOTFIRST               *- No, preserve work word
         XC    STATBYTE,STATBYTE      *Initialize status bits
NOTFIRST EQU   *
         CH    R5,=AL2(20)            *Max scan 20 first buffers
         BL    BELOW20
         OI    STATBYTE,STATSTOP        *Stop further scanning
BELOW20  EQU   *
         TM    STATBYTE,MODDONE         *Have we modified first //* ?
         BO    MODOK                    *- Yes, that has been done
         L     R5,PTRBUF                *-> Start of buffer
         L     R1,PTRJESRF              *-> word with JESRecfm
         L     R1,0(R1)                 *JESRecfm code
         LTR   R1,R1                    *zero means Fixed
         BZ    MODFIXED                 *Registers are OK
         WTO   'FTCHKCMD - JESRecfm is not fixed'
         B     MODSET                   *We skip processing
MODFIXED EQU   *
         CLC   0(3,R5),=CL3'//*'        *Is this a comment stmt ?
         BNE   MODOK                    *- No, we can't use it
*
* -------------------------------------------------------------------
* Remove the comment character in the following MVC Statement and
* replace the literal with your own value, if you want this
* exit to replace the first comment card in this job
* with your installation standard value.
* With the comment character in pos one, this exit does NOT
* modify any comment statements.
* -------------------------------------------------------------------
*
*        MVC   0(80,R5),=CL80'//* Our modified comment stmt.'
*
MODSET   EQU   *
         OI    STATBYTE,MODDONE         *Modify is now done
MODOK    EQU   *
         TM    STATBYTE,STATSTOP        *Job stmt. scanning terminated ?
         BO    ACCEPT                   *- Yes, just return
*
* -------------------------------------------------------------------
*
* The buffer holds a JCL statement with // in col 1 and 2 - or
* SYSIN data.
*
* The code will look for the word JOB in the first valid JCL statement.
*
* If the word JOB is found the code scans for a USER= keyword until
* either a hit is made or the job card statement terminates (It may
* continue over more records).
* Proper considerations to avoid scanning text within apostrophies
* are taken.
*
* -------------------------------------------------------------------
*
         L     R5,PTRBUF                *-> Start of buffer
         LA    R6,1                     *Scan one byte at a time.
         L     R7,PTRBUFLN              *-> fullword w. buffer length
         L     R7,0(R7)                 *L'buffer
         L     R1,PTRJESRF              *-> word with JESRecfm
         L     R1,0(R1)                 *JESRecfm code
         LTR   R1,R1                    *zero means Fixed
         BZ    RECFMFIX                 *Registers are OK
         LA    R5,4(R5)                 *Skip rdw
         SH    R7,=AL2(4)               *Reduce length
RECFMFIX EQU   *
         AR    R7,R5                    *-> First byte after scan area
         BCTR  R7,0                    *-> Last byte to scan
         MVC   JOBUSER,=CL8' '         *Init to space
         MVI   LASTNONB,C' '           *Set to blank
         LR    R8,R5                   *Save original start
         LR    R9,R7                   *Save original end
         TM    STATBYTE,STATJOB        *Are we scanning JOB card
         BO    JOBCONT                 *- Yes, continued job card
         CLC   0(2,R5),=CL2'/*'        *EOF OK
         BE    NEXTREC
         CLC   0(3,R5),=CL3'//*'       *Comments as well
         BE    NEXTREC
         CLC   0(3,R5),=CL3'// '       *Just flushing
         BE    NEXTREC
         CLC   0(2,R5),=CL2'//'        *Must be JCL stmt.
         BNE   NEXTREC                 *Must be there somewhere..
         LR    R7,R5                   *Start of record
         LA    R7,15(R7)               *JOB Must be within first 16
         CR    R7,R9                   *Avoid scanning after buffer
         BNH   JOBLOOP                 *OK
         LR    R7,R9                   *Else reduce to end-buffer
JOBLOOP  EQU   *
         CLC   0(5,R5),=CL5' JOB '     *Is this a JOB Card?
         BE    JOBFOUND                *- Yes, start scan for USER=
         BXLE  R5,R6,JOBLOOP           *Look for JOB
         B     NEXTREC                 *Ask for next buffer
JOBFOUND EQU   *
         OI    STATBYTE,STATJOB        *We found a job card
JOBCONT  EQU   *
         LR    R5,R8                   *-> First byte in record
         LR    R7,R9                   *Only scan buffer
         LA    R6,1                    *Advance one byte
USLOOK1  EQU   *
         CLI   0(R5),C' '              *Is this a blank?
         BE    US1NEXT                 *- Yes, just advance
         MVC   LASTNONB,0(R5)          *Save for Cont. test
         TM    STATBYTE,APOST          *Within apostrophies?
         BO    LOOKAPOS                *We will only recog. ending apos
         CLC   0(5,R5),=CL5'USER='     *USER= keyword?
         BE    USERFND                 *- yes, we've got it!
         CLI   0(R5),C''''             *Starting apost ?
         BNE   US1NEXT                 *- No, just advance
         OI    STATBYTE,APOST          *We are now within aposts
         B     US1NEXT                 *Advance
LOOKAPOS EQU   *
         CLI   0(R5),C''''             *Ending apost ?
         BNE   US1NEXT                 *Advance
         NI    STATBYTE,AB-APOST       *We are now outside apost
US1NEXT  EQU   *
         BXLE  R5,R6,USLOOK1
         CLI   LASTNONB,C','           *Was last a cont. mark?
         BE    NEXTREC                 *- Yes, let's have next buffer
         OI    STATBYTE,STATSTOP       *- No, just stop scanning
NEXTREC  EQU   *
         SR    R15,R15                 *No reason to complain
         B     DONE                    *Give us next buffer
*
* ----------------------------------------------------------------------
*
* We have found the USER= keyword, Now just copy the value
*
```

