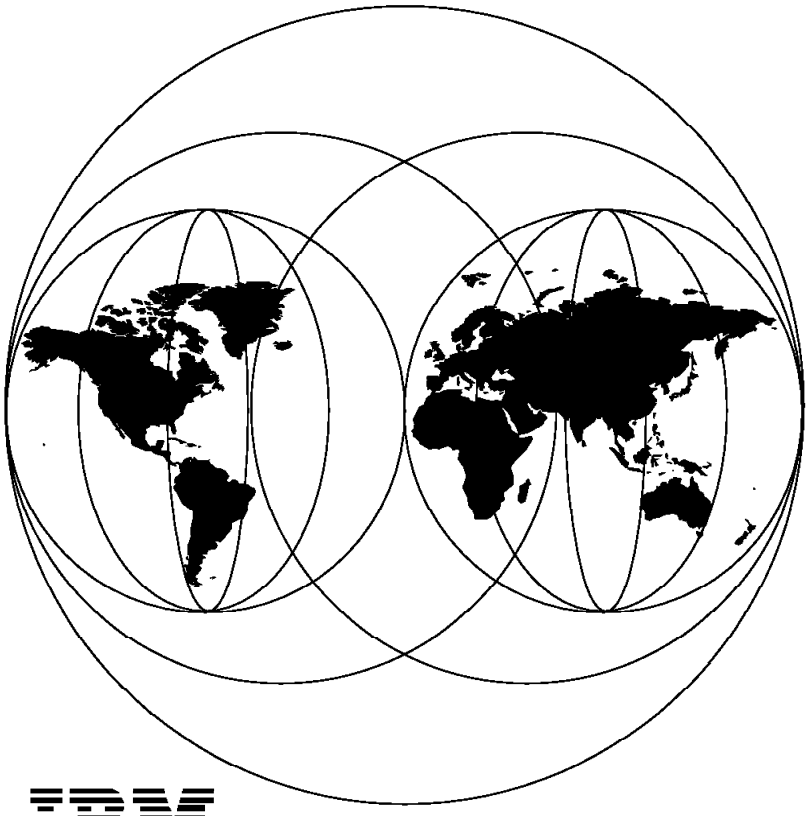
SG24-4854-00

TME 10 Software Distribution - Mobile Clients

January 1997



IBM

International Technical Support Organization
Raleigh Center

IBML International Technical Support Organization

SG24-4854-00

TME 10 Software Distribution - Mobile Clients

January 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 371.

First Edition (January 1997)

This edition applies to TME 10 Software Distribution, 5697-SWD for use with the AIX, OS/2, Windows 3.1, Windows 95 and Windows NT operating systems.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1997. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1997

Figures

© Copyright IBM Corp. 1997

Tables

© Copyright IBM Corp. 1997

Preface

This redbook describes the Mobile Client feature as available in TME 10 Software Distribution. It explains the concept of a mobile software distribution client, gives examples of how to install and configure the Mobile Client in different environments and shows typical change management scenarios.

This redbook was written for IBM and customer technical personnel who are involved in the design and implementation of software distribution networks using mobile clients, for example, in a sales organization with many mobile users.

This redbook should also be useful in getting a general understanding of mobile computing and the involved components.

Several practical examples are presented to demonstrate the function and the usage of the Mobile Client feature.

Some knowledge of software distribution in general as well as a basic knowledge of the underlying operating systems and communication protocols is assumed.

Some knowledge of TME 10 Software Distribution, in particular Software Distribution for AIX and Software Distribution for OS/2, is recommended but not mandatory.

How This Redbook Is Organized

This redbook contains 391 pages. It is organized in five parts. You might want to read this redbook from cover to cover or refer to any area of interest. However, we do recommend that you read Part 1, “Introduction to Software Distribution with Mobile Clients” on page 1 in any case.

The other parts can be read and used separately if desired. However, we will use environments once set up in later chapters.

The parts are as follows:

- Part 1, “Introduction to Software Distribution with Mobile Clients”

  This part provides an introduction to software distribution using the Mobile Client feature of TME 10 Software Distribution. It gives an overview of TME 10 Software Distribution first and then introduces the Mobile Client feature. In particular Chapter 3, “Architecture and Data Flow of the Mobile Client” on page 13 provides a detailed description of the architecture of the Mobile Client and its components, as well as the data and control flow during a change request performed with the Mobile Client.

  We recommend that you read this part before trying to use the Mobile Client in a scenario, since it will provide you with fundamental knowledge that is essential for understanding and using the Mobile Client.

- Part 2, “Setting Up a Mobile Client Environment”

  This part provides guidance on how to set up the Mobile Client feature in different scenarios. We will focus on installation and customization of the

© Copyright IBM Corp. 1997

Mobile Client on different client platforms and give advice on what to look out for. We also point to the differences between the Mobile Client and a standard client as far as installation and configuration are concerned.

- Part 3, “Simple Scenarios Using the Mobile Client”

  This part shows some basic change management scenarios that can be used to familiarize yourself with the Mobile Client. Besides showing basic principles of the Mobile Client, we also provide you with some basic knowledge on how to create and distribute change objects in TME 10 Software Distribution in general.

  We especially recommend you read this part if you are not yet very familiar with either TME 10 Software Distribution or the Mobile Client.

- Part 4, “Advanced Scenarios Using the Mobile Client”

  This part contains some more advanced scenarios using the Mobile Client feature. In particular we show the following scenarios:

  – Using a fully disconnected Mobile Client
  – Using the Mobile Client over a PPP link
  – Using the Mobile Client with IBM 8235 LAN Dialer
  – Using the asynchronous protocol support feature
  – Using change objects with Mobile Clients, for example, CID
  – Using the Mobile Client in a focal point connected environment
  – Using the Mobile Client with dynamic IP

  We recommend you read this chapter if you need to implement the Mobile Client in one of the above scenarios and you are already familiar with the Mobile Client.

  This part also contains detailed descriptions of configuring modems, network protocols and related components.

- Part 5, “Implementing Mobile Clients”

  This part is intended to help you plan and design a software distribution network using Mobile Clients.

  It summarizes the observations and experiences gained in the previous parts and compares the different options you can choose from when planning to implement a software distribution solution.

  This part is intended mainly for project managers, I/T architects and designers who have to design and implement a software distribution solution.

  Also, Chapter 18, “Example Front-End Application” on page 325 contains a sample Hotel application that automates the process of connecting to a TME 10 Software Distribution server from a remote location, for example, from a hotel.

  This application is entirely written in Java and can therefore be used on all client platforms.

- Part 6, “Appendixes” on page 351

  In this part we have included information that may be helpful when using Mobile Clients.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Stefan Uelpenich is an Advisory International Support Representative at the Systems Management and Networking ITSO Center, Raleigh. He writes extensively and teaches IBM classes worldwide on all areas of systems management. Before joining the ITSO, he worked in IBM Germany′s Professional Services organization as an Advisory I/T Architect for Systems Management.

Mark Guthrie has recently joined ISSC Australia from IBM Australia where he worked as a Technical Team Leader. He has worked for the past six years on client/server systems, and has architected several systems management solutions mainly for insurance companies. His most recent work was on client/server performance for an MVS-based banking system.

Sonja Thurau is an I/T Specialist from IBM Germany working in Systems Management Software Support. She has seven years of experience in the area of software distribution and has implemented software distribution solutions in several large projects.

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

David L. Boone
Kathryn Casamento
Wolfgang Geiger
Linda Robinson
Roger Serrette
Shawn Walsh
Gail Doucette Wojton
Systems Management and Networking ITSO Center, Raleigh.

Linda Harrell
Vito Losacco
Donatella Sabellico
IBM Networking Software Laboratory, Rome, Italy

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Part 1. Introduction to Software Distribution with Mobile Clients

This part introduces the Mobile Client which is new to TME 10 Software Distribution Version 3.1.3. We give a short introduction to TME 10 Software Distribution in general and then introduce the Mobile Client feature.

This part also includes a detailed description of the architecture of the Mobile Client that should be helpful in understanding change management using the Mobile Client.

© Copyright IBM Corp. 1997

Chapter 1. TME 10 Software Distribution

TME 10 Software Distribution is Tivoli/IBM's solution for electronic software distribution in the enterprise network.

It is part of Tivoli Management Environment (TME) 10, Tivoli/IBM's solution for enterprise systems management.

At this time TME 10 Software Distribution contains the following products inherited from the former Tivoli TME 3.0 and IBM SystemView families of products:

- Tivoli/Courier 3.0
- Software Distribution for AIX 3.1.3
- Software Distribution for OS/2 3.1.3
- Software Distribution for Windows NT 3.1.3
- Software Distribution clients for a wide range of platforms
- NetView Distribution Manager for AIX 1.2
- NetView Distribution Manager/2 2.1

These products will be integrated into one single product that will combine features of both, Tivoli/Courier and Software Distribution 3.1.3 and will run on top of the Tivoli Management Framework (TMF).

Note

If you would like to know more details about the integration of the above products into the next release of TME 10 Software Distribution you can obtain a copy of the *TME 10 Roadmap* document from your Tivoli/IBM representative.

The Mobile Client feature discussed in this book is currently part of the Software Distribution 3.1.3 family of products which will be the base for migration onto the TMF.

This means that at the moment you should use the Software Distribution 3.1.3 family of products to be able to use the Mobile Client feature.

1.1 The Mobile Client

Increasingly organizations are relying on PCs to run their business applications. Often these PCs are laptops and spend much of their time away from the office. Even when connected to a LAN, laptops may change physical and logical location frequently as their owner moves about within the corporation. Despite the technical difficulties, laptop users require the same level of data currency and timely software delivery that their desk bound colleagues enjoy.

The main issues with software distribution to laptop users are:

- Timely installation of software updates such as table values even when the laptop is not connected to a network.

© Copyright IBM Corp. 1997

- Connection to software distribution services over modems.
- Minimizing connect time to servers when connected over modems.
- Identification of laptops as they move in a network, for example, changes of IP address with location.

To address these issues TME 10 Software Distribution Version 3.1.3 introduces the Mobile Client. The Mobile Client can:

- Operate as a normal client when connected to a server.
- Perform change management activities when disconnected from a server. This is possible because of the Mobile Client's local catalog.
- Operate as a fully disconnected client and never connect to a server.
- Connect to a server over a modem using standard connections such as SLIP or PPP, or using its own high-performance asynchronous support.

1.2 Supported Platforms

The Mobile Client can only operate with a server that understands how to interact with the Mobile Client. Currently the following platforms are supported as clients and servers:

1.2.1 Clients

OS/2 Version 2.1 and above

Windows Version 3.1

Windows 95

Windows NT Version 3.51 or above

AIX Version 3.2 or above

1.2.2 Servers

AIX Version 3.2 or above

OS/2 Version 2.1 or above

Windows NT Version 3.51 or above

Note

Besides the operating systems you will need other prerequisites in order to run TME 10 Software Distribution and the Mobile Client, for example, communications subsystems such as TCP/IP. Refer to Part 2, "Setting Up a Mobile Client Environment" on page 41 for details.

1.2.3 Support for Asynchronous Communications

The asynchronous communications feature allows a software distribution mobile client to connect to a server using a modem without any underlying communications protocol, for example, TCP/IP. In order to do so TME 10 Software Distribution directly controls the modems to establish connections. Refer to Chapter 11, "Using the Asynchronous Protocol Support for Mobile Clients" on page 181 for a detailed description of this feature.

Note

At the time this book was written we only had access to the asynchronous support feature for the AIX server and the OS/2 and Windows 3.1 clients.

Please check with your IBM representative to obtain current availibility information.

However, you are also able to use modems for communications on all other platforms without using direct asynchronous support, for example, by using a PPP link. Refer to Chapter 9, “Using the Mobile Client with PPP” on page 131 for an example.

1.2.4 Support for Communication Protocols

A various number of communication protocols are supported by TME 10 Software Distribution.

The communication protocols available depend on the combination of client and server you choose. Refer to the product manuals for a detailed description on which protocols are available for which platforms.

In this book we use TCP/IP for all scenarios. You can also use different protocols, for example, NetBIOS to connect an OS/2 client to an OS/2 server.

Chapter 2. Uses for a Mobile Client

The Mobile Client opens new opportunities to use change management. A variety of circumstances are ideally suited for using a Mobile Client, this section presents a few.

2.1 Deadline Activation

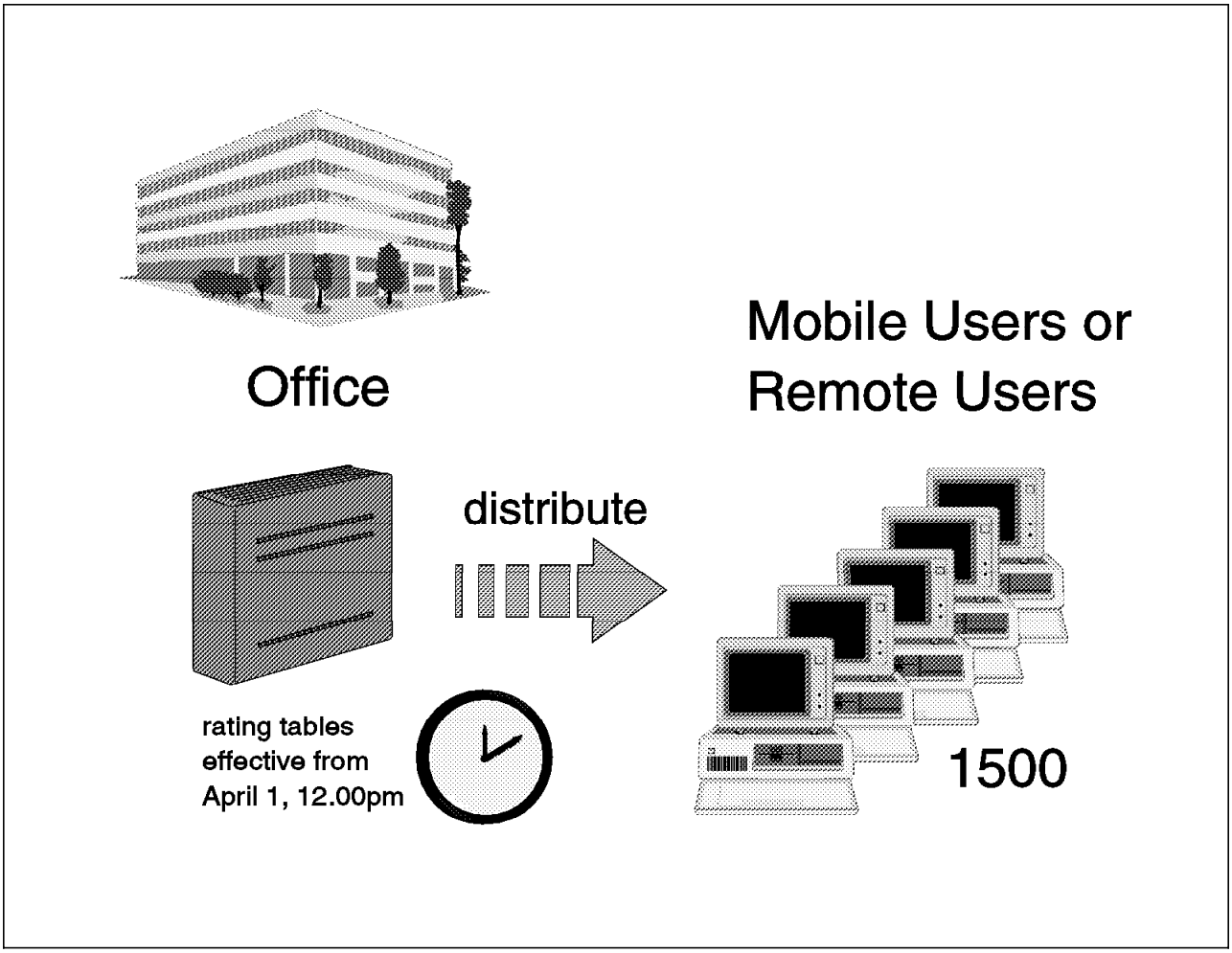


Figure 1. Deadline Activation

Laptop users such as mortgage assessors who work part of the time connected to the office LAN and part of the time visiting clients, require rating table updates to be made effective on certain dates. They may be out of the office for several days at a time, during which period new tables need to be installed. The Mobile Client can receive the change objects containing the table updates while connected to the office software distribution server and apply them automatically when the time is right. Confirmation of the update are passed back to the office server the next time the laptop connects to the LAN.

Another example for deadline activation is when there are many client systems connected to a server through a slow link, as shown in Figure 2 on page 9.

© Copyright IBM Corp. 1997

Consider the following example:

An insurance company has 1500 insurance agents connected to its head office, all using a modem link, that need to use a new form, effective April 1. However, since there is a large number of users and it is very likely that many them are not available all the time, you may not want to start the distribution of the new form only a short time before.

You can, for example, start the distribution to the clients on February 1. Since the Mobile Client can store change objects locally you can schedule the object to be sent to all clients immediately. The server will then connect the clients and send the object if the client is available; if not, the object is sent when the client becomes available.

At the same time you also send the install request to the Mobile Client, but you specify that the install is not to happen before April 1. Since there is a two month period between when we started the distribution and when we need the change to become effective we can be sure that all clients will have received the object by April 1.

Since the Mobile Client can process install requests locally all agent systems will automatically start the install on April 1 and will not rely on a connection to the server. The only pre-requisite for this to happen is that the client computer is switched on and that the Mobile Client is started.

2.2 Modem Connections

Some laptop users visit their own office so infrequently that it is impractical to use a LAN-based solution for software distribution. By installing the Mobile Agent on these laptops and connecting to a software distribution server over a modem, these users can receive software and data updates in the same way as LAN-based users. The Mobile Agent receives the change objects into its catalog from the catalog of the server and installs the software locally. This has the following benefits over a redirected install across a modem:

1. The modem connection is held open for as short a time as possible, thus minimizing cost.
2. The install can be done later at the users convenience, or scheduled to occur automatically.
3. The change object is passed over the connection once only and in a compressed format.
4. The install is performed in exactly the same way as for a LAN-attached install. Many install programs will re-read the same data many times during the install process which is extremely inefficient if the connection to the file is slow. Also some install programs will timeout if access to the files is slow.

We show how to use the Mobile Client with modem connections in Chapter 9, “Using the Mobile Client with PPP” on page 131, Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181 and Chapter 10, “Using the Mobile Client with IBM 8235 LAN Dialer” on page 165.

Another example where modems are used with software distribution is shown below:

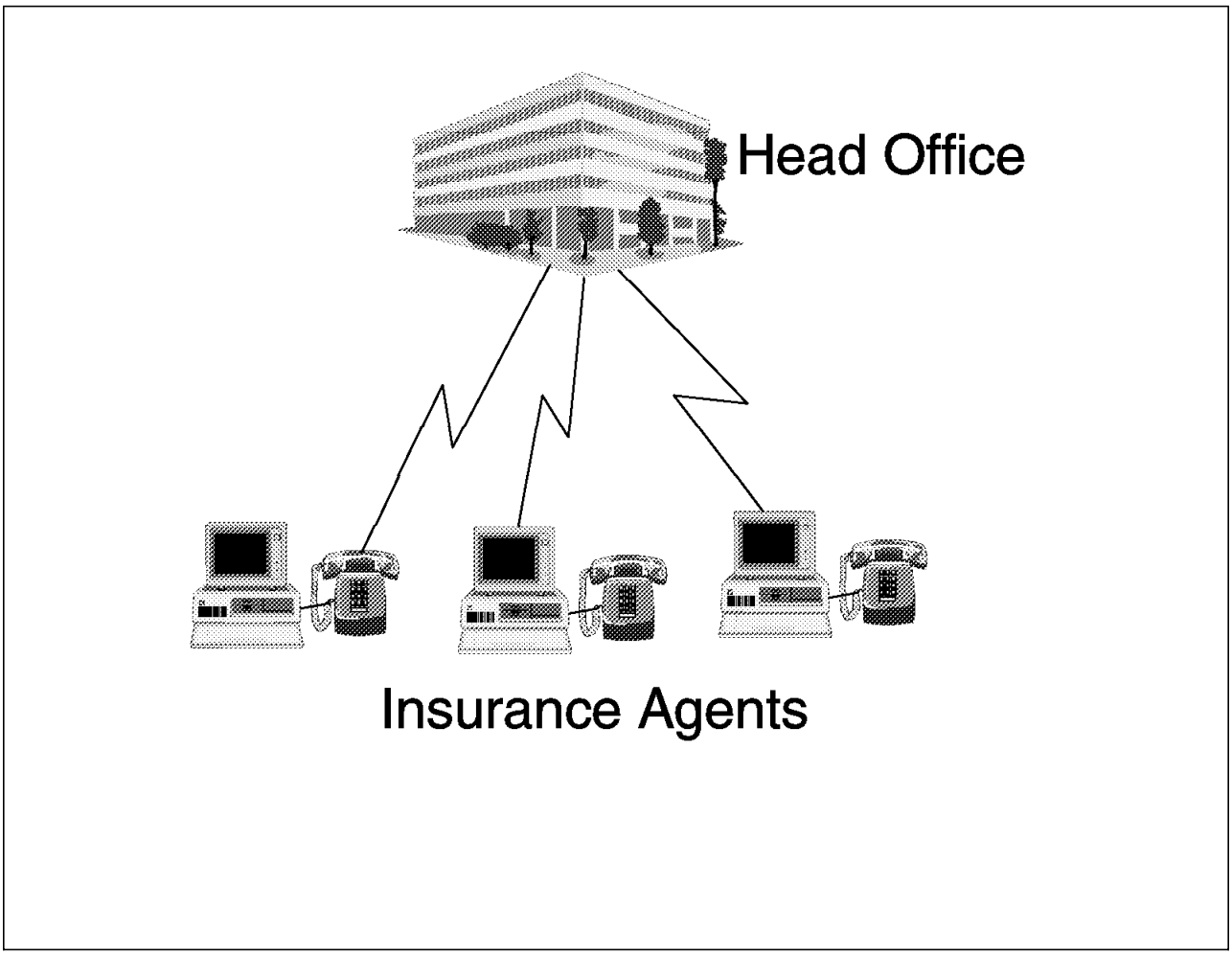


Figure 2. Software Distribution over Modems

In the above example several insurance agents are connected to their head office. The agents work in their own branch offices and never visit the head office. Software changes might occur from time to time, but mostly these are initiated by the head office.

However, the central server might not always be able to reach all agents, so it must be able to initiate a modem connection at a later time.

Also, a connection can be initiated by an agent, for example, to send new contracts to the head office for processing. In this case the client needs to establish a modem connection.

In both cases the Mobile Client is useful because it can directly control the modem links as described in Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181.

Also, for the above example it is quite likely that the modem connection is used with deadline activation.

2.3 Fully Disconnected Client

The fully disconnected client can be used to distribute software to computers that never connect to a TME 10 Software Distribution server.

This feature is useful when you either have no network connection between these systems and the server at all or when you have only a very slow line available.

You can then export the change file at the server and write it to an external media which you ship to the clients.

Since the Mobile Client has its own repository it can import the change file from the external media and then take care of the installation itself.

By reporting the status to the server and updating the change management history manually you still can maintain centralized control.

2.4 Combining LAN and WAN

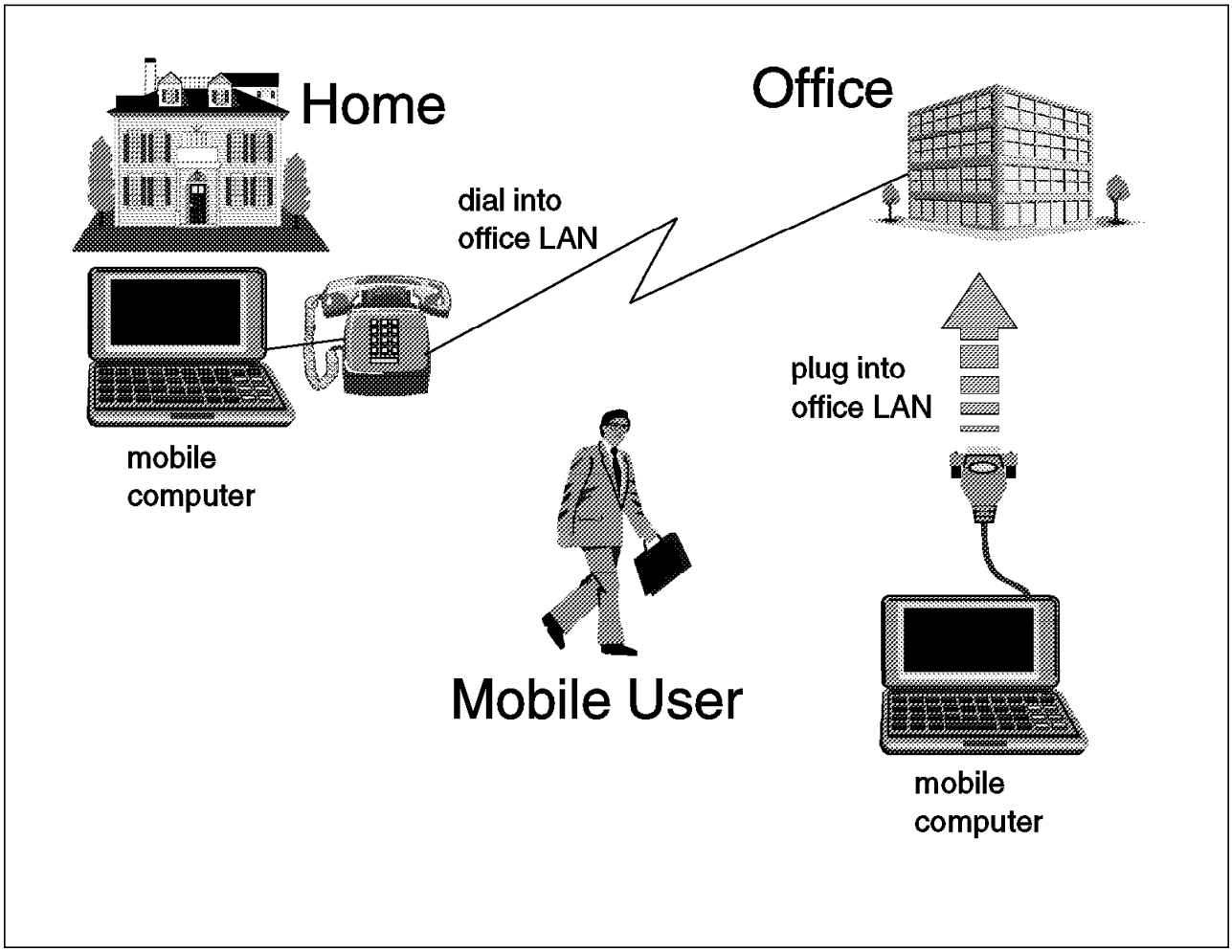


Figure 3. Combining LAN and WAN

Increasingly companies let their employees work part of their time at home while still keeping their office at the company.

Therefore they work part of their time in the office and part of their time at home or other places, for example, when visiting a customer.

This scenario is slightly different from an insurance agent, who maybe never visits his/her head office but is always connected remotely.

When using a WAN/LAN combination it is most desirable that the switch between work locations is as transparent to the user as possible.

The user in Figure 3 on page 10 uses the same notebook computer at home and at work. When he works at home, he/she dials into the office LAN using a modem. When he/she works at the office, he/she plugs in his/her notebook computer into the office LAN, for example, using a docking station.

In both cases the agent does not want to have to deal with any issues related to software distribution.

The Mobile Client is ideally suited for taking care of the software distribution part of this switch when combined with, for example, an IBM 8235 as shown in Chapter 10, “Using the Mobile Client with IBM 8235 LAN Dialer” on page 165.

Chapter 3. Architecture and Data Flow of the Mobile Client

The intention of this chapter is to explain the architecture of the Mobile Client and show the data flow to and from the Mobile Client. The first part introduces the internal product components and the second part explains the concept of the Mobile Client and demonstrates the data flow during change management processing between the server and the Mobile Client.

3.1 Overview of the Internal Components

For a common understanding we give an overview of the internal components of the Software Distribution 3.1.3 Mobile Client product in this section. To describe the new architecture we show the components that are available at the server and at the standard client, before we describe the components of the Mobile Client and their functions.

Figure 4 shows the architecture of the Software Distribution 3.1.3 products. In the following sections we go into detail about the components shown in the figure.

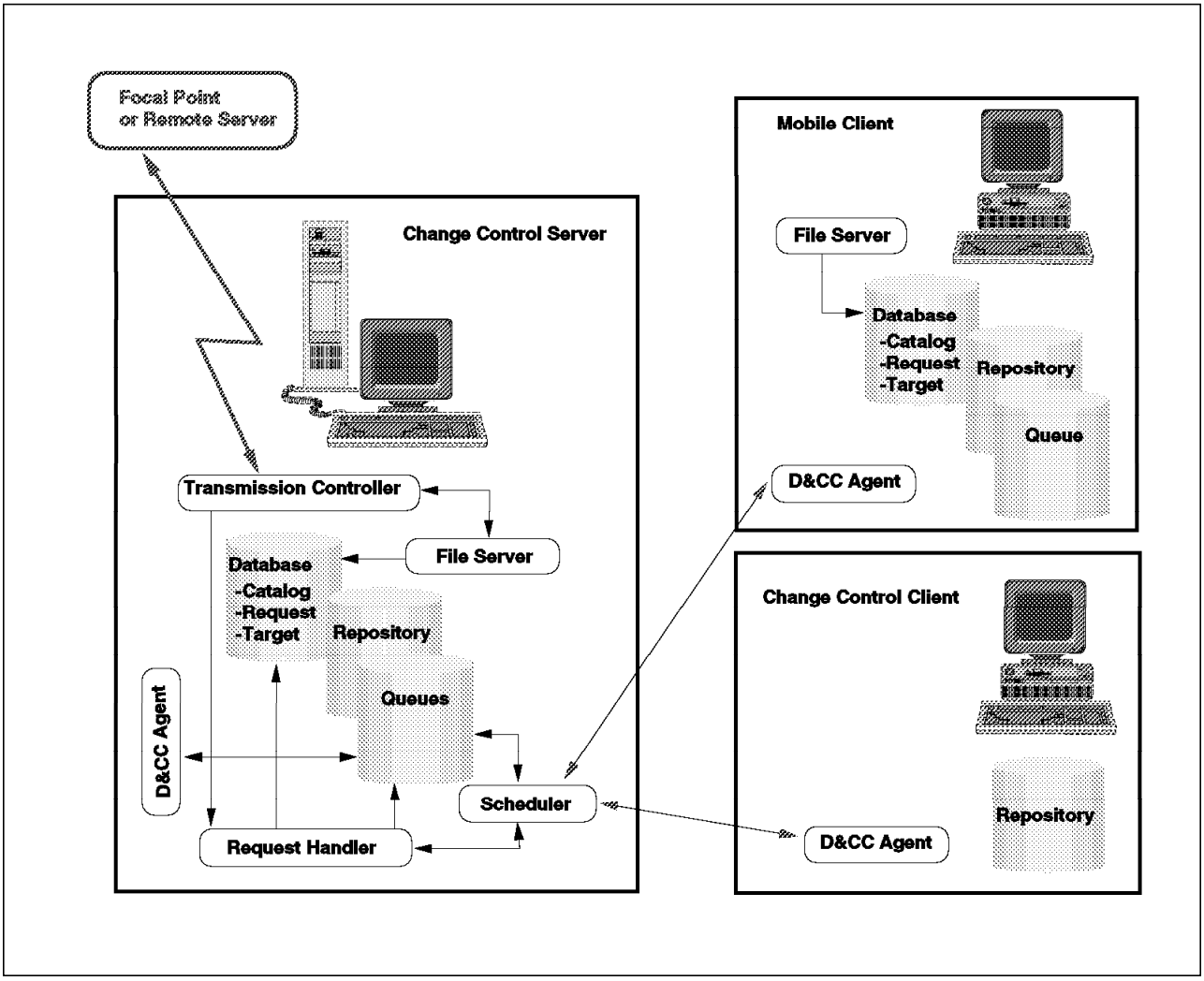


Figure 4. Software Distribution 3.1.3 Architecture

© Copyright IBM Corp. 1997

3.1.1 Internal Components of the Server

In this part we give a brief overview of the internal components shown for the change control server in Figure 4 on page 13 and their functions.

At the server we find the following components:

- Transmission controller

  The transmission controller is controlling the communication with to remote targets, such as a focal point or another server. Any files being sent to or from a remote system are processed by the transmission controller.

- D&CC agent

  The Distribution & Change Control (D&CC) agent runs at the server and each client and handles the distribution and change control requests. The name of the D&CC agent task is fndcmps. Each D&CC agent has an input queue that is maintained on the server. Requests that are placed on the queue are processed by the D&CC agent.

  The D&CC agent at the server handles all request for the server issued by a remote server or focal point. Here the server can be seen as another client.

- Request handler

  The request handler is responsible for the transfer of distribution and change control requests to the database and for the transfer of requests from the database to the scheduler. Reports are returned from the scheduler to the request handler, which stores them in the request database.

  The request handler also manages all the commands that change the status of requests, for example, hold or delete a request in the queue.

- Scheduler

  The scheduler is responsible for the direction of distribution and change control requests. The scheduler processes the requests that are placed in its input queue. If a request is due to be processed, the request handler transfers the request from the database to the scheduler's input queue. Then the scheduler transfers the request to its output queue for local and remote targets. Requests for a local target are placed in the D&CC agent queue for the target.

- File Server

  The file server maintains the catalog and is responsible for receiving data from and sending data to the network under the control of the transmission controller component. The file server also gives the other components of the server access to the catalog.

The Software Distribution 3.1.3 server uses data areas to store objects and workstation information and queue commands and requests in the system. Most of these data areas are databases for which an integrated database manager is provided with the product.

Database Format

The database content of the Software Distribution 3.1.3 product is stored in an internal database format in flat files in different subdirectories of the product directory. The product directory is C:SOFTDIST for OS/2 and all Windows platforms and /usr/lpp/netviewdm for the AIX platform by default.

The following lists the major data areas at the Software Distribution 3.1.3 server:

- Catalog

  The catalog serves as a reference to all change management objects and data files that are available at the server. Each object in the catalog is identified by an object name and has a pointer to the physical location of the object. The catalog holds the information about the target workstations that are authorized for each object.

  Information about Object Authorization

  A target workstation can only install objects in push mode if it is authorized for the object. The authorization can be specified during the creation of an object, by using the AUTHORIZE function to explicitly authorize one target or by authorizing all targets for each newly created object by setting the keyword AUTHORIZE: ALL in the configuration file NVDM.CFG of the server.

  The configuration file is located in the product directory C:SOFTDIST for the OS/2 and Windows NT server and /usr/lpp/netviewdm/db for the AIX server by default.

  The catalog also holds the change control history database that contains the status of change management objects at each target in the domain of the server. The history database also holds the information about data files that were distributed to a target.

- Target database

  The information stored in the target database is:

  – Target definitions

  – Group definitions

  – Installation tokens for targets

  – Hardware information extracted from a target or defined for a target

- Request database

  The request database is used by the request handler to store requests. Every request that is processed at the server is copied from the request handler into the request database. The request status in the database is updated by the request handler after reports are returned from the scheduler.

- Repository

  The repository is used by Software Distribution 3.1.3 to store its change management objects and is located by default in the directory C:SOFTDISTREPOS for OS/2 and Windows platforms and in the path /usr/lpp/netviewdm/repos for the AIX platform.

- Queues

  The server provides queues for request handling. The queues hold the files for Software Distribution 3.1.3 queuing and scheduling mechanisms. The queues at the server are divided into:

  – Local request queue

    Queue for locally attached clients

  – Remote request queue

Queue for remotely attached servers, including the focal point

– High priority request queue

  Queue for requests that are submitted with high priority

3.1.2 Internal Components of the Standard Client

The only component available at a Software Distribution 3.1.3 standard client, also called change control client, is the D&CC agent as shown in Figure 4 on page 13. The D&CC agent handles all the distribution and change control requests for the client. As soon as a request is received from the server the D&CC agent processes the request at the client. The D&CC agent uses its input queue at the server for request processing.

Distribution requests include functions to store, retrieve or delete a file. For a distribution request the agent goes through the following steps:

- Receive the request from the server
- Transfer files to and from the server and store them in the local file system
- Send a report to the server

Change control requests include functions to install, remove or uninstall software, initiate procedures or activate target workstations. The processing of a change control request varies according to the request that was issued. If files have to be processed in a request, they are copied from the server to the client before the installation or execution. The installation or execution itself is triggered on behalf of the client.

A report is always sent back to the server after the request processing is completed. The D&CC agent queue at the server is used to return the reports to the server.

Each client has a repository used to store change management objects locally at the client. When change control objects are sent from the server to the client they are stored in the local repository.

3.1.3 Internal Components of the Mobile Client

The Software Distribution 3.1.3 Mobile Client is capable of performing change management functions without having a connection with the server. This is called disconnected change management or disconnected mode. (See 3.2.1, “General Information about the Mobile Client” on page 18 for more information.)

To be able to perform disconnected change management operations the Mobile Client has its own catalog and other components compared to the connected standard client. As shown in Figure 4 on page 13 we find the following components at the Mobile Client:

- D&CC agent

  The Distribution & Change Control (D&CC) agent handles the distribution and change management requests from the server and the Mobile Client itself. The description of the D&CC agent in 3.1.2, “Internal Components of the Standard Client” is also valid for the Mobile Client, with the only exception that the initiate procedure function is not available while the Mobile Client is disconnected.

- File server

The file server maintains the local catalog of the Mobile Client.

The Mobile Client has its own data areas required to store objects and process requests locally while being disconnected from the server. Most of these data areas are databases managed by an integrated database manager. The databases are stored in an internal format in flat files in different subdirectories of the product directory at the Mobile Client. The default product directory is C:SOFTDIST for OS/2 and Windows platforms and /usr/lpp/netviewdm for the AIX platform. At the Mobile Client we find the following data areas:

- Catalog

  The Mobile Client has a local catalog that contains entries for all the change management objects that are available for the Mobile Client. The objects stored in the local catalog of the Mobile Client are:

  – Objects that were sent to the Mobile Client.

  – Objects for which the target of the Mobile Client is authorized, because all targets are authorized for the object or the administrator authorized the Mobile Client for the object.

  Note

  The objects for which a Mobile Client is authorized are automatically cataloged at the Mobile Client when it synchronizes its databases with the server as shown in Figure 9 on page 28. Notice that the change files are automatically cataloged but not stored at the Mobile Client. We will explain this in more detail in 3.2.2, “Data Flow during Connection Setup” on page 23.

- Objects that were created by the user of the Mobile Client while being disconnected.
- Objects that were exported at the server to an external device and imported at the Mobile Client.

  The catalog also holds the change control history of the objects cataloged locally for the Mobile Client.

- Target database

  The target database at the Mobile Client holds the following information:

  – Target definition of the Mobile Client

  – Installation tokens for the Mobile Client

  – Hardware information stored for the Mobile Client

- Request database

  The local request database is used to store requests that are processed in disconnected mode at the Mobile Client. The status of the requests is updated in the request database after processing is completed.

- Repository

  The Mobile Client and every other client have their own repository. The repository is used by Software Distribution 3.1.3 to store its objects. By default the repository is located in the directory C:SOFTDISTREPOS on the OS/2 and Windows platforms and /usr/lpp/netviewdm/repos on the AIX platform. The repository on the Mobile Client holds:

  – Objects that were sent to and stored at the Mobile Client.

- Objects that were created at the Mobile Client while disconnected.

- Local Queue

  For the execution of disconnected requests a local request queue is available at the Mobile Client. Requests are placed in the local queue when:

  - Requests are issued locally from the user of the Mobile Client while working in disconnected mode.
  - Requests are issued toward the Mobile Client with the options to install while disconnected.

Figure 7 on page 26 shows the data that is transferred between the server and the Mobile Client and where the data is being stored at the Mobile Client.

3.2 Data Flow between Software Distribution 3.1.3 Server and Mobile Client

In this section we introduce the concept of the Mobile Client and demonstrate how data flows between the server and the Mobile Client. After some general information about the Mobile Client we explain which data is being stored at the Mobile Client during the connection setup to the server and then we present scenarios to show the data flow during change management processing.

3.2.1 General Information about the Mobile Client

To have a better understanding about the data flow we give you some general information about the design of the Mobile Client in this section.

The Software Distribution 3.1.3 Mobile Client is designed to work as a stand-alone application that allows users to update software and data installed on their workstation using Software Distribution 3.1.3 change management functions. A connection to the Software Distribution 3.1.3 server is only required for a short time period to update information required for change management processes.

Typically a Mobile Client is installed on workstations using an asynchronous link or ISDN connection for the communication to a central location. The Software Distribution 3.1.3 Mobile Client addresses the need to keep connection costs low for these workstations by reducing the connection time to a minimum required.

The Mobile Client can perform change management operations required to update software or databases at a workstation in disconnected mode, while the standard client works in connected mode. The following functions can be executed in disconnected mode at the Mobile Client:

- Create and catalog change files in the local catalog
- Import change files from an external device into the local catalog
- Install software
- Remove or uninstall software
- Accept software changes
- Activate the workstation
- View information about objects
- View and control pending requests

Note

The initiate procedure function is not available for the Mobile Client while working in disconnected mode.

The Mobile Client product can operate in two different modes:

- Disconnected mode

  The disconnected mode is the default operation mode after the installation of the Mobile Client product. In this mode the Mobile Client usually carries out change management operations while disconnected from the server and only connects to the server to receive updates and send results of operations back to the server. However, a Mobile Client in disconnected mode can function in the same way as a connected standard client and thus allow the server to perform change management operations in connected mode, as demonstrated in 3.2.3.1, “Data Flow during Immediate Connected Installation” on page 27.

- Permanently disconnected mode

  A Mobile Client in permanently disconnected mode or fully disconnected mode is never connected to a server. Every operation is done disconnected and updates can only be received manually from external devices, such as CD-ROMs, diskettes or tapes. Results of change management operations will not be reported to the server, instead the change management status can be updated manually at the server, using the command:

  ```
  nvdm updcm
  ```

  We show the usage of the fully disconnected Mobile Client including the function to update the status in Chapter 8, “Using a Fully Disconnected Client” on page 121.

  This mode allows workstations that have no network connection at all to be integrated in a centralized change management process using Software Distribution 3.1.3.

In the following sections we only refer to the Mobile Client in disconnected mode and explain the usage of the fully disconnected mode in Chapter 8, “Using a Fully Disconnected Client” on page 121.

The default mode of the Mobile Client for change management processing is the disconnected mode. This means there is no active connection between the server and the Mobile Client. When the Mobile Client is started at system startup it comes up in disconnected mode and does not try to connect to the server, instead it waits for incoming requests from the server. Figure 5 on page 20 shows the messages of the OS/2 agent window during the start of the OS/2 Mobile Client.

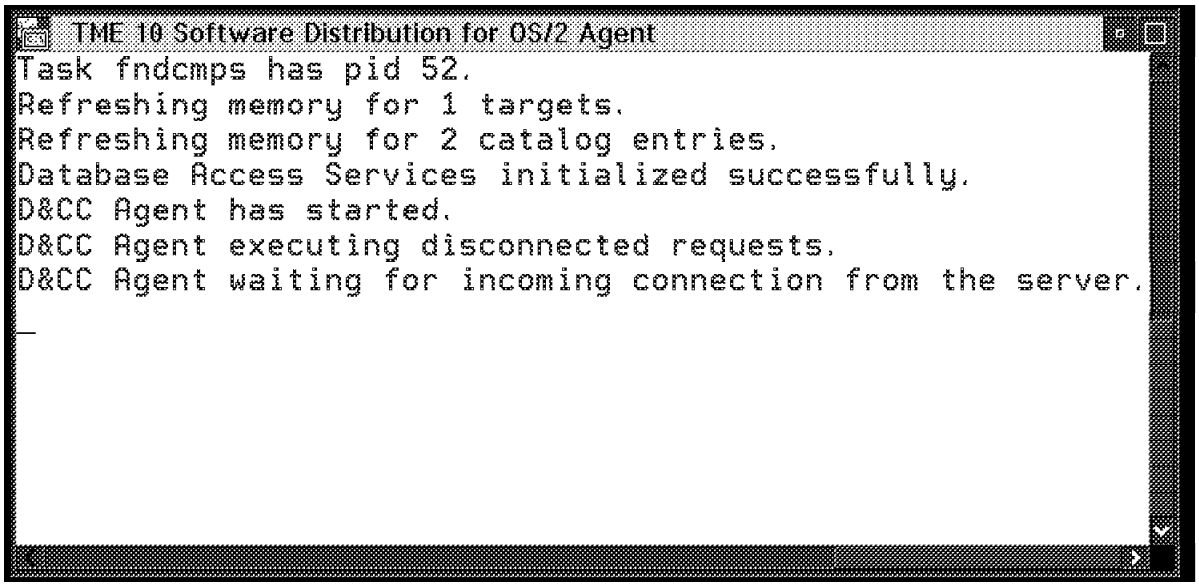


Figure 5. Message Shown at Start of the OS/2 Mobile Client

The first message shows that the D&CC agent program fndcmps is starting with the process identification number 52. The next three messages show that the local target database and catalog are initialized. Then we see the message that the D&CC agent is successfully started and processing disconnected requests, if outstanding. (Read the information provided later in this section and 3.2.3.2, "Data Flow during Disconnected Deferred Installation" on page 31 for more information about this.) After this the agent is ready and waiting for incoming requests from the server.

Note

During the first start of the Mobile Client after the product installation the client connects to the server once to automatically register itself as a target at the server. Refer to 4.5.2, "Using Automatic Client Registration" on page 70 and 5.4.2, "Using Automatic Client Registration" on page 86 for more information about auto-registration.

The client creates a database entry for the first connection to the server, if this entry is not available the client will connect to the server for auto-registration (also see 3.2.3.3, "Data Flow during Automatic Client Registration" on page 36).

A connection has to be established explicitly to process any change management requests initiated from the server or another client toward the Mobile Client. To establish a connection to the Mobile Client mobile1, for our example, the following command must be issued from the command line interface at the server or any other client:

```
nvdm connect mobile1
```

Note

At the time this redbook was written the only server that supported the connect command from the graphical user interface was the OS/2 server.

When the connect command is issued from the command line interface of the Mobile Client it is used by the Mobile Client to specify a time at the server during which the client is available and a connection between the Mobile Client and the server can be established, the connection window. After setting the window the connection is broken and all further communication is initiated from the server.

The following parameters can be specified with the connect command:

```
nvdm connect target -s start -d duration - t phone number

target  specifies the target name for which the connection window is set,
        if not specified the target of the request originator is used.
-s      specifies the start time for the connection window, if
        omitted the current time is used.
-d      specifies the duration for the connection in minutes. If this
        parameter is not specified the default duration of 60 minutes is
        used.
-t      is the telephone number under which the client can be reached.
```

Note

The telephone number is used only with the asynchronous connection. See Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181 for more information.

When the connection window is defined for the Mobile Client the server will schedule two requests:

- Open connection window
- Close connection window

As soon as the open connection window requests is processed a connection to the Mobile Client is established. 3.2.2, “Data Flow during Connection Setup” on page 23 describes the data flow during the connection process.

Warning

We recommend that you always specify the duration parameter and set it to a minimum time required to process outstanding requests.

Notice that the specification of 1 minute for the duration leads to an error. The Open connection window request will expire before getting executed, when -d 1 is specified.

While the default mode for the change management processing is disconnected, the command line interface default mode is server connected. This results in a connection setup to the server for every command that is issued from the command line interface at the Mobile Client. In server connected mode the following command issued at the Mobile Client to check the status of the client will cause a request being processed at the server:

```
nvdm stattg
```

The only commands that will always be processed locally at the Mobile Client, even when the command line interface is server connected are:

```
nvdm start      To start the Mobile Client
nvdm stop       To stop the Mobile Client
```

Note

On a standard client the commands nvdm start and nvdm stop always cause that the server is contacted. The following message is displayed when issuing these commands from the command line interface of a standard client which is connected to server wtr05150:

```
Trying to connect to default server (wtr05150)
Connected to server wtr05150
```

The message shows that the server is connected to process the command.

To change the mode in which the command line interface operates use the command for the server select function as follows:

```
nvdm svr myself
```

The server name myself is a reserved keyword and puts the client into disconnected mode. The command line interface now operates in disconnected mode and every command issued is performed locally at the Mobile Client. The available commands in disconnected mode are limited to the functions that can be performed in this mode. (See 3.2.1, “General Information about the Mobile Client” on page 18 for a list of available functions.)

Notice that the connect command required to establish a connection to the server initiated by the Mobile Client will not work in disconnected mode of the command line interface. The following commands have to be issued at the Mobile Client to connect to the server from disconnected command line interface mode:

```
nvdm svr servername
nvdm connect
```

The first command changes the operation mode for the command line interface to run at the server. If you don’t specify the server name the default server in the configuration file of the Mobile Client is used. The configuration file NVDM.CFG is located by default in the product directory C:SOFTDIST for all OS/2 and Windows clients and in the path /usr/lpp/netviewdm/db for the AIX client.

The second command initiates a connection request to be scheduled at the server for the Mobile Client where the command was originated.

Note

The nvdm svr command results in a connection setup to the server, but this connection only changes the command line interface operation mode and is not sufficient enough to process change management requests. This requires the nvdm connect command.

The nvdm svr command is necessary to know which is the server to connect to later and to have the full command line available.

While the Mobile Client is disconnected it checks its local queue for pending requests at each reboot and executes requests that are due to be processed.

The local queue is also checked when connecting to the server after the synchronization is completed.

Note

The only time request processing is done at the Mobile Client is during the reboot of the workstation which causes a restart of the D&CC agent or when performing a connect to the server.

This is valid for processing of deferred disconnected requests and for any immediate or deferred request issued locally at the Mobile Client. Immediate requests triggered by the server are processed at once, when the connection window is open as shown in 3.2.3.1, “Data Flow during Immediate Connected Installation” on page 27.

On the OS/2, Windows NT, Windows 95 and AIX Mobile Client the restart of the D&CC agent can be forced with the commands:

```
nvdm stop
nvdm start
```

To break an active connection to the server or close the connection window the disconnect command is used as follows:

```
nvdm disconnect target -s start

target  specifies the target name for which the close connection window
        request is executed, if not specified the target of the request
        originator is used.
-s      specifies the start time for the close connection request
        and defaults to immediately if not specified.
```

Before establishing a new connection during an open connection window the Mobile Client must first disconnect.

The following table explains the status a Mobile Client can have at the server:

Table 1. Status of Mobile Client at the Server

Status at server	Explanation
Available	The connection window is open and the synchronization is completed, but the Mobile Client is currently not connected to the server.
Not available	Outside the connection window or not synchronized yet.
Busy	The connection window is open and the Mobile Client is currently connected to the server.

3.2.2 Data Flow during Connection Setup

The following description is intended to help you understand which data is processed during the connection setup between the Mobile Client and the server.