```
* --------------------------------------------------------------------
*
USERFND  EQU   *
         LA    R5,5(R5)              *Past USER=
         LA    R8,JOBUSER            *Copy value here
US2LOOK  EQU   *
         CLI   0(R5),C' '            *A blank terminates value
         BE    US2DONE
         CLI   0(R5),C','            *So does a comma
         BE    US2DONE
         MVC   0(1,R8),0(R5)         *Move one byte at a time
         LA    R8,1(R8)              *Advance target pointer
         BXLE  R5,R6,US2LOOK         *Advance source pointer
*
* --------------------------------------------------------------------
*
* Check the value of USER= keyword against the client's
* FTP Log on user id.
*
* --------------------------------------------------------------------
*
US2DONE  EQU   *
         CLC   JOBUSER,0(R3)         *USER= equals log on ID ?
         BE    ACCEPT                *- Yes, accept the job sub.
DROPIT   EQU   *                     *If not, reject the log on
         MVC   WTOWORK(WTOLISTL),WTOLIST
         L     R15,PTRUSER           *->FTP user ID
         MVC   WTOUSER,0(R15)        *Move in user ID
         WTO   MF=(E,WTOWORK)        *Put message to log
         LA    R15,4                 *Any value which is not zero
         B     DONE
ACCEPT   EQU   *
         OI    STATBYTE,STATSTOP     *No more scanning needed
         SR    R15,R15               *Zero RC means accept it
DONE     EQU   *
         ST    R15,0(R4)             *Return the RC
         TERM
         LTORG
WTOLIST  WTO   'FTCHKJES - Job from FTP userID=xxxxxxxx cancelled by exC
               it',MF=L
WTOLISTL EQU   *-WTOLIST
         END
```

E.5 FTP RDW post process sample program