To process requests in disconnected mode the Mobile Client needs information about its target configuration, information about requests to be executed while disconnected and the objects required for installation requests. The server

needs to update its database with the status of operations that were performed at the Mobile Client before transferring new change management requests to the Mobile Client, because of possible dependencies between change management objects.

This process is called synchronization of the client and server database and is always the first action when the client connects to the server. Figure 6 shows the messages at the OS/2 Mobile Client during the synchronization with the server.

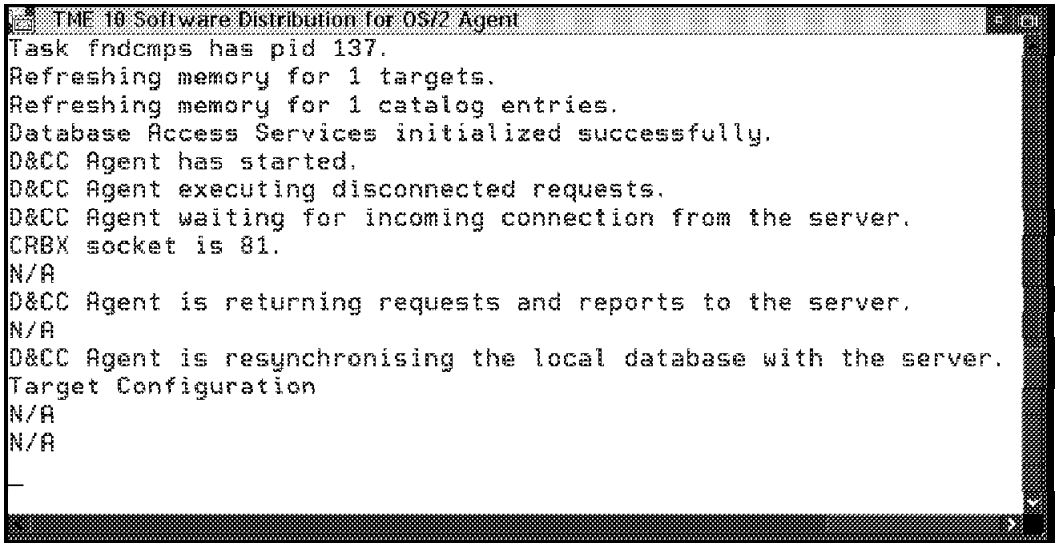


Figure 6. Message during Synchronization of the OS/2 Mobile Client with the Server

During the synchronization of the client database the information listed below is extracted from the server database and stored in the local database of the Mobile Client:

- Installation tokens for the target of the Mobile Client
- Hardware tokens for the target of the Mobile Client
- Change management status of objects in the catalog that are related to the Mobile Client

  New objects that were cataloged at the server for the Mobile Client are added to the local catalog of the Mobile Client. The objects are cataloged during the connect process.

Note

It is important to know that objects are only cataloged at the Mobile Client, but not stored in the repository of the Mobile Client. To execute an installation request disconnected, the Mobile Client also needs the object that is referenced in a catalog entry or else the installation will fail. Therefore it is required that the change management object for a disconnected installation is sent to the Mobile Client before the installation request is issued.

This can be done using the graphical user interface or by executing the following command at the server or any client:

```
nvdm send install.object.ref.1 mobile1
```

Where install.object.ref.1 is the change object and mobile1 the name of the Mobile Client.

No objects need to be stored in the repository of the Mobile Client for the change management requests remove, uninstall and accept. The catalog entry at the local catalog is all that is required to perform these requests disconnected.

During the synchronization of the server database the Mobile Client forwards information about operations processed while disconnected to the server. The database at the server is updated accordingly. The information provided by the Mobile Client includes:

- Reports of change management request
- Requests to catalog objects, that were cataloged locally

  If an object that was created at the Mobile Client while disconnected is not already cataloged at the server, it is cataloged during the connect process.
- All log messages generated at the client since the last synchronization

  Note

  Log messages are only sent to the server if the keyword FORWARD OFF LINE LOGS: YES was specified in the configuration file of the Mobile Client.

  This must be considered carefully for performance reasons.

Figure 7 on page 26 shows the data that is moved when synchronizing the databases of the server and the Mobile Client during the connect process initiated by the nvdm connect command. The data flow shown is numbered and the explanation follows the figure.

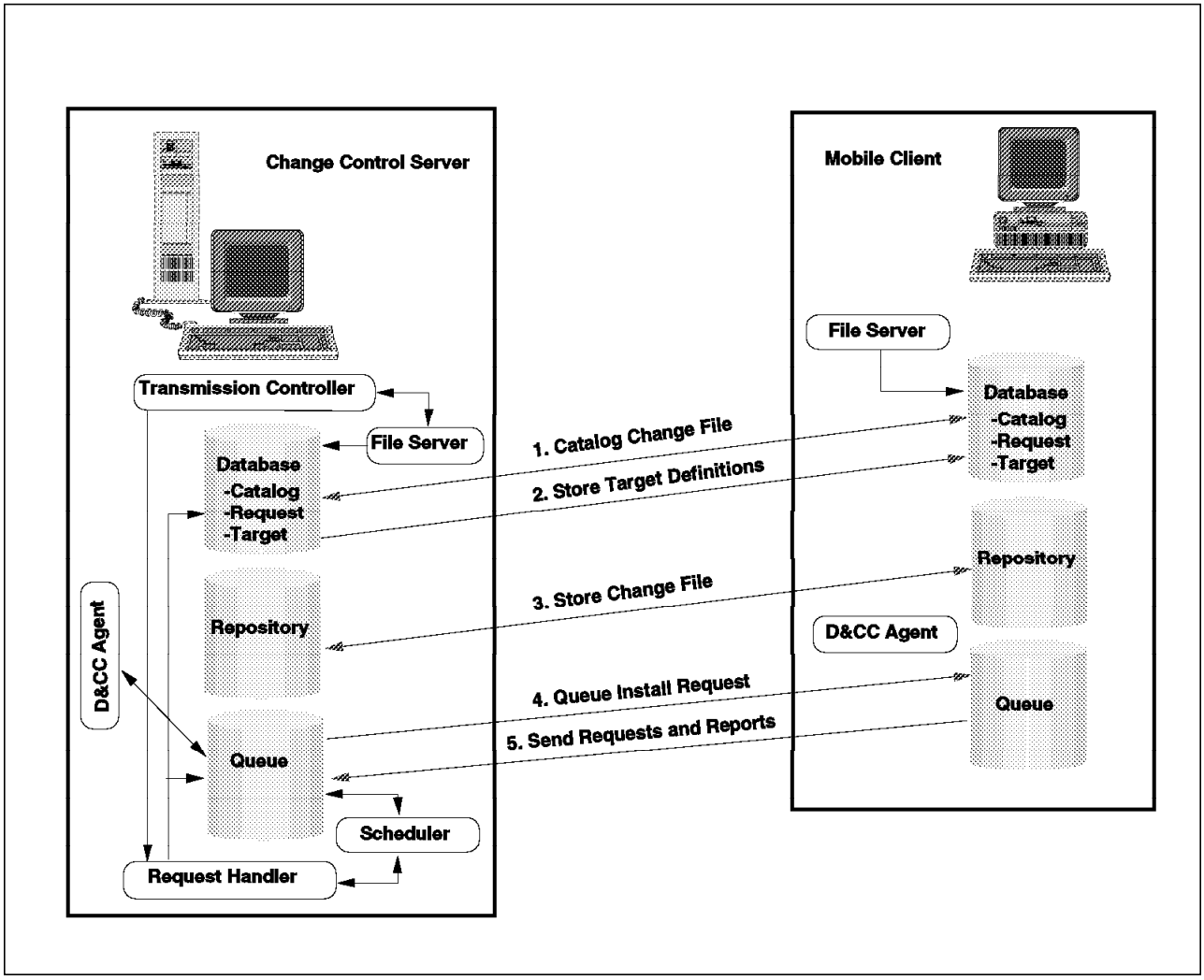


Figure 7. Data Flow between Mobile Client and Server Components during Connection Setup

1. Change files are cataloged in both the servers and Mobile Clients catalog depending on the status of a change object as explained before.
2. The target definitions stored at the server database are always synchronized with the client database during the connect process.
3. When change management objects are sent to the Mobile Client for disconnected installation requests, they are stored in the local repository at the Mobile Client. Remember that the send must be executed explicitly.
4. Any request that is outstanding at the server for disconnected processing at the Mobile Client is sent to the Mobile Client and queued in the local request queue of the Mobile Client.
5. The Mobile Client sends all reports of change management operations performed while disconnected to the server. The results are placed in the schedulers input queue at the server and then sent to the request handler, which stores the results in the database.

Figure 8 on page 27 shows the messages at the OS/2 Mobile Client when cataloging new change objects during the synchronization.

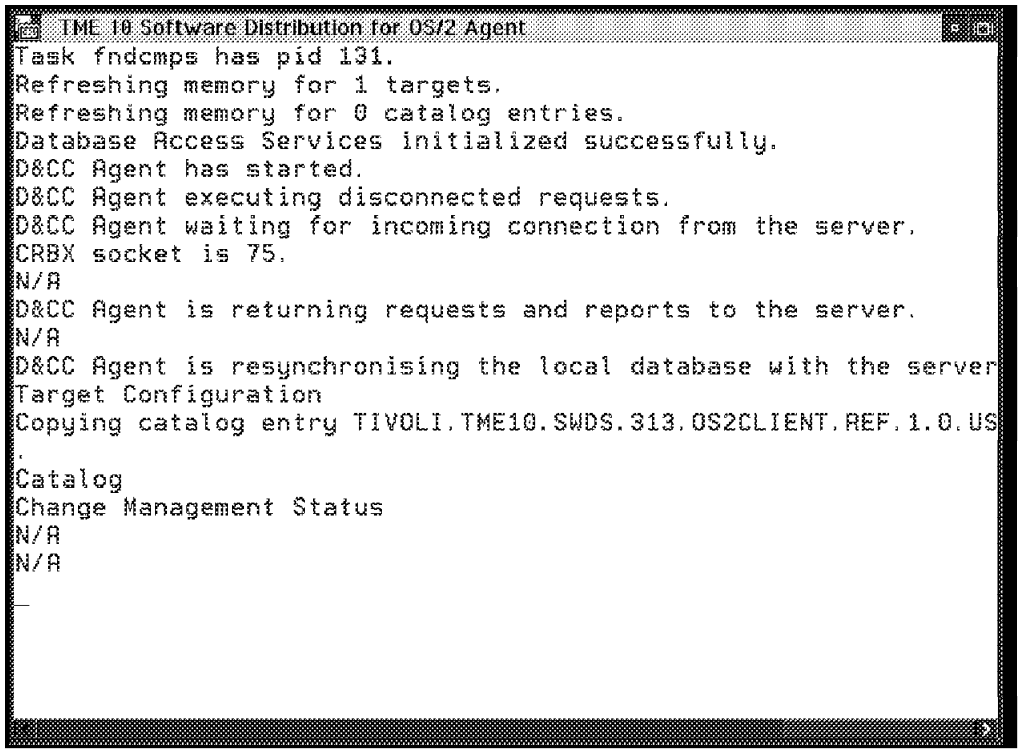


Figure 8. Message at OS/2 Mobile Client - Synchronization with Catalog of New Objects

3.2.3 Data Flow during Change Management

In this section we show scenarios of the data flow between the server and the Mobile Client during change management processes. The following scenarios are covered:

- Immediate connected installation
- Disconnected deferred installation
- Automatic client registration

For each scenario we provide a graphic of the data flow and an explanation about the functions.

3.2.3.1 Data Flow during Immediate Connected Installation

An immediate connected installation is a request for the installation of a change object that is triggered by the server and executed immediately when a connection is established with the Mobile Client. When performing a connected installation the Mobile Client functions in the same way as a connected standard client. The connection between the server and the Mobile Client is active during the installation.

The major difference between the Mobile Client and the connected client is that the request for the Mobile Client is waiting for execution at the server until the connect command is issued either from the server or any other client or from the Mobile Client itself.

Figure 9 on page 28 shows the data flow during an immediate connected installation between the server and the Mobile Client.

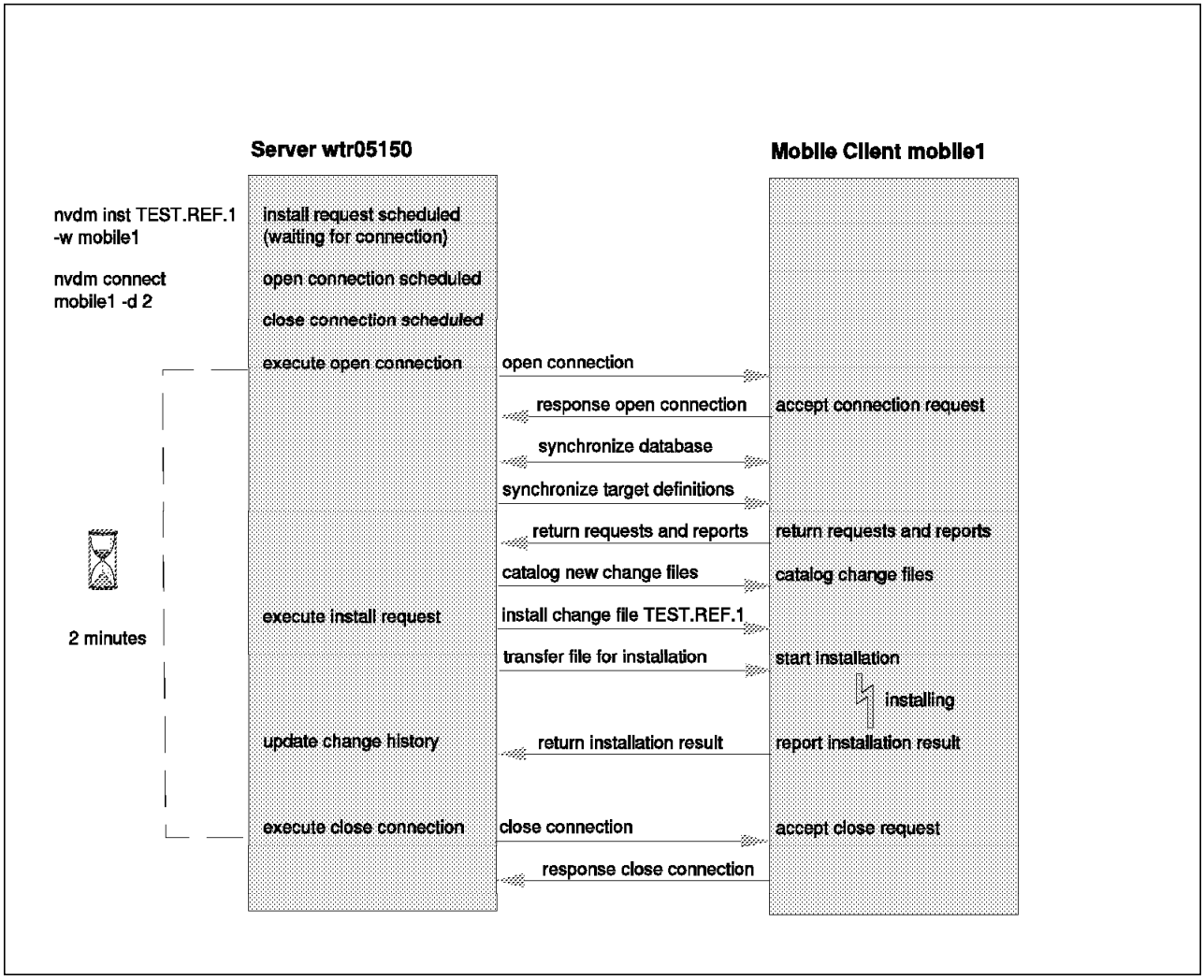


Figure 9. Data Flow - Immediate Connected Installation Using the Mobile Client

The following steps are demonstrated in Figure 9:

1. Initiation of the installation request

   The following command is issued by an administrator at the server or any other client to initiate the installation of a change object with the name TEST.REF.1 on the client mobile1:

   ```
   nvdm inst TEST.REF.1 -w mobile1
   ```

   The installation request for the Mobile Client is scheduled at the server and will not be processed until a connection is established between the server and the client mobile1. The request is waiting for a new connection to the Mobile Client.

2. Connect command

   To establish the connection to the Mobile Client the administrator issues the command to connect to the client mobile1 and sets the connection duration to 2 minutes, just long enough for the install command to get processed. The following command is issued and results in the specification of a connection window for the client mobile1 that is opened at the current time and stays open for 2 minutes:

   ```
   nvdm connect mobile1 -d 2
   ```

3. Connection setup

   As a result of the connect command the server schedules a request to open the connection and a request to close the connection. The open connection request is executed immediately, while the close connection request is scheduled to be due in 2 minutes. The open connection request establishes a connection to the Mobile Client mobile1. The connection is successfully established in this example because the physical communication is available and the Mobile Client is up and running. The Mobile Client sends a response for the open connection request back to the server.

4. Synchronization

   As soon as a connection is active between the server and the Mobile Client the synchronization of the databases at the server and the client is started. During the synchronization the status of change management objects is updated in the clients database. The Mobile Client also retrieves the target definitions for its target from the server database. See 3.2.2, “Data Flow during Connection Setup” on page 23 for an explanation of the synchronization process.

   Reports of change management operations performed disconnected at the Mobile Client are sent to the server by the Mobile Client. Notice there might be none if no reports were outstanding.

   Note

   You will always see the message that shows the synchronization of the reports, even when none are sent:

   `D&CC Agent is returning requests and reports to the server.`

   New change objects that are available for the Mobile Client at the server are cataloged automatically in the local catalog at the Mobile Client. This will only happen if the information about the objects was not already available at the local database of the Mobile Client.

5. Execution of the installation request

   Now that a connection to the Mobile Client is available and synchronization is completed, the installation request that was scheduled at the server is executed. The request is transferred for processing to the D&CC agent of the Mobile Client. Files that are required for the installation are transferred to the Mobile Client and stored temporarily in a work area at the client. The work area is located by default in the directory C:SOFTDISTWORK at the OS/2 and Windows clients and in the path /usr/lpp/netviewdm/work at the AIX client.

6. Installation and reporting of results

   The installation is processed under the control of the D&CC agent at the Mobile Client while the connection window is open. When the installation is completed the installation results are reported back to the server.

   The server receives the results reported by the client and updates the request database and the change management history of the object for the target of the Mobile Client accordingly.

7. Close of the connection

The request to close the connection is executed 2 minutes after the open connection request was executed, since the connection window duration was defined to stay open for 2 minutes.

The close connection request is sent to the Mobile Client which returns a response to show the connection close is accepted. The connection is closed by the server.

Figure 10 on page 31 shows the same change management process with the following differences:

- The connection duration specified (here 2 minutes) is shorter than the installation duration. This demonstrates that the close connection request waits for processing until the running request is completed. The connection will not be closed in the middle of a change management request, yet any following request that is pending at the server will not start processing, even when it was issued before the connect command. The close connection request is executed with a higher priority than any other request. A new connection request is thus required to process following requests.
- No reports are returned to the server, since none were outstanding at the Mobile Client.

  Note

  You will still see the message:

  `D&CC Agent is returning requests and reports to the server.`

  But no reports or requests are sent.

- No change objects are cataloged at the Mobile Client during synchronization, because no new change objects were created for the Mobile Client in this example.

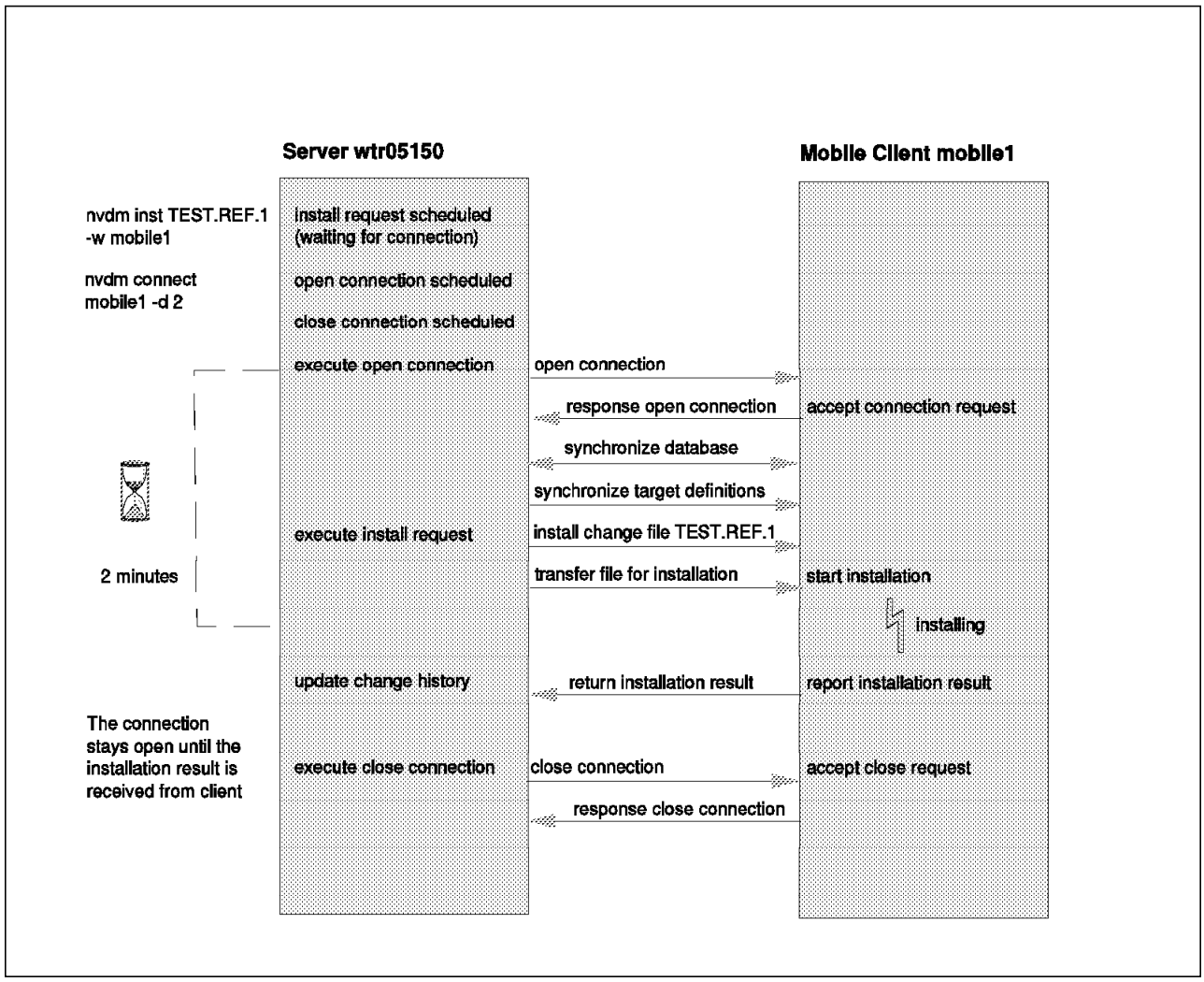


Figure 10. Data Flow - Immediate Connected Installation with Connection Time Shorter than Installation Duration

3.2.3.2 Data Flow during Disconnected Deferred Installation

The typical installation used for a Mobile Client is a disconnected deferred installation, which is also called deadline activation.

The disconnected deferred installation allows an administrator to plan the update of software or data ahead of time for Mobile Clients and thus ensure that the update takes place on the date required. This is an important aspect in the usage of Mobile Clients, since the Mobile Clients are not always physically available it must be possible to distribute files and change management requests once the client is available and execute them even when the client is not connected to a server. The following options are used to perform the deadline activation:

- Disconnected option

  The disconnected installation option specifies that a request is processed at the Mobile Client when it is disconnected from the server, thus avoiding to hold the connection during the installation process. To specify the disconnected option use the parameter:

  `-dc`

- Deferred option

  The deferred option specifies a time or a date when the request is to be processed. If no date or time is specified with a disconnected request, the request is processed immediately when received by the Mobile Client during a connection process. The following parameter can be used to define the deferred option:

  ```
  -db   defines a date at which the request is executed
  -tb   defines a time at which the request is executed
  -da   defines a date at which the request is aborted, if not yet executed
  -ta   defines a time at which the request is aborted, if not yet executed
  ```

A disconnected request without the deferred option is useful when the installation duration takes a long time so it is desirable for the connection window to close before the request is completed. If such a request is issued as an immediate request (without the disconnect option) the connection window will not close until request processing is completed as shown in Figure 10 on page 31.

We demonstrate the process of a disconnected deferred installation request in Figure 11 on page 33 and Figure 12 on page 35 and explain the data flow following each figure. The graphic was split into two figures because of its size.

Figure 11 on page 33 shows the setup of the disconnected deferred installation request.

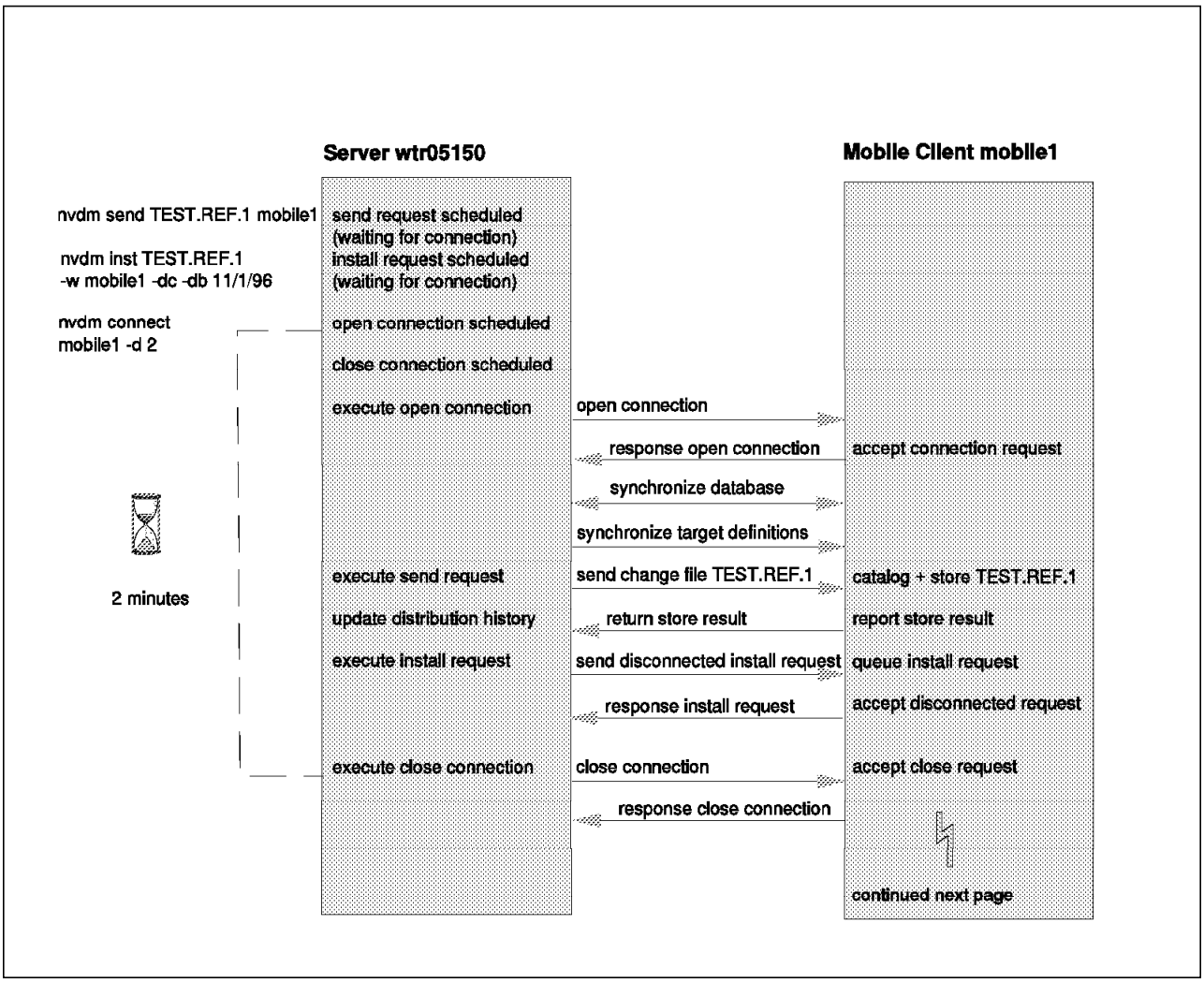


Figure 11. Data Flow - Disconnected Deferred Installation Using the Mobile Client - Part One

The following process is shown in Figure 11:

1. Send of the installation object

   Before executing the disconnected installation request the change management object is sent to the Mobile Client. As described in 3.2.2, “Data Flow during Connection Setup” on page 23 the change object to be installed disconnected must be available in the local repository of the Mobile Client. In our scenario an administrator sends the object with the name TEST.REF.1 to the client mobile1 before the installation, using the command nvdm send TEST.REF.1 mobile1. The send request is scheduled and waiting for a new connection to the Mobile Client mobile1.

2. Initiation of the disconnected installation request

   To initiate a disconnected deferred installation request for the change management object TEST.REF.1 at the client mobile1, the following command is issued by an administrator:

   ```
   nvdm inst TEST.REF.1 -w mobile1 -dc -db 11/1/96
   ```

   The parameter -db 11/1/96 specifies that the installation is executed on November 1, 1996. After entering the command the installation request is scheduled at the server and waiting for a connection to the client mobile1.

3. Initiation of the connect request

   A connect request is initiated by an administrator from the server in our scenario to allow the disconnected request to be transferred to the Mobile Client for processing. Since a Mobile Client is not always connected to the network the installation request is transferred before it is due to be processed. To open a connection window with a duration of 2 minutes for the client mobile1 the administrator issues the following command:

   ```
   nvdm connect mobile1 -d 2
   ```

4. Connection setup

   The server schedules two requests following the connect command, one request to open the connection, which is executed immediately and one request to close the connection, which is delayed. The connection to the client mobile1 is established when the server receives the response for the open connection from the Mobile Client.

5. Synchronization

   The first action during the connect is the synchronization of the databases at the server and the client. The status of change management objects is updated in both databases and target definitions for the target mobile1 are retrieved from the server database. See 3.2.2, “Data Flow during Connection Setup” on page 23 for more information about synchronization.

6. Execution of the send request

   The send of the object TEST.REF.1 that was scheduled earlier is now being processed. The object is transferred to the Mobile Client and stored in the local repository by the D&CC agent at the Mobile Client. A report with the result of the store operation is sent back to server immediately during the open connection and the distribution history of the object is updated for the target mobile1 at the server.

7. Transfer of the disconnected installation request

   The installation request is processed as soon as the send request is completed. Since we are executing a disconnected request the request is transferred to the local queue of the Mobile Client and scheduled for later execution by the D&CC agent of the Mobile Client. The D&CC agent sends a report back to the server when the request is queued.

8. Close of the connection

   When the connection duration of 2 minutes is over the close connection request is executed by the server. The request to close the connection is sent to the Mobile Client and the connection is closed after the response of the Mobile Client to this request is sent back to the server.

Figure 12 on page 35 shows the processing of the disconnected deferred installation when the specified date is reached. In our scenario, the request is due on November 1, 1996 and we assume that the user of the workstation reboots the system on that date.

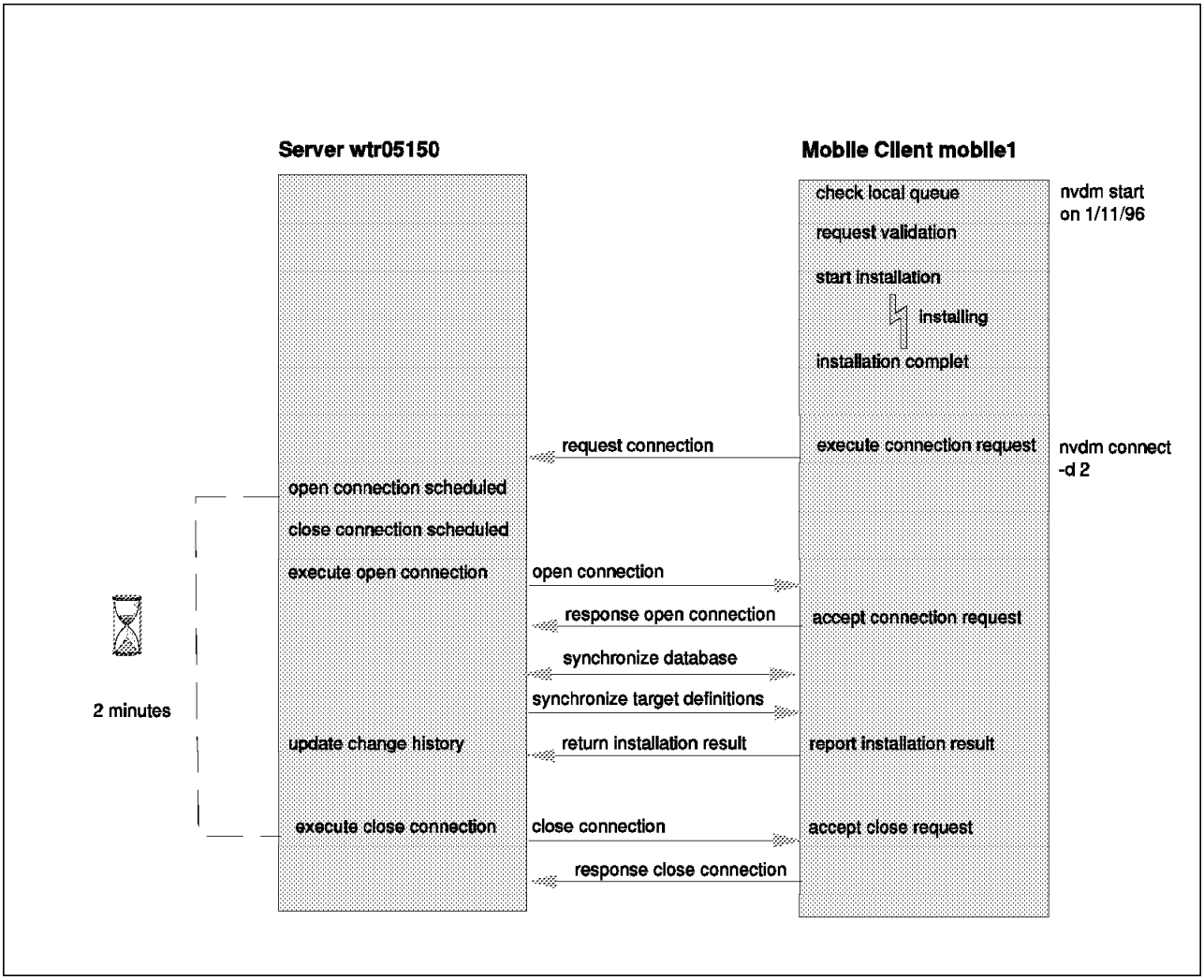


Figure 12. Data Flow - Disconnected Deferred Installation Using the Mobile Client - Part Two

1. Check of the local queue at restart

   The reboot of the system implies an nvdm start. During the restart the D&CC agent program fndcmps checks for requests in the local queue and processes the requests that are due. In our scenario the installation of the object TEST.REF.1 is due on the day of the reboot and thus immediately processed for execution by the agent program.

2. Request validation

   Before the request execution is in disconnected mode the request is validated by the D&CC agent at the Mobile Client. Request validation includes:

   - Check if the change management status is valid.

     The check is done using the change management status information in the local database.

   - Check if software pre-requisites are met.

     If software pre-requisites are specified in the change object, the local history database is used to check if they are met.

   - Check if hardware pre-requisites are met.

If hardware pre-requisites are specified in the change object the local target configuration database is used to check if they are met.

3. Installation of the disconnected deferred request

   The installation of the change object TEST.REF.1 is started after the validation. In our scenario the validation was successful and the installation is executed. When the installation is completed the installation results are queued in the local database. The results are transferred to the server during the next connect process.

4. Initiating a connection from the Mobile Client

   In our scenario we assume that the user of the Mobile Client issues a connect command some time after the installation, using the command:

   ```
   nvdm connect -d 2
   ```

   A connection window with a duration of 2 minutes is specified and a request to open the connection is sent to the server. Notice that no target name is specified since the connect request is for the Mobile Client where the command is issued.

5. Connection setup

   When the connection request from the Mobile Client mobile1 is received at the server, the server schedules the requests to open the connection and to close the connection. The open connection request for mobile1 is executed immediately and the connection is established with the response to that request from the mobile1. The close connection request is scheduled to be due 2 minutes after the open connection request.

6. Synchronization

   When the connection is established the database at the server and the local database at the client are synchronized. After updating the status of change management objects and the target definitions, the Mobile Client sends the result of the disconnected installation request back to the server. The server updates the change management history of the object TEST.REF.1 for the target mobile1 accordingly.

7. Close of the connection

   The close connection is executed when the specified time is reached and the close request responded by the Mobile Client.

3.2.3.3 Data Flow during Automatic Client Registration

We demonstrate the process of automatic client registration for the Mobile Client in the next scenario, because it differs from change management processing using the Mobile Client in the following way:

- The automatic client registration is only processed once, when the client is started for the first time and has not connected the server yet or when the server name defined at the client is changed. See 4.4.1, “Changing the Communication Protocol and/or Server Name” on page 61 for more information about changing the server name.
- The inventory discovery process is started automatically by the server after auto-registration of a client.

Figure 13 on page 37 shows the process of the first initial start and automatic registration of an OS/2 Mobile Client. Refer to 4.5.2, “Using Automatic Client Registration” on page 70 and 5.4.2, “Using Automatic Client Registration” on

page 86 for more information about the automatic client registration function. In this scenario we assume that the Mobile Client is automatically started with the nvdm start command at the first reboot after the product installation. See 4.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 45 for an example of the installation of the OS/2 Mobile Client.

```
wtr05118 43 FNDC0015I: Task nvdm has pid 43.
wtr05118 43 FNDRX020I: Attempting to connect to server wtr05150 on port 729
wtr05118 43 FNDRX001I: New connection 8 from client wtr05118 agent 3.
wtr05118 43 FNDCL785I: The product started successfully.
wtr05118 47 FNDC0015I: Task fndcmps has pid 47.
wtr05118 47 FNDDB005I: Refreshing memory for 1 targets.
wtr05118 47 FNDDB006I: Refreshing memory for 0 catalog entries.
wtr05118 47 FNDDB007I: Database Access Services initialized successfully.
wtr05118 47 FNDCM644I: D&CC Agent performed a cold start.
wtr05118 47 FNDCM645I: D&CC Agent executing disconnected requests.
```

Figure 13. Data Flow - Automatic Client Registration Using the Mobile Client

The following sequence is processed:

1. Check of server definition during restart

   When the Mobile Client is started with the command line interface command it always checks if the server is defined, with the following consequences:

   - Server is defined

     If the server is defined it indicates that the client already registered with the server. In this case the client does not try to connect to the server.

   - Server is not defined

     If the server is not defined, the client has not registered itself as a target at the server yet. Thus the client automatically connects with its command line interface to the server defined in the client configuration file and registers itself.

   In our scenario we start the Mobile Client for the first time, so the server is not yet defined. The name of the server is wtr05150 and the name of the Mobile Client is wtr05118 in our example.

2. Connect to the server with auto-registration request

   The client wtr05118 automatically connects the server wtr05150 with an auto-registration request using its command line interface.

   Note

   Notice that during this connect the client only initiates a request from its command line interface at the server, no synchronization or request processing is done as seen with the nvdm connect command.

   During the auto-registration the client transfers the following information provided in its configuration file to the server:

   - Workstation name indicated by the keyword WORKSTATION NAME
   - Protocol being used indicated by the keyword PROTOCOL
   - Type of target indicated by the keyword CONFIGURATION
   - Client operation mode indicated by the keyword TARGET MODE

- Operating system of the client indicated by the keyword MACHINE TYPE
- Message log level indicated by the keyword MESSAGE LOG LEVEL
- Target address for the client indicated by the keyword TARGET ADDRESS

The configuration file NVDM.CFG is located by default in the product directory C:SOFTDIST for all OS/2 and Windows client platforms and in the path /usr/lpp/netviewdm/db for the AIX client. Also see Table 2 on page 71 for the mapping of the client configuration keywords to the target definition at the server.

3. Auto-registration

   Using the information sent by the Mobile Client with the auto-registration request the server adds a target definition to its database for the client requesting the connection.

   Note

   The auto-registration must be allowed for the server, by setting the following keyword in the servers configuration file:

   ```
   AUTOMATIC TARGET REGISTRATION: YES
   ```

   The client is now successfully registered.

4. Scheduling of the inventory discovery request

   For each client that is newly defined at a server, either manually or through auto-registration, an inventory discovery request is immediately scheduled by the server. The inventory discovery request is used to extract the configuration information about the target from the client workstation. Also see 10.5.2, “Creating an Inventory File” on page 175 for more information.

   The server schedules the inventory request for the Mobile Client after the auto-registration. The request is waiting for processing until a connection is established with the client.

5. Connection setup

   In our scenario an administrator issued the nvdm connect command some time later to set up a connection with the Mobile Client. The server schedules the requests to open and close the connection with the newly registered Mobile Client, where the open connection request is executed immediately and a connection with the Mobile Client is established by the server. The close connection request is scheduled for later execution.

6. Synchronization

   When the connection is opened between the server and the Mobile Client the synchronization of the server and the client database is performed. The status of change management objects is updated in the database of the server and in the local database of the Mobile Client and target definitions for the target of the Mobile Client are retrieved from the server database. See 3.2.2, “Data Flow during Connection Setup” on page 23 for an explanation of the synchronization process.

   Change objects that are available for the Mobile Client at the server are cataloged automatically in the local catalog at the Mobile Client.

7. Execution of inventory discovery request

The inventory discovery process is executed when the synchronization of the databases is completed. Depending on the inventory program that is defined at the Mobile Client the following is performed:

- If no inventory program was specified, the inventory files FNDSWINV, FNDHWINV and FNDTKINV are read at the Mobile Client and the database at the server is updated as follows:
  - The change management history database is updated according to the information in the software inventory file FNDSWINV. A catalog entry is created for each object in the FNDSWINV file that did not already exist at the server.
  - The target workstation database is updated according to the information in the hardware inventory file FNDHWINV, if the file exists.
  - The target configuration database is updated according to the information in the installation token inventory file FNDTKINV, if the file exists.
- If an inventory program was specified, the program is executed under the control of the D&CC agent of the Mobile Client, the inventory files are created and processed as mentioned above.

See 4.4.4, “Defining an Inventory Program” on page 65 for the usage definition of an inventory program.

8. Close of the connection

   The close connection request is executed by the server when the default duration time of 60 minutes is passed. Notice that the default duration is valid here because the connect command was initiated by the administrator without specifying a duration time. The connection is closed when the request is acknowledged by the Mobile Client.

Part 2. Setting Up a Mobile Client Environment

This part describes the steps required to install the TME 10 Software Distribution Mobile Client product on the available client platforms and explains how to connect the Mobile Client in different environments to different servers.

TME 10 Software Distribution Mobile Clients are available on the following client platforms:

- OS/2
- Windows NT
- Windows 3.1
- Windows 95
- AIX

Note

We do not cover the Mobile Client for AIX in this part because we think in most cases the desktop or laptop clients are more likely to run either OS/2 or Windows.

However, you will find a complete description on how to set up the Mobile Client feature on the AIX platform in Appendix A, “Installing and Configuring the Mobile Client for AIX” on page 353.

The environments we show in this part are:

- OS/2 LAN environment with:
  - Software Distribution for OS/2 server
  - OS/2 Mobile Client
  - Windows 3.1 Mobile Client
- AIX LAN environment with:
  - Software Distribution for AIX server
  - OS/2 Mobile Client
  - Windows NT/95 Mobile Client

© Copyright IBM Corp. 1997

Note

In this part we give you some general information about installing and configuring Mobile Clients in a rather simple LAN environment. If you are going to set up your clients in an environment involving, for example, modem connections, you should also refer to Part 4, “Advanced Scenarios Using the Mobile Client” on page 119.

Normally, you will be more likely to use a Mobile Client remotely or at least in a WAN/LAN combination. However, the information in this part can still be applied to these environments.

Sometimes the configuration steps in this chapter can even be used without any change. For example, when we use the Mobile Client with the IBM 8235 in Chapter 10, “Using the Mobile Client with IBM 8235 LAN Dialer” on page 165 the client does not realize that this is not a ″real″ TCP/IP LAN environment. Instead, the remote connection is completely transparent to the Mobile Client.

Chapter 4. Mobile Clients in an OS/2 LAN Environment

Mobile Clients can be installed using various configurations. One basic configuration where the Mobile Client can be installed is a stand-alone OS/2 LAN, for example, travelling users who connect their laptops to a LAN using a docking station to receive software updates. Figure 14 shows the environment we are setting up in our examples.

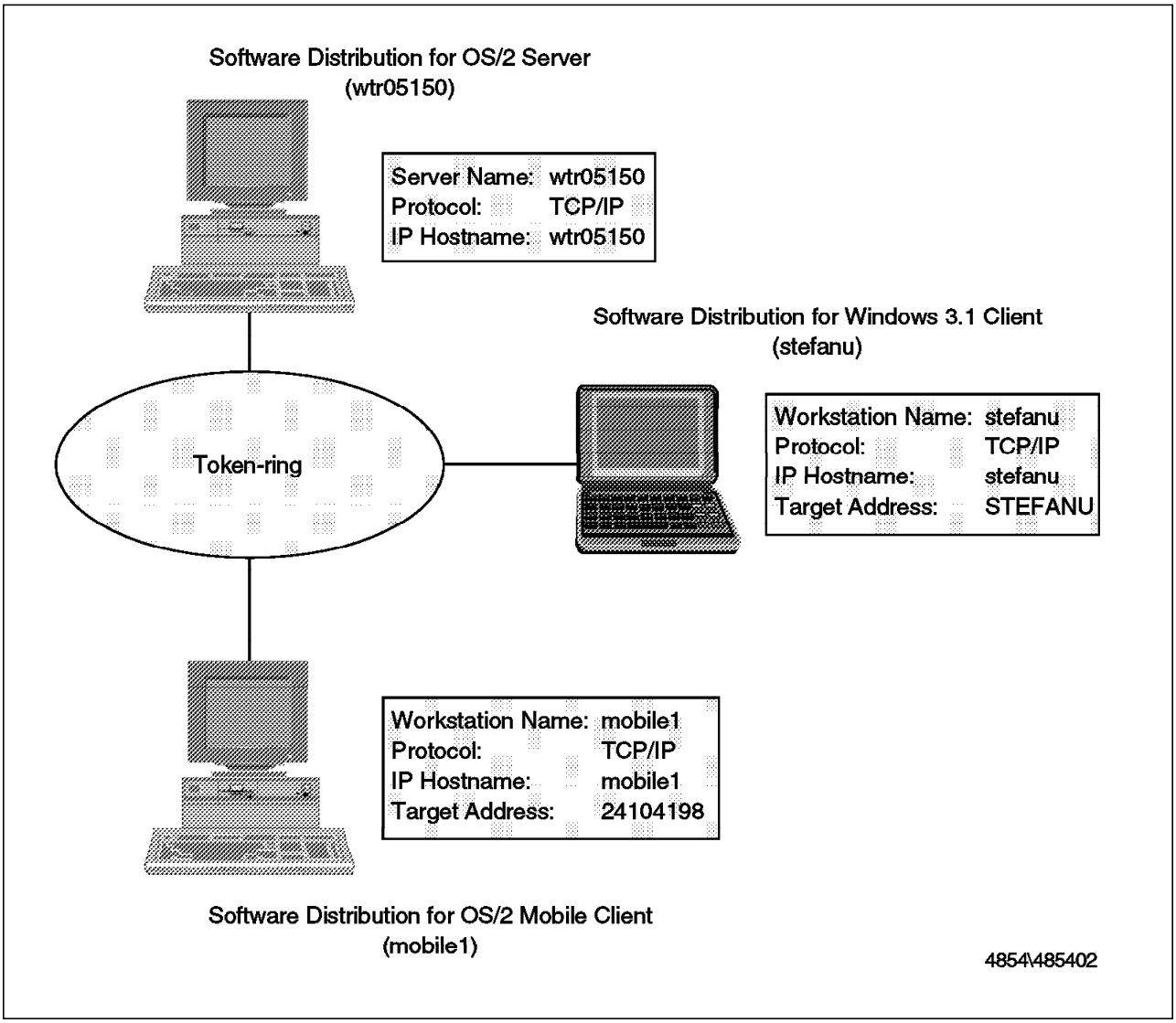


Figure 14. OS/2 LAN with OS/2 and Windows 3.1 Mobile Clients

4.1 Overview

The purpose of this chapter is to give guidelines on the setup of an OS/2 LAN with Mobile Clients connected to it. For the different Mobile Client platforms used in our environment we guide you through the installation of the client product, give you information about the configuration of the client and show you how to set up the Software Distribution for OS/2 server.

© Copyright IBM Corp. 1997

We perform the following steps:

- Install and configure the Mobile Client for OS/2
- Install and configure the Mobile Client for Windows 3.1
- Define both clients to a Software Distribution for OS/2 server

Note

We show how to define the OS/2 client at the server manually and use auto-registration to define the Windows 3.1 client.

4.2 Installation and Setup of the OS/2 Mobile Client

The following outlines the specific installation steps and customization aspects related to the OS/2 Mobile Client. The first section gives an overview of the installation requirements, the second section describes the interactive installation and the third section the installation using response files.

For a full description of the TME 10 Software Distribution client installation procedure see *Tivoli TME10 Software Distribution Clients Installation and Configuration*, SH19-4337.

4.2.1 Installation Requirements

The list below shows the hardware and software requirements needed for the TME 10 Software Distribution for OS/2 Mobile Client.

4.2.1.1 Hardware Requirements

- Minimum of 386 processor
- 16 MB RAM
- 25 MB disk space
- 5 MB disk space needed temporarily for the installation
- Token-ring or Ethernet card

4.2.1.2 Software Requirements

- OS/2 2.11 or OS/2 Warp 3.0 or later
- Multiprotocol Transport Services (MPTS) 1.0 or later (LAPS level WR08000) for the NetBIOS communication protocol support
- IBM TCP/IP for OS/2 2.0 with CSD UN64092 or later installed for the TCP/IP communication protocol support
- Novell NetWare Requester 2.10 or later for the IPX/SPX communication protocol support
- Communication Server, or Communication Manager (CM/2) 1.11 or later for the APPC communication protocol support

4.2.2 Interactive Installation of the OS/2 Mobile Client

In our example we are installing the OS/2 Mobile Client on mobile1, which is connected to the Software Distribution for OS/2 server wtr05150 using the TCP/IP protocol. The TCP/IP communication including the TCP/IP name resolution is already configured in our environment and therefore not further mentioned in this section.