```
**********************************************************************
*                                                                    *
* Name:       FTPRDW                                                 *
*                                                                    *
* Function:   Reconstruct a RECFM=V or VB Data Set from a RECFM=U    *
*             Data Set - holding TCP/IP FTP RDW information.         *
*             Use would be:                                          *
*               1. Transfer a V or VB Data set with FTP using        *
*                  Binary, Stream and RDW option - to a TCP/IP host  *
*                  with a Stream file system.                        *
*               2. Transfer the data set back to MVS with FTP using  *
*                  Binary  Stream and MVS SITE RECFM=U BLKSIZE=some  *
*                  high value                                        *
*               3. Run this program specifying the RECFM=U data set  *
*                  as input and an output data set with the same      *
*                  DCB information as the original data set.          *
*                                                                     *
* Interface:  DD-name: INPUT   The RECFM=U Data Set                   *
*             DD-name: OUTPUT  The reconstructed V or VB Data Set     *
*             DD-name: PRINT   Error and informational messages       *
*                                                                     *
* Logic:      The RECFM=U Data set could look like                    *
*             Block 1: rdw1,data1,rdw2,data2                          *
*             Block 2: data2,rdw3,data3,rdw4                          *
*             Block 3: data4  ,rdw5,data5,rd                          *
*             Block 4: w6,data.                                       *
*             Read the RECFM=U Data Set using BSAM - assemble the     *
*             individual records and write them to the RECFM V or     *
*             VB data set using QSAM.                                 *
*                                                                     *
* Restric-    If the format of the data set does not adhere to the    *
* tions:      format constructed by MVS FTP using the SITE option     *
*             RDW - results may be unpredictable.                     *
*                                                                     *
* Abends:     - none -                                                *
*                                                                     *
* Returncode: RC = 0:  Reconstruction performed without errors        *
*             RC = 16: No output data set written, errors encountered *
*                      See the PRINT file                             *
*                                                                     *
* Written:    July 20'th 1992 at ITSC Raleigh                         *
*                                                                     *
***********************************************************************
         PRINT NOGEN
         DCBD  DSORG=PS
FTPRDW   INIT 'Recreate Record boundaries for VB dataset',             C
               MODE=24
*
* ---------------------------------------------------------------------
*
*  Open DCB's
*
*  Verify that input data set is RECFM=U, Obtain Blksize and
*  GETMAIN storage for one buffer.
*
*  Verify that output data set is RECFM V or VB, Obtain Max. Record
*  Length and GETMAIN storage to build one output record.
*
* ---------------------------------------------------------------------
*
         XC    RC,RC                    *Start optimistic!
         OPEN  (UDCB,(INPUT),VBDCB,(OUTPUT),PRINT,(OUTPUT))
         LA    R10,UDCB                 *RECFM=U DCB
         USING IHADCB,R10
         TM    DCBRECFM,DCBRECU         *Should be RECFM=U
         BO    INPUTISU                 *- Yes, it is.
         LA    R2,MSG01                 *- No, it's not
         MVC   RC,=A(16)                *Bad return
         B     ERROR
INPUTISU EQU   *
         LH    R2,DCBBLKSI              *We need BLKSIZE for buffer
         ST    R2,BUFFERL
         GETMAIN R,LV=(R2)              *Getmain a buffer
         ST    R1,BSTART                *Here our buffer is
         LA    R10,VBDCB                *OUTPUT DCB
         TM    DCBRECFM,DCBRECV         *Must be RECFM=V/VB
         BO    OUTISV                   *- Yes, It is
         LA    R2,MSG02                 *- No, it is not
         MVC   RC,=A(16)                *Bad return
         B     ERROR
OUTISV   EQU  *
         LH    R2,DCBLRECL              *OUTPUT max record length
         LTR   R2,R2                    *Must be GT 0
         BP    RECLOK                   *It is
         LA    R2,MSG03                 *Unlikely.
         MVC   RC,=A(16)                *Bad return
         B     ERROR
RECLOK   EQU   *
         ST    R2,RECORDL               *Room for max. reclen.
         GETMAIN R,LV=(R2)              *Getmain record buffer
         ST    R1,RECORDAD              *Here to build output record
         ST    R1,OUTP                  *Output buffer Pointer
*
* ----------------------------------------------------------------------
*
*  Main Program logic
*
* ----------------------------------------------------------------------
*
         BAL   R14,READBLOK             *Get a block of data
NEXT     EQU   *
         L     R2,BP                    *Current buffer pointer
         LA    R2,4(R2)                 *Past RDW
         C     R2,BEND                  *Partial RDW in this block ?
         BH    PARTRDW                  *- Yes, special attention!
         L     R2,BP
         MVC   ARDW,0(R2)               *Save Actual RDW
         LA    R2,4(R2)                 *Advance past RDW
         ST    R2,BP                    *New position
         B     GOTARDW
PARTRDW  EQU   *
         L     R2,BEND                  *End of buffer
         S     R2,BP                    *Remaining bytes in buffer
         ST    R2,LEN                   *We may need it..
         LTR   R2,R2                    *Anything left?
         BNZ   PARTRDWD                 *- Yes, partial RDW is present
         BAL   R14,READBLOK             *Read next block
         B     NEXT                     *Process next block
RDWMVC   MVC   0(*-*,R3),0(R4)          *Move partial RDW
PARTRDWD EQU   *
         L     R2,LEN                   *So many bytes
         BCTR  R2,0                     *Ready for Execute an MVC
         LA    R3,ARDW                  *Here to move RDW
         L     R4,BP                    *Here to move from
         EX    R2,RDWMVC                *Move partial RDW
         BAL   R14,READBLOK             *Get next block
         LA    R2,4                     *RDW Length
         S     R2,LEN                   *Remains to move
         LR    R5,R2                    *We need it to advance BP.
         BCTR  R2,0                     *For execute
         LA    R3,ARDW                  *
         A     R3,LEN                   *Here to place remaining RDW
         L     R4,BP                    *Here to move from
         EX    R2,RDWMVC                *Move remaining part
         A     R5,BP                  *Advance Buffer pointer
         ST    R5,BP                  *Ready for further process
GOTARDW  EQU   *
         LH    R3,ARDW                *Length in RDW
         LTR   R3,R3                  *Zero length RDW
         BNP   SLUT                   *We are Done.
         SH    R3,=AL2(4)             *Excluding RDW
         ST    R3,ALEN                *This is L'record
         L     R3,OUTP                *Here to build output rec
         MVC   0(L'ARDW,R3),ARDW      *Copy RDW to output record
         LA    R3,4(R3)               *Advance output pointer
         ST    R3,OUTP
*
BUILD    EQU   *
*
         CLC   BP,BEND                *Anything left in buffer ?
         BL    SOMELEFT               *- Yes, some data is left.
         BAL   R14,READBLOK           *- No, Read next block.
         B     BUILD                  *Check again..
SOMELEFT EQU   *
         L     R2,BP                  *Buffer pointer
         A     R2,ALEN                *Plus rest of this record
         C     R2,BEND                *Is it all in the buffer ?
         BH    PARTREC                *Only a part in this buffer
         L     R5,ALEN                *L'records data
         LR    R7,R5                  *From and To length equal
         L     R4,OUTP                *Move to
         L     R6,BP                  *Move From
         MVCL  R4,R6                  *Move records data part
         L     R2,RECORDAD            *Here is record
         PUT   VBDCB,(R2)             *Write output record
         L     R2,RECOUT              *Number of records written
         LA    R2,1(R2)               *Plus One
         ST    R2,RECOUT              *
         MVC   OUTP,RECORDAD          *Reset Output pointer
         L     R2,ALEN                *So much used from buffer
         A     R2,BP                  *Advance buffer pointer
         ST    R2,BP
         B     NEXT                   *Process next record
PARTREC  EQU   *
         L     R5,BEND                *End of buffer
         S     R5,BP                  *Move what remains in buffer
         LR    R7,R5                  *From and To length equals
         LR    R3,R5                  *We need it..
         L     R4,OUTP                *Move To
         L     R6,BP                  *Move From
         MVCL  R4,R6                  *Move partial record
         L     R2,OUTP                *First part went here
         AR    R2,R3                  *Next part goes here
         ST    R2,OUTP                *Advance output pointer
         L     R2,ALEN                *L'whole record
         SR    R2,R3                  *Remaining length
         ST    R2,ALEN                *This remains to be moved
         BAL   R14,READBLOK           *Get next block
         B     BUILD                  *Process remaining part
*
* ---------------------------------------------------------------------
*
*  Read a Block of data and initialize buffer variables
*
```