We use the following hardware and software components on mobile1:

- IBM Value Point PC Model P60/D, equipped with:
  - Intel Pentium Processor
  - 32 MB RAM
  - 405 MB Hard Disk
  - IBM 16/4 Token-Ring Adapter
- IBM OS/2 Warp Version 3.0
- MPTS Version 2.0
- IBM TCP/IP for OS/2 Version 3.0

To install the OS/2 Mobile Client interactively from CD-ROM follow these steps:

1. Insert the CD-ROM.
2. Change to the directory SD4OS2IMAGES.
3. Type `install`.
4. Select **Continue** in the Instructions window.
5. In the Install window select **Update CONFIG.SYS** and choose the **OK** push button.
6. In the Install - directories window you can select several components for the installation. To install the OS/2 Mobile Client select the component **Distribution Mobile Client (with GUI)** as shown in Figure 15 on page 46.

   Also you can select the component **Hw/Sw Discovery Tool** if you want to use the NetFinity scanners supplied with TME 10 Software Distribution for the hardware and software inventory function.

   Selecting the component **Hw/Sw Discovery Tool** installs the NetFinity scanners in the BIN product directory.

   Note

   The NetFinity scanners are used for hardware and software inventory checking when the statement INVENTORY PROGRAM: FNDINV is added to the NVDM.CFG of the client.

   We install the Hw/Sw Discovery tool in our example and show the usage of the tool with the Mobile Client in 10.5.2, “Creating an Inventory File” on page 175. Also see 4.4.4, “Defining an Inventory Program” on page 65.

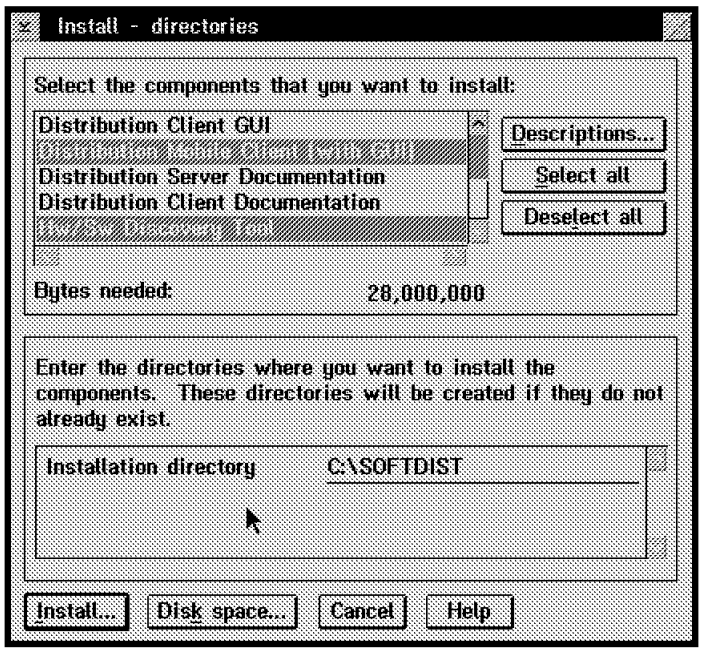


Figure 15. Software Distribution for OS/2 Install - Directories Window

Version Information

With the initial Version 3.1.3 the only component that can be selected with the installation of the OS/2 Mobile Client is the Hw/Sw Discovery tool. There is no online documentation available for the Mobile Client. Selecting the component **Distribution Client Documentation** will lead to the message `EPFIQ184: Installation of the 'Distribution Client Documentation' component requires that the 'Distribution Client' component also be installed.`

The graphical user interface for the Mobile Client and the Software Preparation GUI are installed automatically with the Mobile Client.

However, with CSD XR21052 installed you will be able to select other components, such as online information when installing the Mobile Client. The selection window for the components to be installed will look slightly different from that shown above.

Please check with your IBM representative to determine the version you are running.

7. The default directory where the components are installed is C:SOFTDIST. You can change the name of the drive and the directory simply by overtyping the values in the field Installation directory or change the drive name by selecting the **Disk space...** push button.
8. To check the available disk space click on **Disk space...**. Optionally select a different drive and set the Change directories to selected drive. Click on **OK** to accept the values.
9. Select **Install** in the Install - directories window to start the installation. A progress indicator shows the ongoing progress of the installation and the files being copied during the installation.
10. During the installation the Distribution Configuration notebook is displayed as shown in Figure 16 on page 47.

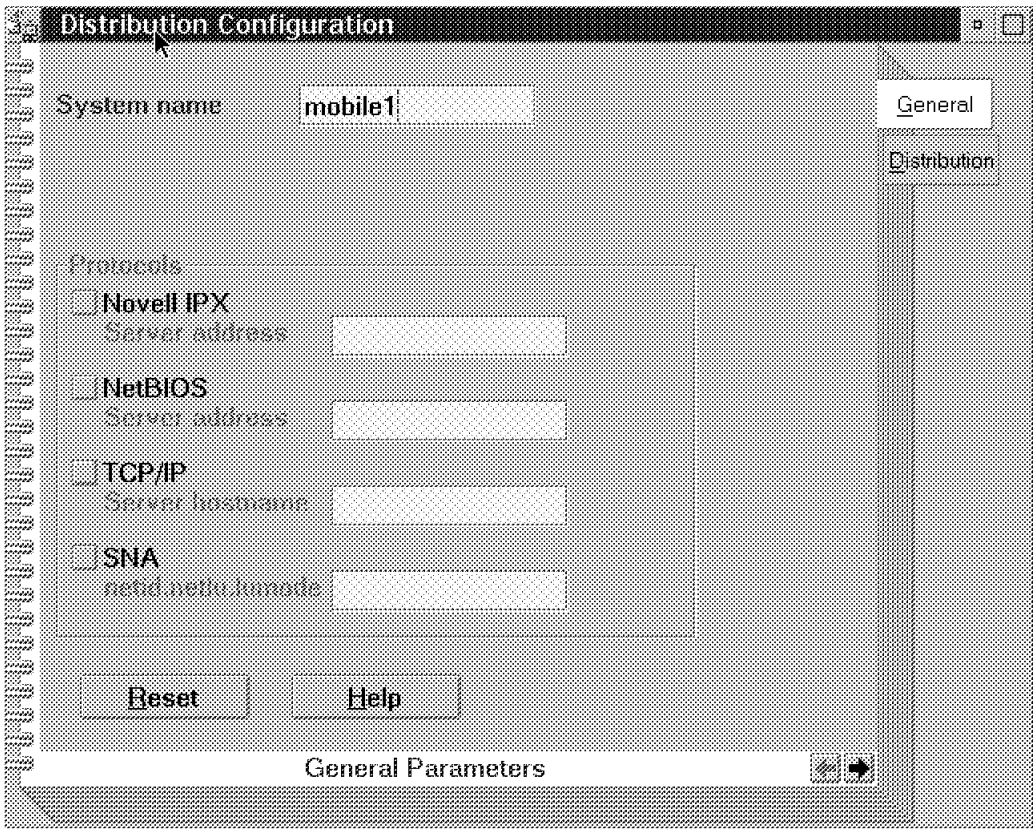


Figure 16. Software Distribution for OS/2 Client Distribution Configuration Notebook General Page

The System name is the workstation name of the Mobile Client and the name under which the client is known at the Software Distribution for OS/2 server. The name you specify here is case sensitive and can be the same as the TCP/IP hostname or NetBIOS name, but this is not required. In our example we choose the system name to be the same as the TCP/IP hostname and set it to `mobile1`.

Note

The Protocols fields cannot be selected because the protocol used for the client depends on the protocol used for the communication with the server, which are defined in the next step.

11. Go to the Distribution Configuration window by selecting the next page of the notebook or the **Distribution** tab.

    In the Distribution Configuration window specify the workstation name of the Software Distribution for OS/2 server in the field System name and the protocol used for the communication between client and server in the field Network Driver as shown in Figure 17 on page 48. In our example the server name is `wtr05150` and the communication protocol used is TCP/IP.

    For the TCP/IP protocol the Network Address must be the TCP/IP hostname of the Software Distribution for OS/2 server, which is `wtr05150` in our example.

    Close the notebook by selecting the close button in the upper left corner of the window.

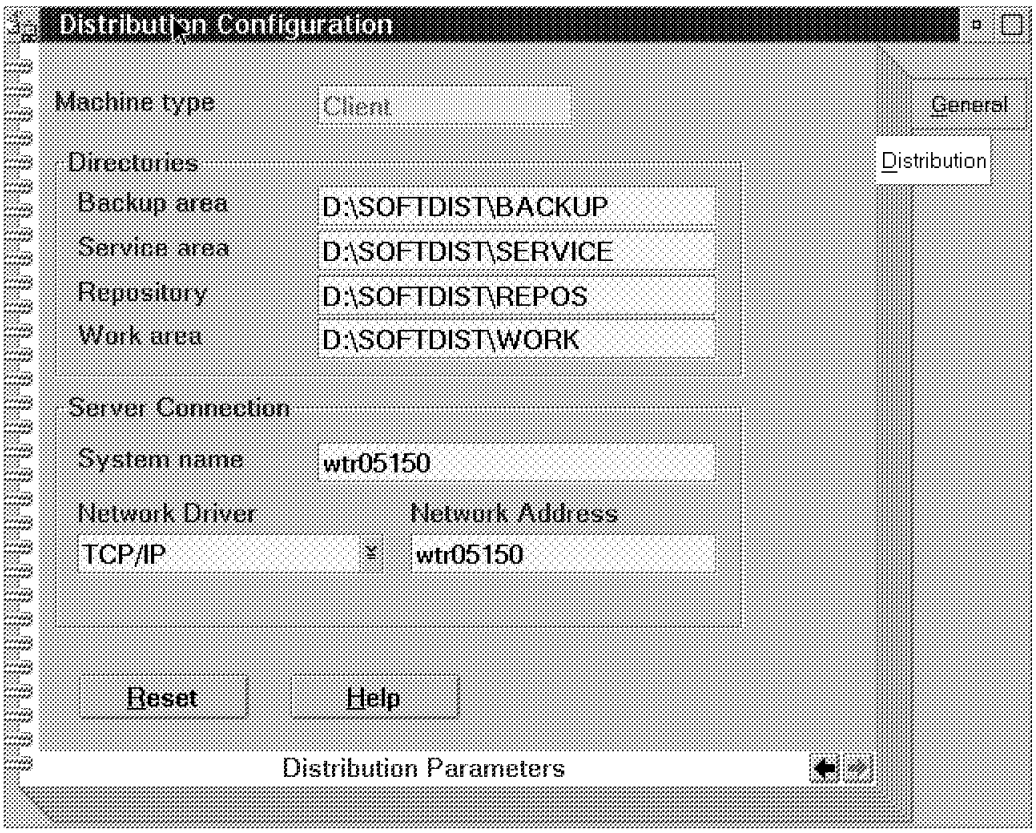


Figure 17. Software Distribution for OS/2 Client Distribution Configuration Notebook Distribution Page

Note

We show how to use the asynchronous support available in TME 10 Software Distribution in Chapter 11, "Using the Asynchronous Protocol Support for Mobile Clients" on page 181. You should notice that at the moment you cannot specify to use asynchronous communications for a client using the configuration notebook shown above. Instead you will have to edit the NVDM.CFG file.

12. Select **Yes** to save the changes in the notebook and continue the installation.
13. A pop-up message will show that the new configuration will not be used until the product is restarted. Select **OK** here.
14. When the installation is finished a pop-up message will inform you that a reboot of your system is required to activate changes in the CONFIG.SYS file. Select **OK** to end the installation process.
15. Select **Exit** to leave the installation window and reboot the system so that changes made in the CONFIG.SYS can be activated.
16. After the installation an icon appears on the OS/2 desktop for the TME 10 Software Distribution for OS/2 folder.

    The folder contains the icons for stopping and starting the product, the icons for the Software Preparation GUI and the graphical user interface as shown in Figure 18 on page 49.

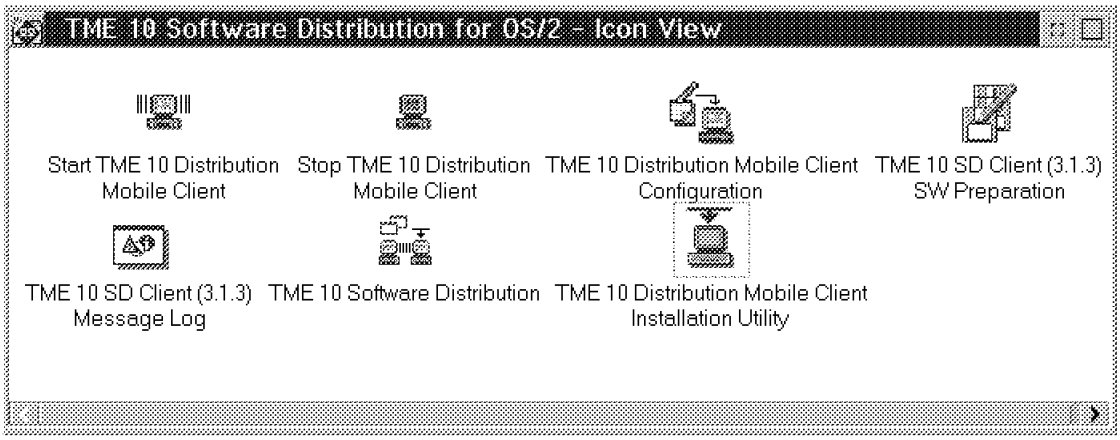


Figure 18. TME 10 Software Distribution for OS/2 Folder

The OS/2 Mobile Client is now ready for use. The Software Distribution agent program FNDCMPS will start automatically at system startup, because the entry Start TME 10 Distribution Mobile Client is placed in the OS/2 Startup folder during the installation.

You can see a task with the name TME 10 Software Distribution for OS/2 Agent in your OS/2 task list. Click on the task window to see messages indicating a successful start of the agent as shown in Figure 19.

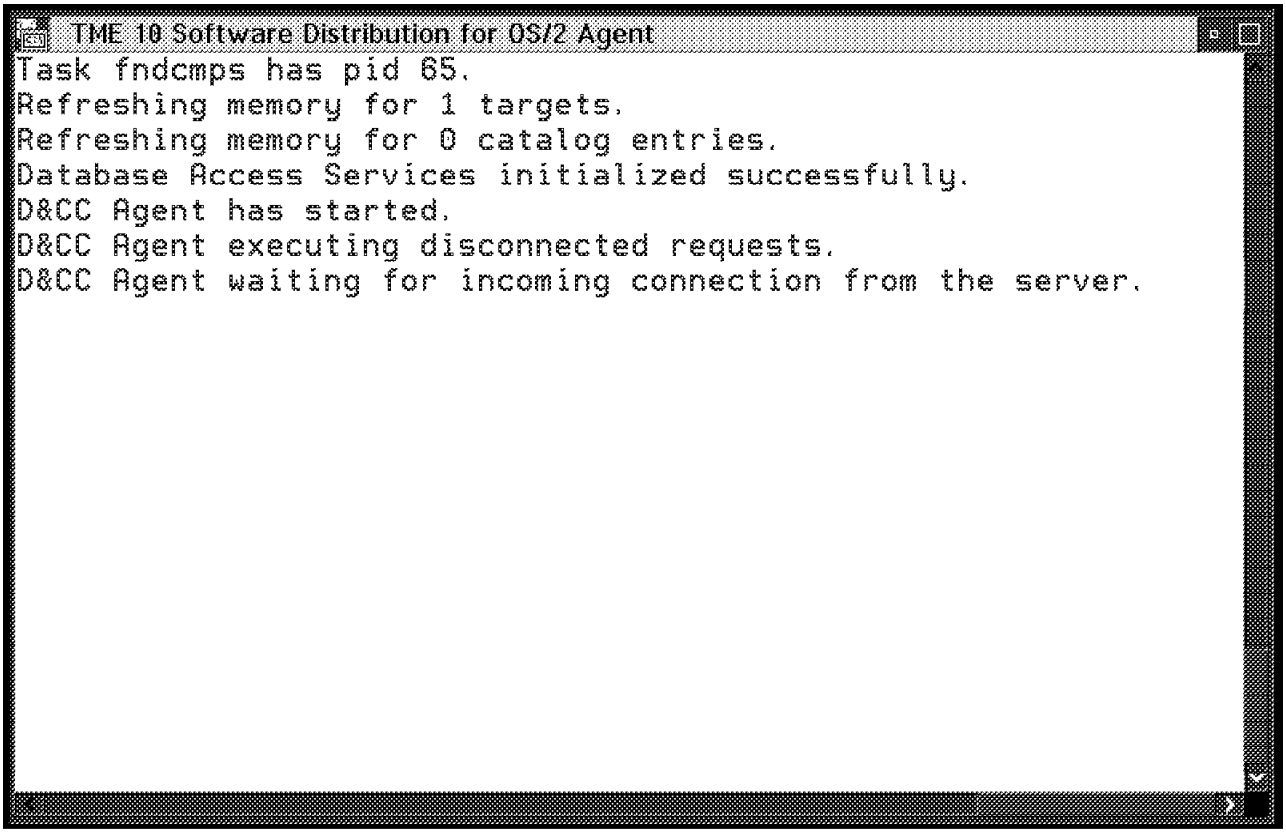


Figure 19. OS/2 Mobile Client Startup Message

4.2.3 Response File Driven Installation the OS/2 Mobile Client

The following describes how to install the OS/2 Mobile Client using a response file.

You can use a response file to install the Mobile Client in batch mode, that is without any user interaction required. This is especially useful when you intend to install many clients.

To install the product in batch mode you must create a response file to supply the user responses you would normally give in interactive mode.

In the directory SD4OS2IMAGES there is a sample response file SDISTCLT.RSP that you can modify for your environment.

The following response file is used to perform the installation described above in batch mode:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; DESCRIPTION:  TME 10 Software Distribution (3.1.3) for OS/2 Client         ;;
;;               sample configuration                                         ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Target path
FILE = C:\SOFTDIST

; Work area
; It is the path for the data directory
WORK = C:\SOFTDIST\WORK

; Software Distribution components to install
COMP = Distribution Mobile Client (with GUI)
COMP = Hw/Sw Discovery Tool
; COMP = Preparation Site Client
; COMP = Distribution Client GUI
; COMP = Distribution Client Documentation
; COMP = Distribution Mobile Client (with GUI)
; COMP = Hw/Sw Discovery Tool

DELETEBACKUP = No
SAVEBACKUP   = Yes
CFGUPDATE    = Auto
OVERWRITE    = Yes

; Software Distribution System Name. This identify the system in the network
; mandatory
SystemName   = mobile1

; Network drivers and Network addresses
; You may specify five types of Driver keywords. The value used is 1 or 0.
; Parm1 keyword is required for NETBIOS Driver.
; mandatory
Driver.TCPIP = 1
; Driver.NETBIOS = 1
; Parm1.NETBIOS = $(Parm1_NetBios)
; Driver.IPX = 1


; Distribution directories
BackupArea  = C:\SOFTDIST\BACKUP
ServiceArea = C:\SOFTDIST\SERVICE
Repository  = C:\SOFTDIST\REPOS
WorkArea    = C:\SOFTDIST\WORK


; Software Distribution Server Connection

; Distribution Server's System Name
ServerName = wtr05150
```

Figure 20. Response File to Install OS/2 Client (Part 1)

```
; Network driver (TCP/IP for TCP/IP, NB for NETBIOS, TLI for IPX)
ServerDriver = TCP/IP

; Distribution Server's address (hostname for TCP/IP, NETBIOS/IPX address
; for NETBIOS/IPX)
ServerAddress = wtr05150

; Distribution Client's hostname.
TCP.Hostname = mobile1

; Distribution Client's Target address. Warning: this field should be
; max 8 characters in length.
TargetAddress = 24104198
```

Figure 21. Response File to Install OS/2 Client (Part 2)

You can use the above response file in combination with the INSTALL command, also located in SD4OS2IMAGES.

The command to install the client in batch mode using the above response file is:

```
INSTALL /A:I /L1:C:\MESSAGE.LOG /X
/L3:C:\ERROR.LOG /X /R:C:\OS2CLT.RSP

/A:I - Action = Install
/L1  - Destination for message output
/L3  - Destination for error output
/X   - Run in batch mode
/R   - Response file
```

Note

The above example assumes that the customized response file is stored in OS2CLT.RSP.

After a successful installation the message file MESSAGE.LOG should look similar to the following:

```
1996-11-05 19:01:52.41  The requested components of
Tivoli TME 10 Software Distribution (3.1.3) for OS/2
are successfully installed.  Elapsed time was 00:03:40.
Your CONFIG.SYS file has been modified.
You must reboot your system to activate the changes.
```

4.3 Installation and Setup of the Windows 3.1 Mobile Client

This section describes the steps that need to be performed to install the Mobile Client on a Windows 3.1 workstation using the TCP/IP protocol. Before describing the installation we list the hardware and software requirements for Windows 3.1 and explain how to retrieve and install the Win32s support.

For a full description of the TME 10 Software Distribution client installation procedure see *Tivoli TME10 Software Distribution Clients Installation and Configuration*, SH19-4337.

4.3.1 Installation Requirements

The following is a list of the hardware and software requirements for the TME 10 Software Distribution for Windows 3.1 Mobile Client.

4.3.1.1 Hardware Requirements

- Minimum of 386 processor. (A 486 processor is recommended when installing the GUI).
- At least 12 MB RAM without the graphical user interface, 16 MB RAM with the graphical user interface.
- 6 MB disk space for the Mobile Client (including the base code).
- 10 MB disk space for the graphical user interface.
- 4 MB disk space for the online documentation.
- 4 MB disk space for Hw/Sw Discovery tool.
- Token-ring or Ethernet card.

4.3.1.2 Software Requirements:

- MS-DOS Version 6.2 or IBM DOS Version 6.3 or PC-DOS Version 7.0
- A Winsock compliant TCP/IP Protocol Stack for TCP/IP communication
- LAN Protocol Support Program 1.36 for the NetBIOS communication protocol support
- Novell Netware Requester for DOS for the IPX/SPX communication protocol support
- Microsoft Windows 32 Bit support (Win32s support)

4.3.2 Installation of the Win32s Support

The following gives you the information needed to retrieve and install the Win32s support, which is required to be installed before the installation of the TME 10 Software Distribution for Windows 3.1 Mobile Client.

Note

The Win32s support is required to run a 32-bit application on a Windows 3.1 system. The TME 10 Software Distribution for Windows 3.1 Version 3.1.3 is a 32-bit application and therefore has the Win32s support as an installation requirement.

The Win32s support can be retrieved without a fee from the Internet. Figure 22 on page 54 shows an example how to search for the right Web page to download Win32s support using the Lycos search tool. Lycos is available under the Web address http://www.lycos.com.

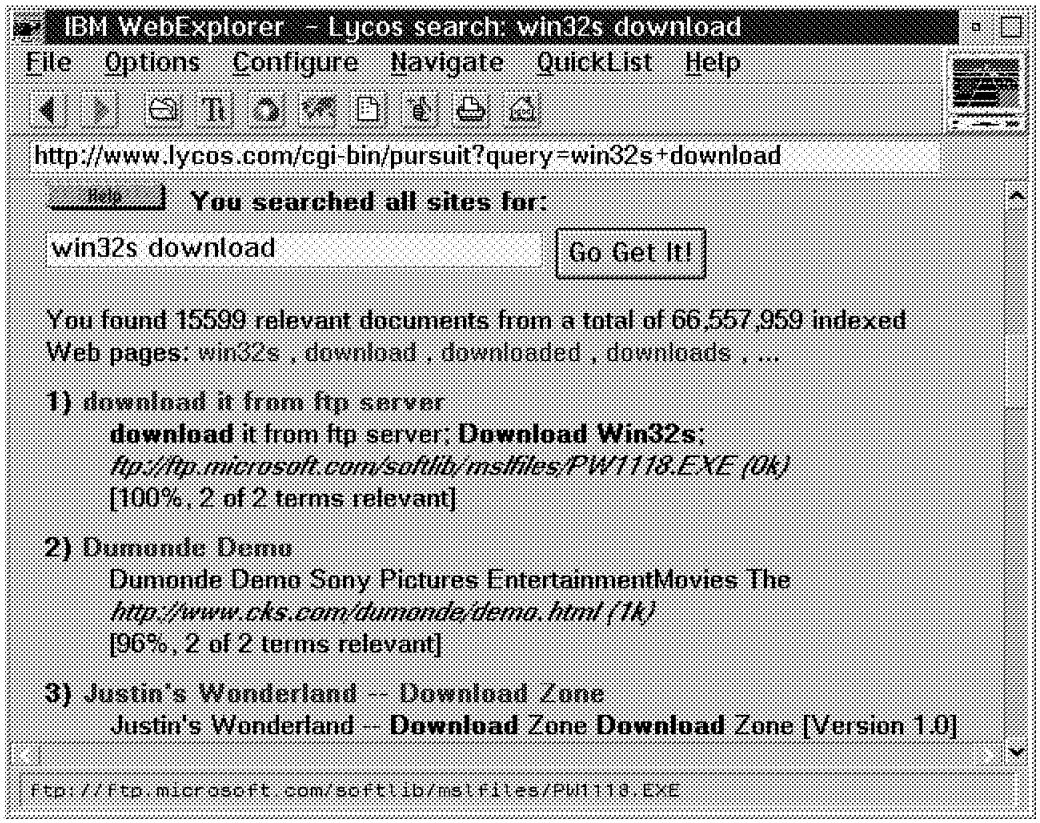


Figure 22. Using WebExplorer to Search for Win32s Support

1. Select a search tool from your Web browser and specify win32s download as the search criteria, for example use Lycos as shown in Figure 22.
2. From the search results select the document that says Download Win32s. In our example in Figure 22 this is the first document found. Position the mouse on **download it from ftp server** and press Enter to start the download.
3. A message window will pop up as shown in Figure 23 asking you whether you want to copy the file for the Win32s support to your local disk. Select **OK** to copy the file.

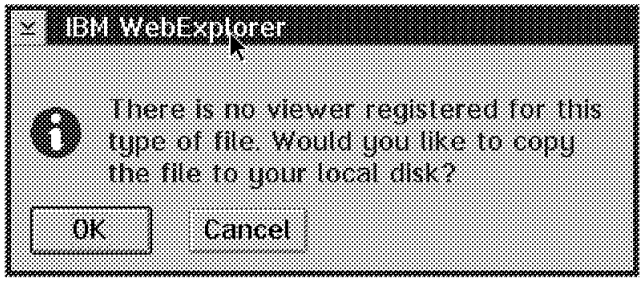


Figure 23. WebExplorer for Win32s Support - Message Window

4. A window will appear as shown in Figure 24 on page 55, in which you specify the drive and directory name for storing the file. Specify a directory name of your choice and select **OK**.

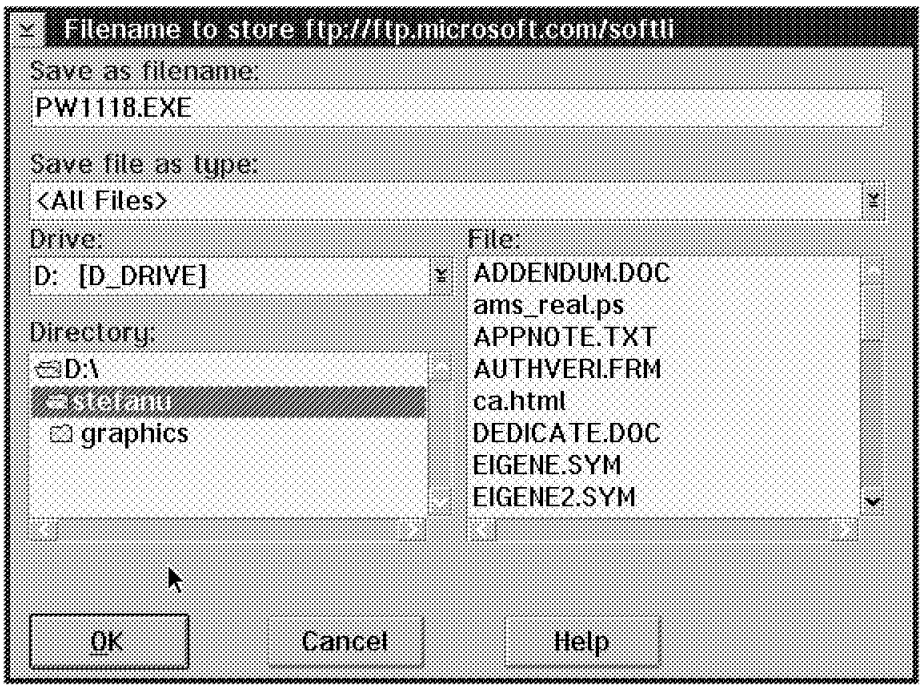


Figure 24. WebExplorer for Win32s Support - Store File Window

5. When the download is finished exit the Web browser and copy the file onto your Windows 3.1 system.
6. At your Windows workstation execute the downloaded file named PW1118.EXE. The file is a self extracting DOS program and will create the files needed for the installation of the Win32s support.
7. From the Windows Program Manager menu select the options **File** and **Run** from the pull-down menu.
8. Specify directorysetup in the Run window, where directory is the directory where you extracted the PW1118.EXE.
9. The Win32s support is installed and a folder named Win32 Application is added in the Program Manager menu containing a sample 32-bit application.

Note

In the above example we use IBM WebExplorer to download the file but you can use any other web browser.

4.3.3 Installation of the Windows 3.1 Mobile Client

In this part we show how to install the TME 10 Software Distribution for Windows 3.1 Mobile Client, which we connect to a Software Distribution for OS/2 server using the TCP/IP protocol. The setup of the TCP/IP communication was already completed and is therefore not included in this section.

We use the following hardware and software components:

- IBM ThinkPad Model 755C, equipped with:
  - 32 MB RAM
  - 800 MB hard disk

  - Docking Station IBM Dock II with an IBM 16/4 Token-ring Adapter
- IBM PC DOS Version 6.3
- IBM TCP/IP for DOS Version 2.11
- Windows 3.1
- Windows 32-bit support (Win32s)

Note

In order to use IBM TCP/IP for DOS you must not use a PC DOS version later than 6.3.

Also, when using IBM TCP/IP for DOS Version 2.11 you must apply CSD UB10767 which will update the product to level 2.1.1.4 in order to run the Windows 3.1 client successfully. We experienced that Windows will hang when the Mobile Client is started without the CSD applied.

Follow these steps to install the Windows 3.1 Mobile Client:

1. Check if the statement SHARE.EXE is added in the AUTOEXEC.BAT file.
2. Insert the CD-ROM.
3. Start Windows and select **File** from the Windows Program Manager menu.
4. Select **Run** from the File pull-down menu.
5. In the Run window that is displayed enter driveSD4W31install where drive is the drive letter for your CD-ROM drive. Select **OK** to start the installation program.
6. The Information Window appears. Read the information provided and select **Continue** to continue with the installation.
7. In the Install window select **OK** to update the CONFIG.SYS and AUTOEXEC.BAT files.
8. Select the components you want to install in the Install - directories window as shown in Figure 25 on page 57. For the installation of the Windows 3.1 Mobile Client select the following components:
   - Distribution Client Mobile for the Mobile Client including the base code
   - Distribution GUI for the graphical user interface
   - Distribution Documentation for the online documentation
   - Hw/Sw Discovery tool

   Note

   The graphical user interface for the TME 10 Software Distribution for Windows 3.1 Mobile Client was not yet available at the time this redbook was written. The selection of the component Distribution GUI will create the icon for the Software Preparation GUI in the TME 10 Software Distribution folder on the Windows 3.1 desktop. However, when you try to start the Software Preparation window you will only get a message that it is not available yet.

   The component Hw/Sw Discovery tool will install the NetFinity scanners for Windows 3.1 in the BIN product directory. The NetFinity scanners are supplied with TME 10 Software Distribution and can be used to do hardware

and software inventory discovery. See 4.4.4, “Defining an Inventory Program” on page 65 for more information.

In our example we install the Hw/Sw Discovery tool on the Windows 3.1 Mobile Client. We show the usage of the tool with the Mobile Client in 10.5.2, “Creating an Inventory File” on page 175.

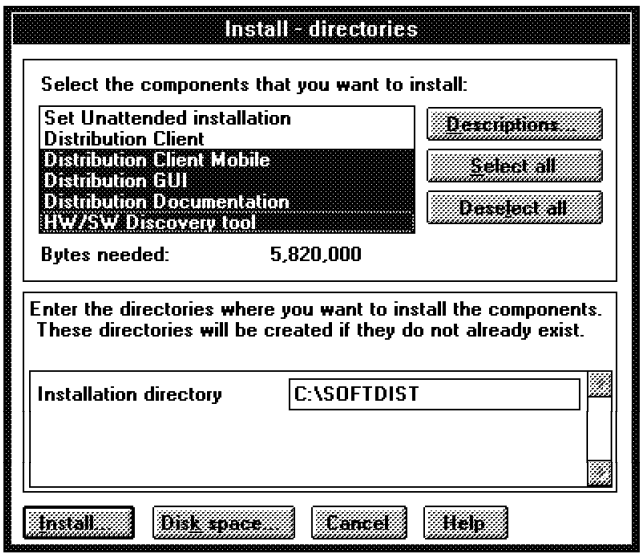


Figure 25. Software Distribution Client for Windows Install - Directories Window

9. The default directory where the components are installed is C:SOFTDIST. To change the name of the directory enter a new name in the Installation directory field.
10. Select the option **Disk space...** if you want to check if enough disk space is available on the selected drive. In the Disk space window that appears you can select a different drive and set the **Change directories to selected drive** check box to change the drive for the installation directory. Select **OK** to accept the values.
11. Select **Install** in the Install - directories window to start the transfer of the product files.
12. The Server Name window is displayed as shown in Figure 26. Enter the name of your Software Distribution for OS/2 server and select **OK** to continue. In our example the server name is wtr05150.

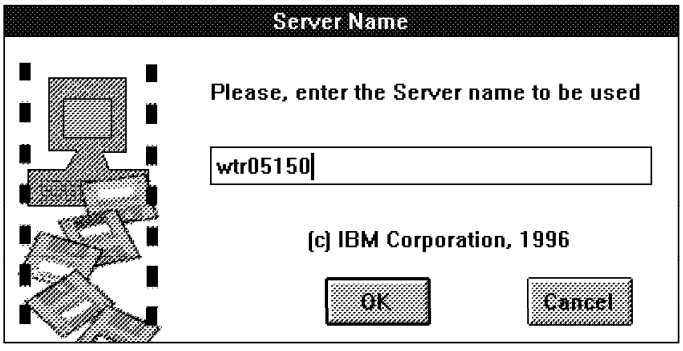


Figure 26. Software Distribution Client for Windows Server Name Window

Note

If you forget to add the SHARE.EXE statement in the AUTOEXEC.BAT file, the Server Name window will not be displayed. The configuration of the Windows 3.1 Mobile Client will not be completed and you have to set the name of your TME 10 Software Distribution server in the NVDM.CFG of the client. See 4.4.1, “Changing the Communication Protocol and/or Server Name” on page 61 for more information on defining a server name.

13. The Installation and Maintenance window will appear and indicate the successful installation after the transfer of the files is completed. Select **OK** to finish the installation.
14. Select **Exit** to leave the Installation window and restart the Windows workstation to activate changes made in the CONFIG.SYS and AUTOEXEC.BAT.

Note

The installation adds some environment variables to the AUTOEXEC.BAT file for the Windows 3.1 Mobile Client. Make sure that these variables appear before the last executable, for example WIN, in the file. If the variables appear after the last executable they will not be set during the reboot.

15. The installation adds a folder named TME 10 Software Distribution in your Windows Program Manager menu as shown in Figure 27.

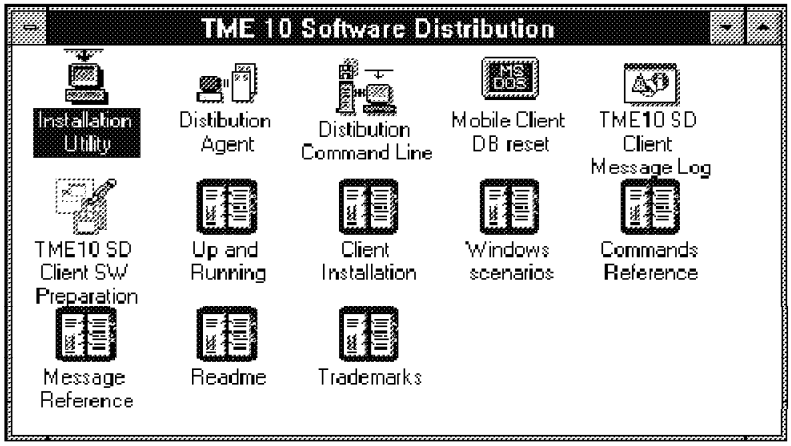


Figure 27. TME 10 Software Distribution Folder on Windows 3.1

16. The Software Distribution agent program fndcmps for the Windows Mobile Client is automatically started with the start of Windows because of the statement run=FNDCMPS.EXE that was added to the C:WINDOWSWIN.INI file during the installation.

Note

We experienced some problems with the initialization of the database on the Windows 3.1 Mobile Client after the installation. The database access services could not be initialized during the start of fndcmps as the following messages in the fndlog indicate:

```
FNDC0022I: Initializing trace and logging.
FNDC0015I: Task fndcmps has pid 5327.
FNDDB005I: Refreshing memory for 0 targets.
FNDDB006I: Refreshing memory for 0 catalog entries.
FNDDB007I: Database Access Services initialized successfully.
FNDDB035E: Cannot initialize Database Access Services; routing table
           missing or unreadable.
FNDC0016I: Task fndcmps has returned with return code 0.
FNDC0149I: The last product task is exiting.
```

To prevent this from happening select the **Mobile Client DB reset** icon from the TME 10 Software Distribution folder to re-initialize the database. After restarting Windows, the agent program fndcmps should start successfully.

4.3.4 Verifying the Installation of the Windows 3.1 Mobile Client

This section gives you some information on how you can verify if the installation and configuration of the Windows 3.1 Mobile Client was performed successfully.

To check if the Mobile Client can connect to the server, select the icon **Distribution Command Line** from the TME10 Software Distribution folder, which starts the command line interface of the Windows 3.1 Mobile Client. The TME 10 Software Distribution Command Line window is displayed with the command prompt `nvdm>`. Enter the following command to open a connection to the server:

```
svr
```

If the configuration of the server was successful you receive the message `Connected to server wtr05150` as shown in Figure 28 on page 60, where `wtr05150` is the name of our TME 10 Software Distribution server.

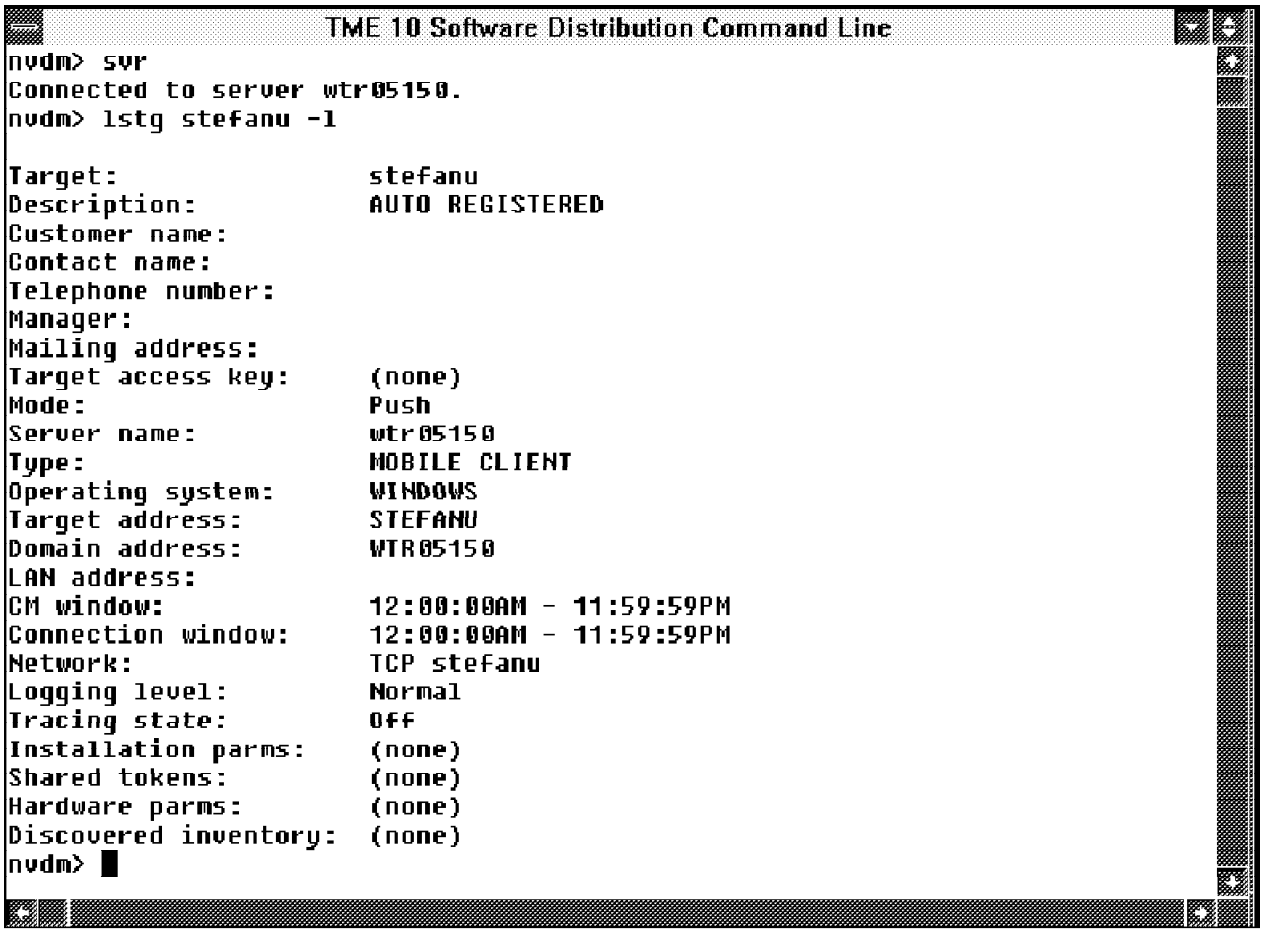


Figure 28. TME 10 Software Distribution Command Line Window

The configuration of the Windows 3.1 Mobile Client at the server was done using automatic client registration. See 4.5.2, “Using Automatic Client Registration” on page 70 for more information. Enter the following command to list the target definition of the Mobile Client at the Software Distribution for OS/2 server:

```
lstg stefanu -l
```

The name of the target in our example is stefanu. An example of the result of the command is shown in Figure 28.

4.4 Changing the Configuration of the Mobile Client

Normally no changes have to be made to the Mobile Client after the installation. But there are some situations where you need to make changes to the clients configuration:

- When you want to change the protocol used for communicating with the server.
- When you rename your server.
- When you rename your client you have to change the workstation name.
- When you want change the target address after the installation.

- When you want to define a program being invoked for inventory discovery function.
- When you want to increase the size of the log file.
- When you want to increase the size of the trace files.
- When you want define a permanently disconnected client.

This section explains how you change the configuration of the Mobile Client. As an example we show you how to make the changes using an OS/2 Mobile Client.

Note

On some systems where the NetBIOS protocol is used you might have to set the PROTOCOL keyword in the NVDM.CFG to reflect the correct NetBIOS name. The default used for the NetBIOS name and the target address during the installation is the NetBIOS adapter address, but the adapter address is not always identified correctly. Therefore, check the entries in the NVDM.CFG and change them accordingly as shown in 4.4.1, “Changing the Communication Protocol and/or Server Name” and 4.4.3, “Changing the Target Address” on page 64.

4.4.1 Changing the Communication Protocol and/or Server Name

If you want to change the protocol being used for the communication to the server or choose a different server name, you can do this by:

1. Selecting the icon **TME 10 Distribution Mobile Client Configuration** from the TME 10 Software Distribution for OS/2 folder located on your client desktop after the installation (see Figure 18 on page 49). This will bring up the distribution configuration notebook as shown in Figure 16 on page 47.

   Select the second page of the notebook and enter the new values for the server and/or protocol as shown in Figure 17 on page 48.

2. Editing the NVDM.CFG located in the product directory. The default directory name is C:SOFTDIST.

   Figure 29 on page 62 shows the contents of the NVDM.CFG file. In our example the protocol used is TCP/IP so the PROTOCOL keyword is set to PROTOCOL: TCP mobile1 729 50. The second keyword value is the TCP/IP hostname of the Mobile Client, the value 729 is the IP port being used by the product and 50 is the maximum number of connections.

   Note

   The IP port number 729 is added in the ETCSERVICES file on the client workstation during the installation.

   In case of using IBM TCP/IP for DOS this is the file C:TCPDOSETCSERVICES.

   The SERVER keyword is set to wtr05150 TCP wtr05150, where the first value is the workstation name of the server, the second keyword indicates that the TCP/IP protocol is used and the third keyword is the TCP/IP hostname of the server.

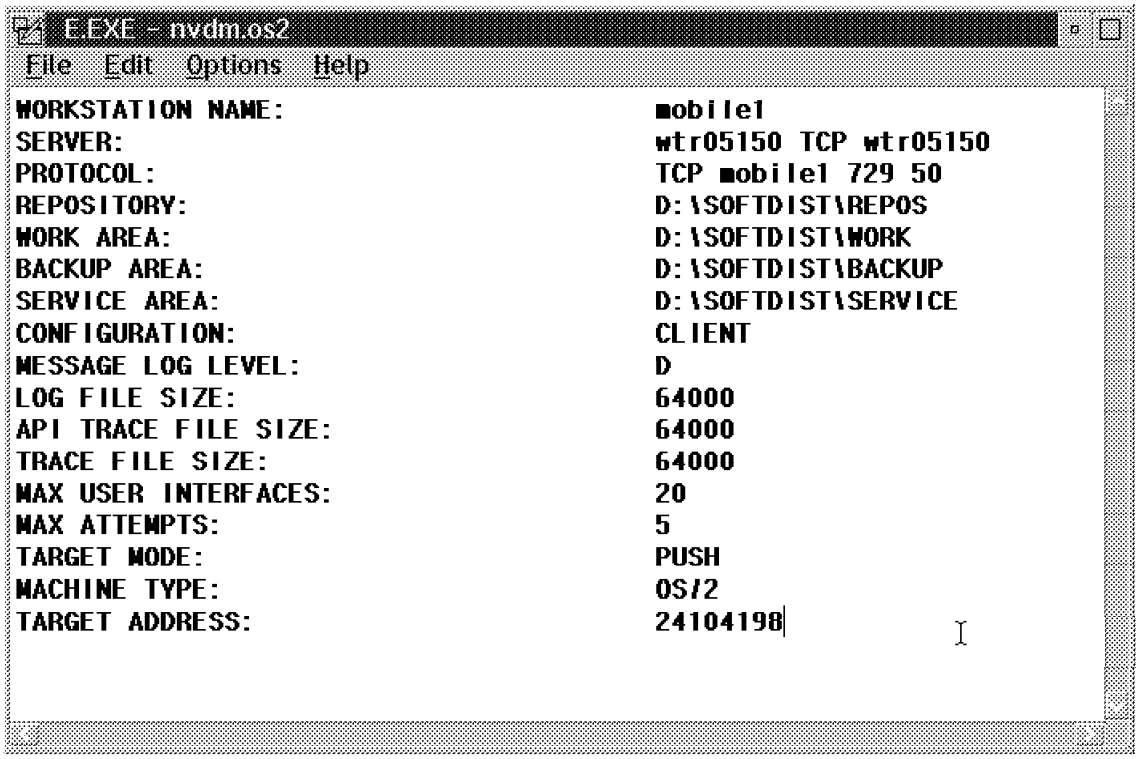


Figure 29. OS/2 Mobile Client NVDM.CFG for the Connection to the Software Distribution for OS/2 Server

The following list gives an example for each communication protocol and shows the according values for the PROTOCOL and SERVER keywords.

- Example for the NetBIOS protocol:

  SERVER: wtr05150 NBI nbwtr05150

  PROTOCOL: NBI 5A48674D 0 20

  For the SERVER keyword specify the workstation name of the server, NBI to indicate NETBIOS protocol is being used and the NetBIOS name of the server. For the PROTOCOL keyword specify NBI and the NetBIOS name of the Mobile Client, the adapter number (default is 0) and the maximum number of connections (default is 20).

- Example for the TCP/IP protocol:

  SERVER: wtr05150 TCP wtr05150

  PROTOCOL: TCP mobile1 729 50

  For the SERVER keyword specify the workstation name of the server, TCP to indicate TCP/IP protocol is being used and the IP hostname of the server. For the PROTOCOL keyword specify TCP and the IP hostname of the Mobile Client, the port number (default is 729) and the maximum number of connections (default is 50).

- Example for the IPX/SPX protocol:

  SERVER: wtr05150 IPX 0000000240004XC5B9B3869F

  PROTOCOL: IPX 4000ADC5A3BC

  For the SERVER keyword specify the workstation name of the server, IPX to indicate IPX/SPX protocol is being used and the IPX address of the server.

For the PROTOCOL keyword specify IPX and the IPX address of the Mobile Client.

- Example for the asynchronous protocol:

  ```
  SERVER:    rs600012 ASY 3013456
  PROTOCOL:  ASY 3013395
  ```

  For the SERVER keyword specify the workstation name of the server, ASY to indicate asynchronous protocol is being used and the telephone number of the server. For the PROTOCOL keyword specify ASY and the telephone number of the Mobile Client.

  Note

  We show an example of how to configure asynchronous support in Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181.

- Example for the SNA protocol:

  ```
  SERVER:    wtr05150 SNA ITIBM0PC.LT0539A0.MLU62A
  PROTOCOL:  SNA LT0541A0
  ```

  For the SERVER keyword specify the workstation name of the server, SNA to indicate SNA protocol is being used and the SNA address of the server. The SNA address is specified as NETWORKID.LUNAME.LUMODE. For the PROTOCOL keyword specify SNA and the local LU alias of the client as defined in the communication manager of the OS/2 Mobile Client.

  Experiencing Problems

  If you have established a connection to the server before you changed the server name you will experience problems after changing the server name. You probably receive the following message:

  ```
  FNDCLD29E: Failed to connect to server wtr05150
  ```

  Where wtr05150 is the server name before it was changed.

  To resolve this problem delete all the files in the directory C:SOFTDISTUICFG. The product adds the last contacted server name in files for the command line interface and graphical user interface. The files have the user ID that was used as the first extension and either .cli or .gui as the second extension, respectively for the command line interface or graphical user interface.

  After deleting these files you should be able to successfully connect to the new server.

4.4.2 Changing the Clients Workstation Name

There are two ways to change the workstation name of the Mobile Client:

1. You can use the graphical user interface by selecting the icon **TME 10 Distribution Mobile Client Configuration** from the TME 10 Software Distribution for OS/2 folder (see Figure 18 on page 49). This will bring up the distribution configuration notebook as shown in Figure 16 on page 47.

   Change the clients workstation name by typing in a new name in the field System name. Close the notebook and restart your Mobile Client by clicking

on the icons **Stop TME 10 Distribution Mobile Client** and **Start TME 10 Distribution Mobile Client** in the TME 10 Software Distribution for OS/2 folder.

2. You can edit the clients configuration file NVDM.CFG located in the product directory. The default directory name is C:SOFTDIST.

   Figure 29 on page 62 shows the contents of the OS/2 Mobile Client NVDM.CFG file. Change the value of the keyword WORKSTATION NAME to your new workstation name and restart the Mobile Client as mentioned earlier or by issuing the commands:

   ```
   nvdm stop
   nvdm start
   ```

Note

If you change the client workstation name you must delete the target definition previously made at the server and define a new target for the client as explained in section 4.5.1, “Defining a Target Using the Graphical User Interface or the Command Line Interface” on page 67 and 4.5.2, “Using Automatic Client Registration” on page 70.

You can delete the old target definition using the graphical user interface or by typing in the command:

```
nvdm deltg mobile1
```

Where mobile1 is the workstation name of the client before it was changed. Enter y at the prompt `Delete target mobile1 [y/n]?`.

Experiencing Problems

If you experience some problems after changing the workstation name of the client, try to delete all the files in the directory C:SOFTDISTUICFG. You probably receive the following message:

```
FNDDB081E: Failed to add target mobile1 because the maximum number
of targets,1, has already been defined.
```

Where mobile1 is the workstation name before it was changed.

The message is misleading, but you should be able to successfully restart the client after the deletion of the files.

4.4.3 Changing the Target Address

The target address is a unique address or name to identify a client at the server and is used to define the client at any other server in connected domains, including the focal point. The target address in the NVDM.CFG is set automatically during the installation. When TCP/IP is used as the communication protocol this address is set to the IP address (leaving out the periods) as shown in Figure 29 on page 62. Sometimes you might want to set it to be the same value as the workstation name.

To change the target address edit the NVDM.CFG located in the product directory and change the value of the keyword TARGET ADDRESS to a value of your choice.

Note

After changing the target address of the client you must update the target definition of the client at the server to reflect the new target address.

You can do this by using the graphical user interface or by typing the command:

```
nvdm updtg workstationname -s targetaddress
```

Where workstationname is the name under which your client is known at the server and targetaddress is the new target address.

4.4.4 Defining an Inventory Program

An inventory program is used to scan information about the hardware and software installed at a client workstation. The information discovered by the program has to be placed into the TME 10 Software Distribution inventory files in the product directory of the client using a special format. These files are:

- FNDSWINV

  For software inventory information

- FNDHWINV

  For hardware inventory information

- FNDTKINV

  For installation tokens

During the inventory discovery process of TME 10 Software Distribution the data in those files is read and stored in the client workstation database at the server. The inventory discovery process is initiated from the Targets window of the graphical user interface at the server or client or through the command line interface. From the command line interface issue the command:

```
nvdm inv -w targetname
```

The information stored can be used to do hardware or software pre-requisite checking during change management processes, which we show in 10.5.2, “Creating an Inventory File” on page 175.

You can write your own inventory program or use the program FNDINV supplied in the BIN product directory with the Software Distribution for OS/2 Mobile Client product.

To invoke the fndinv program when the nvdm inv command is issued specify the statement: INVENTORY PROGRAM: FNDINV in the NVDM.CFG located in the clients product directory; normally this is C:SOFTDIST.

If you selected the component Hw/Sw Discovery Tool during the installation of the Mobile Client as shown in 4.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 45, the program FNDINV invokes the NetFinity scanners for the hardware and software inventory and creates the files FNDSWINV, FNDHWINV and FNDTKINV.

Hint

To discover previously installed software with the inventory discovery the NetFinity software dictionary file DEFAULT.SID has to be customized and placed in the clients BIN product directory. The files FNDSWINV and FNDTKINV are created and contain information about software and installation tokens depending on the information defined in DEFAULT.SID. To customize the software dictionary used by the NetFinity Manager, refer to *LAN Management Processes Using NetFinity*, SG24-4517 for more information.

If the file DEFAULT.SID is not available at the client, the following message is displayed during the inventory discovery process: `SYS0002 the system cannot find the file specified`. The hardware discovery will still run successfully and create the file FNDHWINV.

If no inventory program is specified, which is the default, or when the Hw/Sw Discovery Tool is not installed, the only file being processed when invoking the nvdm inv command is FNDSWINV, which is created during the installation containing a single entry for the TME 10 Software Distribution client product.

4.4.5 Changing the Size of the Log File

If you need to change the size of the client message log file you can do this by editing the NVDM.CFG located in the client product directory. The keyword LOG FILE SIZE specifies the size of the message log file in bytes. The message log file is named FNDLOG and is located in the clients product directory.

When the message log is full, it is automatically backed up in a file called FNDLOG.BAK. So there is no real need to change the size unless you want to avoid repeated backups when a lot of entries are written to the log file. This depends on the activities going on between the client and the server and on the message log level specified in the keyword MESSAGE LOG LEVEL. Selecting D for Diagnostic increases the amount of messages written to the log. Remember that a larger value for the log file requires more disk space.

4.4.6 Changing the Size of the Trace File

The size of the clients trace file can be changed by editing the clients configuration file NVDM.CFG located in the clients product directory.

There are two types of traces available for the Mobile Client:

- Application trace between client and server (API TRACE)

  The application trace controls the on-going activities between the server and the client. This might be useful in some cases where the information in the message log does not seem to be sufficient.

  The keyword API TRACE FILE SIZE specifies the size of the application trace files in bytes. There are two trace files allocated in the clients product directory when the application trace is running, named fndapi1 and fndapi2.

  The application trace is initiated in the graphical user interface by selecting the option **Selected** in the Software Distribution for OS/2 Targets window for the target that is traced and choosing **Trace** from the pull-down menu. Either the choice On or Off can be selected. To initiate the trace from the command line interface enter the following command at the client that you want to trace:

```
nvdm tron
```

To stop the trace enter the following command:

```
nvdm troff
```

- Internal trace

  The keyword TRACE FILE SIZE specifies the size of the internal trace files in bytes. There are two trace files allocated in the clients product directory when tracing is initiated, named fndtrc1 and fndtrc2.

  Internal tracing is only required for diagnosis when serious problems occur and have to be reported to an IBM representative. It is initiated by specifying the statement SET FNDITRC=0 or SET FNDITRC=1 in the CONFIG.SYS. The value 0 sets the deepest trace level available.

4.5 Setting Up the Software Distribution for OS/2 Server

Every workstation in the network has to be defined as a target at the Software Distribution for OS/2 server. In this section we explain the different methods that can be used to define targets at the Software Distribution for OS/2 server. The Target can either be defined by an administrator using the graphical user interface or the command line interface as shown in 4.5.1, “Defining a Target Using the Graphical User Interface or the Command Line Interface” or by using the function automatic client registration as shown in 4.5.2, “Using Automatic Client Registration” on page 70.

Note

In our example the Software Distribution for OS/2 server has already been installed an configured on machine wtr05150. If you need guidance on how to do that in your environment, refer to *Tivoli TME 10 Software Distribution (3.1.3) for OS/2*, SH19-4334.

4.5.1 Defining a Target Using the Graphical User Interface or the Command Line Interface

This section explains how to define the Mobile Client as a target at the Software Distribution for OS/2 server using the graphical user interface and gives an example of the same operation using the command line interface. In our example we show how to define the OS/2 Mobile Client installed in section 4.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 45.

To define a Mobile Client as a target at your server select the option **Target** in the Software Distribution for OS/2 Targets window and from the pull-down menu select **New target...** as shown in Figure 30 on page 68.

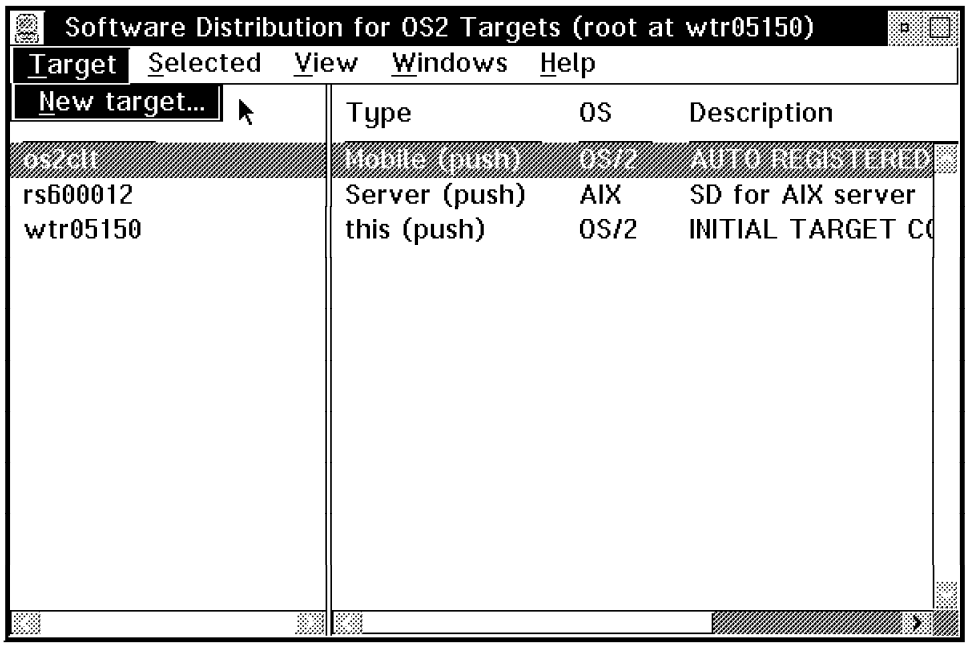


Figure 30. Software Distribution for OS/2 Targets Window

The New Target window as shown in Figure 31 on page 69 is presented.

In the Name field type in the workstation name for the Mobile Client under which it is known at the server. In our example the workstation name is mobile1.

Note

The name specified here must match the value of the keyword WORKSTATION NAME in the NVDM.CFG of the Mobile Client as shown in Figure 29 on page 62.

An optional description for the client can be specified in the field Description.

The option Change management defines the target mode. The following values can be selected:

- Push

  Push mode allows that change management is initiated from the server, the client itself or any other target.

- Pull

  Pull mode only allows change management initiated from the client itself.

- Manager

  Manager mode is selected when defining another server as a target.

- Focal

  Focal mode is selected when the target to be defined is a server that acts as a focal point for this server.

In our example we are defining a client that allows change management from the server and other targets, so the selected value is set to Push.

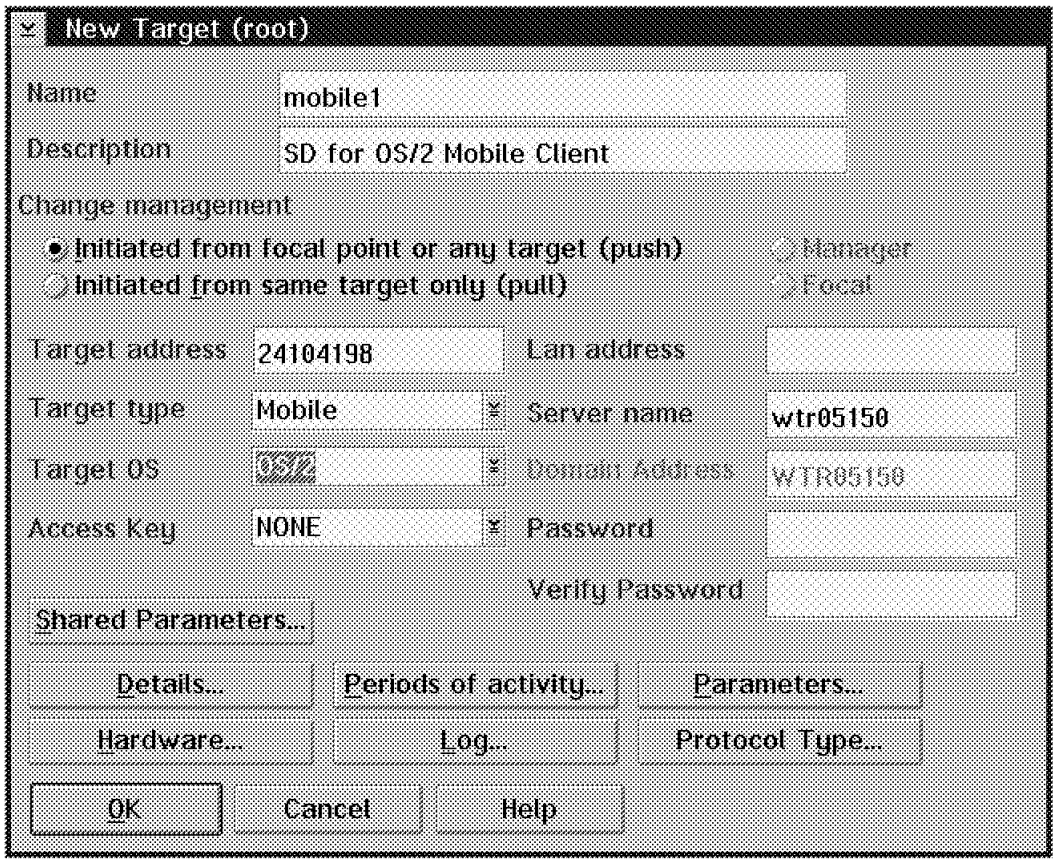


Figure 31. New Target Window

The Target address is a unique address for the target at the server and is used to identify the target at any other server in connected domains, including the focal point. You can set it to be the same as the TCP/IP hostname. In our example the target is set to the TCP/IP hostname, which is `mobile1`.

Note

The target address field only allows capital letters. The value must match the value of the keyword TARGET ADDRESS in the NVDM.CFG of the Mobile Client as shown in Figure 29 on page 62.

If you choose a different name for the target address than the actual TCP/IP hostname, then you have to select the push button **Protocol Type...** which will open the Protocol Type window. In this window select **TCP** for the protocol and specify the hostname in the field Host name.

Set the Target type to Mobile for the Mobile Client. The target type is the only keyword which distinguishes the Mobile Client from a normal client at the server. This value is set to Client when defining a normal client or a fully disconnected client. We explain the usage of a fully disconnected client in Chapter 8, “Using a Fully Disconnected Client” on page 121.

Note

The corresponding keyword for the target type in the clients configuration file NVDM.CFG is the keyword CONFIGURATION. Notice that this keyword is always set to Client for the Mobile Client.

For the Target OS select the operating system type of your Mobile Client. In our example we set this value to OS/2.

No other fields have to be specified for the definition of the Mobile Client. The field Server name and Domain address are automatically set to the name of the server where you are defining the target, as soon as you select the target type to be mobile or client. In our example these fields are automatically set to wtr05150.

Command Line Example

To define the above client using the command line interface enter the command `nvdm addtg` with the following parameters and values:

```
nvdm addtg mobile1 -d "Software Distribution for OS/2 Mobile Client" -m
push -b mobile -s mobile1 -y OS/2 -tp tcp:mobile1
```

- The first parameter is the name of the new target.
- -d is the optional description (put text in quotes " ").
- -m is the target mode.
- -m is the target mode.
- -b is the target type.
- -s is the target address (or shortname).
- -y is the target OS (operating system).
- -tp is the protocol type, with the value for TCP/IP protocol and the IP hostname.

4.5.2 Using Automatic Client Registration

You can allow new targets to be automatically defined at the server when they connect the server for the first time. This process is also called auto-registration.

To use auto-registration at your server you must add the statement AUTOMATIC TARGET REGISTRATION: YES in the NVDM.CFG of the server. In the clients configuration file you must specify the keywords TARGET ADDRESS and TARGET MODE as shown in Figure 29 on page 62.

During the first start of the Mobile Client the client registers itself at the server. During the registry a target entry is automatically created at the server with the information provided in the clients NVDM.CFG file. Figure 32 on page 71 shows the Target Details window of an auto registered target. Note that the description field is always set to AUTO REGISTERED.

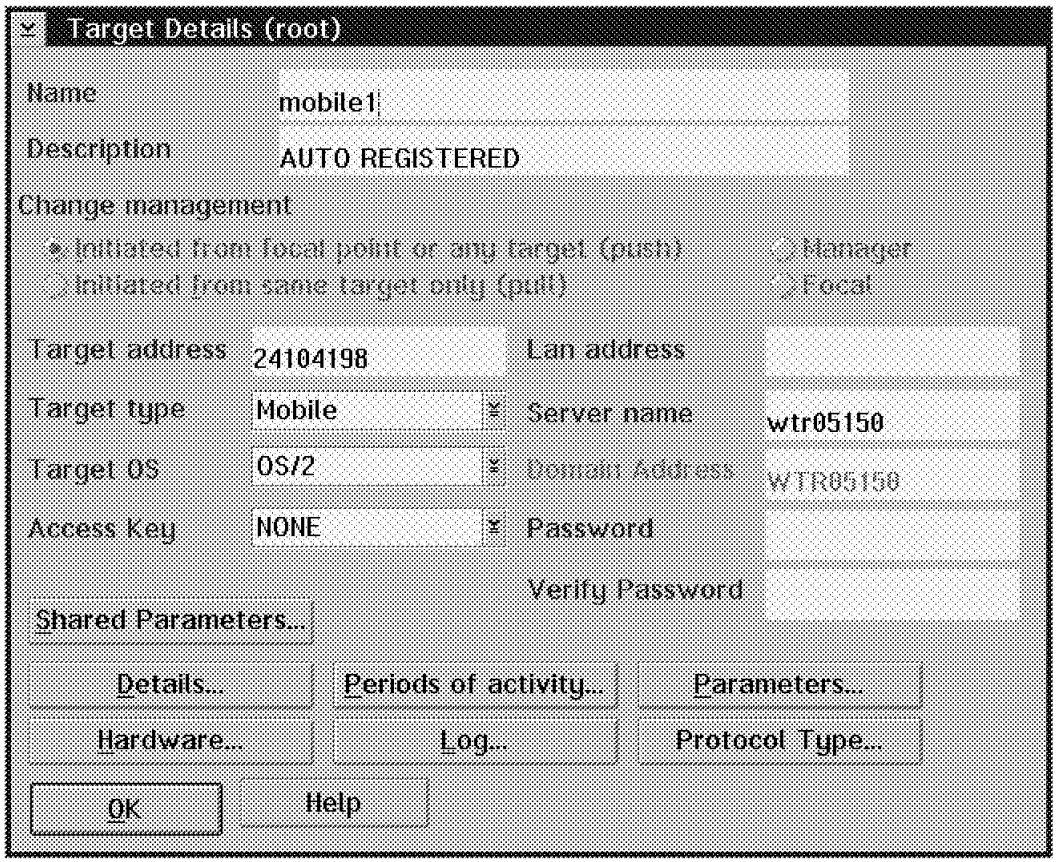


Figure 32. New Target Window

Table 2 shows how the keywords in the NVDM.CFG of the client map to the target definition at the server.

Table 2. Mapping of NVDM.CFG Keywords into Auto-Registered Client Definition

Client configuration keyword	**Target keyword at server**	**Explanation**
WORKSTATION NAME	Name	Workstation name of the client
PROTOCOL	Protocol type keywords, for example TCP and host name	Communication protocol used
CONFIGURATION	Target type	Type of target
MESSAGE LOG LEVEL	Log level selection	The level used for message log
TARGET MODE	Change management (Push, Pull, Manager or Focal)	The mode in which client operates
MACHINE TYPE	Target OS	The type of operating system of the client
TARGET ADDRESS	Target address	The unique target address for the client

Chapter 5. Mobile Clients in an AIX LAN Environment

This chapter describes the setup of an AIX LAN with OS/2 and Windows NT/95 Mobile Clients connected to a Software Distribution for AIX server, as shown in Figure 33.

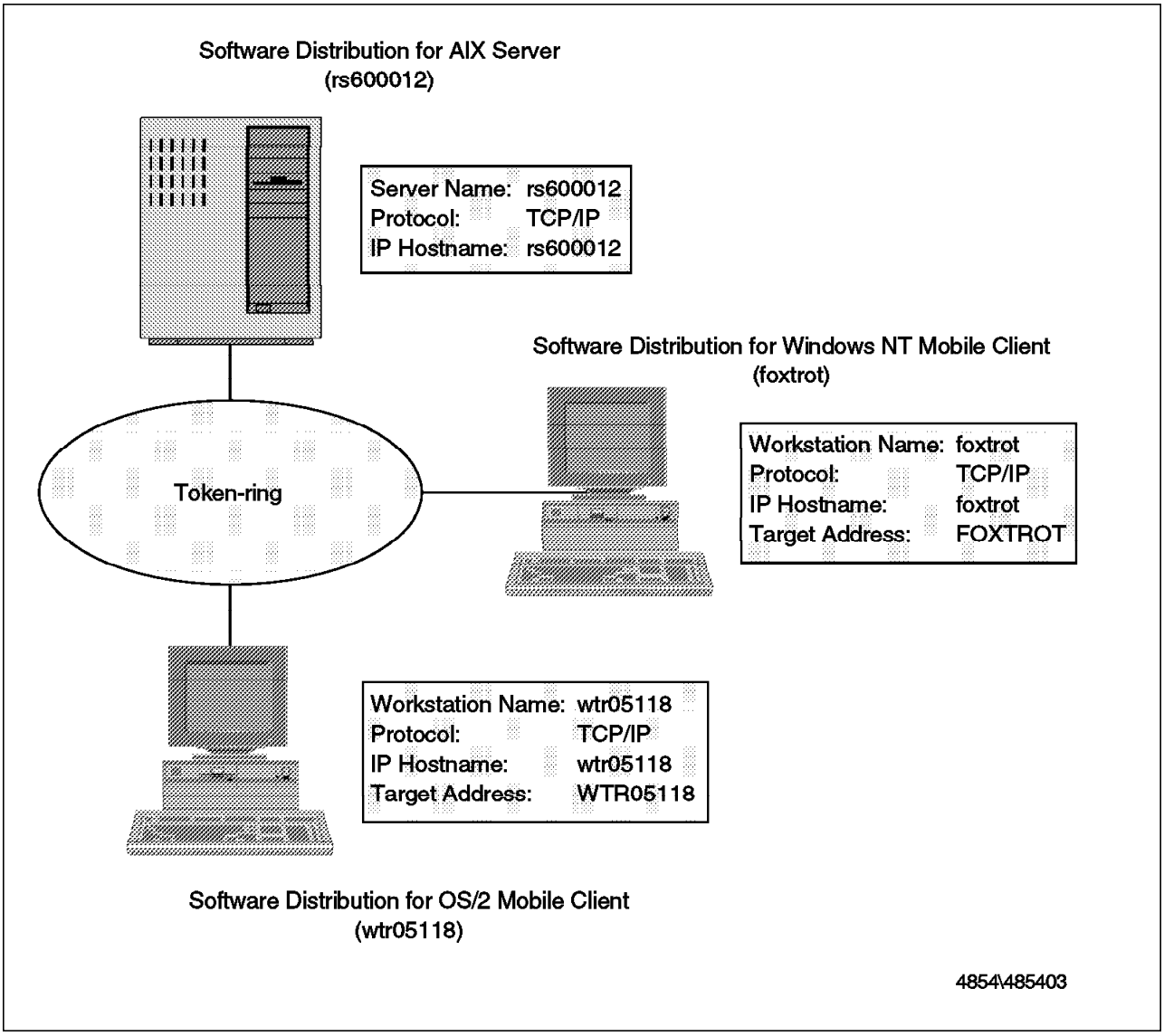


Figure 33. AIX LAN with OS/2 and Windows NT Mobile Clients

5.1 Overview

This section contains the information needed to connect OS/2 and Windows NT/95 Mobile Clients to a Software Distribution for AIX server. It guides you through the installation of the TME 10 Software Distribution client product on the OS/2 and Windows NT/95 platforms, describes how to change the configuration of the Mobile Clients and shows you how to define the Mobile Clients to the Software Distribution for AIX server.

© Copyright IBM Corp. 1997

5.2 Installation and Setup of the OS/2 Mobile Client

The following outlines the specific installation steps and customization aspects related to the OS/2 Mobile Client.

For a full description of the TME 10 Software Distribution client installation procedure see *Tivoli TME10 Software Distribution Clients Installation and Configuration*, SH19-4337.

5.2.1 Installation Requirements

Refer to 4.2.1, “Installation Requirements” on page 44 for a full list of hardware and software requirements for the TME 10 Software Distribution for OS/2 Mobile Client.

5.2.1.1 Software Requirements

The only communication protocols available for the connection of the OS/2 Mobile Client to a Software Distribution for AIX server are:

- TCP/IP

  Which requires that IBM TCP/IP for OS/2 2.0 with CSD UN64092 or later is installed.

- SNA (APPC)

  Which requires that Communication Server, or Communication Manager (CM/2) 1.11 or later is installed.

5.2.2 Interactive Installation of the OS/2 Mobile Client

We are installing an OS/2 Mobile Client on a desktop workstation which we connect to a Software Distribution for AIX server. The protocol used in the AIX LAN is TCP/IP.

To install the OS/2 Mobile Client interactively from CD-ROM perform the steps described in 4.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 45. The only difference in the installation of the OS/2 Mobile Client in the AIX LAN are the values defined in the configuration notebook as shown in Figure 34 on page 75.

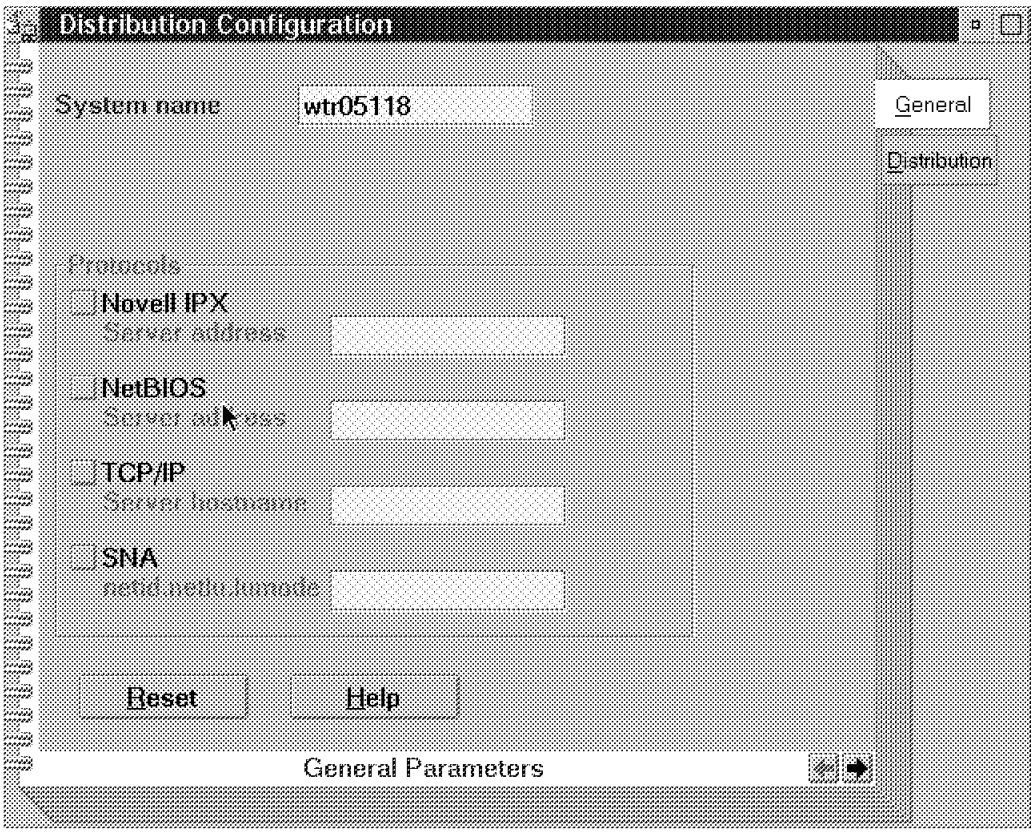


Figure 34. Software Distribution for OS/2 Client Distribution Configuration Notebook General Page

The System name entered here is the workstation name of the Mobile Client under which it is known at the Software Distribution for AIX server. In our example we choose the system name to be the same as the TCP/IP hostname and set it to wtr05118.

In the Distribution Configuration window, as shown in Figure 35 on page 76, specify the workstation name of the Software Distribution for AIX server in the field System name and TCP/IP as the protocol in the Network Driver field. The network address must be the TCP/IP hostname of the server. In our example the name of the Software Distribution for AIX server is rs600012 the same as the TCP/IP hostname.

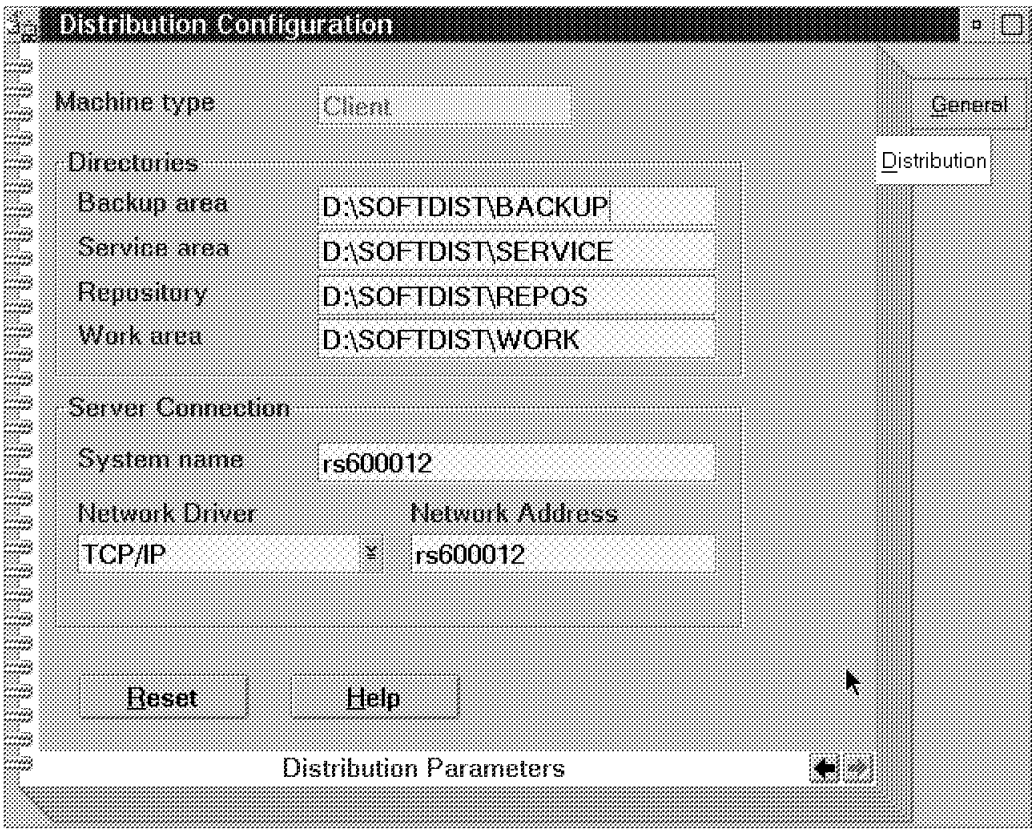


Figure 35. Software Distribution for OS/2 Client Distribution Configuration Notebook Distribution Page

5.3 Installation and Setup of the Windows NT/95 Mobile Client

This section describes the requirements for installing the Windows NT/95 Mobile Client and shows how to install the Mobile Client interactively.

For a full description of the TME 10 Software Distribution client installation procedure see *Tivoli TME10 Software Distribution Clients Installation and Configuration*, SH19-4337.

Note

The client code for Windows NT and Windows 95 is the same. Therefore, although we show the setup on Windows NT in our example, you will also be able to use it on Windows 95.

5.3.1 Installation Requirements

The list below shows the hardware and software requirements needed for the TME 10 Software Distribution for Windows NT Mobile Client.

5.3.1.1 Hardware Requirements

- 486 processor
- At least 16 MB RAM without graphical user interface, 20 MB RAM with graphical user interface
- 6 MB disk space for the Mobile Client (5 MB for the client and 1 MB for the Mobile functionality)
- 8 MB disk space for the graphical user interface
- 4 MB disk space for the Hw/Sw Discovery tool
- 4 MB disk space for the documentation
- Token-ring or Ethernet card

5.3.1.2 Software Requirements

- Windows NT Version 3.51 is required.
- The communication support for the following protocols is included in Windows NT Version 3.51:
  - NetBIOS
  - TCP/IP
  - IPX/SPX

5.3.2 Installing the Windows NT Mobile Client

In the following example we are installing the Windows NT Mobile Client using the TCP/IP protocol and connecting it to the Software Distribution for AIX server. The TCP/IP communication support is already configured at the Windows NT client after the installation of the operating system. Therefore the communication setup is not described in this section.

To install the Windows NT Mobile Client interactively from CD-ROM complete the following steps:

1. Insert the CD-ROM.
2. Change to the directory SD4WNT95.
3. Type setup and the Welcome window appears.
4. Select **Next** to continue the installation.
5. In the Select Installation window choose to install the **Software Distribution Client**, and then select **Next**.
6. In the Select Components window you can select the components you want to install, as shown in Figure 36 on page 78. To install the Windows NT Mobile Client with the graphical user interface and the documentation select the following components:
   - Distribution Client Mobile for the Mobile Client including the client base component
   - Distribution GUI for the graphical user interface
   - Distribution Documentation for the online documentation

   Use the cursor keys to scroll up and down and press the space bar to select the component.

Note

The only graphical user interface for the TME 10 Software Distribution for Windows NT Mobile Client that was available at the time this rebook was written was the Software Preparation GUI.

Check with your IBM representative to obtain current version information.

Optionally you can select the component **Hw/Sw Discovery** to install the NetFinity scanners that support hardware and software inventory discovery for the Windows NT client.

Note

The NetFinity scanners are used for hardware and software inventory checking when the statement INVENTORY PROGRAM: FNDINV is added in the NVDM.CFG of the Windows NT Mobile Client.

We install the Hw/Sw Discovery tool in our example and show the usage of the tool with the Mobile Client in 10.5.2, “Creating an Inventory File” on page 175. Also see 4.4.4, “Defining an Inventory Program” on page 65.

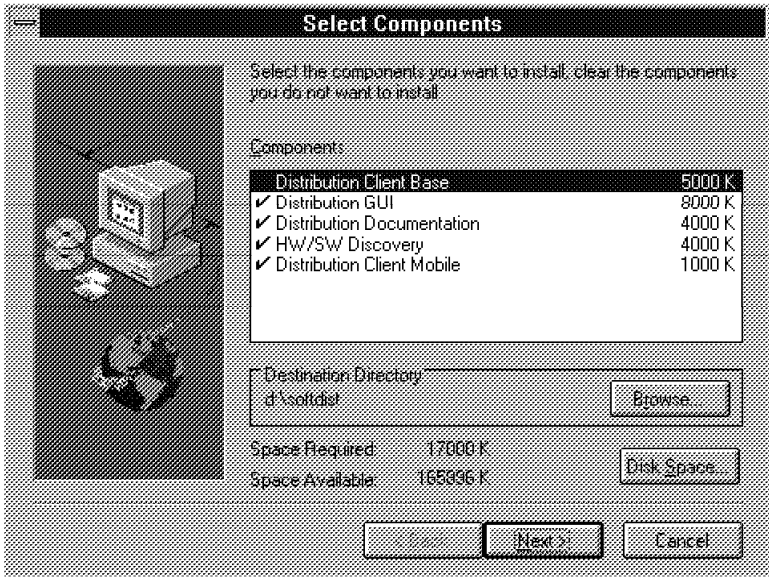


Figure 36. Software Distribution for Windows NT Client Select Components Window

7. The default directory where the components are installed is C:SOFTDIST. You can change the name of the drive and the directory by selecting **Browse...**. The Choose Directory window appears and you can change the directory here.

   Note

   If the directory does not exist, a pop-up window will show up indicating that the directory specified does not exist. Select **Yes** to create the directory.

8. To check the available disk space click on **Disk space...** in the Select Components window. In the Available Disk Space window that appears you can see the amount of disk space that is available on the selected drive and

the space required for the installation. You can select a different drive for the installation. Select **OK** to return to the Select Components window.

9. Select **Next** in the Select Components window to continue with the installation.
10. The Select Program Folder window appears. The default folder that is created on the Windows NT desktop for the Windows NT Mobile Client product is named Tivoli TME 10 Software Distribution. You can choose a different folder by typing in a folder name in the Program Folders field or by selecting an existing folder from the list offered in the Select Folder window. Select **Next** to continue.
11. The Client Configuration window appears as shown in Figure 37.

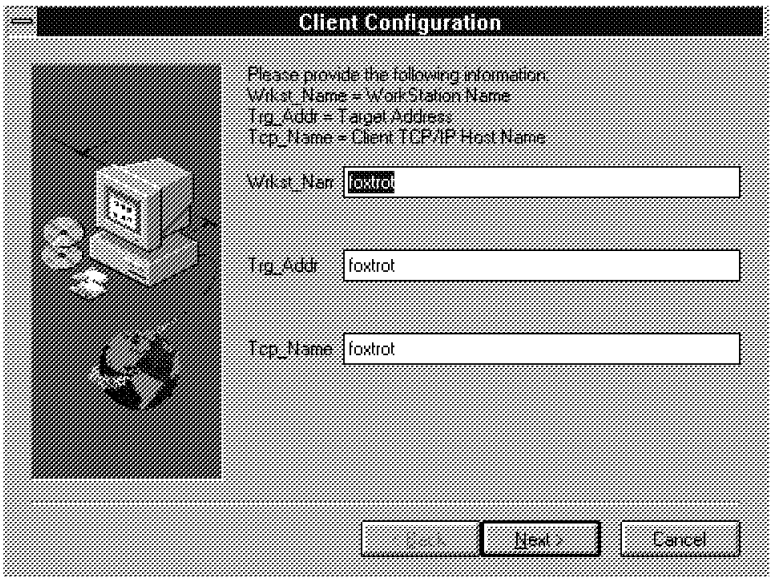


Figure 37. Software Distribution for Windows NT Client Configuration Window

In this window specify the name of your Mobile Client workstation in the field Wrkst_Name, the target address of the Mobile Client in the Trg_Addr field and the TCP/IP hostname in the Tcp_Name field.

The workstation name is the name under which the Mobile Client is known at the server and the target address assigns a unique address to the Mobile Client. The default is that all values are set to the TCP/IP hostname. In our example the hostname is foxtrot and we accept the default settings.

Note

If you change any value here remember that these fields are case sensitive. Select **Next** to go to the Server Configuration window.

12. Specify the workstation name of the Software Distribution for AIX server in the Srv_Name field and the TCP/IP hostname of the server in the Tcp_Name field of the Server Configuration window as shown in Figure 38 on page 80. Use the Tab key to move between fields. In our example the server name is rs600012, the same as the TCP/IP name of the server. Select **Next** to proceed with the installation.

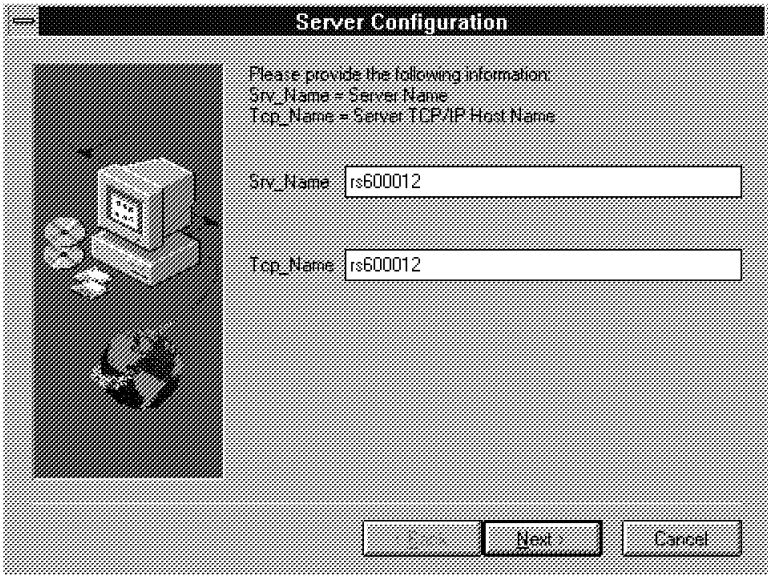


Figure 38. Software Distribution for Windows NT Server Configuration Window

13. The Start Product Selection window appears. Select **Automatic Startup** to define a Windows NT service that will automatically start the Mobile Client with the Windows NT start or select **Manual Startup** if you explicitly want to start the product. We suggest selecting Automatic Startup here. To start the installation of the product select **Next**.
14. The Copying Profiles window shows the progress of the installation.
15. When the installation is complete the Setup Complete window appears. Here you can select to restart your computer now or later. Select **Restart now** and **Finish** to exit the installation window.
16. The system is rebooted automatically now.
17. After the installation a program folder is placed on the Windows NT desktop for the TME 10 Software Distribution for Windows NT Client.

    The folder contains the icons for stopping and starting the product and the icons for the Software Preparation GUI as shown in Figure 39.

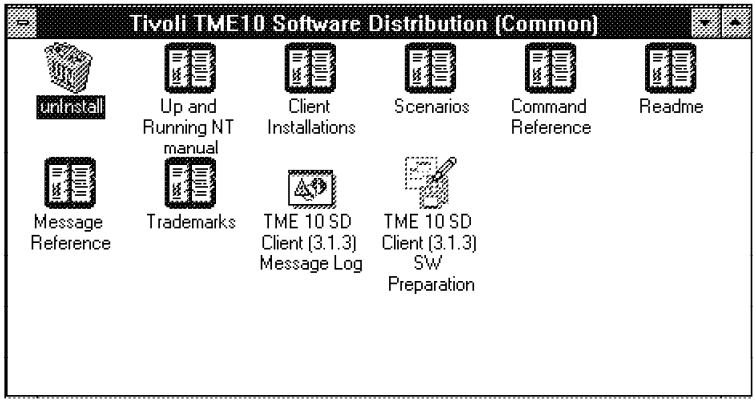


Figure 39. Tivoli TME 10 Software Distribution Folder on the Windows NT Client Desktop

5.3.3 Verifying the Installation of the Windows NT Mobile Client

This section gives an example of how you can verify that the Windows NT Mobile Client was installed correctly and is up and running.

The TME 10 Software Distribution product at the Windows NT platform does not provide a running task window for the TME 10 Software Distribution agent program. The only way to check if the agent is started is to look at the Windows NT services window as shown in Figure 40. Select **Control Panel** from the Main window of the Windows NT desktop and then **Services**. You should see a service called NetViewDM with the status started.

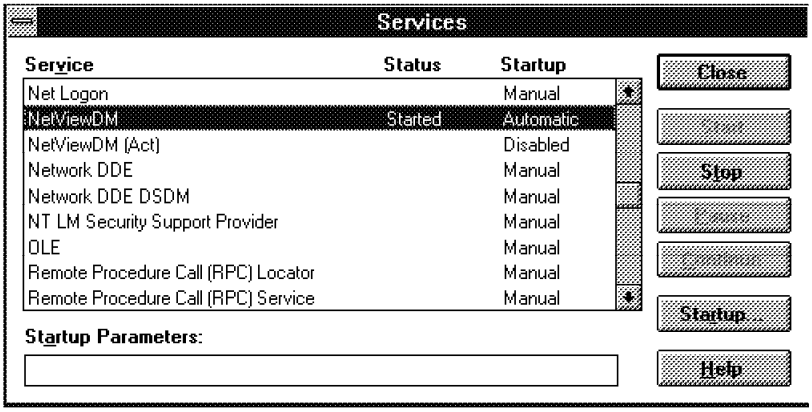


Figure 40. Windows NT Services Showing the Running Agent

To check if the connection to the Software Distribution for AIX server is running properly enter the following command from a command prompt:

```
nvdm lstg foxtrot
```

If the client installation and configuration was correct as in our example, you should see an output such as the one in Figure 41 on page 82. In our example the client was automatically registered, so the Description field says AUTO REGISTERED.

```
Target:                foxtrot
Description:           AUTO REGISTERED
Customer name:
Contact name:
Telephone number:
Manager:
Mailing address:
Target access key:     (none)
Mode:                  Push
Server name:           rs600012
Type:                  MOBILE CLIENT
Operating system:      WINDOWS_NT
Target address:        FOXTROT
Domain address:        RS600012
LAN address:
CM window:             12:00:00AM - 11:59:59PM
Connection window:     12:00:00AM - 11:59:59PM
Network:               TCP foxtrot
Logging level:         Normal
Tracing state:         Off
Installation parms:    (none)
Shared tokens:         (none)
Hardware parms:        (none)
Discovered inventory:  (none)
```

Figure 41. Output of List Target from Windows NT Mobile Client

Figure 42 shows the client configuration file NVDM.CFG of the Windows NT Mobile Client after the installation.