```
* ----------------------------------------------------------------------
*
READBLOK EQU   *
*
         ST    R14,SAVER14             *Remember return address.
         L     R11,BSTART              *Read into buffer
         READ  MYDECB,SF,UDCB,(R11),'S' *Read one block
         CHECK MYDECB                  *Wait for completion
         LA    R11,MYDECB              *-> DECB
         L     R11,16(R11)             *-> IOB
         L     R14,BUFFERL             *Max blksize
         SH    R14,14(R11)             *Minus residual from IOB
         ST    R14,BUFLEN              *Gives actual blk length
         A     R14,BSTART              *plus buffer start
         ST    R14,BEND                *Gives end of buffer
         MVC   BP,BSTART               *Reset current buffer pointer
         L     R14,RECIN               *Number of records read
         LA    R14,1(R14)              *Plus one more
         ST    R14,RECIN
         L     R14,SAVER14             *Restore return address
         BR    R14                     *Back to main logic
*
ERROR    EQU   *
         PUT   PRINT,(R2)
SLUT     EQU   *
         L     R2,RECIN                *Blocks read
         CVD   R2,DORD
         OI    DORD+7,X'0F'
         UNPK  MSG04NO,DORD            *For message
         PUT   PRINT,MSG04
         L     R2,RECOUT               *Records written
         CVD   R2,DORD
         OI    DORD+7,X'0F'
         UNPK  MSG05NO,DORD            *For message
         PUT   PRINT,MSG05
         CLOSE (UDCB,,VBDCB,,PRINT)
         L     R2,BSTART
         L     R3,BUFFERL
         FREEMAIN R,A=(R2),LV=(R3)
         L     R2,RECORDAD
         L     R3,RECORDL
         FREEMAIN R,A=(R2),LV=(R3)
         L     R15,RC                  *The return code to pass
         TERM  RC=R15
         LTORG
DORD     DC    D'0'
RC       DC    A(0)                    * Return Code
SAVER14  DC    A(0)                    * Return from READ routine
BSTART   DC    A(0)                    *-> Start of buffer
BEND     DC    A(0)                    *-> First byte after buffer
BUFLEN   DC    A(0)                    *L'Block
BP       DC    A(0)                    *-> Current buffer position
ARDW     DC    A(0)                    * Actual RDW
ALEN     DC    A(0)                    * Actual record length
LEN      DC    A(0)                    * Work
OUTP     DC    A(0)                    * Output record pointer
BUFFERL  DC    A(0)
RECORDAD DC    A(0)
RECORDL  DC    A(0)
RECIN    DC    A(0)
```

```
RECOUT   DC    A(0)
UDCB     DCB   DSORG=PS,RECFM=U,DDNAME=INPUT,MACRF=(R),EODAD=SLUT,     C
               BLKSIZE=32760
VBDCB    DCB   DSORG=PS,RECFM=VB,MACRF=(PM),DDNAME=OUTPUT
PRINT    DCB   DSORG=PS,RECFM=F,MACRF=(PM),DDNAME=PRINT,               C
               LRECL=80
MSG01    DC    CL80'RDW001 - Input Data Set must be RECFM=U'
MSG02    DC    CL80'RDW002 - Output Data Set must be RECFM=V or VB'
MSG03    DC    CL80'RDW003 - Output Data Set LRECL Must be greater thanC
                zero'
MSG04    DC    C'RDW004 - Blocks read from Input Data Set: '
MSG04NO  DC    CL7' '
         DC    CL(80-(*-MSG04))' '
MSG05    DC    C'RDW005 - Records written to Output Data Set: '
MSG05NO  DC    CL7' '
         DC    CL(80-(*-MSG05))' '
         END
```

E.6 assembler entry code (INIT MACRO)

This macro generates standard housekeeping code in an assembler routine. See the macro comments for details on the syntax.

```
.* --------------------------------------------------------------------------------
         MACRO
&NAME INIT &TEXT,&BASE=R12,&FLOAT=NO,&RENT=NO,&WORKLEN=0, C
               &PRINT=NOGEN,&MODE=31,&EQUATES=YES
.* ===================================================================
.*
.* This macro is used to generate the normal assembler routine
.* housekeeping code - CSECT, TITLE, Base registers, save area
.* chaining etc.
.*
.* It is assumed to be used together with it's companion macro
.* called TERM for exit from an assembler routine. The INIT macro
.* sets some GLOBAL variables that are tested by TERM.
.*
.* Parameters are:
.*
.* TEXT         The text value to be used in a TITLE stmt.
.* BASE=        One or more base registers. BASE=R12 or
.*              BASE=(R12,R11,R10).  Default is BASE=R12.
.* FLOAT=YES/NO Indicate whether floating point registers
.*              should be saved upon entry and restored upon
.*              exit.
.*              Default is FLOAT=NO.
.* RENT=YES/NO  Indicate whether the macro should generate
.*              reentrant code or not.  Default is RENT=NO.
.* WORKLEN=     Length of extension to getmained savearea to
.*              use as workarea for the program. R13 is base
.*              starting with 18F savearea.  If FLOAT=YES an
.*              additional 32 bytes are reserved for savearea.
.* PRINT=GEN/NOGEN Print macro expansions or not
.* MODE=24/31   If MODE=24 the macro generates AMODE and
.*              RMODE 24.  If MODE=31 the macro generates
.*              AMODE 31 and RMODE ANY.
.* EQUATES=YES/NO Generate bit equates - default is YES
.*
.*
```

```
.* ====================================================================
&NAME TITLE &TEXT
&NAME CSECT
         AIF   ('&MODE' EQ '31').MODE31
&NAME AMODE 24
&NAME RMODE 24
         AGO   .MODEOK
.MODE31 ANOP
&NAME AMODE 31
&NAME RMODE ANY
.MODEOK ANOP
         GBLB  &EQUDONE,&FLOATREG,&INIRENT
         GBLC  &FLOATSAV
         GBLA  &GETLEN
         PRINT &PRINT              *DEFAULT PRINT OPTION
         AIF   (&EQUDONE).EQOK
*
* GENERAL PURPOSE REGISTER EQUATES
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         AIF   ('&FLOAT' EQ 'NO').NOFLOAT
*
* FLOATING POINT REGISTER EQUATES
*
FR0      EQU   0
FR2      EQU   2
FR4      EQU   4
FR6      EQU   6
.NOFLOAT ANOP
         AIF   ('&EQUATES' EQ 'NO').NOBITEQ
*
* BITTESTING EQUATES
*
AB       EQU   X'FF'
BIT0     EQU   X'80'
BIT1     EQU   X'40'
BIT2     EQU   X'20'
BIT3     EQU   X'10'
BIT4     EQU   X'08'
BIT5     EQU   X'04'
BIT6     EQU   X'02'
BIT7     EQU   X'01'
.NOBITEQ ANOP
*
&EQUDONE SETB 1
```