```
WORKSTATION NAME:              foxtrot
SERVER:                        rs600012 TCP rs600012
#SERVER:                        rs600012 NBI rs600012
#SERVER:                        rs600012 IPX (sNetId)(sAdpAdd)(sAplAdd)
PROTOCOL:                      TCP foxtrot 729 50
#PROTOCOL:                      NBI foxtrot 0 50
#PROTOCOL:                      IPX (NetId)(AdpAdd)(AplAdd)
REPOSITORY:                    d:\softdist\repos
SERVICE AREA:                  d:\softdist\service
BACKUP AREA:                   d:\softdist\backup
WORK AREA:                     d:\softdist\work
CONFIGURATION:                 CLIENT
MESSAGE LOG LEVEL:             D
LOG FILE SIZE:                 500000
API TRACE FILE SIZE:           500000
TRACE FILE SIZE:               1000000
MAX USER INTERFACES:           20
MAX ATTEMPTS:                  5
TARGET MODE:                   PUSH
MACHINE TYPE:                  WINDOWS_NT
INVENTORY PROGRAM:             fndinv
TARGET ADDRESS:                foxtrot
```

Figure 42. Configuration File of the Windows NT Mobile Client

5.4 Setting Up the Software Distribution for AIX Server

The Software Distribution for AIX server is already installed and running in our environment. The configuration of the Software Distribution for AIX server does not need to be modified for the connection of Mobile Clients. The mobile support is already available with the installation of the server product. No additional components have to be selected during the server installation.

In this section we show as an example how to define a new Windows NT workstation as a target at the Software Distribution for AIX server. Examples demonstrate how to use the graphical user interface and command line interface to define targets and how targets can be defined automatically using the automatic client registration.

5.4.1 Defining a Target Using the Graphical User Interface or the Command Line Interface

This section explains how an administrator can define the Mobile Client as a target using the graphical user interface. At the end of this section we provide the information needed to perform the same task using the command line interface.

To define the OS/2 Mobile Client as a target at your server select the option **Target** in the Software Distribution for AIX Targets Window and from the pull-down menu select **New target...** as shown in Figure 43.

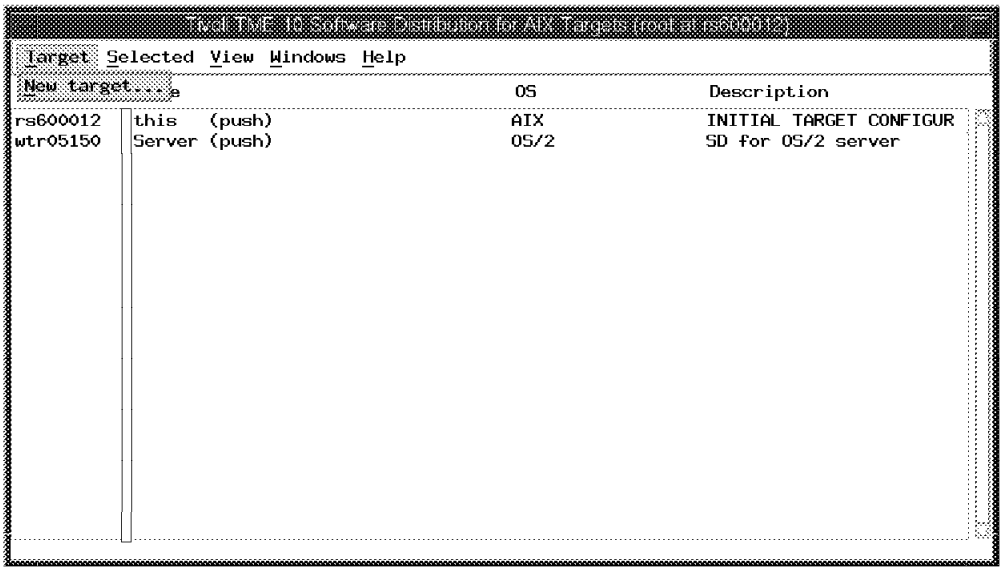


Figure 43. Software Distribution for AIX Targets Window

The New Target window as shown in Figure 44 on page 84 is presented.

In the Name field type in the workstation name for the Windows NT Mobile Client.

Note

The name specified here must match the value of the keyword WORKSTATION NAME in the NVDM.CFG of the Windows NT Mobile Client, which is foxtrot in our example.

The Description field is optional. You can define any comment here. In our example we specified as information for the administrator that this is a Windows NT Mobile Client.

The option Change Management defines the target mode. The following values can be selected:

- Push

  Push mode allows that change management is initiated from the server, the client itself or any other target.

- Pull

  Pull mode only allows change management initiated from the client itself.

- Manager

  Manager mode is selected when the defined target is another server.

- Focal

  Focal mode is selected when the defined target is a server that acts as a focal point for this server.

In our example we are defining a client that allows change management from the server and other targets, so the value selected is Push.

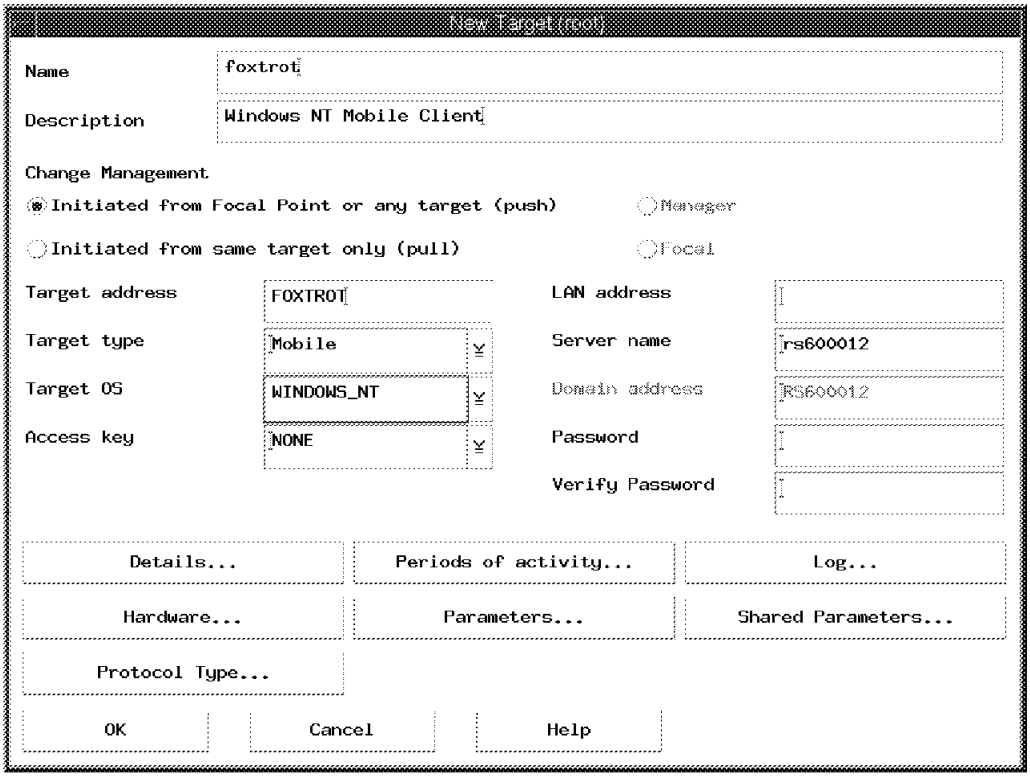


Figure 44. New Target Window at Software Distribution for AIX Server

The Target address is a unique address for this target at the server. You can set it to be the same as the TCP/IP hostname. In our example the target is set to the TCP/IP hostname, which is foxtrot.

Note

The Target address field only allows capital letters and must match the TARGET ADDRESS keyword in the NVDM.CFG of the Mobile Client as shown in Figure 42 on page 82.

If you choose a different name for the target address than the actual TCP/IP hostname then you have to select the push button **Protocol Type...** which will open the Protocol Type window as shown in Figure 45. In this window select TCP for the protocol and specify the hostname in the field Host name. In our example the hostname is foxtrot.

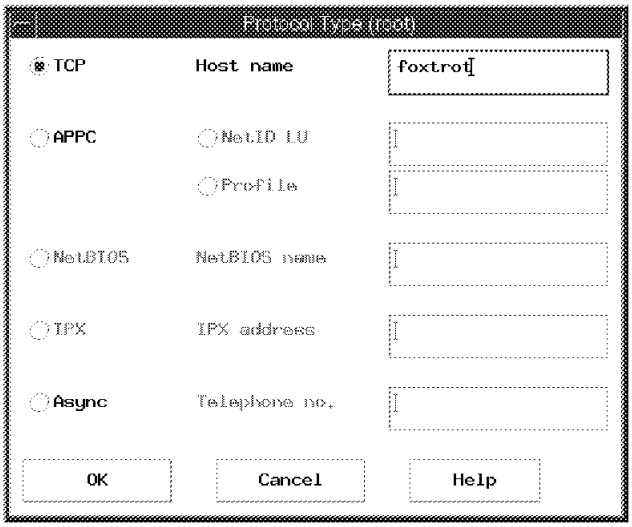


Figure 45. Protocol Type Window

The Target type for the definition of a Mobile Client must be set to Mobile. This keyword is set to Client if a non-mobile client is defined or when a permanently disconnected client is defined. We show how to use a permanently disconnected client in Chapter 8, “Using a Fully Disconnected Client” on page 121.

Since we are defining a Windows NT workstation as a target, the selected Target OS is WINDOWS_NT.

No other fields have to be specified for the definition of the Mobile Client. The fields Server name and Domain address are automatically set to the name of the Software Distribution for AIX server, after the target type of mobile or client is selected. In our example the server name is rs600012.

Command Line Example

To define the above client at the Software Distribution for AIX server using the command line interface enter the command `nvdm addtg` with the following parameters and values:

```
nvdm addtg foxtrot -d "Windows NT Mobile Client" -m push -b mobile -s
foxtrot -y WINDOWS_NT -tp tcp:foxtrot
```

- The first parameter is the name of the new target.
- -d is the optional description (put text in quotes " ").
- -m is the target mode.
- -m is the target mode.
- -b is the target type.
- -s is the target address (or shortname).
- -y is the target OS (operating system).
- -tp is the protocol type, with the value for TCP/IP protocol and the IP hostname.

5.4.2 Using Automatic Client Registration

You can allow new targets to be automatically defined at the Software Distribution for AIX server when they connect the server for the first time. This process is also called auto-registration.

To define that automatic target registration is allowed at the Software Distribution for AIX server add the statement AUTOMATIC TARGET REGISTRATION: YES to the configuration file of the server. The configuration file for the Software Distribution for AIX is named nvdm.cfg and is located in the directory /usr/lpp/netviewdm/db.

Refer to 4.5.2, “Using Automatic Client Registration” on page 70 for a complete description of how to set up the server and client to use the automatic client registration.

Part 3. Simple Scenarios Using the Mobile Client

In this part we look at using the Mobile Client in simple scenarios. In Part 4, “Advanced Scenarios Using the Mobile Client” on page 119 we investigate more complex scenarios based on the foundations built here.

It is not possible to cover every combination of client and server platform in this redbook. The chapters that follow are therefore ordered by change management function rather than software platform, so you will find that we begin by introducing how to create a change object. For this we use OS/2 clients with a Software Distribution for OS/2 server. Later we use a Windows NT client with a Software Distribution for AIX server as an example. Across all platforms (both client and server) the command line interface is consistent; the significant differences come in the graphical interfaces of the different operating systems.

Note

The experienced reader may wish to skip directly to Chapter 7, “LAN-Connected Mobile Client” on page 103. Earlier chapters are intended for readers who are not familiar with TME 10 Software Distribution.

© Copyright IBM Corp. 1997

Chapter 6. First Steps

This chapter introduces TME 10 Software Distribution. First we create a change object that contains one file. Once created we install this to a standard client to show how this works. In the following chapters we work with the Mobile Client.

Note

You can skip this section and go directly to Chapter 7, “LAN-Connected Mobile Client” on page 103 if you already have a change object defined in your catalog. This section is intended for people who are not familiar with creating change objects in TME 10 Software Distribution.

6.1 Environment

We set up a LAN as shown in Figure 46 on page 90 with three OS/2 PCs. All machines are running OS/2 Warp Version 3 and were installed as described in Part 2, “Setting Up a Mobile Client Environment” on page 41 using TCP/IP. All references to machines in this chapter use the IP host name. The laptop was an IBM ThinkPad 755cx.

© Copyright IBM Corp. 1997

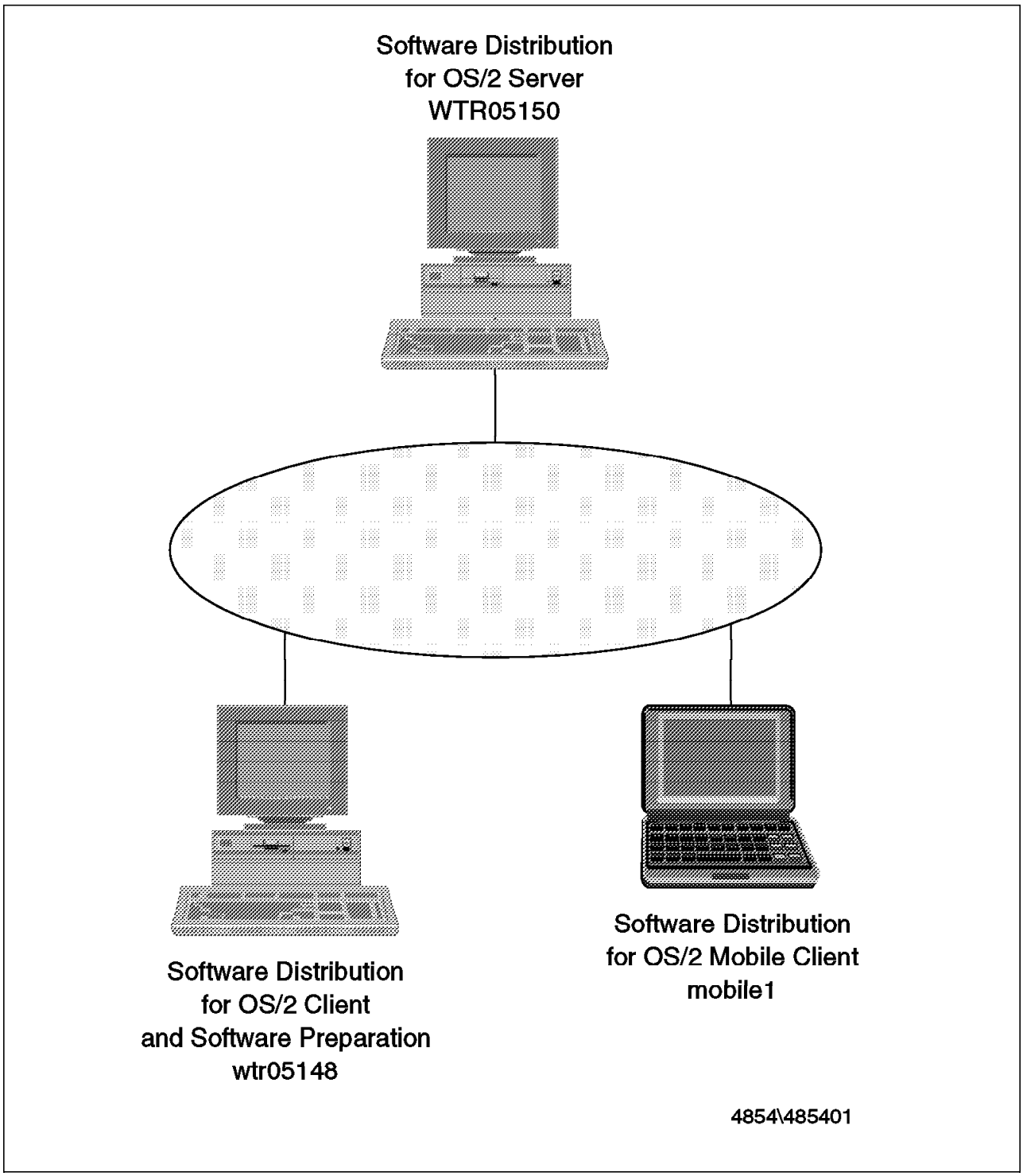


Figure 46. Test Environment at ITSO Raleigh

Note

The configuration of wtr05148 is not described in Part 2, “Setting Up a Mobile Client Environment” on page 41. This is a standard client that we use to prepare change objects.

To recreate the examples that follow you will need to create an environment similar to ours.

Operating Systems

The examples show OS/2 clients and servers. If you wish to perform these examples using other client or server platforms, you should experience no problems if you use the command line options shown in boxes.

6.2 Creating a Change Object

First we need to create a change object. The steps that follow are the same regardless of the complexity and type of the change object, so for simplicity we create a simple generic object consisting of a single file. Descriptions of other change objects can be found in Chapter 12, “Using Change Objects with Mobile Clients” on page 215.

We create our change object on a standard client using the Software Preparation feature. We could just as easily use a Mobile Client, but by working with a normal client first we can see the difference when we introduce the Mobile Client. The normal client is installed very much like the Mobile Client. For further information refer to *Tivoli TME 10 Software Distribution Client Installation and Configuration*, SH19-4337.

To create a change object follow these steps:

1. From the OS/2 Desktop, open the TME 10 Software Distribution for OS/2 folder. You should see something similar to Figure 47.

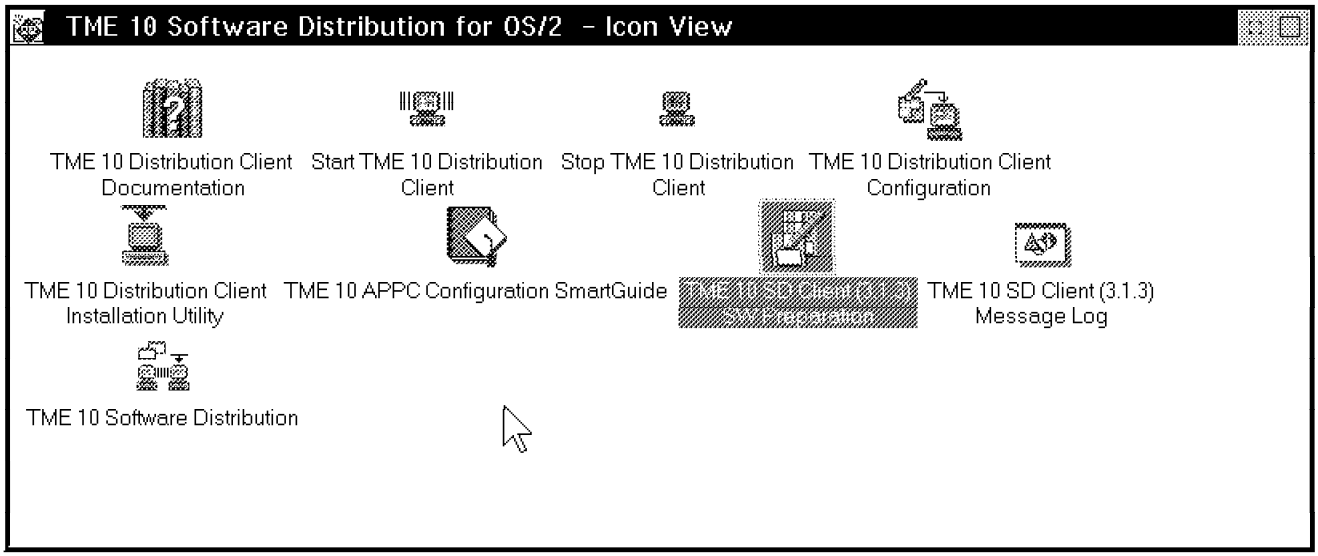


Figure 47. TME 10 Software Distribution for OS/2 Folder

2. Double-click on the **TME 10 SD Client (3.1.3) SW Preparation** icon to start the Software Preparation application.
3. From within the Software Preparation container select the **Generic Software** icon as shown in Figure 48 on page 92.

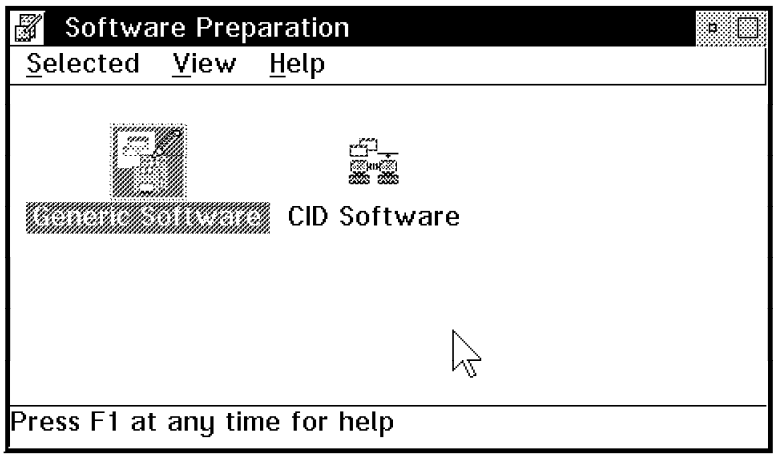


Figure 48. Software Preparation Container

4. You are prompted to log on to your server as shown in Figure 49. If you have not set up the security then the default user ID is root with no password. If you have connections to more than one server, then you must select the server that you wish to use.

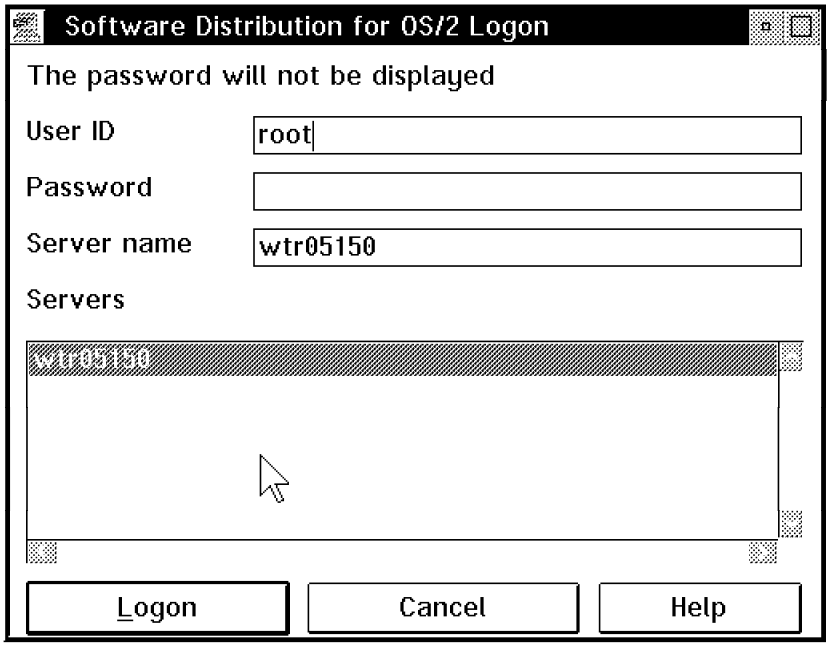


Figure 49. Server Logon

Note

If you experience problems at this stage, then you should check your environment. In particular, check the NVDM.CFG file, both on the server and client. This file is located in the root of the install directory, which is C:\SOFTDIST by default. Remember that the server name is case sensitive. If you are using TCP/IP as the transport protocol then check that you can resolve the server name by opening an OS/2 Window and typing: `HOST SERVERNAME` where SERVERNAME is the IP name of the Software Distribution for OS/2 server. Other problems are likely to be network related. Check in the FNDLOG file on both machines for further information. The FNDLOG is also located in the root of the install directory.

5. You should now see the Software Object Profiles window (Figure 50) with the Help window alongside. Initially this window will be empty. To create a new change object select **Software** from the menu bar. From the pull-down menu choose **New** and then **Create new...**. This will bring up the New Software Profile - Create another window as shown in Figure 51 on page 94.

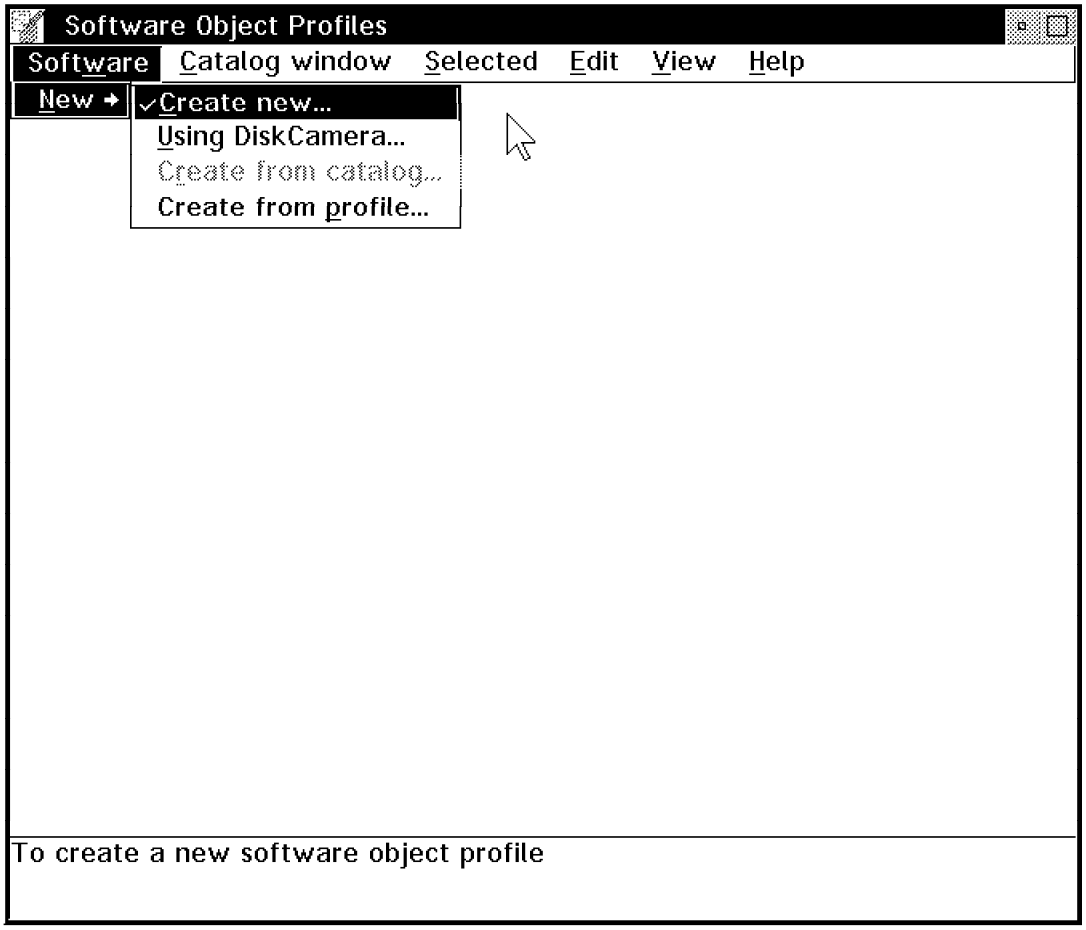


Figure 50. Software Object Profiles

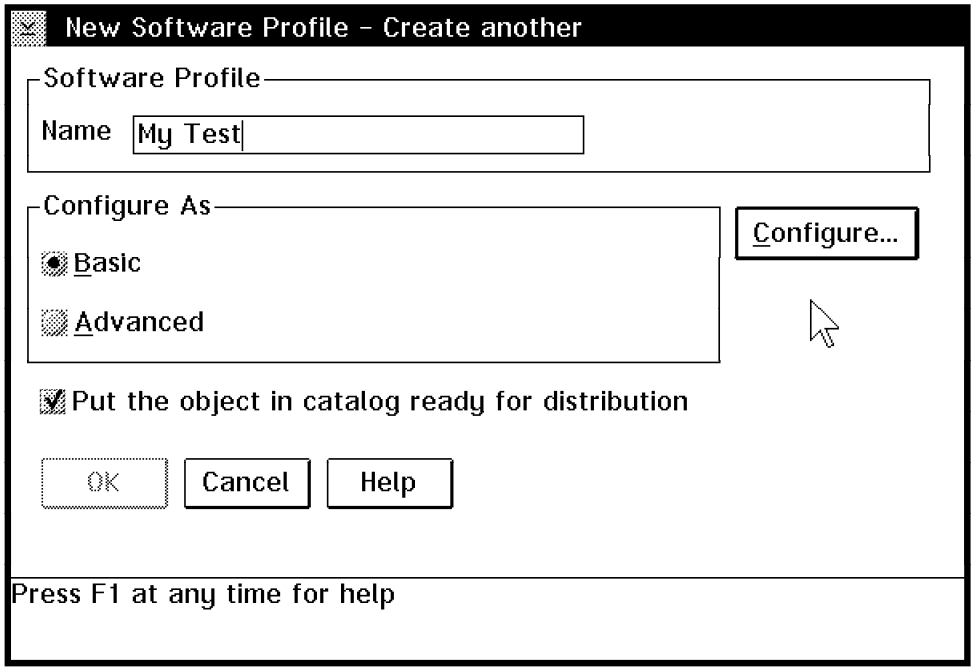


Figure 51. Create Another Window

6. In the Software profile group box fill in the Name field. This is a free format description. Enter My Test. Ensure that the check box is selected so that the object will go into the server's catalog when complete.
7. Click on the **Configure...** push button. For the moment you can leave the level of configuration as basic. If you choose advanced configuration you are presented with many more choices than we need at this stage.
8. You will now be presented with the Basic Configuration notebook as shown in Figure 52 on page 95. On the General page, leave the product type as Refresh. (This indicates that this object can stand on its own and is not an upgrade or a fix for a previously installed product.) Overwrite Component Name with a name for your change object, for example OS2.TEST.OBJECT. You can leave the level and version at the default values. These are used to differentiate between different software levels and country versions of software. For example, LAN.SERVER.REF.3.US may represent the US version of LAN Server 3, while LAN.SERVER.REF.4.UK may represent the UK version of LAN Server 4. In our case we only have one level and version.

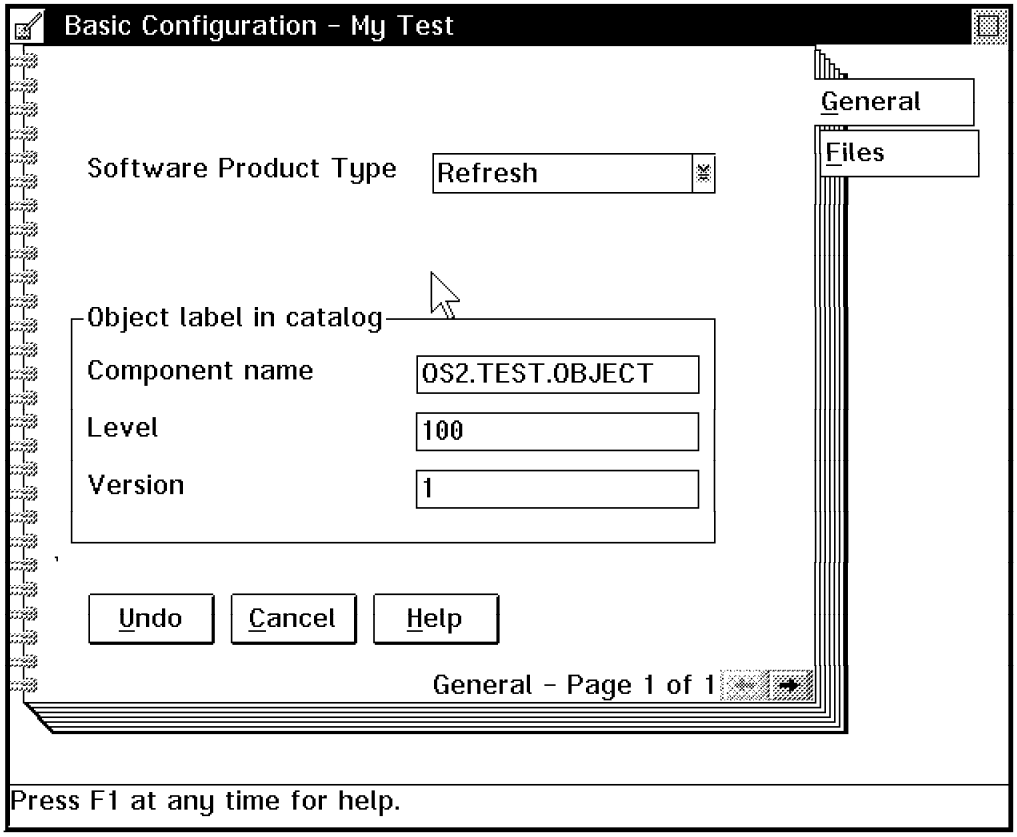


Figure 52. Basic Configuration Notebook - Page 1

9. Select the **Files** tab to go to the next page of the notebook (Figure 53 on page 96.) Enter a file name or click on the **Find...** push button to search for a file. In our example we are using CONFIG.SYS from the C: drive. Leave the action as COPY FILES and select **Add**. Now select **Options** and change the destination file to be C:WORKS.OK to avoid overwriting the CONFIG.SYS when it comes to install it.

   Information

   What you are doing is selecting the files (in this case C:\CONFIG.SYS) to put inside the change object. When this is installed, a file is created on the target machine called C:\WORKS.OK, which will have the contents of your current C:\CONFIG.SYS.

10. Press ALT+F4 to close the notebook and click **OK** to close the New Software Profile - Create another window. Your object will now be created and copied to the Software Distribution for OS/2 server and cataloged.

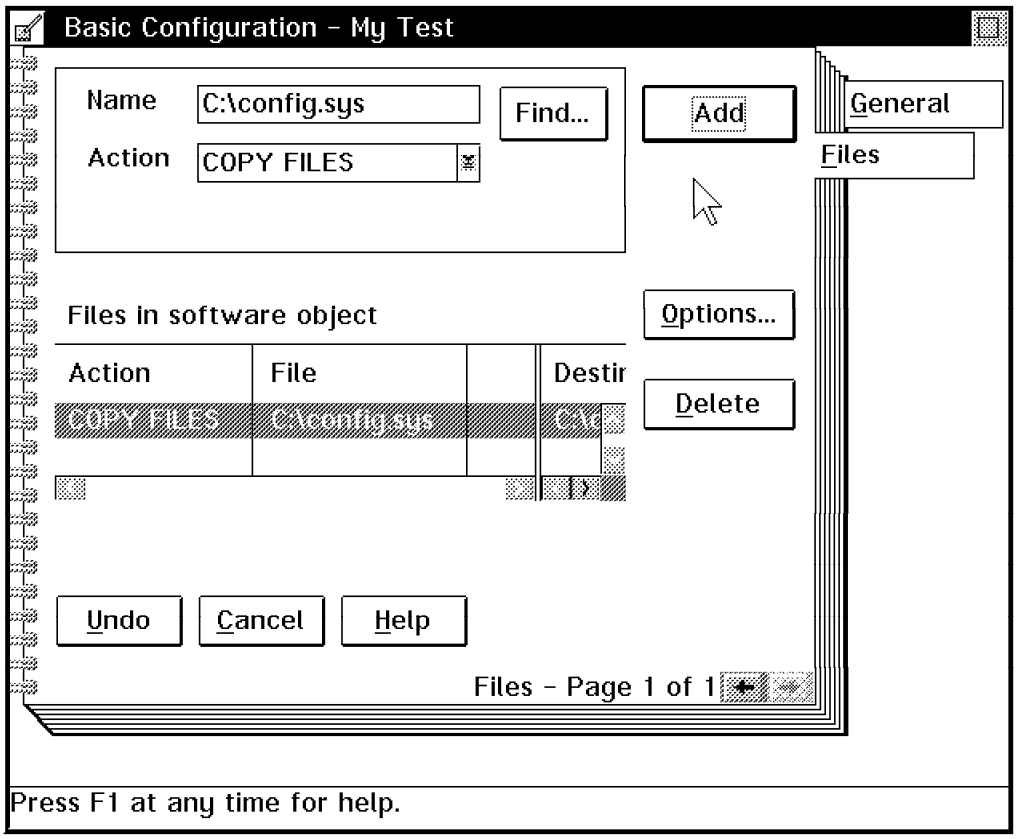


Figure 53. Basic Configuration Notebook - Page 2

Note

If you receive the message:

```
FNDSG029E: Error reading the software profile.
```

then you have an early version of the code that contains a bug. You should request a fix from your Service Representative. In the meantime you can continue by selecting **OK** and editing the profile file in SOFTDIST\SWPRO. Remove the invalid line:

```
INSTALLATION DURATION:          12:00 AM
```

Click your object with the right mouse button and click on **Build** from within the Software Object Profiles window.

You have now successfully built a change object and stored it on your server.

Command Line Option

To achieve the same result from the command line, use a text editor to create the following file and save it as MY.PRO:

```
GLOBAL NAME:                OS2.TEST.OBJECT.REF.100.1
DESCRIPTION:                My Test
CHANGE FILE TYPE:           GEN
COMPRESSION TYPE:           LZW
REBOOT REQUIRED:            NO
REMOVABLE:                  YES
ACTIVABLE:                  YES
INTERACTIVE:                NO
AUTHORIZE:                  NONE
SW HISTORY RESET:           NO
COST:                       0
PACK FILES:                 NO
SECURE PACKAGE:             NO
OBJECT:
    SOURCE NAME:            c:\config.sys
    TARGET NAME:            c:\works.ok
    TYPE:                   FILE
    ACTION:                 COPY
    INCLUDE SUBDIRS:        NO
    GENERAL ATTR:           -A----
    UNIX ATTR:              ----------
    NETWARE ATTR:           -------
```

The above file is called a change file profile. Now from the directory containing MY.PRO run:

```
NVDM BLD MY.PRO
```

Enter the user ID and password when prompted.

Note

You can avoid being prompted for the user ID and password when using the command line interface by setting them in the CONFIG.SYS. For example, add the line:

```
SET FNDUSER=root
```

If you have a password set then also add the line:

```
SET FNDPASSWORD=mypasswd
```

6.3 Installing on a Distribution Client

Note

The experienced user may wish to skip this section and go directly to Chapter 7, “LAN-Connected Mobile Client” on page 103. This section is intended for readers who are not familiar with installing change objects using TME 10 Software Distribution.

In this section we describe how to install the change object that you have just created onto a standard Distribution Client. This is an important step in identifying the differences between the distribution client and the Mobile Client.

You have built a change object using a preparation client. The next step is to install this object onto a distribution client. A preparation client is used to build the change object, which can then be sent to a distribution client for installation.

The only difference between the two kinds of client is that the preparation client has additional software to allow the building of change objects.

We now use the same client as a distribution target for our change object. To do this you need to start the graphical user interface on a machine connected to the Software Distribution for OS/2 server and push the change object to the distribution client.

In our case we use the distribution client itself to do this. This is in fact a pull operation. Remember that a pull is a special case of a push, where the request to perform the operation must originate from the target machine.

1. Start the graphical user interface on the distribution client by selecting **TME 10 Software Distribution** from the main Software Distribution for OS/2 folder or by running:

   ```
   NVDMGI
   ```

   from an OS/2 window.

   Hint

   If the graphical user interface appears to hang for any reason it may be because of the configuration files that are maintained for each user. It is worthwhile deleting all of the files in \SOFTDIST\UICFG to see if this helps.

   Also, if you move your client to point at another server, then you should clean up this directory or information here will cause the client to attempt to connect to the old server.

2. The logon window will appear as before. Enter the user ID and password and select the server.

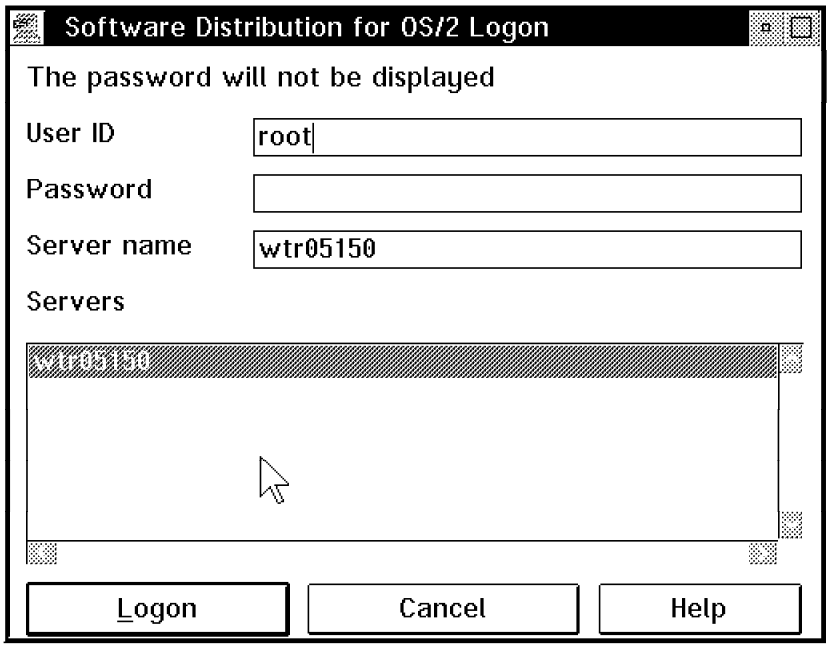


Figure 54. Server Logon

3. You are now presented with the catalog (Figure 55). If this is the first time that you have used Software Distribution for OS/2, then the catalog will contain only the object that you have just created.

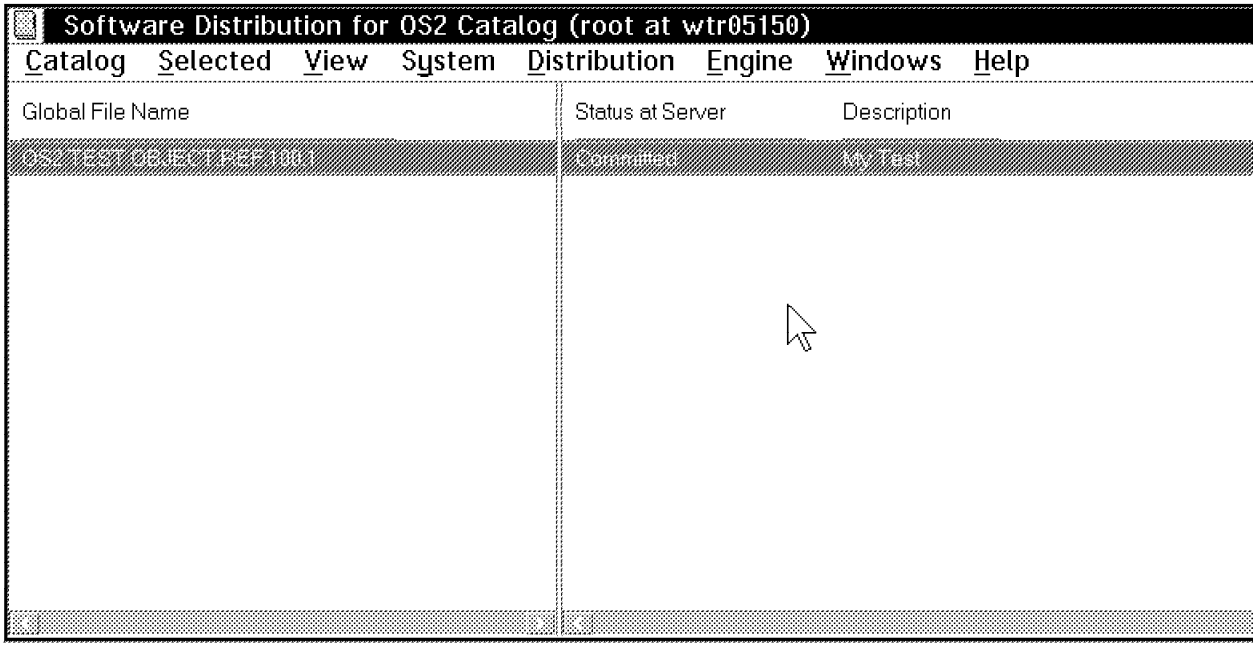


Figure 55. Software Distribution for OS/2 Catalog

4. Select your object and from the menu bar choose **Selected** and **Install...**. This will display the Install Change Files window as you see in Figure 56 on page 100.

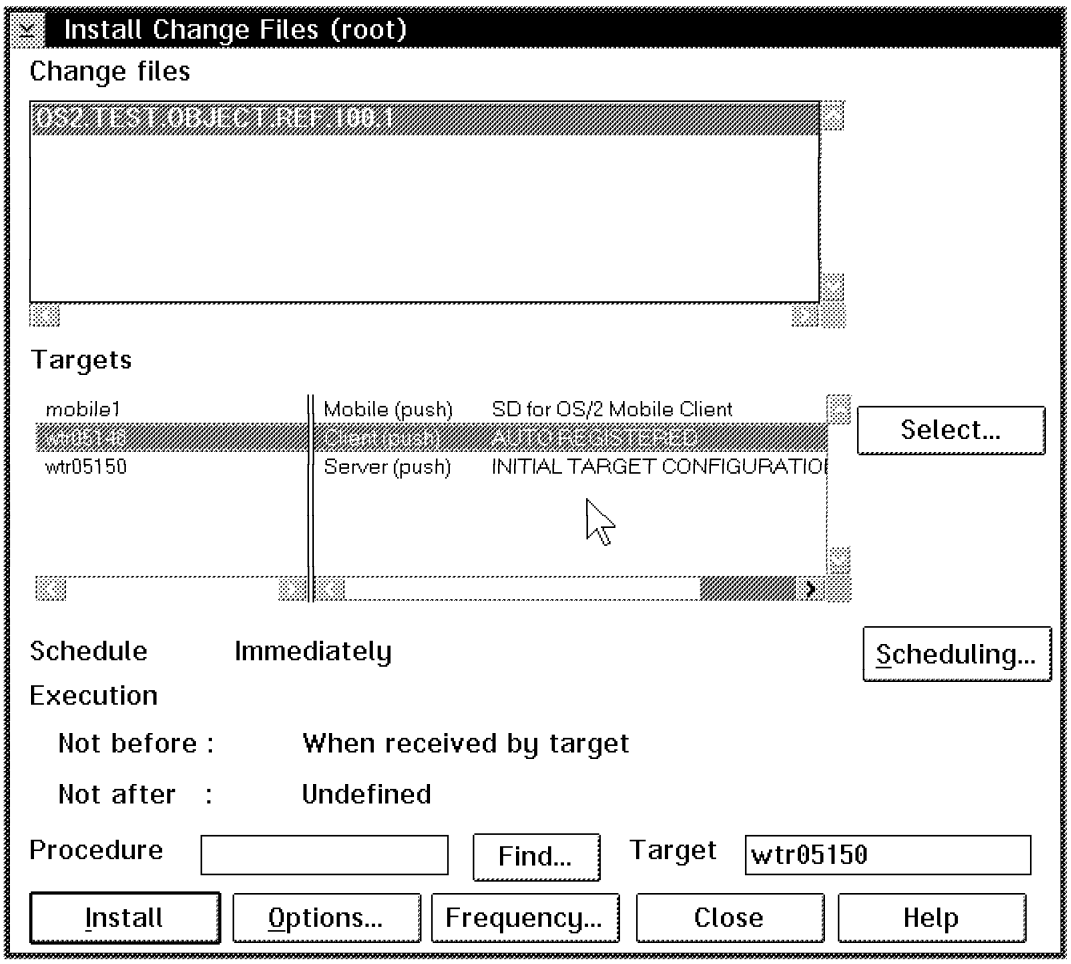


Figure 56. Install Change Files Dialog

5. Select your PC from this list. In our case this was wtr05148, with wtr05150 being the server. Click the **Install** push button to install this object. You should now see a confirmation dialog box similar to Figure 57.

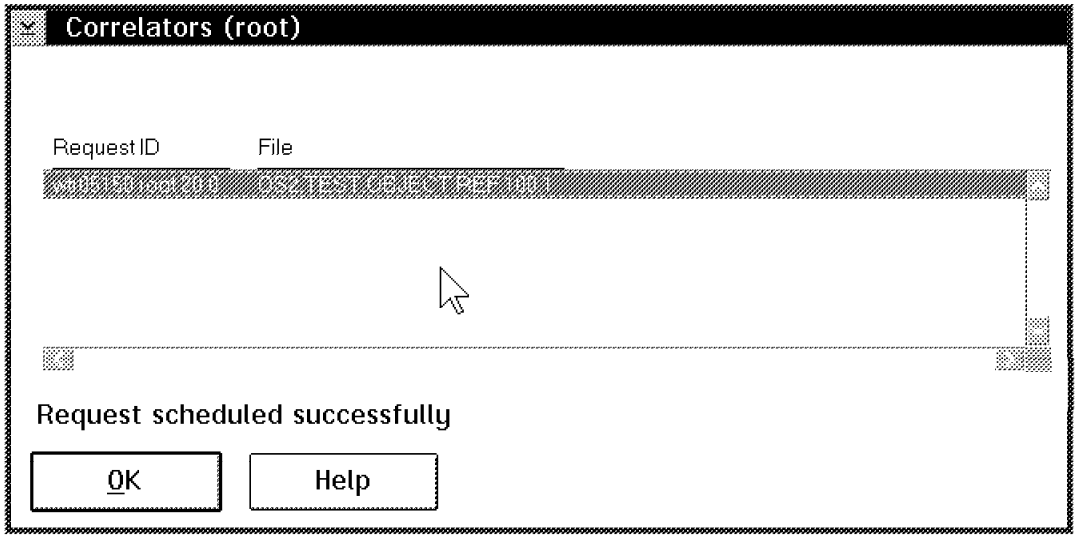


Figure 57. Correlator Window

You have now successfully submitted a change object for installation. You should shortly find a new file on the C: drive of your distribution client as shown in Figure 58 on page 101.

```
C:\>dir *.ok

The volume label in drive C is OS2.
The Volume Serial Number is 6692:8C15.
Directory of C:\

10-16-96   2:05p      4369           0  works.ok
        1 file(s)       4369 bytes used
                     1350144 bytes free

C:\>
```

Figure 58. The File Successfully Installed

Command Line Option

To perform the same action from the command line type:

```
NVDM INST OS2.TEST.OBJECT.REF.100.1 -w wtr05148
```

Replacing wtr05148 with the name of your distribution client.

Chapter 7. LAN-Connected Mobile Client

So far you have created a change object that installs a file and verified that it works by using a distribution client. While a Mobile Client is connected to the LAN, it expects to be treated in the same way as a distribution client and to receive any change objects that are directed at it. The first step with the Mobile Client is to see how this can happen.

7.1 Basic Installation

Installation on the Mobile Client follows the same process as for installation on a Distribution Client as described in 6.3, “Installing on a Distribution Client” on page 97.

Operating Systems

As before, if you wish to follow these examples using operating systems other than OS/2, then follow the command line examples.

7.1.1 Submitting the Install Request

Use the following procedure to install OS2.TEST.OBJECT on a Mobile Client. Note that if you have not already done so you must install a Mobile Client as described in Part 2, “Setting Up a Mobile Client Environment” on page 41.

1. Start the graphical user interface on the distribution client by selecting **TME 10 Software Distribution** from the main Software Distribution for OS/2 folder or by running:

   ```
   NVDMGI
   ```

   from an OS/2 window. This can be done from the distribution client as before.

2. The logon window will appear. Enter the user ID and password and select the server.

© Copyright IBM Corp. 1997

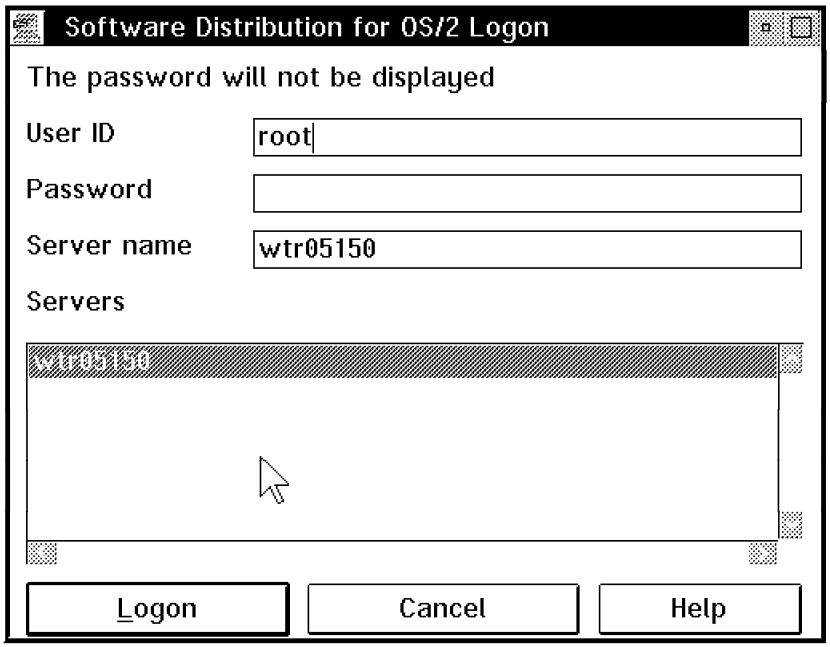


Figure 59. Server Logon

3. You are now presented with the catalog (see Figure 60).

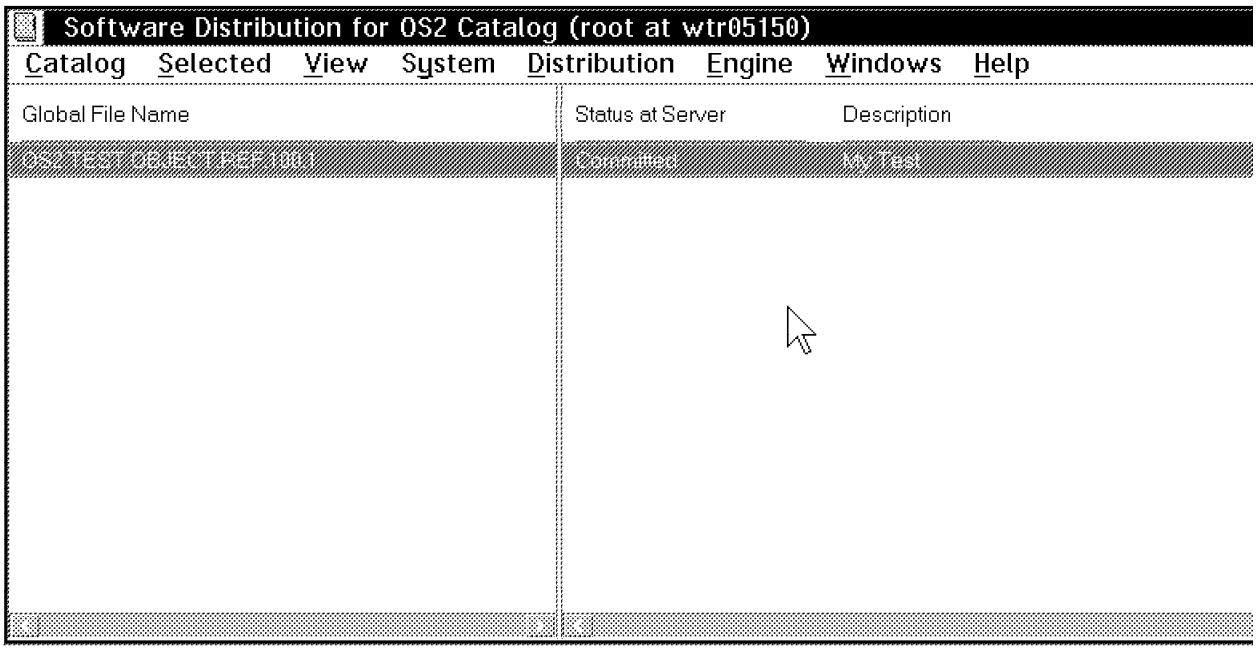


Figure 60. Software Distribution for OS/2 Catalog

4. Select your object and then from the menu bar choose **Selected** and **Install...**. This will display the Install Change Files window as shown in Figure 61 on page 105.

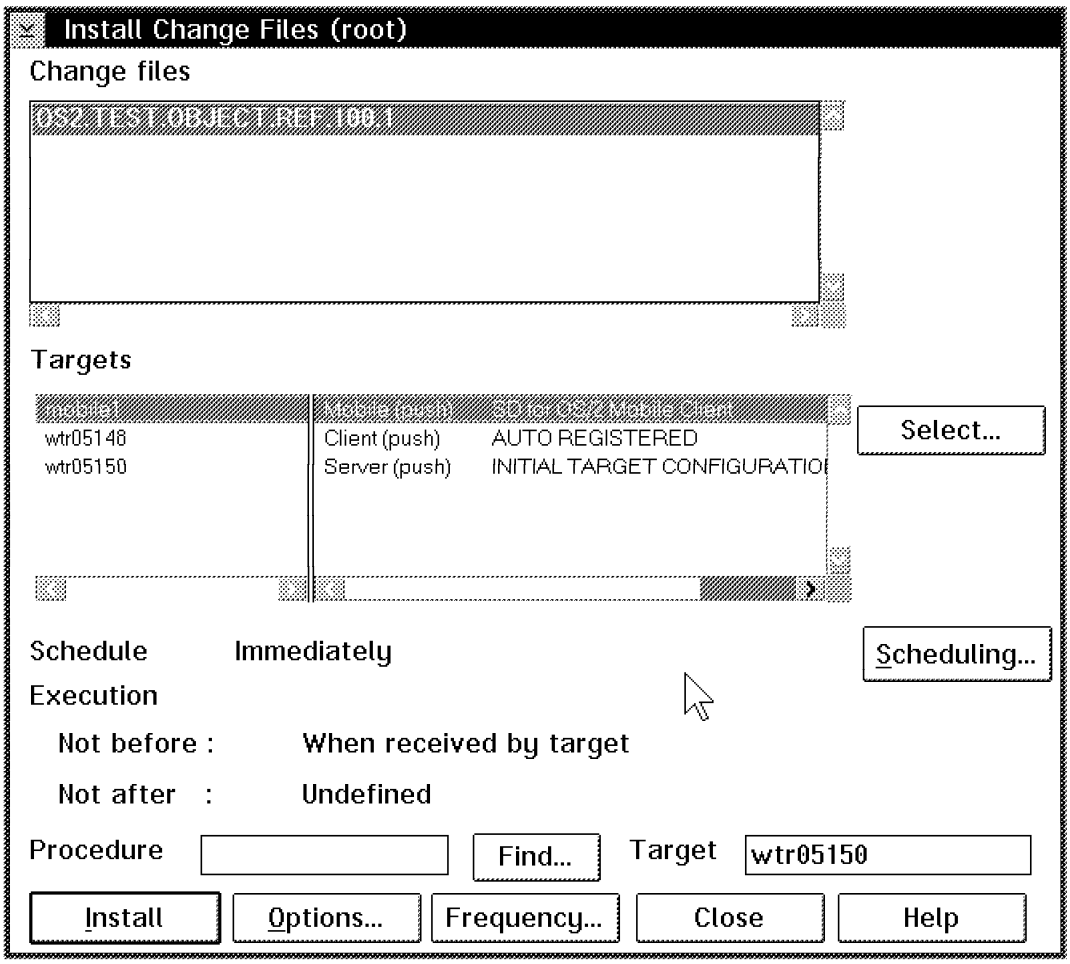


Figure 61. Install Change Files Dialog

5. Select the Mobile Client PC from this list. In our case this was mobile1. Click the **Install** push button to install this object. You should now see a confirmation dialog box similar to Figure 62.

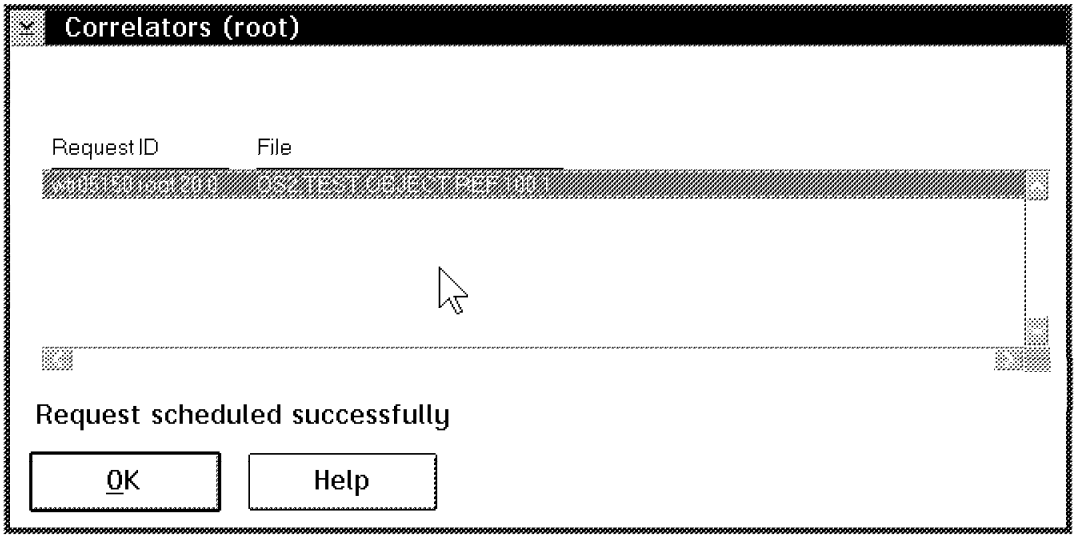


Figure 62. Correlator Window

You have now successfully submitted a change object for installation.

Command Line Option

To perform the same action from the command line type:

```
NVDM INST OS2.TEST.OBJECT.REF.100.1 -w mobile1
```

Replace mobile1 with the name of your Mobile Client.

This can be performed from the Software Distribution for OS/2 server or any connected client.

7.1.2 Connecting to the Mobile Client

The install that you have just submitted has been added to a queue. It will not take place immediately because the Mobile Client is not connected to the server. To have your request executed you will need to force a connection between the client and the server.

To connect from the server:

1. Start the graphical user interface and log on. This can be done from the Software Distribution for OS/2 server or from the distribution client.
2. From the Software Distribution for OS/2 Catalog window you need to move to the Targets window. From the menu bar select **Windows** and then **Targets**. This should now display the Software Distribution for OS/2 Targets window as shown in Figure 63.

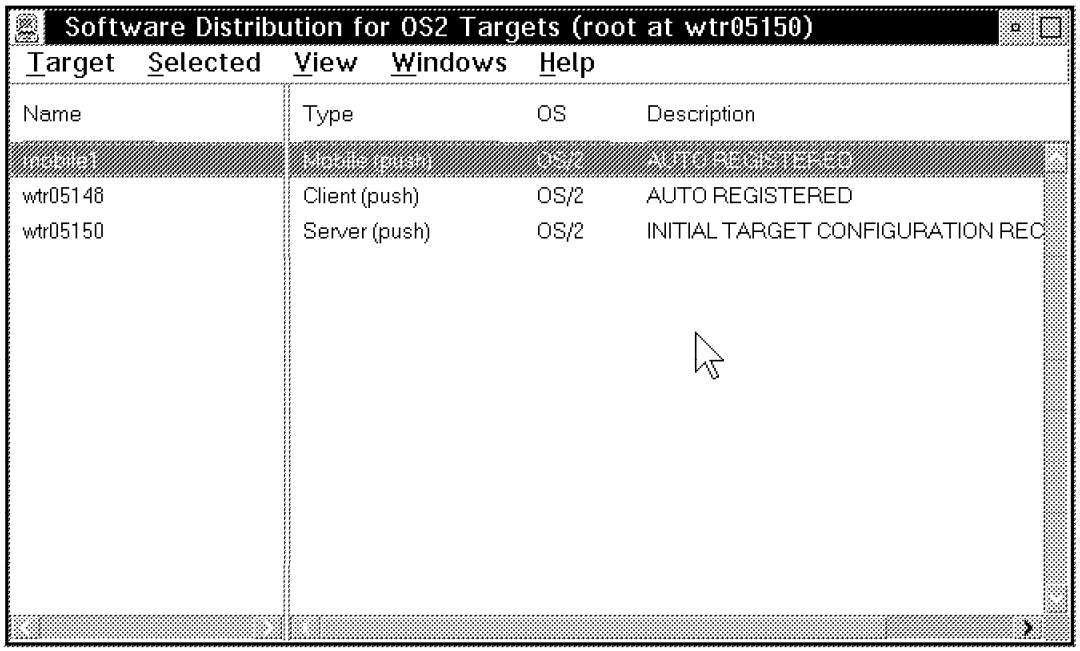


Figure 63. Software Distribution for OS/2 Targets Window

3. Now to confirm the status of the Mobile Client select it and choose **Selected** and then **Status...** from the menu bar. This will display the status window as shown in Figure 64 on page 107. You will notice that the status is Not Available. This is because there is currently no connection window open between the Mobile Client and the server. Click **OK** to dismiss the window.

Command line option

To view the Mobile Client status from the command line, type:

```
NVDM STATTG mobile1
```

Replace mobile1 with the name of your Mobile Client.

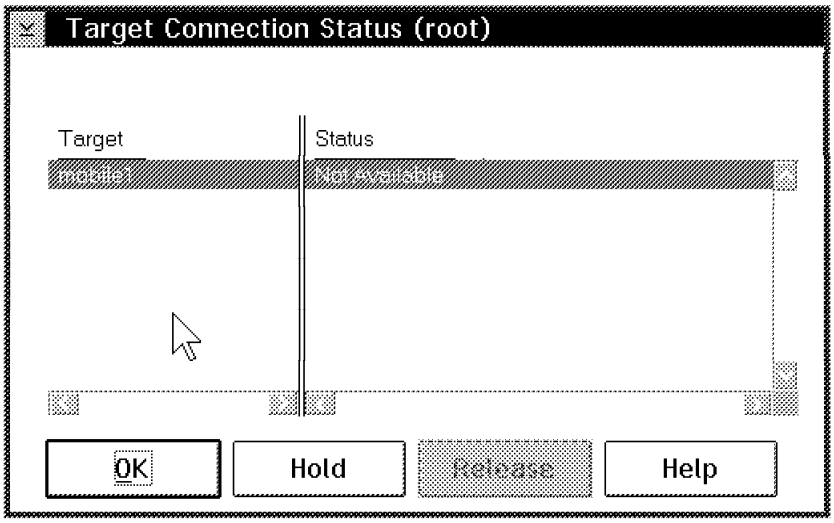


Figure 64. Target Connection Status Window

4. To connect to the client, ensure the client is selected and choose **Selected** and then **Connect...** from the menu bar on the Targets window. This will display the Mobile Client Connect window as you see in Figure 65 on page 108. The time defaults to the current time and the duration defaults to one hour. Click **OK** to connect to the client.

Note

On the AIX server there are currently no items available in the pull-down menu to connect or disconnect a client as shown here for the OS/2 server.

So, at the moment you will have to use the command line interface on the AIX server, using the nvdm connect and nvdm disconnect commands.

Please check with your IBM representative to obtain current version information.

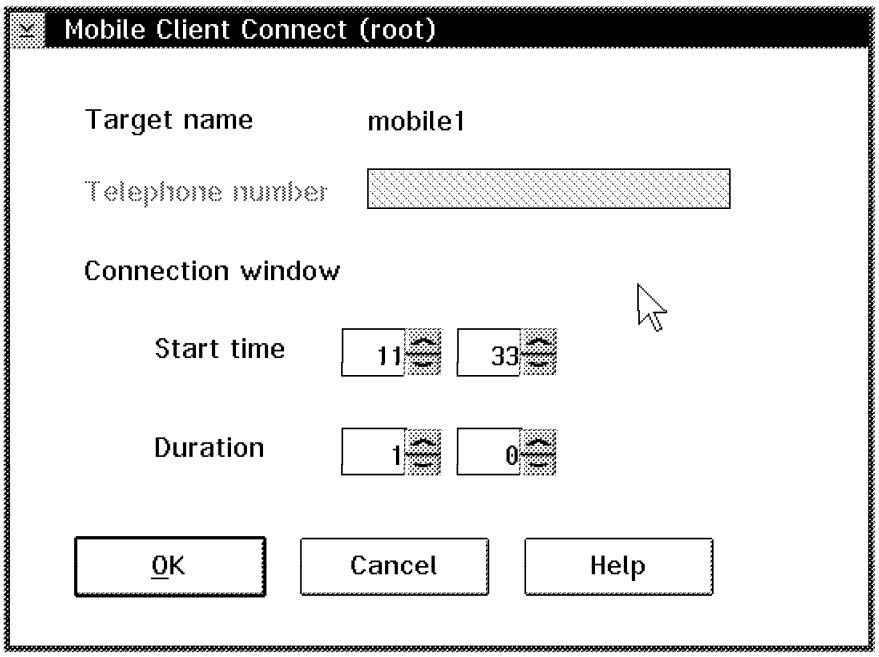


Figure 65. Mobile Client Connect Window

5. If you now check the status of the Mobile Client you will see that it is active as shown in Figure 66. You should shortly find a new file on the C: drive of your Mobile Client as shown in Figure 67 on page 109.

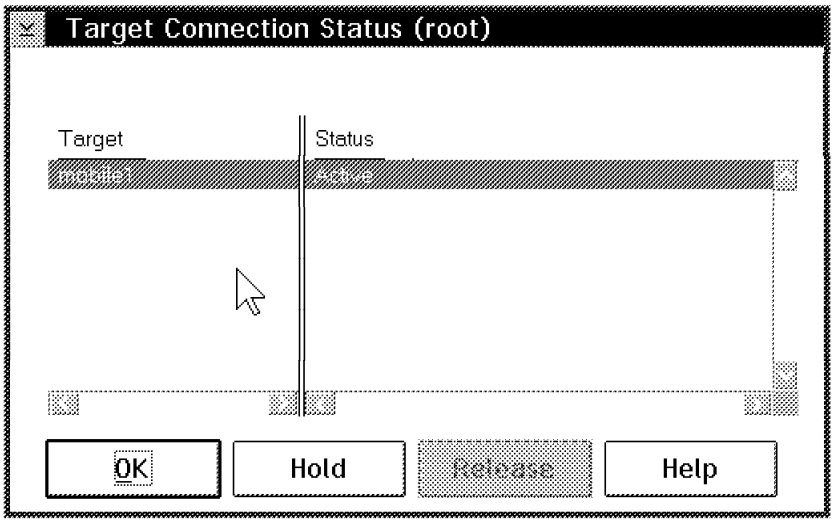


Figure 66. Target Connection Status Window

```
C:\>dir *.ok

The volume label in drive C is DRIVE_C.
The Volume Serial Number is 9343:8123.
Directory of C:\

10-16-96   2:05p      4369           0  works.ok
        1 file(s)       4369 bytes used
                     3629173 bytes free

C:\>
```

Figure 67. The File Successfully Installed

Command Line Option

To perform the same action from the command line type:

```
NVDM CONNECT mobile1
```

Replace mobile1 with the name of your Mobile Client.

This can be performed from the Software Distribution for OS/2 server or any connected client.

You can also connect from the client by typing:

```
NVDM CONNECT
```

The main point to note here is that no communication will take place between the Mobile Client and its server unless a connection window is open.

7.2 Disconnected Immediate Installation

In this section we look at how to install an object using the resources of the Mobile Client only to control the installation. We use the Software Distribution for OS/2 server to send the catalog entry to the Mobile Client's repository and then submit the install request to happen upon disconnection.

7.2.1 Preparation

Before we do this we need to either create a new change object or remove OS2.TEST.OBJECT from mobile1. To create a new object follow the steps outlined in 6.2, "Creating a Change Object" on page 91.

7.2.1.1 Removing the Object from the Mobile Client

To remove the object:

1. First ensure that a connection window is open to the client. Either use the graphical user interface as described in 7.1.2, “Connecting to the Mobile Client” on page 106 or enter:

   ```
   NVDM CONNECT mobile1
   ```

   from the server or a machine connected to the server.

   Note: If you receive the message:

   ```
   FNDCLE36E: A connection window cannot be redefined once it is open.
   ```

   then a connection already exists to the Mobile Client.

2. Now start the graphical user interface and display the catalog window as shown in Figure 60 on page 104. Select the object **OS2.TEST.OBJECT** and from the menu bar choose **Selected**. From the pull-down menu choose **Open** and then **History...**. This will display the history window as shown in Figure 68.

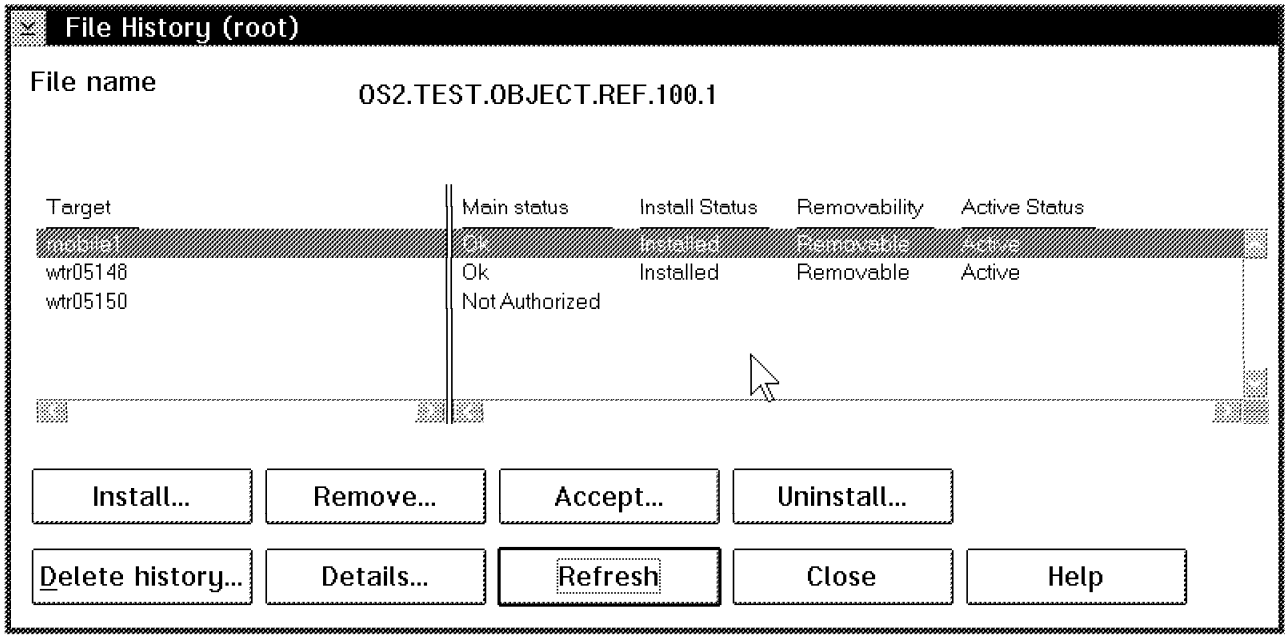


Figure 68. File History Window

3. Select the Mobile Client and click the **Remove...** push button. This will display the Remove Change Files window as seen in Figure 69 on page 111.

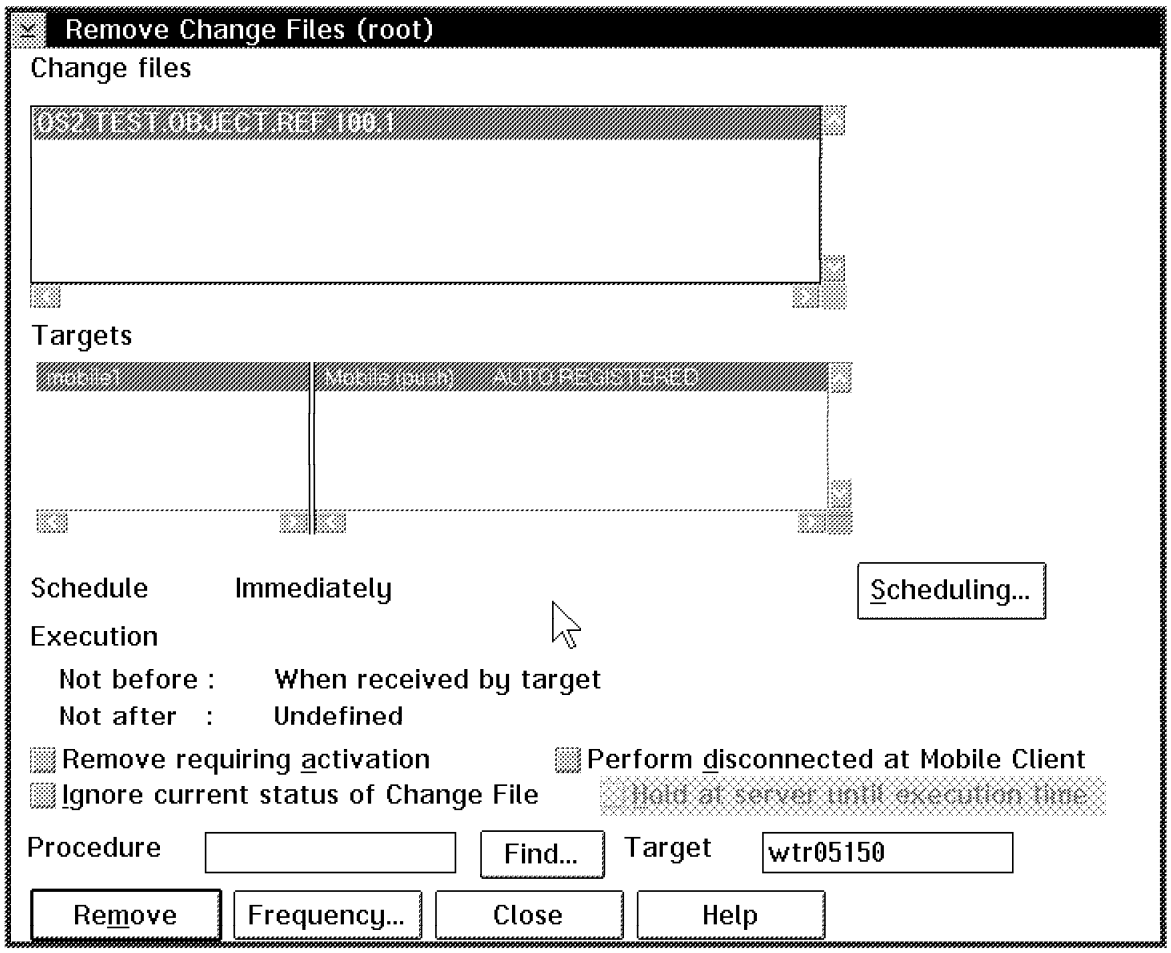


Figure 69. Remove Change Files Window

4. Select the **Remove** push button to remove OS2.TEST.OBJECT immediately. Shortly you should see some activity on your Mobile Client and the file C:\WORKS.OK is removed.
5. To leave the Mobile Client in its default, disconnected state do one of the following:
   a. Select the client from the Targets window (see Figure 63 on page 106.) and from the menu bar select **Selected** and **Disconnect...**
   b. Issue:

      ```
      NVDM DISCONNECT mobile1
      ```

      from the command line.

Command Line Option

To perform the same action entirely from the command line at the server or a client type:

```
NVDM CONNECT mobile1
```

Replace mobile1 with the name of your Mobile Client. Then type:

```
NVDM REM OS2.TEST.OBJECT.REF.100.1 -w mobile1
```

and

```
NVDM DISCONNECT mobile1
```

7.2.2 Sending the Change Object

In order for the Mobile Client to install an object without connecting to the server, it will need to find the object in its local repository. For this to be the case, we must send the change object from the server to the Mobile Client.

1. Using the graphical user interface display the catalog window as shown in Figure 60 on page 104. Highlight the object **OS2.TEST.OBJECT** (or your new object if you chose to create one rather than remove OS2.TEST.OBJECT). From the menu bar select **Selected** and then **Send...**. This will display the Send Files window as you see in Figure 70.

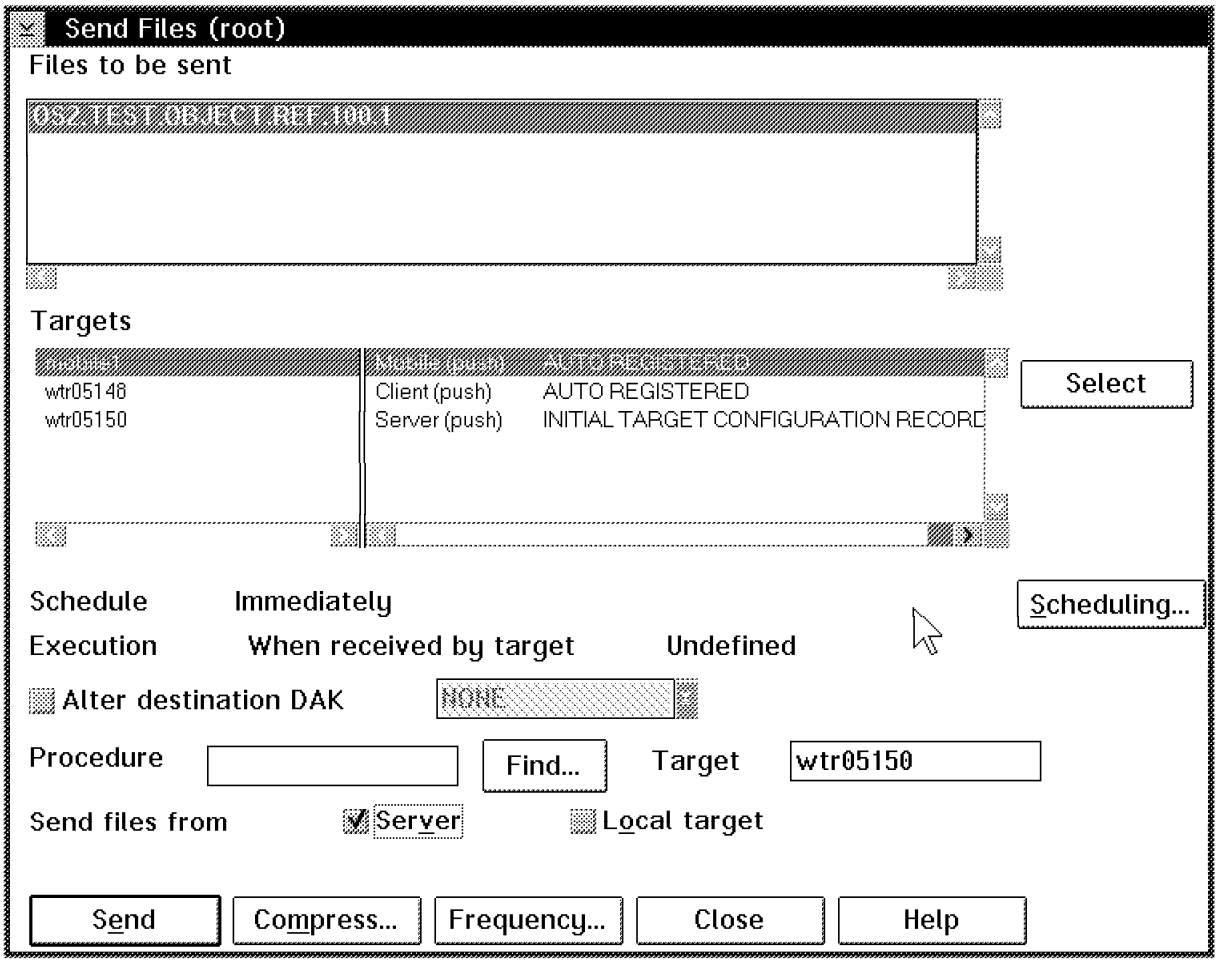


Figure 70. Send Files Window

Select the Mobile Client and ensure that the Send files from Server check box is selected. Click on the **Send** push button to send the object to the Mobile Client.

Note

The fact that you can ″send″ a change file to the Mobile Client is a major difference between a local client and the Mobile Client.

Since the Mobile Client has its own catalog it will allow a change file to be sent to it and stored in its repository. This operation is normally possible only between Software Distribution servers, for example, by sending a change file from a focal point system to a staging server.

2. You will again need to force a connection to the Mobile Client by using the graphical user interface or entering:

   ```
   NVDM CONNECT mobile1
   ```

3. Remember to close the connection when the object has been copied across.

Hint

To see the change objects on the Mobile Client look in the directory \SOFTDIST\REPOS. You should see something like this:

```
C:\>dir \softdist\repos

The volume label in drive C is DRIVE_C.
The Volume Serial Number is 9343:8123.
Directory of C:\

10-21-96   9:02p  11159   0  OS2.TEST.OBJECT.REF.100.1
        1 file(s)       4369 bytes used
                     3620122 bytes free
```

Command Line Option

To perform the same action entirely from the command line, type:

```
NVDM SEND OS2.TEST.OBJECT.REF.100.1 mobile1 -w wtr05150
```

Replace mobile1 with the name of your Mobile Client and wtr05150 with the name of your server. Then type:

```
NVDM CONNECT
```

and:

```
NVDM DISCONNECT
```

7.2.3 Requesting the Install

Now that our Mobile Client has the change object stored locally we can proceed to install it.

1. Using the graphical user interface display the catalog window as shown in Figure 60 on page 104. Highlight the object **OS2.TEST.OBJECT** (or your new object if you chose to create one rather than remove OS2.TEST.OBJECT).

From the menu bar select **Selected** and then **Install...**. This will display the Install Change Files window as you see in Figure 71 on page 114. This is the same window that we have used before, only this time we select additional options.

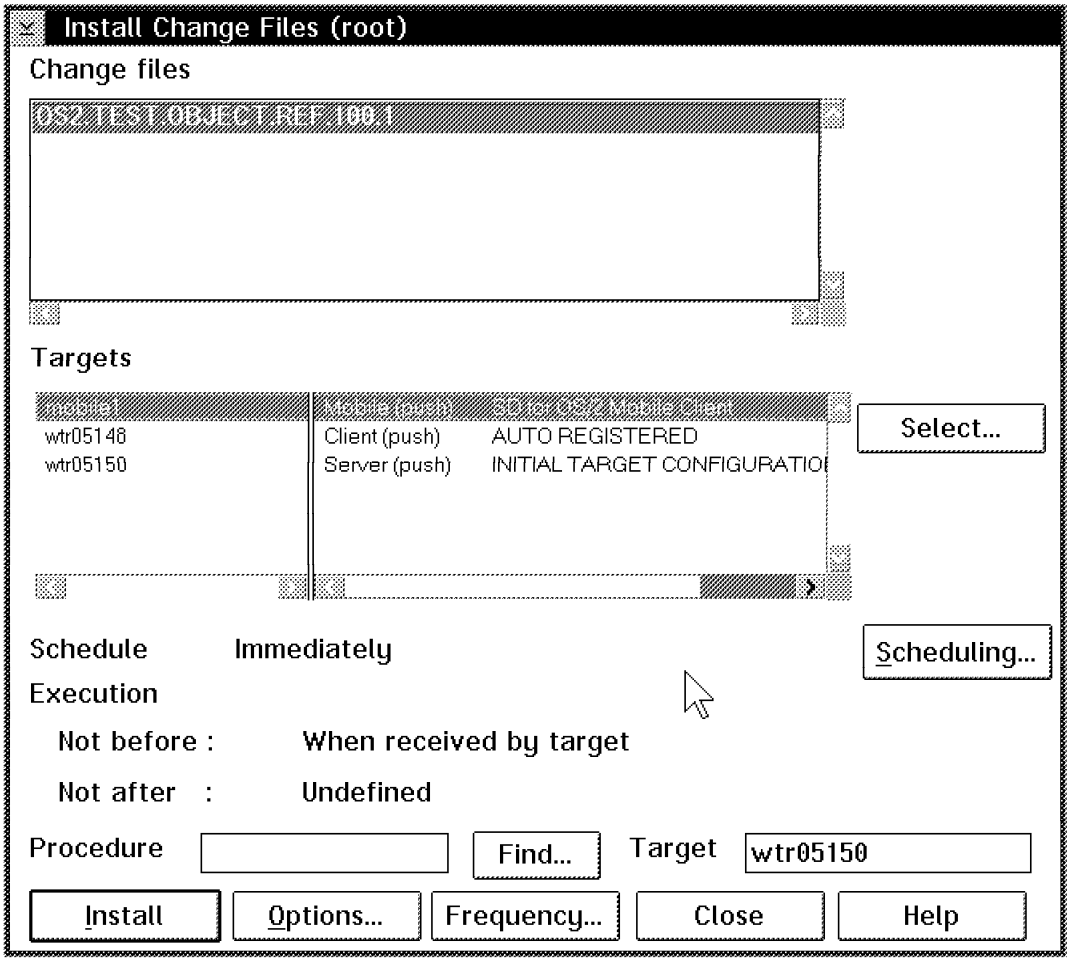


Figure 71. Install Change Files Dialog

2. Click the **Options...** push button to display the Install Options window as shown in Figure 72 on page 115.

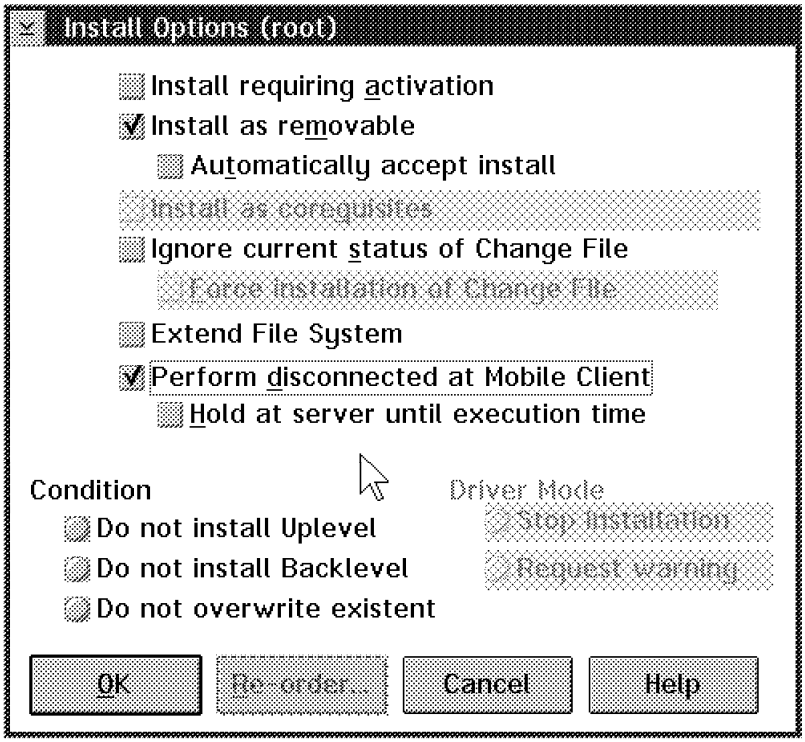


Figure 72. Install Options Dialog

3. You will notice that Install as removable is selected by default. This is why we were able to remove the change object from the Mobile Client. Select the **Perform disconnected at Mobile Client** check box. Click **OK** to perform the action.

Command Line Option

To perform the same action from the command line type:

```
NVDM INST OS2.TEST.OBJECT.REF.100.1 -w mobile1 -dc
```

Replacing mobile1 with the name of your Mobile Client.

This can be performed from the Software Distribution for OS/2 server or any connected client.

7.2.4 Making the Install Happen

So far you have sent the change object to the client and queued a request from the server. The request is to happen in disconnected mode, meaning that a connection to the server is not required to perform the installation.

To tell the Mobile Client that a request is waiting, you need to issue a connect request. This can be done from the graphical user interface or the command line and can be done from the Software Distribution for OS/2 server, the Mobile Client itself or any client connected to the server.

To do this from the server or a client issue:

```
NVDM CONNECT mobile1 -d 2
```

To do this from the Mobile Client enter:

```
NVDM CONNECT -d 2
```

Here we are using the option -d 2 to specify a connect window duration of two minutes. This allows the request to be sent to the Mobile Client but will only keep the connection window open for a short time.

Note

When a connection window is opened, a request to close it is placed on the request queue at the server. When this request is processed the connection is closed. If another request is active then the window is closed after the request has finished.

In the case of a disconnected install as we are doing here, the window can be closed while the request is running. You will see that the install now takes place immediately on the Mobile Client. When we specify that the object is to be installed immediately and disconnected we *allow* the install to occur disconnected but do not insist that the Mobile Client waits until it disconnects to begin the install.

The install should now have completed but the Mobile Client has not told the server whether it was successful or not. To re-synchronize the databases enter:

```
NVDM CONNECT mobile1 -d 2
```

7.3 Deadline Activation

Imagine you are responsible for software distribution to all desktop and laptop PCs within a Sales and Marketing division. Every PC that you look after has the current price list for your company's products held in a Lotus 1-2-3 spreadsheet called C:\SALES\PRICES.WK4. Every month a new price list is issued to take effect on the first day of the month and you are given this file two weeks ahead of time so that you can install it.

The laptop users may not be in the office when the new prices take effect so you have to ensure that the update will happen whether they connect to the LAN that day or not. You can use a process known as deadline activation.

When you receive the file you create a change object on your Software Distribution for OS/2 server. You schedule this to be installed on the desktop PCs on the first day of the following month. For the laptop PCs that are equipped with the Mobile Client you send the change object to their catalogs. Next you schedule an install on the Mobile Clients to happen on the first day of the next month in disconnected mode.

Now even if the laptop is not connected to your Software Distribution for OS/2 server on that date, the machine is updated with the new price list.

Note

Here the change history at the server is updated the next time the Mobile Client is connected to the server.

Let us see how to do this:

1. Build the change object using the new file. This is the same process as we used before to create OS2.TEST.OBJECT. See 6.2, "Creating a Change Object" on page 91 if you need to remind yourself of the technique. We

created an object called SALES.PRICELIST.JAN97.REF.1.US using the profile shown which we then built using the command:

```
NVDM BLD JAN97.PRO
```

```
GLOBAL NAME:                    SALES.PRICELIST.JAN97.REF.1.US
DESCRIPTION:                    Sales Pricelist January 1997
CHANGE FILE TYPE:               GEN
COMPRESSION TYPE:               LZW
REBOOT REQUIRED:                NO
REMOVABLE:                      YES
ACTIVABLE:                      YES
INTERACTIVE:                    NO
AUTHORIZE:                      NONE
SW HISTORY RESET:               NO
COST:                           0
PACK FILES:                     NO
SECURE PACKAGE:                 NO
OBJECT:
    SOURCE NAME:                c:\prod\sales\data\jan97.wk4
    TARGET NAME:                C:\SALES\PRICES.WK4
    TYPE:                       FILE
    ACTION:                     COPY
    INCLUDE SUBDIRS:            NO
    GENERAL ATTR:               -A----
    UNIX ATTR:                  ----------
    NETWARE ATTR:               -------
```

Figure 73. JAN97.PRO

2. Now you must send the change object to your Mobile Client. You can use the graphical user interface as described above, or from the command line enter:

   ```
   NVDM SEND SALES.PRICELIST.JAN97.REF.1.US mobile1 -w wtr05150
   ```

   Remember that this will not actually go to the client until a connect is issued. For simplicity we will not connect now but instead wait until the install is also queued and send both requests together.

3. Next we want to submit the install, which from the command line has the format:

   ```
   NVDM INST SALES.PRICELIST.JAN97.REF.1.US -w mobile1 -dc -db "01-01-97"
   ```

   Here the option -dc specifies an install when disconnected from the server. The option -db "01-01-97" means we want this to happen on January 1, 1997.

   Note

   Take care when entering dates. The format that will be accepted is the format that is defined on the system you are using to issue the command. To check the correct date format open an OS/2 window and type:

   ```
   C:\ >date
   Current date is: Thu 10-24-1996
   Enter the new date: (mm-dd-yy)
   ```

   The prompt (mm-dd-yy) shows that this is the format to use for this PC.

4. Now issue a connect to send the requests to the Mobile Client:

   ```
   NVDM CONNECT mobile1 -d 1
   ```

5. To test that it works from the command prompt set the date on the Mobile Client to January 1, 1997 by entering:

   ```
   DATE 01-01-97
   ```

   Now reboot the system and watch what happens. You should see the file installed. Deadline activation will only take place when the client is first started, either automatically at boot time or by entering NVDM START. Only then is the queue read to see if the request should be processed. If the Mobile Client is not restarted then the request will not take place.

Part 4. Advanced Scenarios Using the Mobile Client

In this part we use the Mobile Client in some more complex scenarios. If you have not used the Mobile Client before you might want to refer to Part 2, “Setting Up a Mobile Client Environment” on page 41 and Part 3, “Simple Scenarios Using the Mobile Client” on page 87 first.

In particular we show the following scenarios:

- Using the Mobile Client in fully disconnected mode
- Using the Mobile Client with PPP
- Using the Mobile Client with IBM 8235 LAN Dialer
- Using the asynchronous support in TME 10 Software Distribution
- Using change objects, for example, CID objects with the Mobile Client
- Using the Mobile Client in a focal point connected environment
- Using the Mobile Client with dynamic IP

© Copyright IBM Corp. 1997

Chapter 8. Using a Fully Disconnected Client

In this chapter we show an example using a fully disconnected client.

A fully disconnected client is a special kind of Mobile Client that never connects to a server. Since a Mobile Client has its own catalog, in fully disconnected mode this catalog can be updated from an external media, for example, a diskette and then the change request is performed locally.

Note

The fully disconnected client is also called permanently disconnected client.

© Copyright IBM Corp. 1997

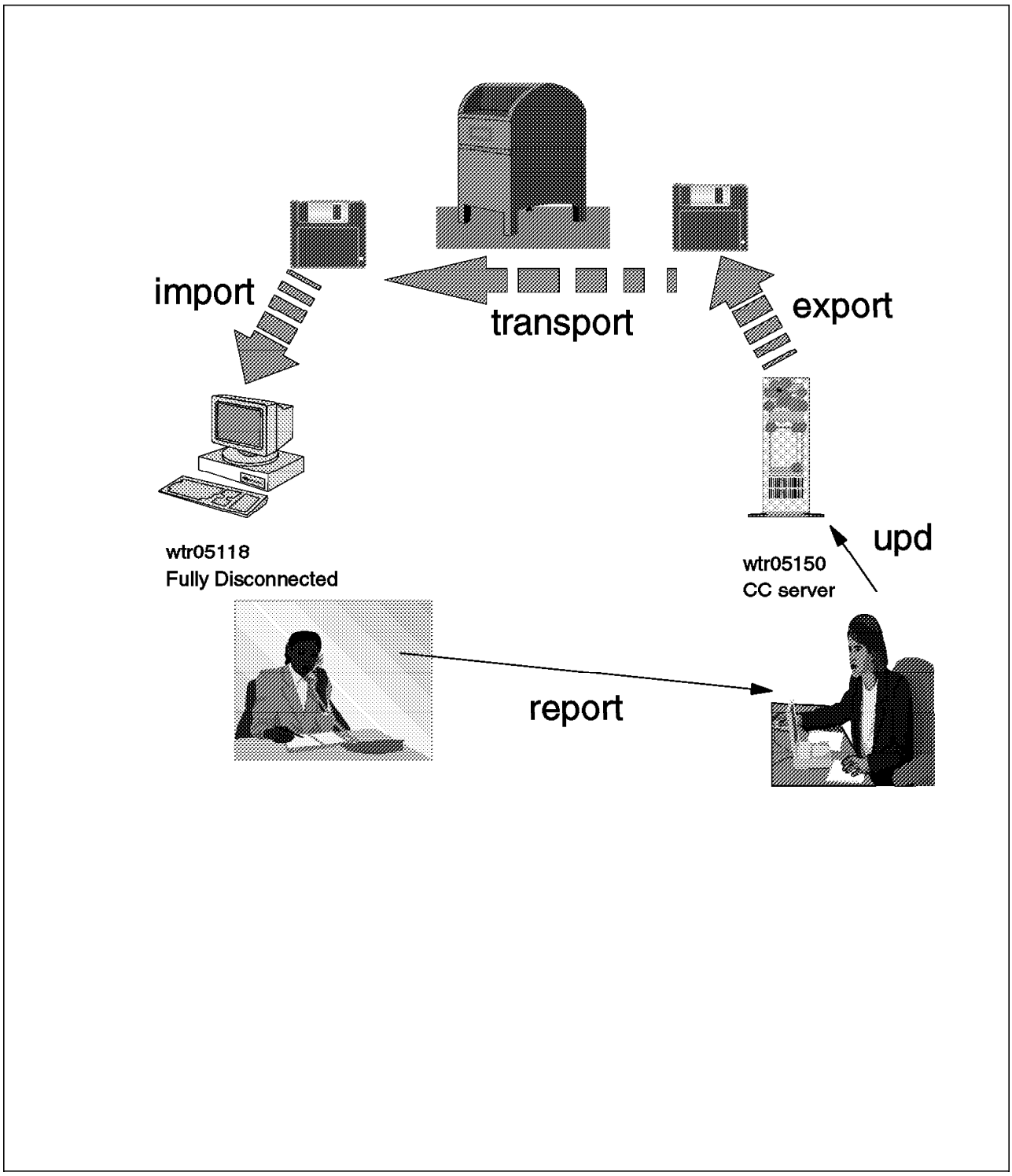


Figure 74. Fully Disconnected Client Scenario

In our example we perform the following activities:

- Configure a fully disconnected client
- Define a fully disconnected client on the Software Distribution for OS/2 server
- Create a change file on the server and export it to a diskette

- Import the change file on the fully disconnected client
- Install the change file on the fully disconnected client
- Update the change control status on the server

8.1 Configuring a Fully Disconnected Client

In our example we reconfigure the OS/2 Mobile Client wtr05118 that has been installed and configured in Part 2, “Setting Up a Mobile Client Environment” on page 41 to be a fully disconnected client.

Since we are reconfiguring an installed client we have to stop the client first. We type the following on wtr05118:

```
nvdm stop
```

After the agent process has stopped, we need to edit the base configuration file NVDM.CFG to make the following changes:

```
WORKSTATION NAME:                wtr05118
FULLY DISCONNECTED:              Y
SERVER:                          wtr05150
PROTOCOL:                        TCP WTR05118 729 50
REPOSITORY:                      d:\SOFTDIST\REPOS
WORK AREA:                       d:\SOFTDIST\WORK
BACKUP AREA:                     d:\SOFTDIST\BACKUP
SERVICE AREA:                    d:\SOFTDIST\SERVICE
CONFIGURATION:                   CLIENT
MESSAGE LOG LEVEL:               D
LOG FILE SIZE:                   64000
API TRACE FILE SIZE:             64000
TRACE FILE SIZE:                 64000
MAX USER INTERFACES:             20
MAX ATTEMPTS:                    5
TARGET MODE:                     PUSH
MACHINE TYPE:                    OS/2
TARGET ADDRESS:                  24104106
```

Figure 75. Base Configuration File for Fully Disconnected Client

Note

The client we used would not start if we used the keyword FULLY DISCONNECTED: YES as recommended in the manual. Instead we had to specify FULLY DISCONNECTED:Y.

You should also notice that you have to specify a PROTOCOL entry for the client, although it will not perform any communications. If you omit this field, the agent process will not start.

Please check the version you are running and with your IBM representative to determine if this behavior is applicable to your environment.

If you reconfigure a Mobile Client that has been connected to a server before, such as we are doing in our example, we experienced that you need to erase

the files in the UICFG, DB, QUEUES and WORK directories in order to make the agent work in fully disconnected mode.

Note

We experienced, that the client still tries to contact the CC server if you don't erase the files as mentioned above, because its database entries still tell it that it is connected.

Type the commands as shown below:

```
D:
CD SOFTDIST
CD UICFG
DEL *.*
CD ..
CD DB
DEL *.*
CD ..
CD QUEUES
DEL *.*
CD ..
CD WORK
DEL *.*
```

Note

Even if you want to use the client in fully disconnected mode just after installing the client product for the first time you should issue the above commands. This is necessary since the configuration notebook used during installation does not have an option to configure a fully disconnected client.

Since we have erased the database directory above we need to rebuild the database by typing:

```
FNDDBINI
```

You can now start the fully disconnected client by typing:

```
NVDM START
```

The OS/2 window in which FNDCMPS.EXE is running should look similar to the following:

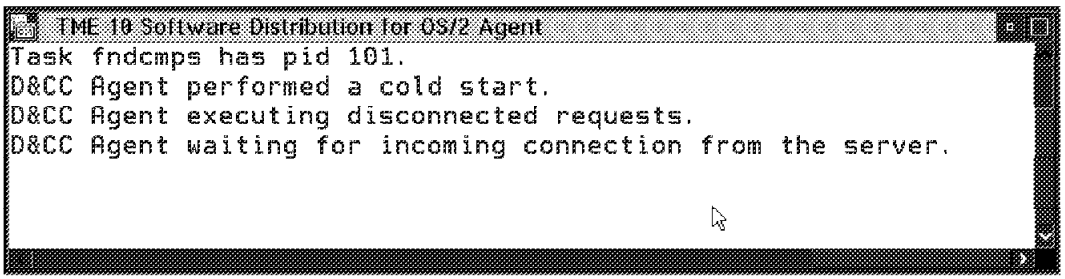


Figure 76. FNDCMPS.EXE for Fully Disconnected Client

Note

You should notice that this window still displays the message: `D&CC Agent waiting for incoming connection from the server.` although the client will never talk to the server again.

To test if your client is really in disconnected mode, you should start the command line interface at the client by typing `nvdm` and then issue the command svr as shown below:

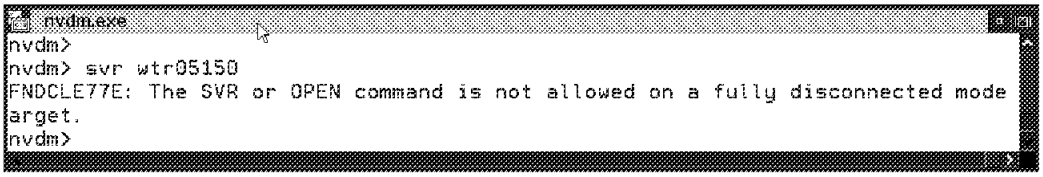


Figure 77. Output of nvdm svr Command

The output confirms that our client is in fully disconnected mode.

8.2 Defining the Fully Disconnected Client at the CC Server

Defining a fully disconnected client at the server is identical with defining a Mobile Client as described in Part 2, “Setting Up a Mobile Client Environment” on page 41 except for one difference:

Although the fully disconnected client is a special kind of Mobile Client its Target type field must be set to CLIENT and not to MOBILE.

The following figure shows the target definition at the server used for our example.

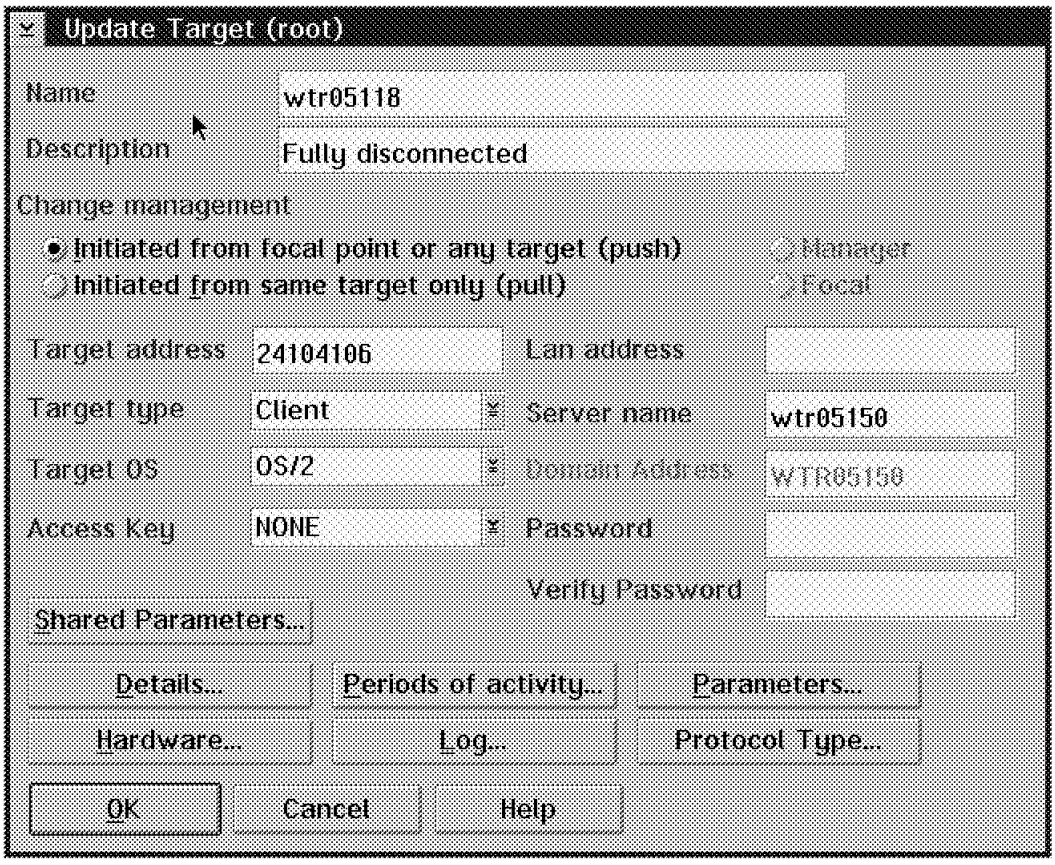


Figure 78. Target Definition of Fully Disconnected Client

8.3 Creating and Exporting a Change File at the CC Server

We need to export a change file at the server and write it to an external media in order to be able to install it at the fully disconnected client.

In this example we do not create a new change file but use the change file OS2.TEST.OBJECT.REF.100.1 that was created in 6.2, “Creating a Change Object” on page 91. In your environment you might want to create a new change file first.

The process for creating a change file that is to be distributed to a fully disconnected client is not different from creating any other change file.

In our example we want to export the change file OS2.TEST.OBJECT.100.1 to a diskette. In order to do so we start the command line interface at the server by typing `nvdm` and the following line:

```
NVDM EXP OS2.TEST.OBJECT.REF.100.1 A:HELLO.CF
```

Note

Remember to insert a diskette into drive A before starting the above command. Also notice that the EXP command is not available from the server GUI but only through using the command line interface.

In order to verify that the change file has been written to the diskette, issue the following command:

```
DIR A:
```

The output should look similar to the following:

```
Volume in drive A has no label.
The Volume Serial Number is 6581:C814.
Directory of A:\

HELLO    CF     11159  10-21-96  10:11a
        1 file(s)      11178 bytes used
                     1446400 bytes free
```

8.4 Importing the Change File on the Fully Disconnected Client

In order to be able to install the change file at the fully disconnected client, we first need to import it into the clients catalog.

In order to do so we insert the diskette that we have prepared in 8.3, “Creating and Exporting a Change File at the CC Server” on page 126 into the diskette drive of the client and start the command line interface by typing `nvdm`.

Warning

In the version we were using we found a problem when we tried to import the change file using the following command:

```
nvdm> imp a:hello.cf -n
```

The command line interface responds with the message:

```
FNDCLE58E: The parameter -N is not valid on a connected mobile client.
```

However, when you type:

```
svr myself
```

the command line interface will respond with the message:

```
FNDCLE77E: The SVR or OPEN command is not allowed on a fully
disconnected mode target.
```

This is the way a fully disconnected client is supposed to respond, since it does not have any possibility to connect a server.

In order to make the client recognize itself as being fully disconnected client we type the following before using the import command:

```
addpm wtr05118 -h test=100
```

This command is only supported on the disconnected client. After it has been issued, the import will work. However, if the command line interface is stopped and started again, you will have to do the above procedure again.

Please check the version you are using and with your IBM representative to determine if the behavior described above is applicable to your environment.

Now we can import the change file:

```
nvdm> imp a:hello.cf -n
```

The parameter -n will only catalog the global file name but leave the change file itself on the diskette. This option is only supported on a fully disconnected client and is useful on clients where local disk space is limited.

In order to verify that the change file has been cataloged, type the following:

```
nvdm> lscm *
```

The client should respond similar to the following:

```
Global file name: OS2.TEST.OBJECT.REF.100.1

    Target:          wtr05118
    Status:          Available
```

8.5 Installing the Change File on the Fully Disconnected Client

In order to install the change file on the fully disconnected client type the following:

```
nvdm> inst OS2.TEST.OBJECT.REF.100.1
```

When you type lsq you will see that the install request has been added to the queue of the local client. However, the install will not take place until you stop and then restart the client again.

In order to do so type the following:

```
nvdm stop
nvdm start
```

When the agent starts the queued request to install the change file is executed. You should observe the OS/2 window in which FNDCMPS.EXE is running in order to see if the change file is successfully installed. You can also look in FNDLOG.

Note

You should also see that the light of the diskette drive is lit when the agent is started. This is because the change file itself resides on the diskette and is installed from there when the install request is processed.

To verify that the change file has been successfully installed type:

```
nvdm lscm *
```

The output should look similar to:

```
Global file name: OS2.TEST.OBJECT.REF.100.1

    Target:          wtr05118
    Status:          Installed, removable, active
```

8.6 Updating the Target History at the CC Server

Since the server has no network connection to the fully disconnected client, we will have to update the target history manually.

This is important because although we have not installed the software using a network we still want to be able to track the target history for all our clients.

In practice, for example, the successful install at the fully disconnected client could be reported by phone from an administrator at the remote site to the administrator of the CC server.

On the CC server, we type the following command to update the target history:

```
nvdm updcm OS2.TEST.OBJECT.REF.100.1 -w wtr05118 -n ira
```

Note

In the above command the parameter -n ira stands for installed, removable, active.

To verify that the status has been changed, type the following at the server:

```
nvdm lscm OS2.TEST.OBJECT.REF.100.1 -w wtr05118
```

The output should look similar to:

```
Global file name: OS2.TEST.OBJECT.REF.100.1

    Target:          wtr05118
    Status:          Not Authorized, Installed, removable, active
```

Note

The way of reporting the change management status by phone does not seem to be feasible for many clients.

In that case you might send out a few hundred diskettes to the client systems and a single administrator at the CC server would not be able to keep track of the history.

Therefore you might want to write the change management status to the diskette on each client and then send these diskettes back to the site where the CC server resides. For example, you could use the output of the nvdm lscm command.

On the server you could have the incoming diskettes processed by a simple script that reads the diskette and updates the status.

Chapter 9. Using the Mobile Client with PPP

One of the primary justifications for the Mobile Client is to be able to perform change management activities involving laptops when they are not LAN attached. In this chapter we look at one approach to using the Mobile Agent over a modem connection.

The Point-to-Point Protocol (PPP) of TCP/IP enables a link between two computers at the IP level. TME 10 Software Distribution can use this link in exactly the same way as it uses a link over token-ring or Ethernet. We look at how to configure PPP between a Software Distribution for OS/2 Mobile Client and a Software Distribution for AIX server. Examples are also given of configuring Windows 95 and Windows NT clients.

A working PPP configuration is a good basis for setting up the native asynchronous support in TME 10 Software Distribution which we describe in Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181.

Before deciding to use PPP, the reader should refer to Chapter 17, “Security Considerations” on page 313 and Chapter 16, “Designing a Network” on page 307 for discussion of the alternatives. We found PPP to be one of the hardest configurations to set up of the ones that we tried.

9.1 Overview of PPP

The Point-to-Point Protocol (PPP) is a peer-based protocol allowing the connection of two TCP/IP hosts across standard asynchronous modems. The connection is made at the inter-networking layer in TCP/IP terms. See Figure 79 on page 132 for a diagram of the TCP/IP architecture layers as they affect us.

© Copyright IBM Corp. 1997

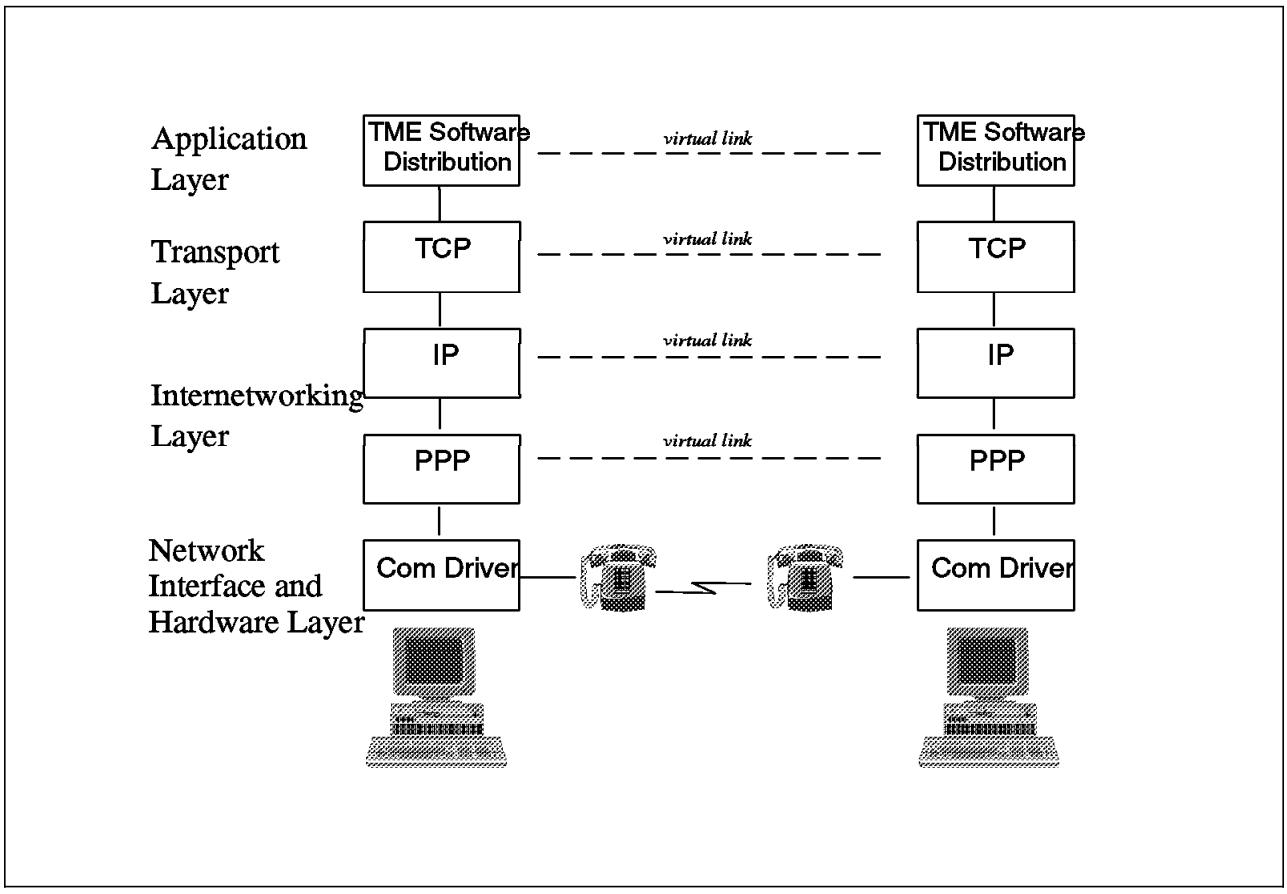


Figure 79. Basic PPP Architecture

Each layer in TCP/IP, although it has a virtual link to the equivalent layer on other hosts, in reality only has a link to the layer below it in the protocol stack. This is why it is possible to swap out the token-ring component and replace it with a modem and PPP without affecting the application layer in any way other than performance.

PPP is strictly peer based. There is no architectural difference between the host that initiates the connection and the host that accepts the connection. However, for simplicity we will regard the machine that dials in as being a PPP client and the machine that accepts the call as being a PPP server. This fits well with our use of PPP since in general the Mobile Client will initiate the session and will therefore also be the PPP client.

PPP is a formal TCP/IP protocol as opposed to Serial Line IP (SLIP) which is only a de facto standard. PPP offers several benefits over the very simplistic SLIP:

- Inclusion of a link control layer allowing automatic configuration of IP addresses.
- Compression of IP and TCP headers (Van Jacobsen Header Compression).
- Support for protocols other than IP datagrams. (This is also available with some SLIP implementations.)
- Frame error detection.

PPP works through a terminal logon. The host acting as a PPP server is configured to allow standard user logon through a modem connection. The PPP server must also have support for PPP installed. The PPP client logs in through

a modem link using a standard user ID and password as understood by the PPP server operating system. In our first example this is an AIX user account. Then the PPP process is started from the command line of the terminal window and control is passed at both ends to the PPP tasks. The initiation of PPP at the server may be done directly from the command line or it may be configured in a login profile. In most implementations the logon to the server is scripted by a process on the client, the script having been set up to match the server operating system logon dialog.

For a fuller discussion of PPP and TCP/IP in general, refer to the redbook *TCP/IP Tutorial and Technical Overview*, GG24-3376.

9.2 Setting Up a Link from OS/2 to AIX

In this section we describe how to configure a PPP link between OS/2 and AIX. We first set up the modem under AIX and then check that we can log in from our OS/2 Client. Once this is working we start the PPP subsystem on AIX and create a user with a login profile to automatically start PPP. Finally we create an attachment script for PPP from OS/2.

Note

The instructions that follow are the steps that worked for us in our environment. They are provided here as a guideline. For detailed instructions refer to the appropriate manuals for your configuration.

Software Levels

Not all versions of OS/2 support PPP. We found that Warp Version 4 (Merlin), Warp Connect and Warp Server worked fine but Warp Version 3 did not, even though it had IBM TCP/IP for OS/2 Version 3.0 at the same CSD level as Warp Connect. To see if it will work open an OS/2 window and type:

```
ppp
```

If you receive the message:

```
Sorry - PPP is not supported by this stack revision
```

then PPP will not work for you. You can try using SLIP, or upgrade your operating system.

9.2.1 Enabling Terminal Login

As described above in 9.1, “Overview of PPP” on page 131, PPP works by logging in through a terminal session before handing control over to the PPP processes. The first step, therefore, is to ensure that we can log on through a modem-connected terminal session. There are two steps to this:

1. Setting Up the TTY Port on AIX.
2. Running a terminal program.

We deal with these in order.

9.2.1.1 Setting Up the TTY Port on AIX

You will need to be logged on to your AIX system with sufficient authority to perform some of the following actions.

1. First connect a modem to a communications port on the RS/6000 and connect a telephone line to the modem. In our case we used serial port 1 and an IBM 7855 modem. (We also had another modem connected to serial port 2 but this is not used until the next chapter.) Verify that the modem is set up to receive AT commands and is configured for 8 data bits, no parity, and 1 stop bit. We recommend that you begin with a speed of 9600 baud and increase this later if required.

2. From a command prompt start SMIT and go to the TTY section by typing:

   ```
   smit tty
   ```

   If you are not using X-Windows then run the character based version of SMIT by entering:

   ```
   smitty tty
   ```

3. You should now see a window similar to Figure 80. Note that if you are using the character interface this will look somewhat different.

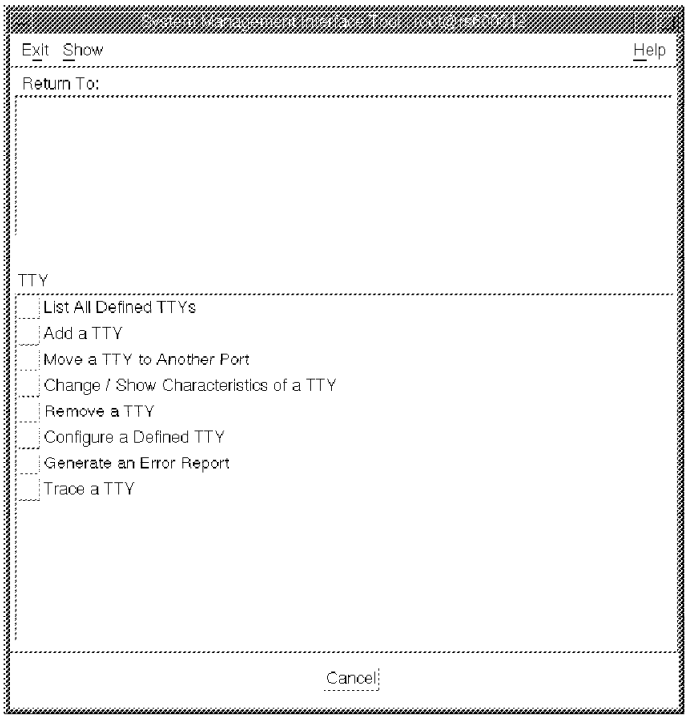


Figure 80. SMIT TTY Panel

4. The first step is to define the serial port to the system. If the serial port is already defined then you can skip this part. To check if the port is defined select **List All Defined TTYs**. If the port is already defined you should see a message similar to Figure 81 on page 135. Remember that we have 2 ports defined. You only need one for this chapter.

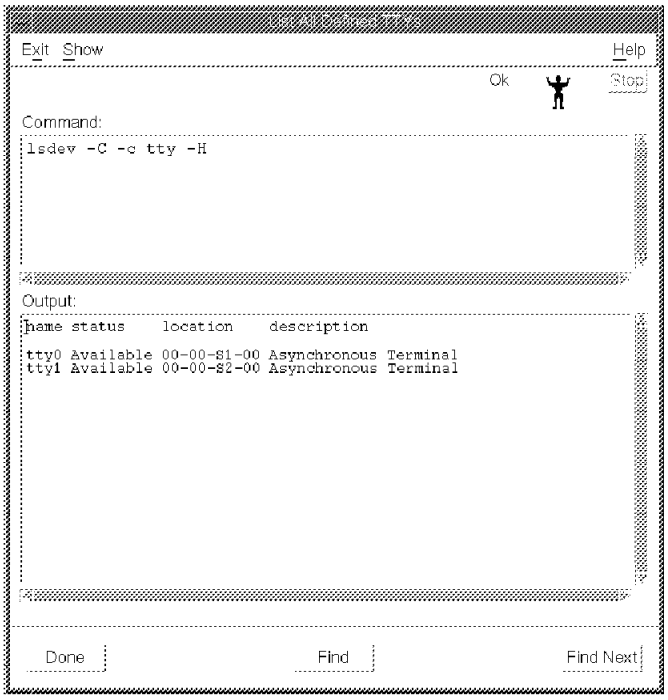


Figure 81. SMIT Defined TTYs Panel

Select **Done** to return from this panel and re-display the SMIT TTY panel (Figure 80 on page 134). If your port is already defined, then proceed to point 5.

To add a new port definition choose **Add a TTY** and select an RS232 asynchronous terminal. Choose the physical port that your modem is connected to; in our case, it was serial port 1. You should now see the Add a TTY window as shown in Figure 82.

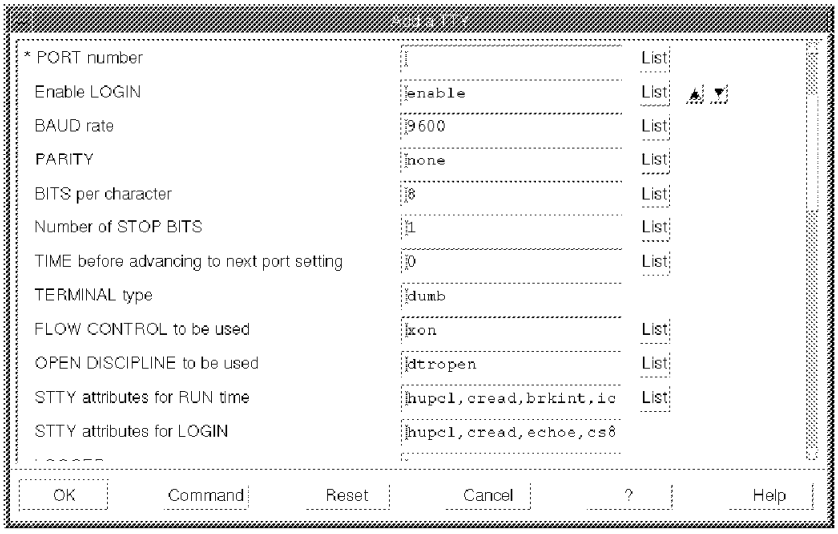


Figure 82. SMIT Add a TTY Dialog

The only value that you need to enter at this stage is the port number which in our case we set to s1. Detailed configuration is covered in the next step. Click on **OK** to create your port and you should receive a successful

message. For any problems at this stage you should refer to your AIX documentation. Select **Done** to return from this panel.

5. Having confirmed that we have a serial port defined, we now need to configure it. From the SMIT TTY panel (see Figure 80 on page 134) select **Change / Show Characteristics of a TTY** and then choose your port (ours was tty0) to bring up the dialog shown in Figure 83. The values that you enter here are largely dictated by the modem that you are using. In our case we set the flow control to be RTS and removed IXON and IXOFF from the STTY login attributes. Ensure also that logins are enabled on this port and select **OK** to change the settings.

   Note

   When we tested the scenarios in this book, we found problems with PPP. In order to get the terminal login to work we had to modify the STTY logmodes. We found that if we entered:

   ```
   chdev -l 'tty0' -a logmodes=''
   ```

   and then tried a connection, the login failed. If we then entered:

   ```
   chdev -l 'tty0' -a logmodes='hupcl,cread,echoe'
   ```

   then the login would work once. These steps had to be repeated each time a connection was required.

   We do not believe this to be a general problem with PPP, but instead we think it had something to do with our specific environment.

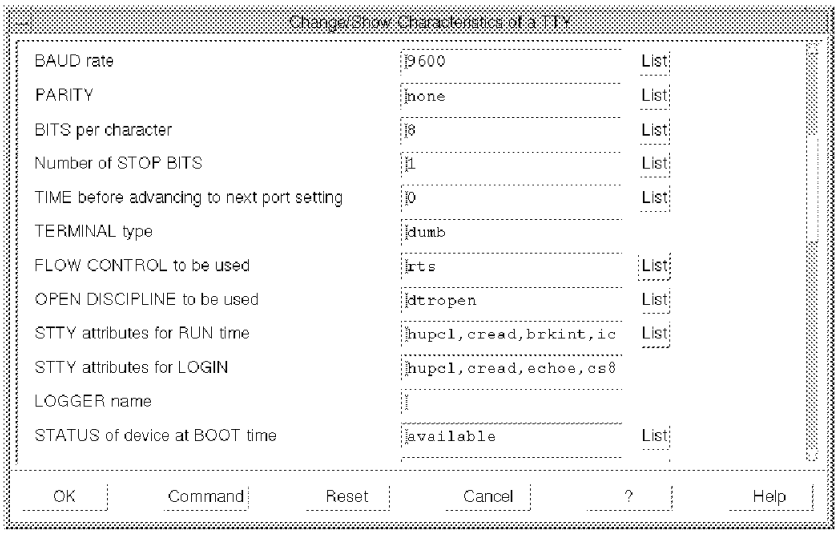


Figure 83. SMIT Change/Show Characteristics Dialog

6. You should now see a confirmation window similar to Figure 84 on page 137.

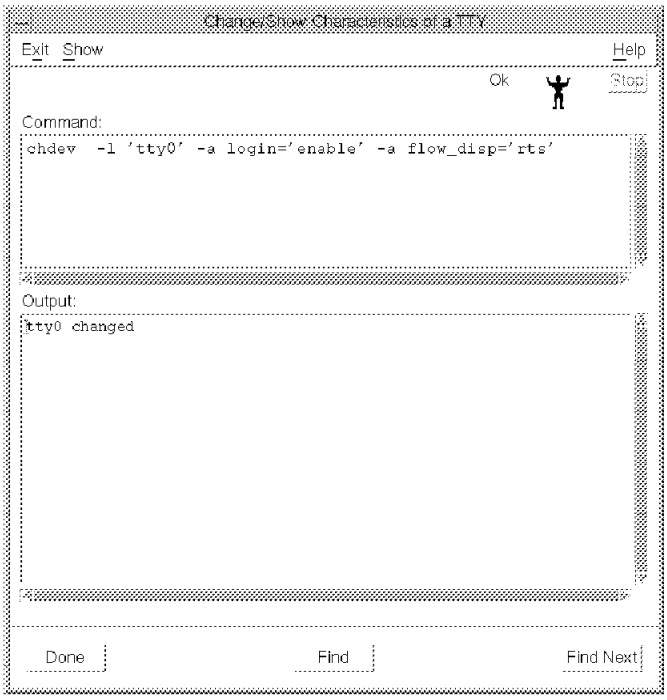


Figure 84. SMIT Change/Show Characteristics Output Window

Note

The command generated by SMIT shows only the changes that were made. This depends on the values that were already set and hence may be different in each configuration. So we are not providing a command line option.

Checking Your Configuration

There are several things to look for to confirm that AIX is able to talk to your modem and allow logins. First check the lights on your modem if you have any. The getty process which you have just started by setting login to enabled, controls logins over serial ports and periodically talks to the modem. You should see that the data ready light on the modem is lit and occasional flickers of the transmit data and read data lights should occur as the getty task communicates with the modem. If you have problems later then check if getty is running:

```
ps -ef | grep getty
```

should return something like:

```
root 14478     1  0   Oct 27   lft0  0:00 /usr/sbin/getty /dev/console
root 22248     1  0 08:43:18      0  0:00 /usr/sbin/getty /dev/tty0
root 30448 23396  1 08:43:34  pts/4  0:00 grep getty
```

To test the connection to the modem, remove getty by disabling logins using the Change / Show Characteristics of a TTY dialog as above and then try entering:

```
echo "AT">/dev/tty0
```

This should cause the lights on the modem to flicker and should return immediately to the command prompt. If this hangs, then you do not have a connection from /dev/tty0 to the modem.

9.2.1.2 Running a Terminal Program

Note

When we originally tried this using a terminal program for OS/2 we did not see the prompts echoed on the screen. You should be able to do the terminal connection using any communication program, however if you run into problems it might be worth trying different programs to see the results.

To confirm that the terminal logins are correctly set up, use a PC communications program such as HyperTerminal which comes with Windows 95 to connect to the AIX system. We ran Windows 95 on the the same laptop as OS/2. This was an IBM ThinkPad 755cx with an IBM PCMCIA modem as used in other parts of this book. This proved that the hardware was functional and meant that once this part was successful the only configuration necessary was of PPP within OS/2.

Start HyperTerminal and provide it with the telephone number only. Leave all other settings at the default. HyperTerminal should now connect to the AIX server and display the screen shown in Figure 85.

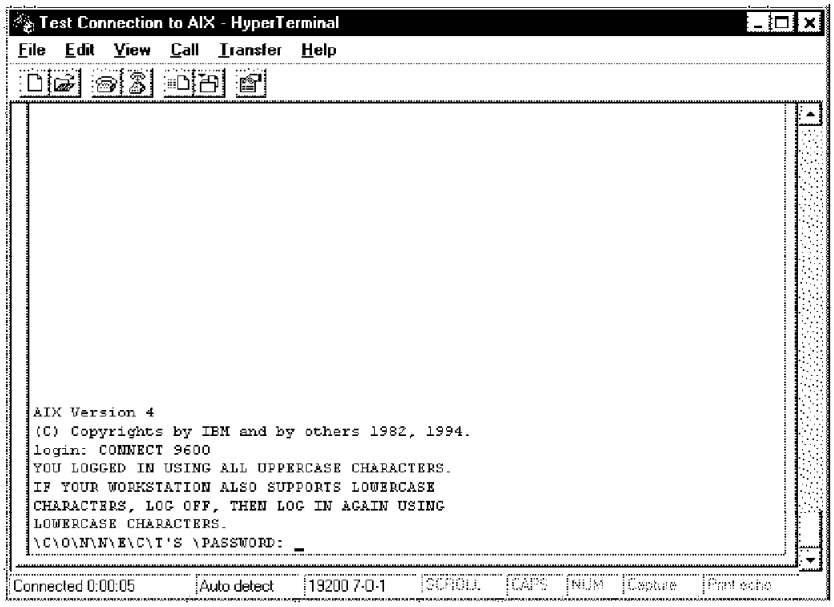


Figure 85. HyperTerminal Connected to AIX

Note

The connect message CONNECT 9600 has been incorrectly sent to the login prompt as a user ID. This also happens with PPP from OS/2 and the attachment script had to be written to cope with this.

This confirms that the TTY logins are properly enabled and that the modems are communicating. It also provides input to developing the login script for automatic connection with PPP.

9.2.2 Getting PPP to Work

Now that we have proved that the logins work okay we need to go to the next step and use PPP. This involves three steps:

1. Start PPP on AIX.
2. Create a user profile on AIX.
3. Create an attachment script on OS/2.

We address these in order:

9.2.2.1 Starting PPP on AIX

To enable PPP on an AIX log on with sufficient privileges and follow these steps:

1. Start SMIT and go to the PPP section by entering:

   ```
   smit ppp
   ```

   This will display the panel shown in Figure 86.

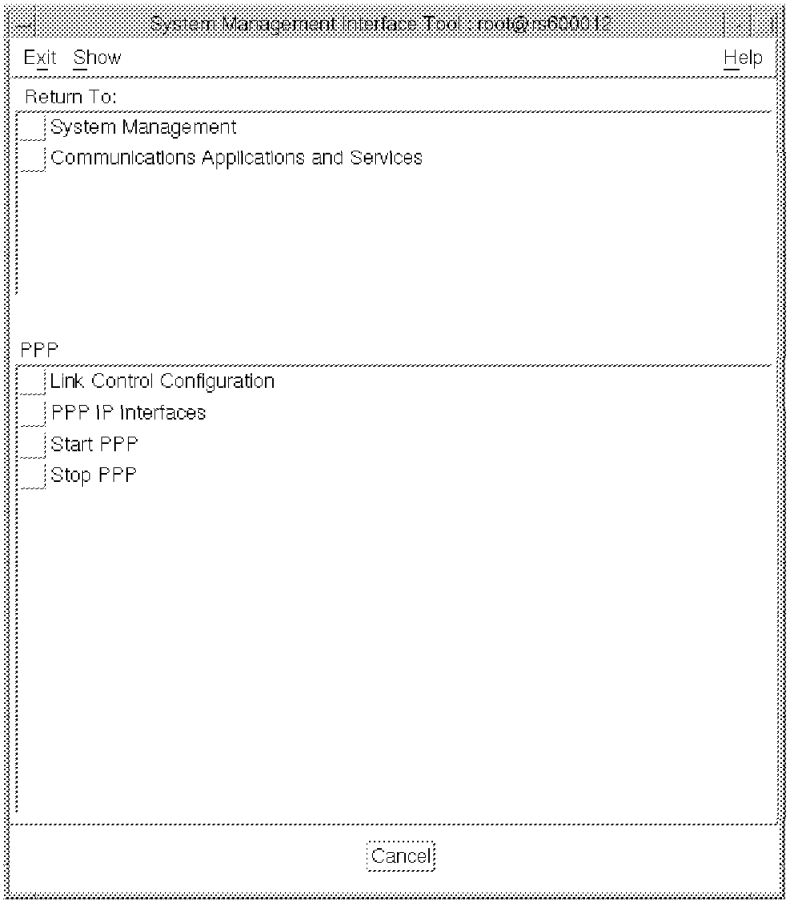


Figure 86. SMIT PPP Panel

2. Select **Link Control Configuration** and then **Add a Link Configuration** to display the panel shown in Figure 87 on page 140.

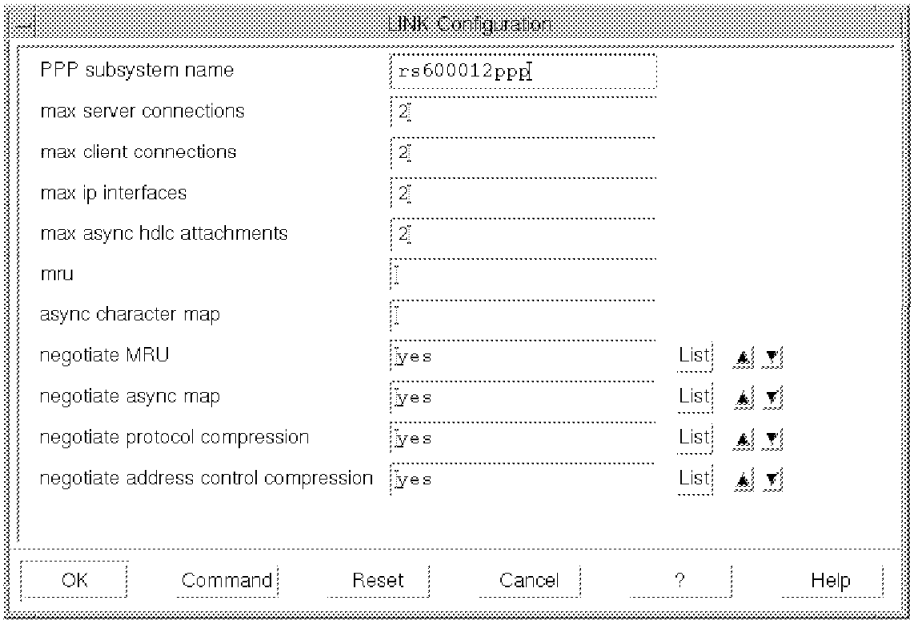


Figure 87. SMIT Link Configuration Panel

Enter a name for your system and values for the number of connections. In our test system we chose low values. Select **OK** to create the link definition. Then choose **Done** to acknowledge the confirmation window.

3. Now return to the PPP menu and choose **PPP Interfaces** and then **Add a Server Interface**. This will bring up the window shown in Figure 88.

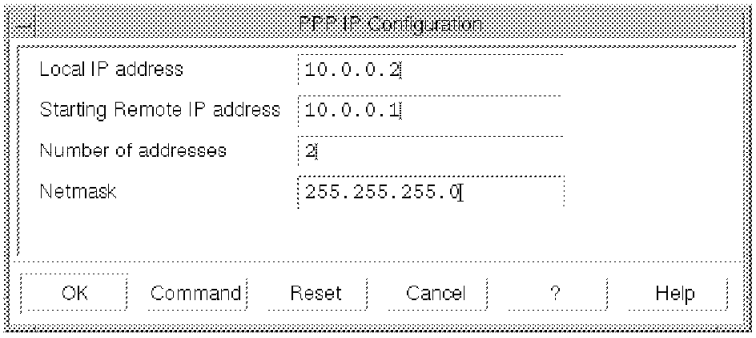


Figure 88. SMIT PPP IP Configuration Panel

Enter an IP address for the AIX machine to use in the local IP address field and an address for the OS/2 host in the Starting Remote IP address field. Specify a subnet mask and a number of addresses to use. This can be one if only one PPP client is connected at a time. Click on **OK** to create the IP definition.

Caution

If your AIX system is not a gateway then IP forwarding should be turned off to avoid any conflicts between the PPP subnet and the LAN subnet. If you want the PPP client to be able to see all of the hosts beyond the PPP server, then this will have to be enabled. This is not required by TME 10 Software Distribution. If you do use IP forwarding, then ensure that you use a valid range of IP addresses for your establishment. If in doubt, see your network administrator.

4. Again return to the PPP panel and this time select **Start PPP**. Choose Both to start PPP now and on system restart and click **OK**. You should see a confirmation similar to Figure 89.

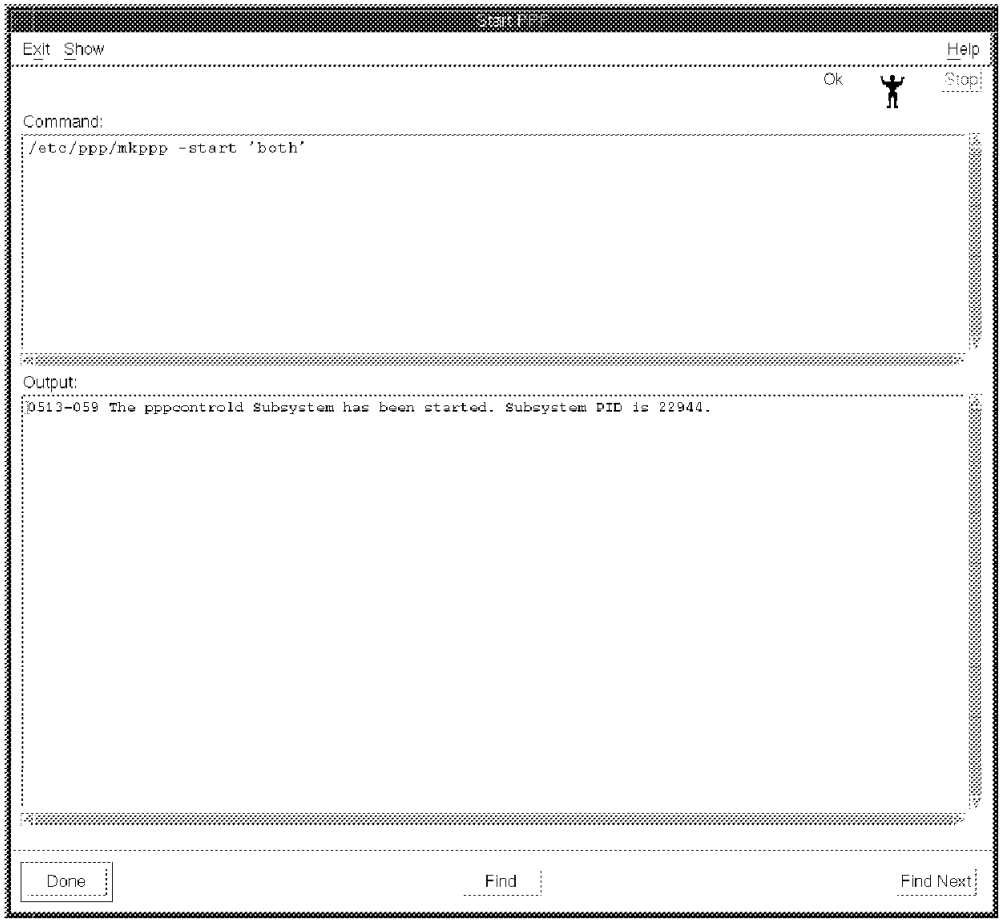


Figure 89. SMIT PPP Start Confirmation

You have now successfully started PPP.

9.2.2.2 Creating a User Profile for PPP

We suggest you create a new user and modify the .profile file to include a command to start PPP. We created a user called pppuser with a password of pppuser. The profile for pppuser is shown in Figure 90 on page 142. The important line is:

```
/usr/sbin/pppattachd server 2>/dev/null
```

```
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.

export PATH

if [ -s "$MAIL" ]           # This is at Shell startup.  In normal
then echo "$MAILMSG"        # operation, the Shell checks
fi                          # periodically.

/usr/sbin/pppattachd server 2>/dev/null
```

Figure 90. /home/pppuser/.profile

If you are using a shell other than the Korn shell, then modify the appropriate configuration file.

9.2.2.3 Creating an Attachment Script on OS/2

You now need to connect the OS/2 Mobile Client to the Software Distribution for AIX server over PPP. To do this we use IBM Dial-Up for TCP/IP which can be found in the TCP/IP folder on the OS/2 desktop or can be started from the command line by typing:

```
SLIPPM
```

Once started this looks like Figure 91.

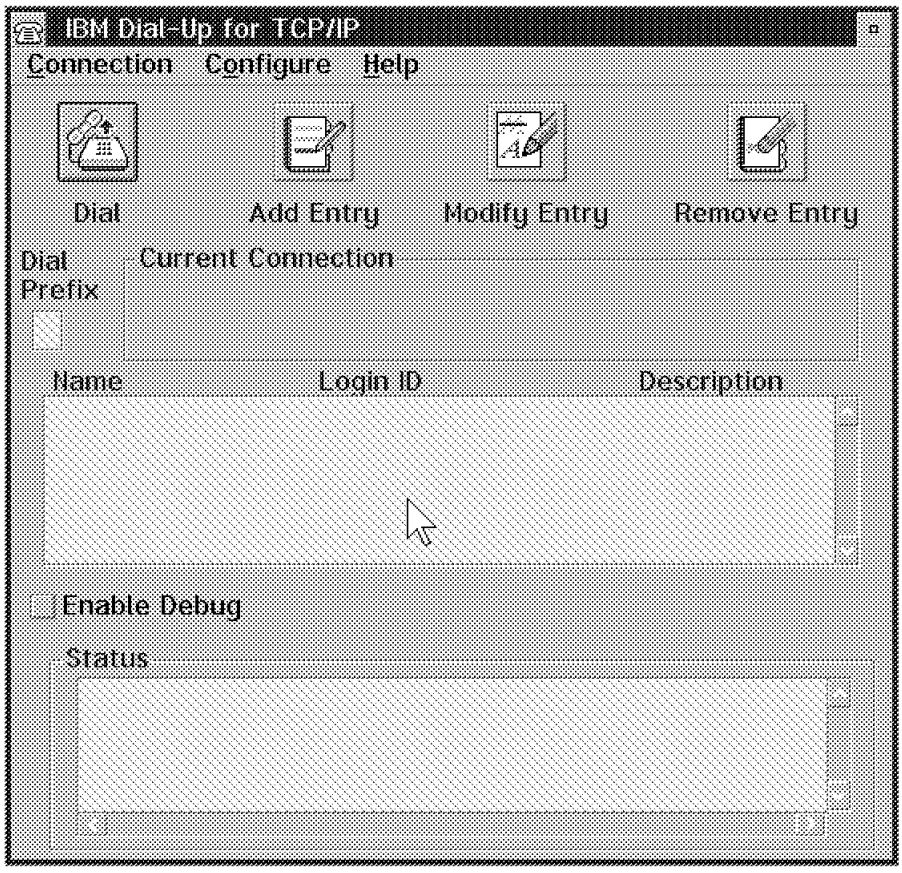


Figure 91. IBM Dial-Up for TCP/IP

Select the **Add Entry** push button to create a new entry. This will display page one of the Add Entry panel as seen in Figure 92 on page 143.

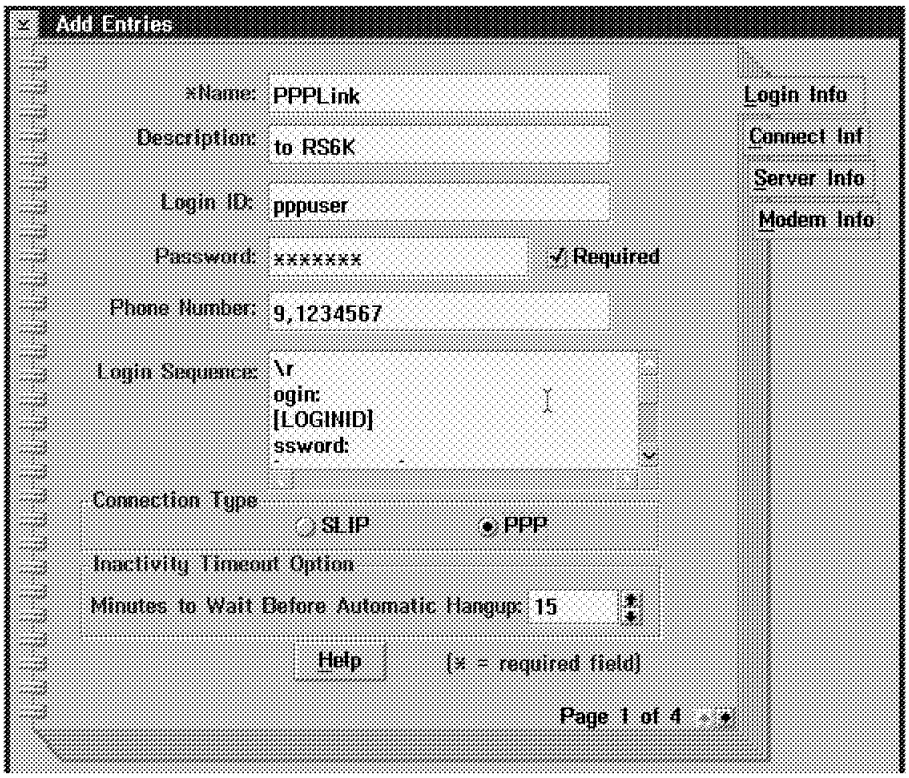


Figure 92. Add Entries - Login Information

Enter a name and a description for your connection. Enter a login ID and password that match the values used when you created the AIX account. Enter the phone number and select the **PPP** radio button. For the Login Sequence you must enter an attachment script that will connect to your AIX system. This will vary from system to system. For our server we used the following:

```
\r
ogin:
[LOGINID]
ssword:
[PASSWORD]
```

This script is run once a connection is established. Table 3 describes each of these commands.

Table 3. Sample Attachment Script Commands

Command	**Action**
\r	Sends enter to bring up login dialog.
ogin:	Waits for the characters "ogin:". This is part of the login prompt.
[LOGINID]	Sends the login ID.
ssword:	Waits for the password prompt.
[PASSWORD]	Sends the password.

Having completed page one of the notebook, click the **Connect Inf** tab to move to the second page. Enter the values that match your environment. Our values are shown in Figure 93 on page 144. Note that you must enter a domain name server even if you will not be using one. In our example we will edit the hosts files on the machine rather than using DNS.

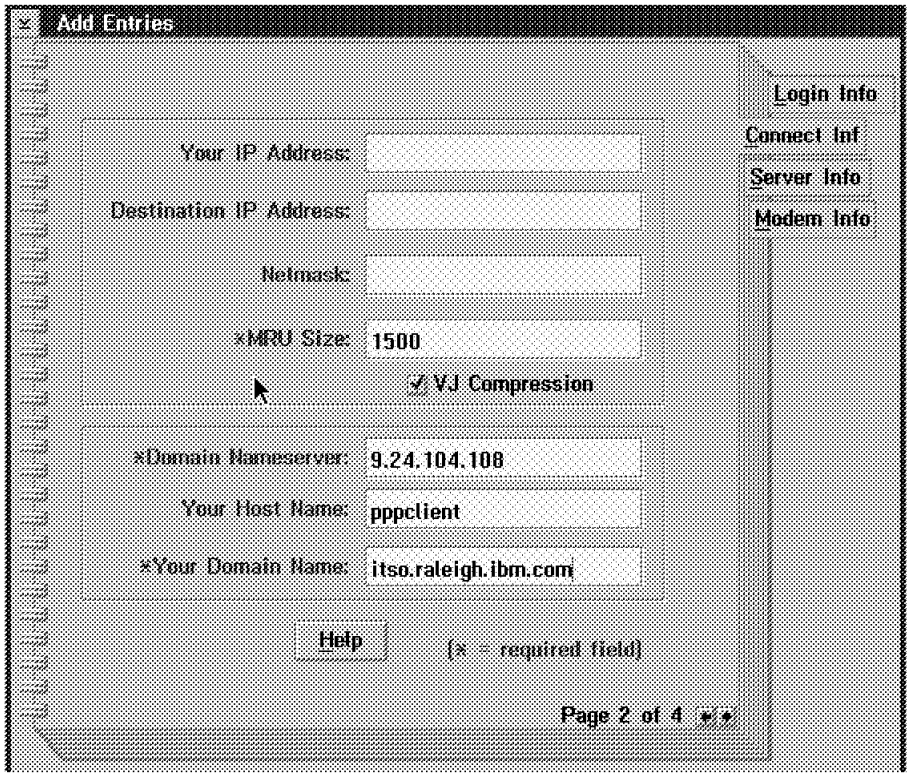


Figure 93. Add Entries - Connection Information

You can skip the third page of the notebook and move to page four to define the modem information. Select your modem and COM port and set the mode to dial. An example is shown in Figure 94 on page 145.

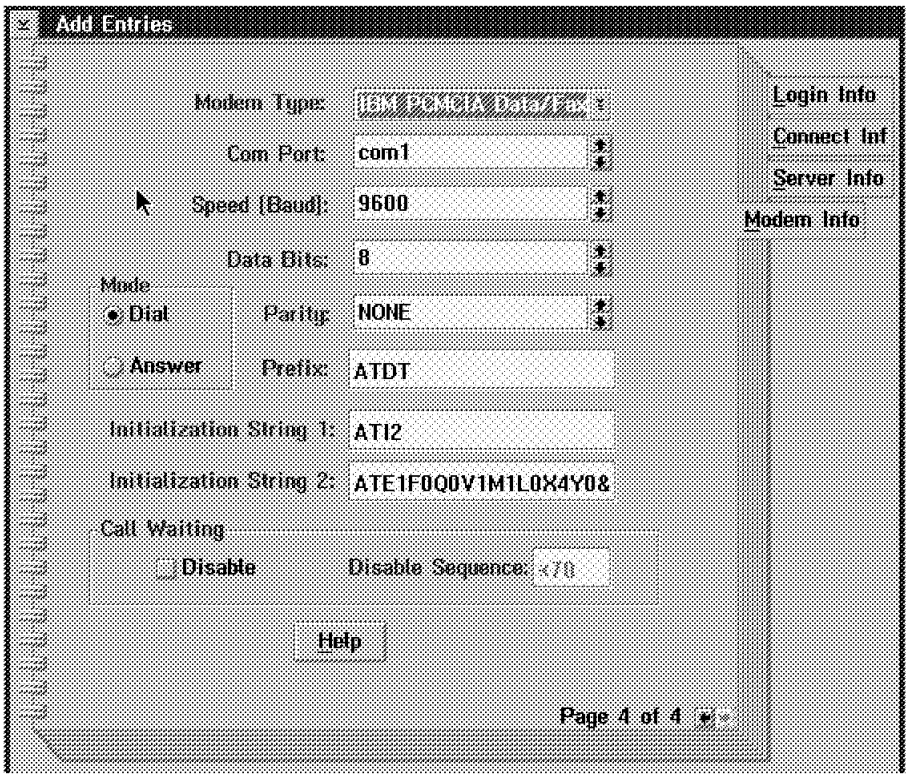


Figure 94. Add Entries - Modem Information

Now close this window answering save when asked if you wish to save your changes.

You can now select **Dial** to connect to the server. If successful, you should see a connection made as shown in Figure 95.

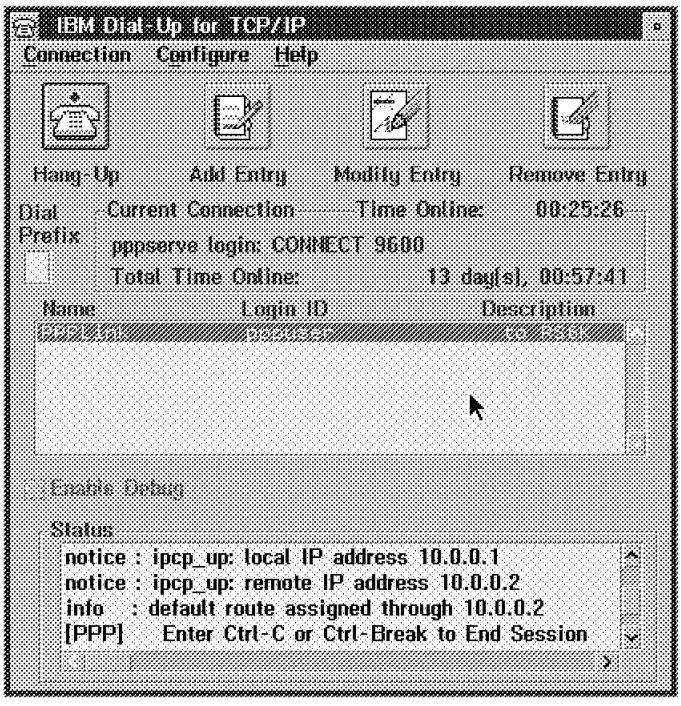


Figure 95. Successful Connection

The last piece of configuration necessary is to hard-code the host names in the hosts files on each machine. This step is optional; you can choose to use the dotted decimal IP addresses rather than host names, or you can go the entire way and define the PPP network to your domain name server and use DNS. If you choose to do this remember to define the AIX system as a gateway.

Edit the hosts file on the OS/2 PC to add the PPP link to the AIX system. This is found in the directory pointed to by the ETC environment variable, usually this is C:\MPTN\ETC. Our file is shown in Figure 96.