```
.EQOK ANOP
.*
.* SETUP ADRESSABILITY, GENERATE ID AND CHAIN SAVEAREA
.*
         USING *,R15
         B     INITA&SYSNDX.
         DC    AL1(INITA&SYSNDX.-*)
         DC    C'&NAME. - &SYSTIME. &SYSDATE.'
         CNOP  0,4
INITA&SYSNDX. EQU *
         STM   R14,R12,12(R13)        *SAVE CALLERS REGS
         AIF   ('&BASE'(1,1) NE '(').ONEBASE
         LR    &BASE(1),R15
         DROP  R15
         USING &NAME.,&BASE(1) *FIRST BASE
         LCLA  &IDX,&BASVAL
&IDX SETA  2
.BASLOP ANOP
         AIF   (&IDX GT N'&BASE).BASDONE
&BASVAL SETA 4095
&BASVAL2 SETA 4095*(&IDX-1)
         LA    &BASE(&IDX),&BASVAL.(&BASE(&IDX-1))
         USING &NAME+&BASVAL2.,&BASE(&IDX) *SUBSEQUENT BASE
&IDX SETA  &IDX+1
         AGO   .BASLOP
.ONEBASE ANOP
         LR    &BASE.,R15
         DROP  R15
         USING &NAME.,&BASE.
.BASDONE ANOP
         AIF   ('&RENT' EQ 'NO').NORENT
&INIRENT SETB 1
&GETLEN SETA (18*4)+&WORKLEN
         AIF   ('&FLOAT' EQ 'NO').RNFL
&GETLEN SETA &GETLEN+(8*4)
.RNFL ANOP
         LA    R0,&GETLEN
         GETMAIN R,LV=(R0)
         LR    R15,R1                  *START GETMAINED AREA
         L     R1,24(R13)              *RESTORE PARM REG
         USING INITS&SYSNDX.,R15
INITS&SYSNDX. DSECT
         DC    18F'0'
INITF&SYSNDX. DC 4D'0'
&NAME. CSECT
         AGO   .DOSAVE
.NORENT ANOP
         B     INITB&SYSNDX.
INITS&SYSNDX. DS 0F
         DC    18F'0'                  *SAVE AREA
         AIF   ('&FLOAT' EQ 'NO').NOFLAR
INITF&SYSNDX. DS 0D
         DC    4D'0'
.NOFLAR ANOP
INITB&SYSNDX. EQU *
.DOSAVE ANOP
         ST    R13,INITS&SYSNDX.+4 *BACKWARD CHAIN
         LA    R15,INITS&SYSNDX.
         ST    R15,8(R13)              *FORWARD CHAIN
         LR    R13,R15                 *READY FOR GO
         AIF   ('&FLOAT' EQ 'NO').NOFLSAV
&FLOATSAV SETC 'INITF&SYSNDX'
         STD   FR0,&FLOATSAV.
         STD   FR2,&FLOATSAV.+8
         STD   FR4,&FLOATSAV.+16
         STD   FR6,&FLOATSAV.+24
&FLOATREG SETB 1
.NOFLSAV ANOP
         AIF   ('&RENT' EQ 'NO').USOK
         DROP  R15
.USOK ANOP
.*
         MEND
```

E.7 assembler exit code (INIT MACRO)

This macro generates standard exit code from an assembler routine.

```
         MACRO
         TERM  &RC=
         GBLB  &FLOATREG,&INIRENT
         GBLA  &GETLEN
         AIF   ('&RC' EQ '').NORC
         AIF   ('&RC'(1,1) EQ 'R').REG
         LA    R15,&RC             *SET RETURN REGISTER
         AGO   .NORC
.REG ANOP
         LR    R15,&RC             *SET RETURN REGISTER
.NORC ANOP
         AIF   (&FLOATREG EQ 0).NOFL
         LD    FR0,18*4(R13)           *FR0
         LD    FR2,18*4+8(R13)         *FR2
         LD    FR4,18*4+16(R13)        *FR3
         LD    FR6,18*4+24(R13)        *FR6
.NOFL ANOP
         AIF   (NOT &INIRENT).NOR
         LR    R1,R13                  *REMEMBER FOR FREEMAIN
.NOR ANOP
         L     R13,4(R13)              *POP SAVEAREAS
         AIF   (NOT &INIRENT).NORENT
         LR    R6,R15                  *SAVE RC
         LA    R0,&GETLEN          *LENGHT TO FREEMAIN
         FREEMAIN R,A=(R1),LV=(R0)
         LR    R15,R6                  *REMENER RC
.NORENT ANOP
         L     R14,12(R13)             *RESTORE REG14
         LM    R0,R12,20(R13)          *RESTORE REMAINING REGS
         BR    R14                     *RETURN....
         MEND
```

E.8 FTPOSTPR user exit

```
/***************************************************************/
/*
  Communications Server for OS/390, Version 2, Release 10

  Copyright:    Licensed Materials - Property of IBM
                "Restricted Materials of IBM"
                5647-A01
                (C) Copyright IBM Corp. 2000.
                US Government Users Restricted Rights -
                Use, duplication or disclosure restricted by
                GSA ADP Schedule Contract with IBM Corp.

  Status:       CSV2R10

  Function: Sample FTP User exit that allows for post-FTP
            processing

  This is unsupported code which is provided on an "as is" basis.

  Parameters being passed in from the FTP server via the
  parameter list:

   +0 --   Pointer to the word with the user exit return code
   +4 --   Pointer to the number of parameters passed in
   +8 --   Pointer to the 8-byte buffer containing the USERID
   +12--   Pointer to the 4-byte client IP address
   +16--   Pointer to the 2-byte client port number
   +20--   Pointer to the 4 byte character string with current
             directory type:
               MVS or HFS (left justified)
   +24--   Pointer to a buffer that contains the current directory
             value, the first two bytes hold the length of the
             remaining buffer.
   +28--   Pointer to the 4 character byte field that contains the
             current filetype (SEQ, JES, SQL), left justified
   +32--   Pointer to the 3 character byte field that contains the
             current FTP reply code
   +36--   Pointer to buffer that contains FTP reply string; first
             two bytes contain the length of the remaining buffer
   +40--   Pointer to the 4 byte field that contains the current
             FTP command code
   +44--   Pointer to the 1 char byte field that contains the current
             CONDDISP setting-
               C for catalog, D for delete
   +48--   Pointer to the 4 byte binary field that contains the close
             reason code:
               0 -- transfer completed normally
               4 -- transfer aborted before data connection was
                      established
               8 -- transfer aborted with socket communication errors
              12 -- transfer aborted after data connection was
                      established
              16 -- transfer aborted with SLQ file errors after data
                      connection was established
*/
/***************************************************************/
/*                                                             */
/*       FTPOSTPR USER EXIT                                    */
/*                                                              */
/****************************************************************/

#pragma linkage(FTPOSTPR, fetchable)


#define _XOPEN_SOURCE_EXTENDED 1

#include <stdio.h>
#include <stdlib.h>
#include <syslog.h>



/* set up structure needed for current directory value         */
typedef struct {
                 short dirlen;
                 char  dirname[1100];
               }currdir;

/* set up structure needed for reply string value              */
typedef struct {
                 short replylen;
                 char  reply[1200];
               }replystr;



/* beginning of FTPOSTPR function                              */
int FTPOSTPR(int *exitrc, int *numparms, char exitusrid[8],
             unsigned long *clientIP, unsigned int *clientport,
             char dirtype[4], currdir *cwd, char filetype[4],
             char replycode[3], replystr *rs, char cmdcode[4],
             char *conddispvalue, int *closerc)
{

   char userid[9];
   memset(userid,'\0',9);
   memcpy(userid,exitusrid,8);
   /* write message to syslog                                  */
     syslog(LOG_INFO,"FTPOSTPR: FTP process completed with rc of "
            "%d, userid '%s', client IP %0.8X, client port %d, "
            "reply code '%s', and reply string '%s'",
            *closerc, userid, *clientIP, *clientport, replycode,
            rs->reply);
}
```

E.9 Sample JCL to compile and link-edit the FTPOSTPR user exit

```
//CCFTPOST JOB (KAKKY,1A1767),'T.KAKIMOTO',MSGCLASS=X,NOTIFY=&SYSUID
//*
//PROCS   JCLLIB ORDER=CBC.SCBCPRC
//CLG  EXEC EDCCL,
//  INFILE=KAKKY.TCPIP.SRCJCL(FTPOSTPR),
//  CPARM='LIST,SOURCE'
//COMPILE.SYSLIB DD
```

```
//               DD DISP=SHR,DSN=TCPIP.SEZACMAC
//LKED.SYSLIB DD DISP=SHR,DSN=TCPIP.SEZACMTX
//            DD DISP=SHR,DSN=CEE.SCEELKED
//LKED.SYSLMOD DD DISP=SHR,DSN=TCPIP.TCPPARMS.VTAMLIB(FTPOSTPR)
//LKED.SYSIN  DD *
  NAME FTPOSTPR(R)
```

F

FTP client sample REXX program

This appendix contains the REXX source code to execute the FTP client. This REXX program can be executed in both a TSO environment and UNIX System Services environment.

F.1 REXX sample program for FTP client

```
/* rexx                               */
/*                                    */
/*  Example REXX exec for FTP         */
/*                                    */

parse arg infile outfile .    /* get command line input */
parse source . . . . . . . . env .   /* check if running under OE  */
                                     /* env='OpenMVS' if invoked   */
                                     /* from OE, otherwise env=' ' */
input_file = ''
output_file = ''

/**************************************/
/*    Process input file             */
/**************************************/
if infile = '' then                  /* input file not specified   */
  do
    say 'Input file name is required.'
    exit 12                          /* return error               */
  end
else
  do
       /* for OE check if input file exist and add redirection to  */
       /* FTP command issued to OE shell.                          */
    if env = "OpenMVS" then
      do                             /* OE environment.            */
        address syscall "stat (infile) fstat." /* input file exist? */
        if fstat.0 = 0 then
          do
            say 'Input file:' infile 'not found.'
            exit 28
          end
        input_file = '<' infile
      end
       /* For TSO allocate DD INPUT               */
    else
      do                              /* TSO environment            */
        ADDRESS TSO "ALLOC DA('"&lor.&lor.infile&lor.&lor."') DD(INPUT) SHR REUSE"
        if rc &notsym.=0 then             /* ALLOC failed              */
          do
            say "*ERROR: ALLOC" infile "failed with return code =" rc
            exit 12                   /* return error               */
          end
      end
/*************************************/
/*    Process output file           */
/*************************************/
if outfile <> '' then
       /* for OE add redirection to FTP command issued to OE shell */
  if env = "OpenMVS" then
    output_file = '>' outfile
       /* For TSO allocate DD OUTPUT              */
  else
    do                                /* TSO environment            */
      ADDRESS TSO "ALLOC DA('"&lor.&lor.outfile&lor.&lor."') DD(OUTPUT) SHR REUSE"
      if rc &notsym.=0 then
        do
          say "*ERROR: ALLOC" outfile "failed with return code =" rc
          exit 12                     /* return error               */
        end
    end
/* say " ALLOC : IN ->" infile " OUT ->" outfile */
/*************************************/
/*    Invoke FTP client             */
/*************************************/
"ftp -v -p TO3ATCP" input_file output_file

if env <> "OpenMVS" then
  do
    "FREE DD(INPUT)"                  /* free the DD(INPUT)         */
    if outfile <> '' then
      "FREE DD(OUTPUT)"               /* free the DD(OUTPUT)        */
  end

say "FTP client return code is:" rc  /* print client return code   */
exit 0                                /* return to invoker          */
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 552.

- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration,* SG24-5227
- *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*, SG24-6516
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance,* SG24-6517
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840
- *TCP/IP in a Sysplex*, SG24-5235
- *Managing OS/390 TCP/IP with SNMP*, SG24-5866
- *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *Networking with z/OS and Cisco Routers: An Interoperability Guide*, SG24-6297
- *zSeries HiperSockets*, SG24-6816

Other resources

These publications are also relevant as further information sources:

- *z/OS V1R2.0 UNIX System Services Planning,* GA22-7800
- *z/OS V1R2.0 UNIX System Services User's Guide*, GA22-7801
- *z/OS V1R1.0-V1R2.0 MVS Initialization and Tuning Guide*, SA22-7591
- *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765
- *z/OS V1R2.0 C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS V1R2.0 CS: IP Migration*, GC31-8773
- *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776
- *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780

- *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781
- *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782
- *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB)*, SC31-8784
- *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM)*, SC31-8786
- *z/OS V1R2.0 CS: IP Application Programming Interface Guide*, SC31-8788

Referenced Web sites

These Web sites are also relevant as further information sources:

- The z/OS Web pages

  http://www-1.ibm.com/servers/eserver/zseries/zos/installation/installz12.html

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

IBM

Redbooks

Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

Communications Server for z/OS V1R2 TCP/IP Implementation Guide

Provides a detailed survey of CS for z/OS applications and APIs

Covers DNS, FTP, syslogd, otelnet, sendmail, and much more

Uses scenarios to ease your application deployment

The Internet and enterprise-based networks have led to the rapidly increasing reliance upon TCP/IP implementations. The zSeries platform provides an environment in which critical business applications flourish. The demands placed on these systems is ever-increasing and such demands require a solid, scalable, highly available, and highly performing Operating System and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality.

The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS. *Volume 2* covers the UNIX applications shipped as part of Communications Server for z/OS IP. In this volume, we classify z/OS applications and provide a detailed survey of the protocols and implementation issues associated with each. These applications provide a rich set of functionality, including remote execution with otelnet and file transfers with FTP and TFTP. In addition, we cover important network functions such as DNS, Dynamic IP, syslogd, and NFS. We provide scenario-based discussions to aid in application deployment.

Because of the varied scope of CS for z/OS, this volume is not intended to cover all aspects of the topic. The main goal of this volume is to provide an insight into the different applications provided by CS for z/OS and, more specifically, into the protocols they use and the mechanisms to deploy them. For more information, including on applications available with CS for z/OS IP, please reference the other volumes in the series.

SG24-5228-03 ISBN 073842692X