```
10.0.0.2        pppserver
```

Figure 96. C:\MPTN\ETC\HOSTS on OS/2

Also edit the hosts file on the AIX system to provide a link back to the OS/2 PC. This file can be found in /etc. An example is shown in Figure 97.

```
# @(#)47 1.1  src/bos/usr/sbin/netstart/hosts, cmdnet, bos411,
#
# COMPONENT_NAME: TCPIP hosts
#
# FUNCTIONS: loopback
#
# ORIGINS: 26  27
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1989
# 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/hosts
#
127.0.0.1       loopback localhost  # loopback (lo0) name/address
9.24.104.124    rs600012
10.0.0.1        pppclient
```

Figure 97. /etc/hosts on AIX

9.3 Using PPP from the Mobile Client

Once you have established the connection, this can be used by TME 10 Software Distribution in the same way as a LAN connection. You must provide a configuration file that contains the correct names for the server and client. Figure 98 on page 147 shows the configuration that we used to connect from our OS/2 Mobile Client to the AIX server.

```
WORKSTATION NAME:              pppmobile
SERVER:                        rs600012 TCP pppserver
PROTOCOL:                      TCP pppclient 729 50
REPOSITORY:                    C:\SOFTDIST\repos
SERVICE AREA:                  C:\SOFTDIST\service
BACKUP AREA:                   C:\SOFTDIST\backup
WORK AREA:                     C:\SOFTDIST\work
CONFIGURATION:                 CLIENT
MESSAGE LOG LEVEL:             N
LOG FILE SIZE:                 500000
API TRACE FILE SIZE:           500000
TRACE FILE SIZE:               1000000
MAX USER INTERFACES:           20
MAX ATTEMPTS:                  5
TARGET MODE:                   PUSH
MACHINE TYPE:                  OS/2
TARGET ADDRESS:                pppmobile
```

Figure 98. Sample NVDM.CFG for PPP

Notice that WORKSTATION NAME and TARGET ADDRESS can be anything unique. SERVER must be the actual server name, the protocol and the TCP/IP host name used from the client to reach the server. PROTOCOL must contain the TCP/IP name used by the server to find the client.

So from the client we can type:

```
PING pppserver
```

From the server we can type:

```
PING pppclient
```

It is worth providing help to users who will have to start PPP before using any of the services of the Mobile Client that require access to the server. In Chapter 18, “Example Front-End Application” on page 325 we show the XYZ Hotel Application which is an example of a front-end menu for remote users of the Mobile Client who require some help with enabling the transport layer and starting TME 10 Software Distribution on their computers. This example can be used for PPP or other connections as described in this book.

9.4 Configuring Other Operating Systems

Now we look at configuring Windows 95 and Windows NT for use with PPP. We assume in this section that you have already configured the AIX server as described above.

9.4.1 Windows 95 Client

In this section we will describe how to establish a PPP connection from a Windows 95 client.

9.4.1.1 Prerequisites

Before starting, ensure that you have installed the Dial-Up Networking option and the Dial-Up Scripting Tool; also that you have a working modem connected to your PC and defined to Windows 95. If these are in place then you can skip to 9.4.1.2, “Creating the Object” on page 150.

To install Dial-Up Networking open the **My Computer** folder from the desktop and double-click on **Control Panel**. Select **Add/Remove Programs** and click the **Windows Setup** tab. This will display a window similar to Figure 99.

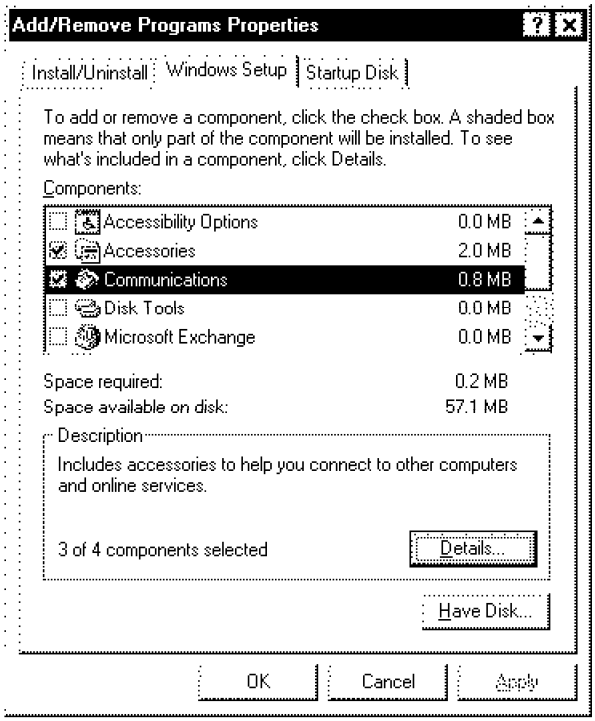


Figure 99. Windows 95 Add/Remove Programs Window

Select **Communications** and click on the **Details...** push button. You should see a window like Figure 100 on page 149. Ensure that Dial-Up Networking is selected and choose **OK**.

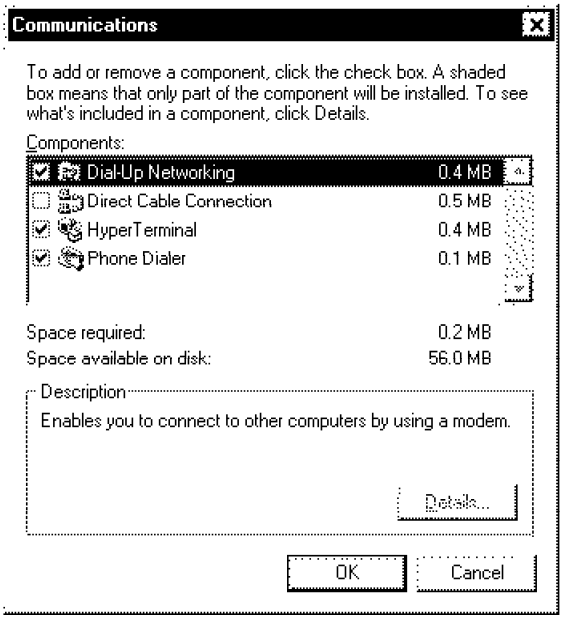


Figure 100. Windows 95 Add/Remove Programs Window

You have now returned to the Add/Remove Programs window. Click on **OK** to install the component.

The Dial-Up Scripting Tool is also installed from the Add/Remove Programs utility. Select **Install...** from the first page of the Add/Remove Programs window to bring up the install program dialog (see Figure 101).

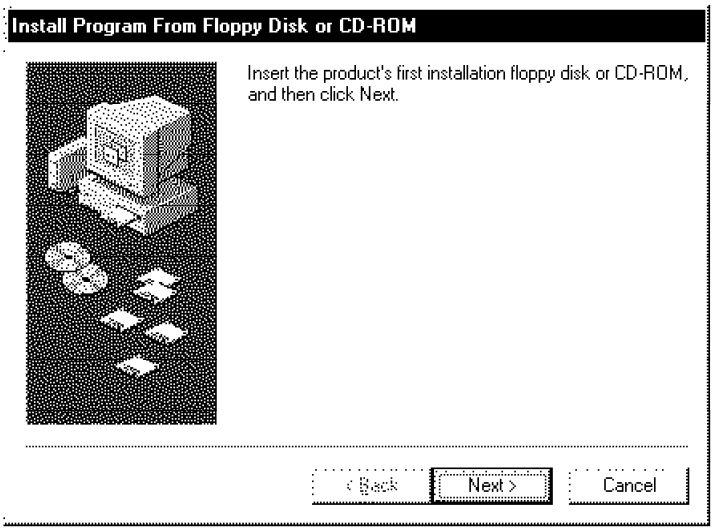


Figure 101. Windows 95 Install Program Dialog

Select **Next>** and enter:

```
D:\ADMIN\APPTOOLS\DSCRIPT\RNAPLUS.INF
```

in the command line field. If you are not installing from CD-ROM or your CD-ROM is not drive D, then adapt this as appropriate.

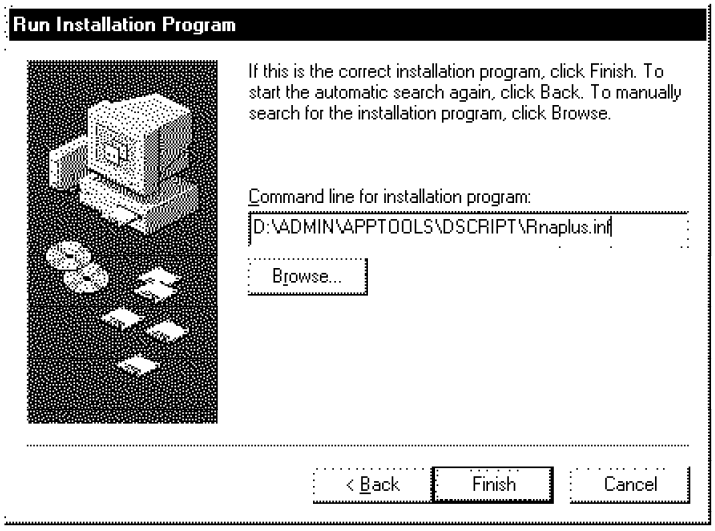


Figure 102. Windows 95 Run Installation Program Screen

Select **Finish** to install this option.

9.4.1.2 Creating the Object

From the Start bar, select **Programs**, then **Accessories** and **Dial-Up Networking**. This will display a container as shown in Figure 103.

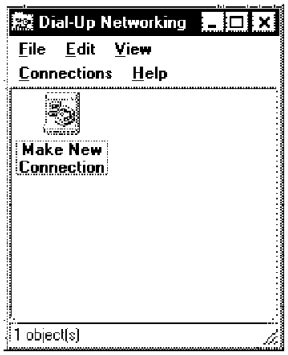


Figure 103. Dial-Up Networking Container

Double-click on **Make New Connection** to create a link to your AIX system. You will see a window like Figure 104 on page 151.

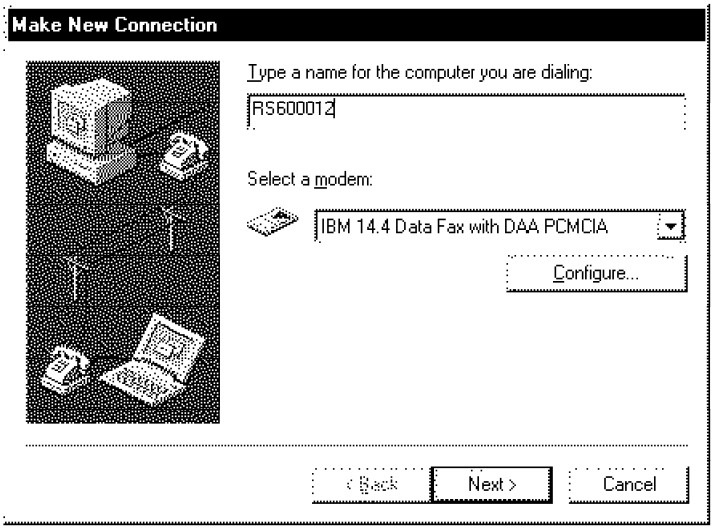


Figure 104. Make New Connection Window

Enter a name and select your modem. Move to the next screen by clicking **Next>** and key in the phone number for the AIX system. Press Enter twice more to complete the dialog.

If you run just this dialog then the default dial-up script is invoked and you will not be able to log in to the AIX system. You now need to create a script using the Dial-Up Scripting Tool. From the Start bar, select **Programs**, then **Accessories** and **Dial-Up Scripting Tool**. This will display a window as shown in Figure 105. The purpose of this dialog is to associate a script with a connection.

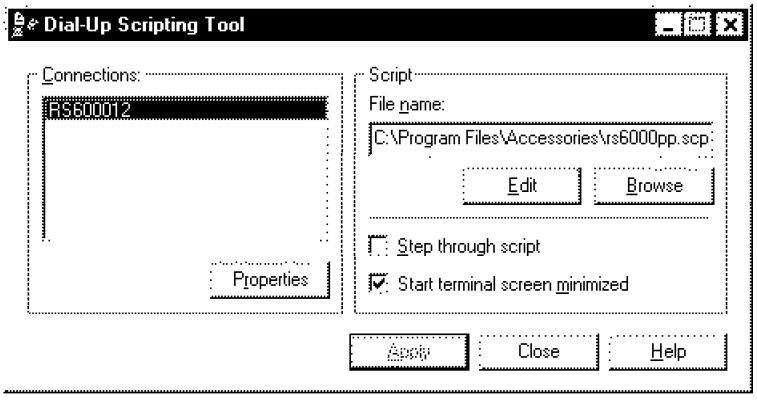


Figure 105. Dial-Up Scripting Tool

Enter a file name in the field and choose **Edit** to create a script. An example script is shown in Figure 106 on page 152. Once your script has been built select **Apply** to create the association.

```
;*********************************************
;* Sample Script to logon to AIX and start     *
;* PPP from a Windows 95 Client                *
;*                                             *
;* Mark Guthrie - 30th October 1996            *
;*********************************************

proc main

;*********************************************
;* Pause for 3 seconds to allow AIX to settle  *
;* Then send enter                             *
;*********************************************
delay 3
transmit "^M"

;*********************************************
;* Wait for the login prompt, then send userid *
;* and enter. Userid is substituted for us.    *
;*********************************************
waitfor "ogin:"
transmit $USERID
transmit "^M"

;*********************************************
;* Wait for the password prompt, then send     *
;* password and enter.                         *
;* Password is also substituted for us.        *
;*********************************************
waitfor "assword:"
transmit $PASSWORD
transmit "^M"

;*********************************************
;* That's it!                                  *
;* PPP is started by logon profile on AIX.     *
;*********************************************
endproc
```

Figure 106. Sample Script File

You can now double-click your connection in the Dial-Up Networking container to bring up the connect dialog as you see in Figure 107 on page 153. Key in the user ID and password and click **Connect** to be connected to your server.

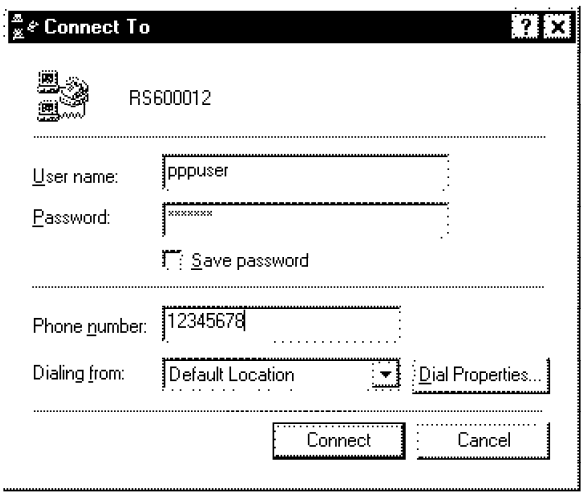


Figure 107. Connect to Window

As described above for OS/2, you can use the PPP connection in the same way as a LAN connection once it has been established. The required NVDM.CFG file is very similar to the one shown for OS/2 (Figure 98 on page 147) with the MACHINE TYPE changed to WIN95. Remember again to edit the hosts file to add a connection to the Software Distribution for AIX server over PPP.

One problem from an administrative and support point of view with PPP on all of the Windows platforms, is that there is no published API or command line interface to PPP. If you wish to use something like the XYZ Hotel Application as described in Chapter 18, “Example Front-End Application” on page 325, then you will either need to try to develop something in a tool such as Visual Basic to start PPP, or investigate using third party PPP products for Windows that have command line interfaces. Alternatively you can leave it up to the user to navigate the screens and establish the connection manually.

9.4.2 Windows NT Client

Software Levels

We performed the following configuration on a Windows NT 3.51 Server. We also tested this on Windows NT Workstation 4.0. This also worked but there were slight differences, which are described below. A similar setup should be possible on other versions of NT but this was not tested.

In this section we describe how to create a PPP connection from Windows NT. We describe installation of the NT Remote Access Service (RAS), and then cover how to configure the link.

9.4.2.1 Installing Remote Access Service (RAS)

To add PPP support to Windows NT open the **Main** folder from Program Manager as shown in Figure 108 on page 154.

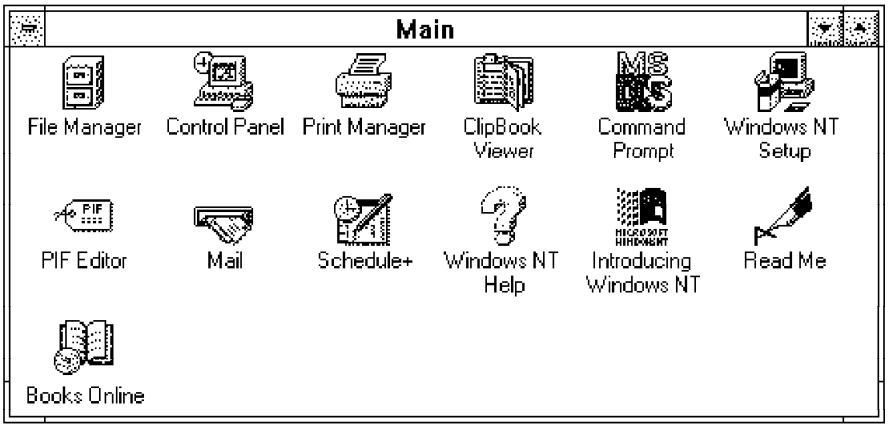


Figure 108. Windows NT Main Folder

Double-click on **Control Panel** to bring up the window shown in Figure 109.

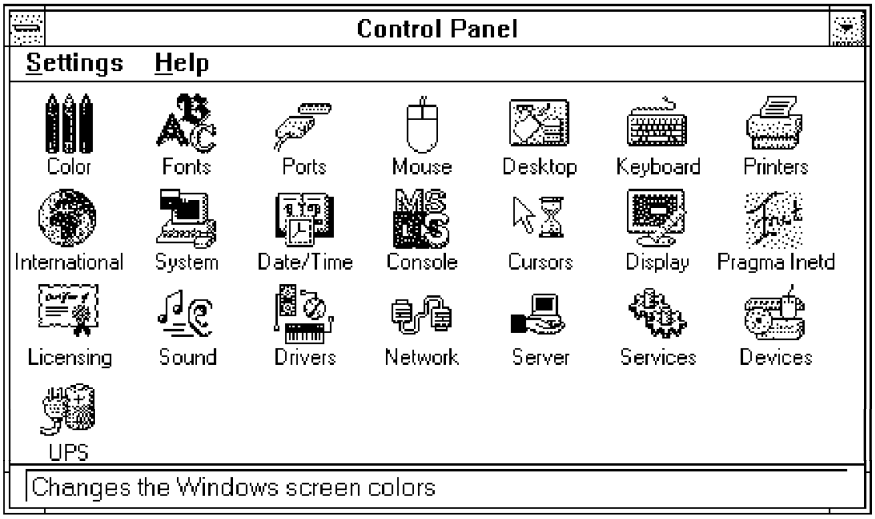


Figure 109. Windows NT Control Panel

Now select **Network** to open the network settings control panel as you see in Figure 110 on page 155.

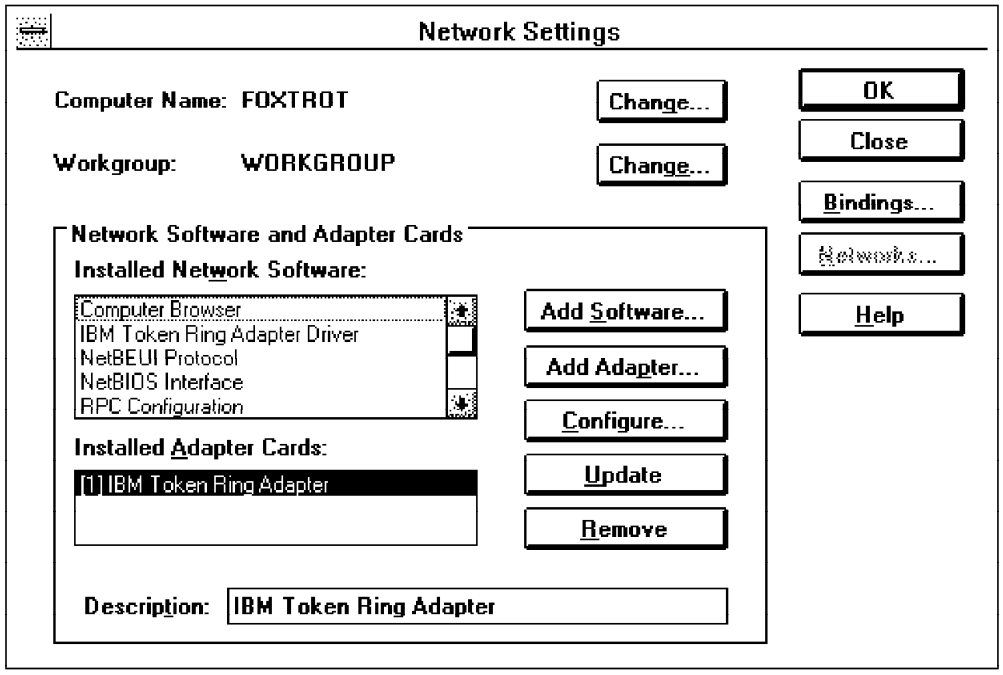


Figure 110. Network Settings

Click on the **Add Software...** push button and select **Remote Access Service** from the selection list as shown in Figure 111.

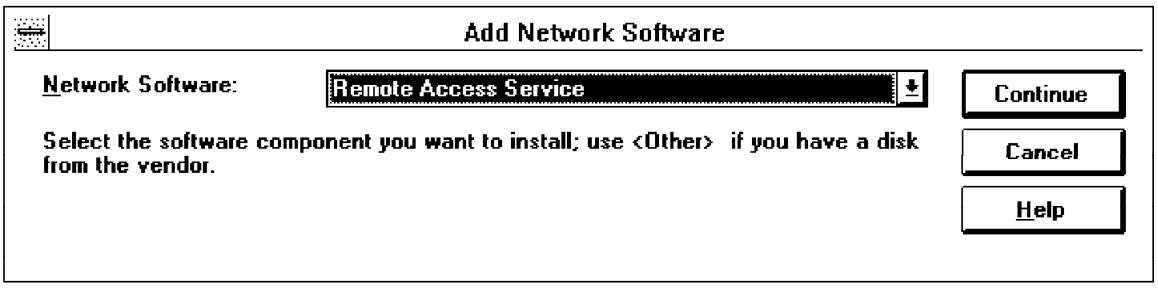


Figure 111. Add Network Software

Select **Continue** and the source directory for your Windows NT distribution files. Once the software is installed you are asked which COM port to use. In our case the modem was connected to COM1 (see Figure 112).

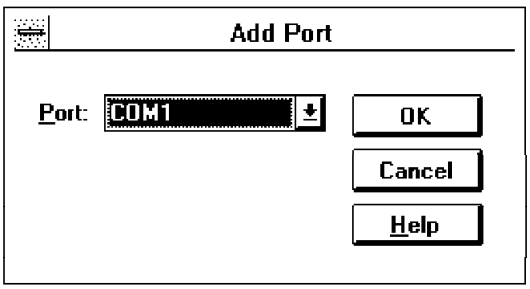


Figure 112. Add Port Window

Remote Access Setup will now try to auto-detect your modem. Ensure that it is switched on and select **OK** when the dialog shown in Figure 113 on page 156 is displayed. If you wish you may decline and manually configure the modem.

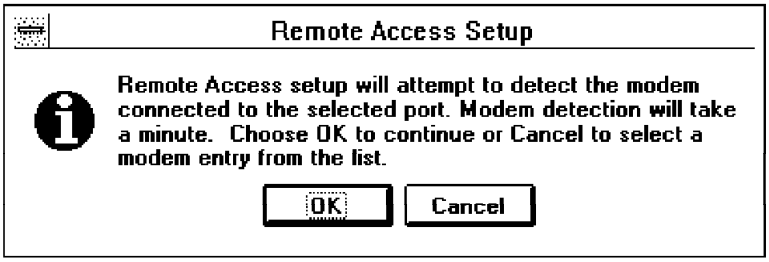


Figure 113. Modem Auto-Detection

In our case the auto-detection failed to identify our IBM 7855 modem. We selected a Hayes Compatible 9600 modem from the list as shown in Figure 114.

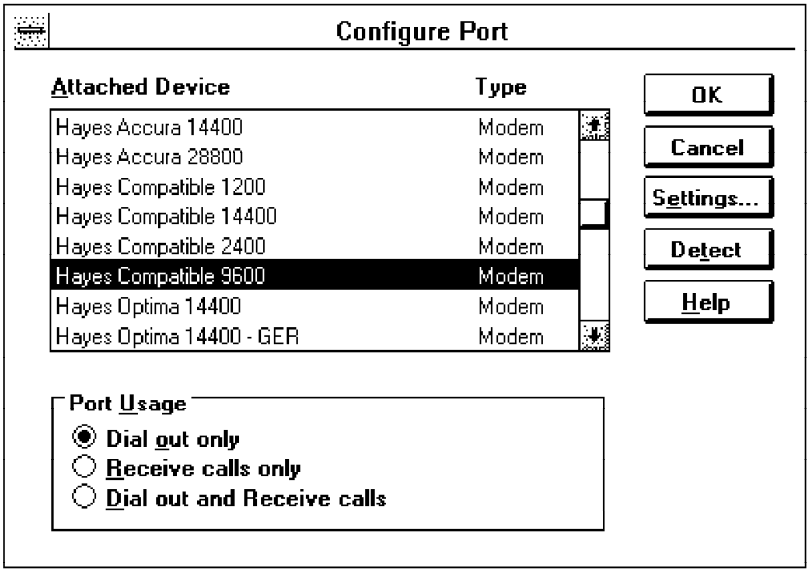


Figure 114. Configure Port

Select the correct port usage and choose **OK**. Select **Continue** from the next screen (Figure 115 on page 157) and Remote Access will finish installing.

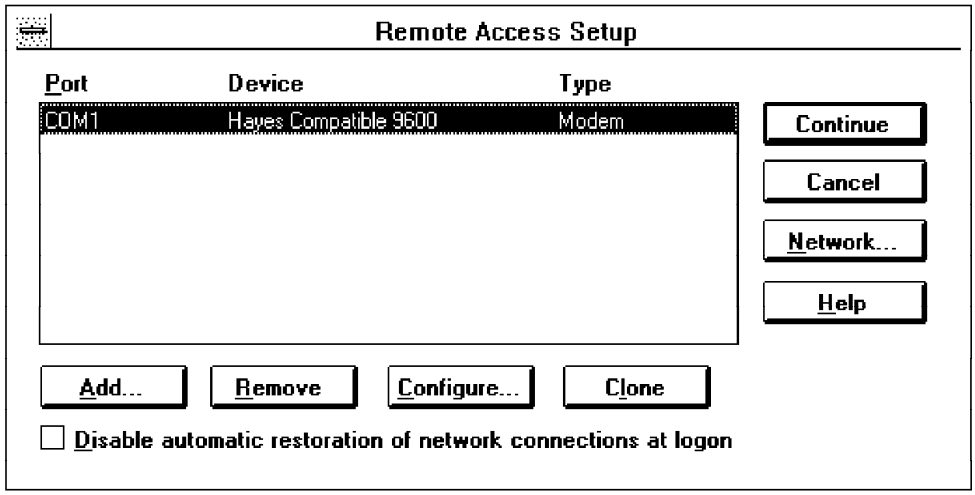


Figure 115. Remote Access Setup

You should now be at the Network Settings window (Figure 110 on page 155). Select **Close** to exit and you are asked to reboot the system as seen in Figure 116.

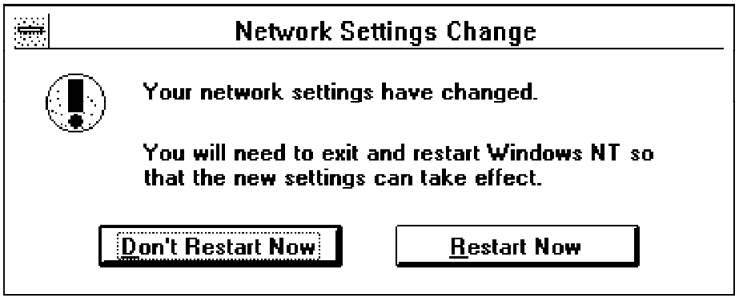


Figure 116. Network Settings Change

Select **Restart Now** to restart your system.

9.4.2.2 Configuring PPP

When your system has restarted you will find a new folder in Program Manager called Remote Access Service. Inside this folder you will find a number of icons as shown in Figure 117 on page 158.

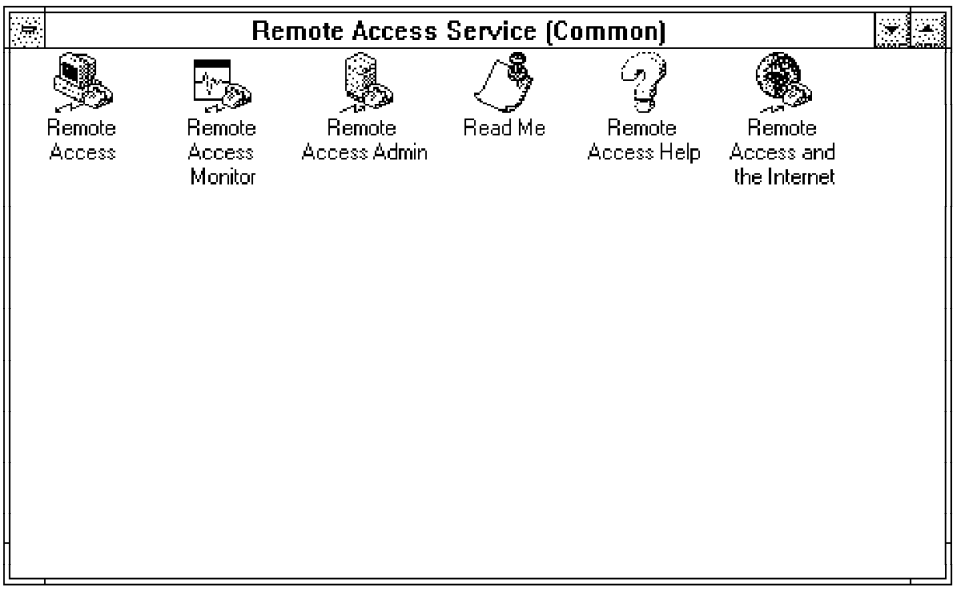


Figure 117. Remote Access Service Folder

The Remote Access Services of NT are particularly well documented so you should find adequate help online if you require it.

Before you go on to define the link, it may be worth modifying the login script to cope with the problem described above where the CONNECT 9600 was echoed to the login prompt.

This problem can be solved by sending an extra Enter before trying to do the logon. If you do not suffer from this problem in your environment then obviously you do not need to perform these steps. The reason for doing this here is that the file containing the login scripts is only read once, at program start.

Using a text editor such as EDIT modify the file C:\winnt35\system32\ras\switch.inf adding the line:

```
COMMAND=<cr>
```

immediately after the line:

```
COMMAND=<cr>
```

Alternatively you can back up this file and create a new copy with the contents as shown in Figure 118 on page 159.

```
[AIX Logon]
  COMMAND=
  COMMAND=<cr>

  OK=<match>"ogin:"
  LOOP=<ignore>

  COMMAND=<username><cr>

  OK=<match>"assword:"
  LOOP=<ignore>

  COMMAND=<password><cr>

  OK=<ignore>
```

Figure 118. C:\winnt35\system32\ras\switch.inf

Now return to the Remote Access Service Folder and double-click on the **Remote Access** icon to show the Remote Access window as shown in Figure 119.

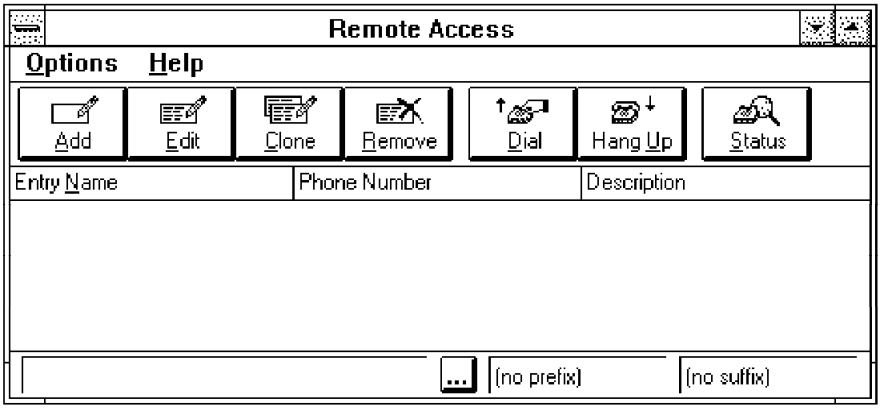


Figure 119. Remote Access Window

The first time in, you will see a warning that the phone book is empty. Click on **OK** to create a new entry. Select **Advanced>>** to display the window shown in Figure 120 on page 160.

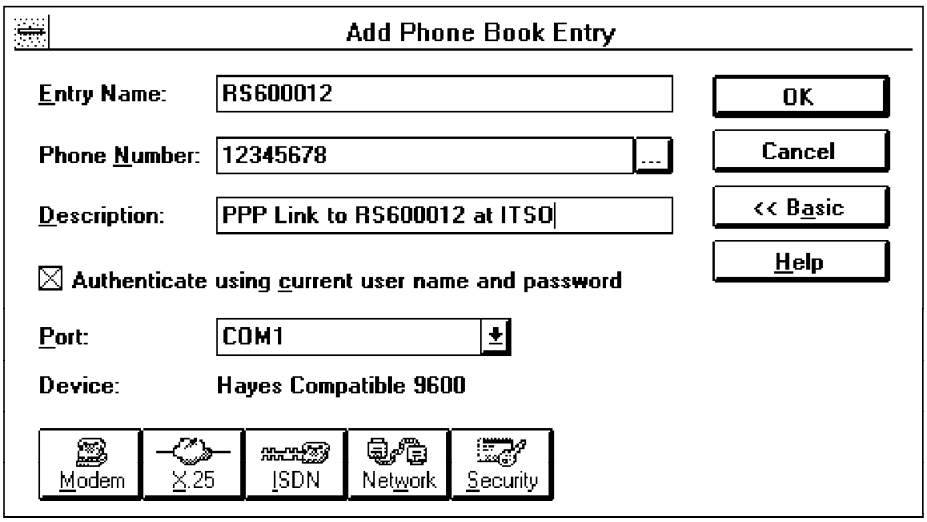


Figure 120. Add Phone Book Entry

Enter a name, description and phone number here. Remove the tick from the Authentication check box and click on **Network** to bring up the dialog shown in Figure 121.

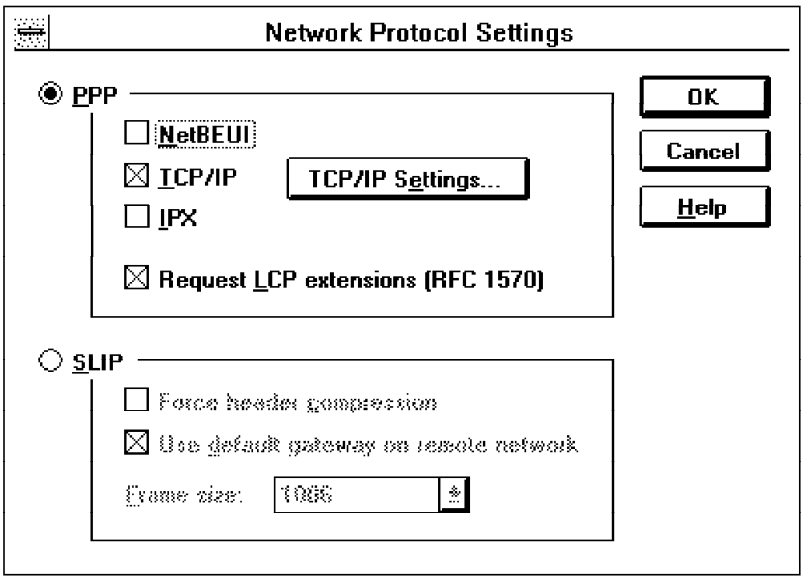


Figure 121. Network Protocol Settings

Ensure that PPP is selected and that TCP/IP is the only protocol. You can leave the TCP/IP settings at the defaults. Click on **OK** to return to the Add Phone Book Entry window (Figure 120) and select the **Security** push button. This will pop up the Security Settings screen as shown in Figure 122 on page 161.

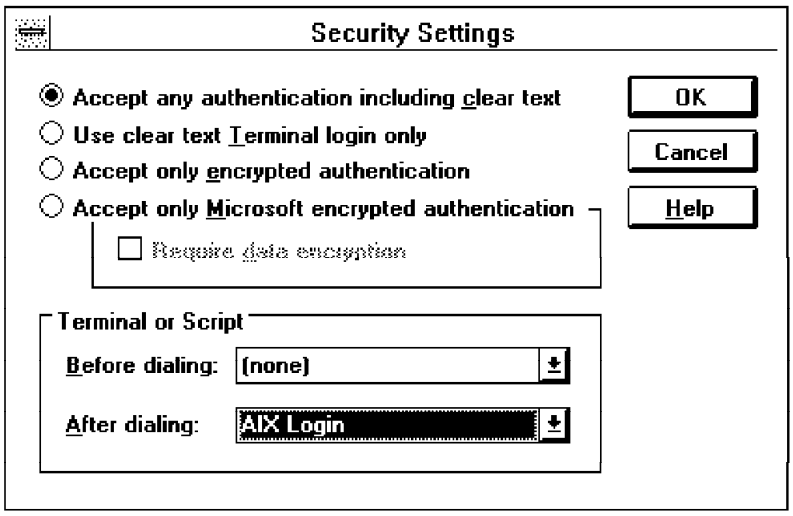


Figure 122. Security Settings Window

From the After dialing drop down list choose Generic Login or if you chose to create the file shown in Figure 118 on page 159, then choose AIX Logon. Now click on **OK** to leave the Security Settings and then click on **OK** again to leave the Edit Phone Book Entry window.

You can now click on **Dial** to connect to the server. You are prompted for a user ID and password. Since we are not using NetBIOS you can leave the Domain field empty. Once the connection has been established and your user ID and password have been confirmed using the login script, you should see a confirmation window as shown in Figure 123.

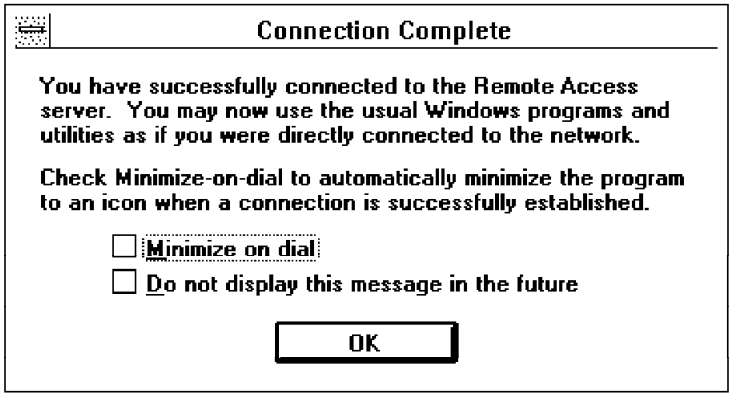


Figure 123. Connection Complete

Windows NT Workstation 4.0

Setup is similar for NT Workstation 4.0. The scripting language has changed somewhat and now more closely resembles the Windows 95 language. An example script is shown here and should be placed in the ras directory.

```
; ************************************
;
; Sample Windows NT 4.0 Script.
;
; Mark Guthrie - November 1996
;
; ************************************
proc main
; ******************************
; As usual send enter twice
; ******************************
   transmit "^M"
   transmit "^M"

; ******************************
; Wait for prompt and send userid
; ******************************
   waitfor "ogin:"
   transmit $USERID, raw
   transmit "^M"

; ******************************
; send password and we're done
; ******************************
   waitfor "assword:"
   transmit $PASSWORD, raw
   transmit "^M"

endproc
```

To use this you must associate it with the connection using the Edit Phonebook Entry notebook within RAS.

9.5 Further Comments

In our examples we have created a link directly to the server using PPP. This means that we do not need to do any IP forwarding from the server and can restrict access to the resources of the server only.

An alternative is to have a separate PPP server, possibly with a bank of modems attached. This could be configured for IP forwarding, with access to the TME 10 Software Distribution server being achieved beyond the PPP link. There are security implications with this which should be considered. Security is discussed in Chapter 17, “Security Considerations” on page 313.

You may have noticed that in our examples we use only one client at a time over PPP. This ensures that we always get the same IP address when we connect to the server. It is essential that the Software Distribution for AIX server can resolve the hostname to a single target and that this target is always the same machine. If we had a pool of IP addresses for use with PPP, then it would be likely that each time we connected we would be given a different IP address.

This would confuse Software Distribution for AIX and we could not perform software distribution.

Within PPP it is possible for the client to request an IP address during the connection. In theory this should allow you to allocate IP addresses from the PPP pool as if they were static addresses, and to use the same address for a machine each time. The version of PPP that we used on the server did not support this feature (sometimes called client network protocol).

Chapter 14, “Using the Mobile Client with Dynamic IP” on page 275 discusses the related topic of dynamic IP which may be of interest to the reader.

Chapter 10. Using the Mobile Client with IBM 8235 LAN Dialer

In this chapter we demonstrate how to use the Mobile Client in combination with the IBM 8235 Remote Access to LANs Server.

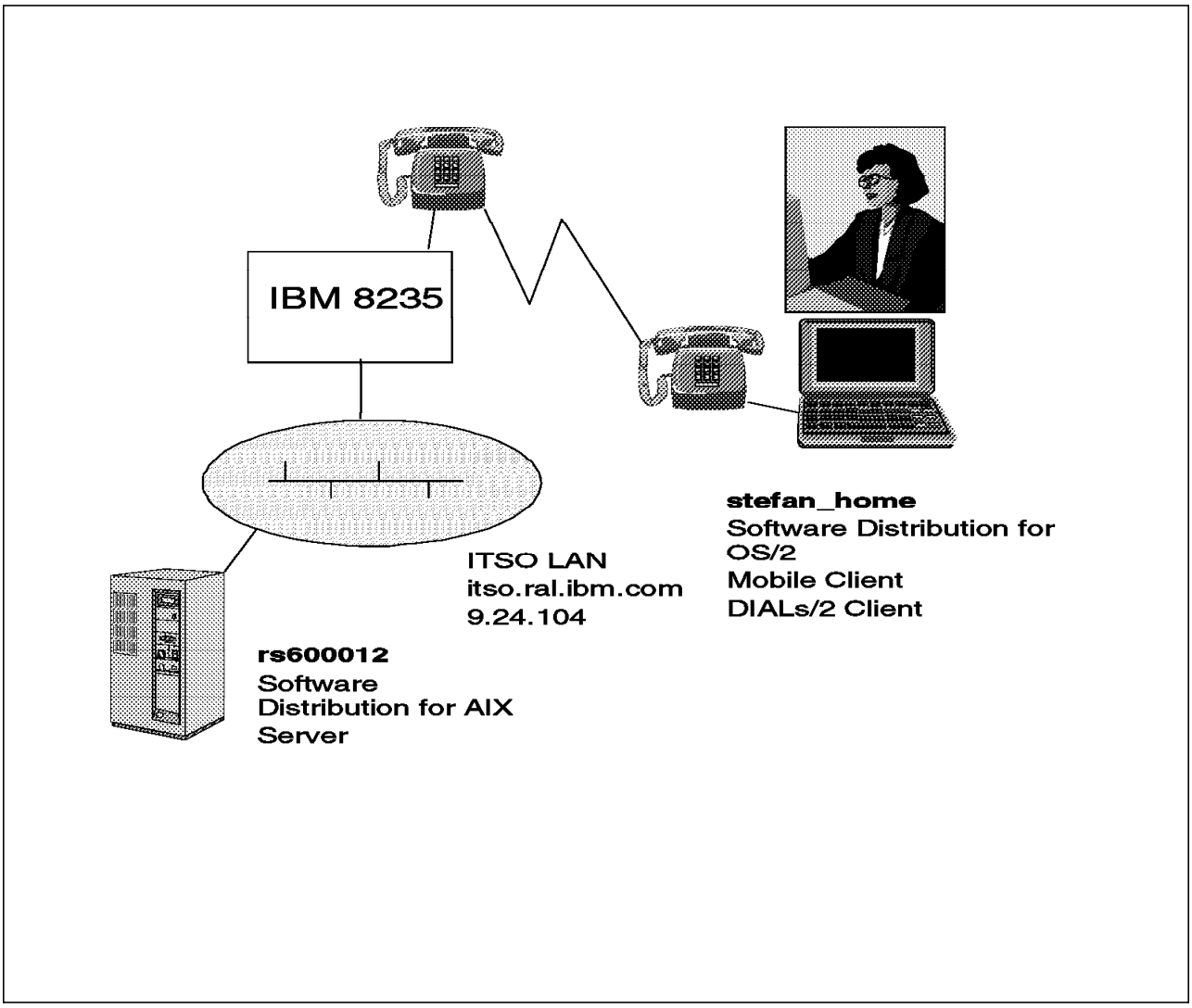


Figure 124. Using the Mobile Client with IBM 8235 Scenario

We show how to install the DIALs Client for OS/2 and then show an example of using TME 10 Software Distribution over a modem link with DIALs.

We, however, do not explain how to set up the IBM 8235 system itself but assume that this has already been configured. In fact, we use the IBM 8235 unit that is used at ITSO for remote access to its LANs.

If you need guidance on how to configure the IBM 8235, you should refer to the redbook *IBM 8235 Dial-In Access to LANs Server Concepts and Implementation*, SG24-4816.

It is important to notice that in this scenario there is no direct connection using a modem link between the Mobile Client and the TME 10 Software Distribution

© Copyright IBM Corp. 1997

server. Instead the IBM 8235 provides transparent access to the LAN where the server resides.

Thus for the client it looks like it is connected to a normal LAN with the exception that the LAN is not permanently available.

10.1 Prerequisites

Besides the prerequisites for the Mobile Client mentioned in Part 2, “Setting Up a Mobile Client Environment” on page 41, we need the following additional prerequisites in order to use DIALs:

- An IBM 8235 at the site we want to connect to that is readily configured for remote LAN Access.
- The DIALs client software. Refer to the IBM 8235 documentation on which version to use.

Further, we assume that the client system we use is readily configured to be used in a LAN environment and that it has the corresponding software, for example, MPTS installed.

10.2 Installing the DIALs Client for OS/2

Insert the diskette with the DIALs Client for OS/2 and type:

```
A:SETUP2
```

The following panel will appear:

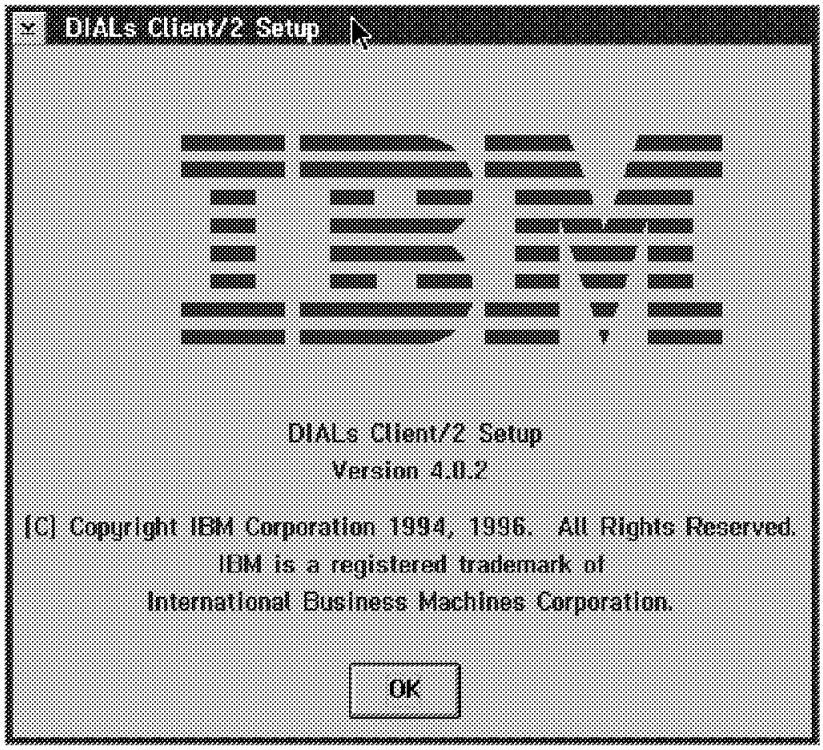


Figure 125. DIALs Client/2 Setup Window

Select the **OK** button. The following window will pop up asking you for the directory in which you wish to install the product.

Figure 126. DIALs Install Directory Window

Type in a directory of your choice and then select the **OK** button. The installation will start and the window shown below will indicate the progress of the installation

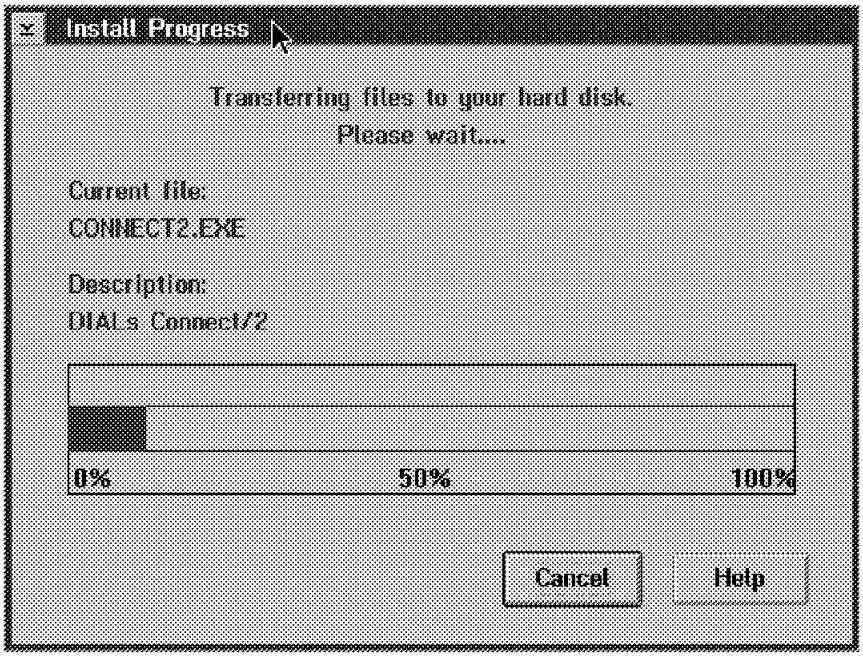


Figure 127. DIALs Install Progress Window

After the files have been copied to your disk you are asked to specify the serial port and the modem type you are using with DIALs:

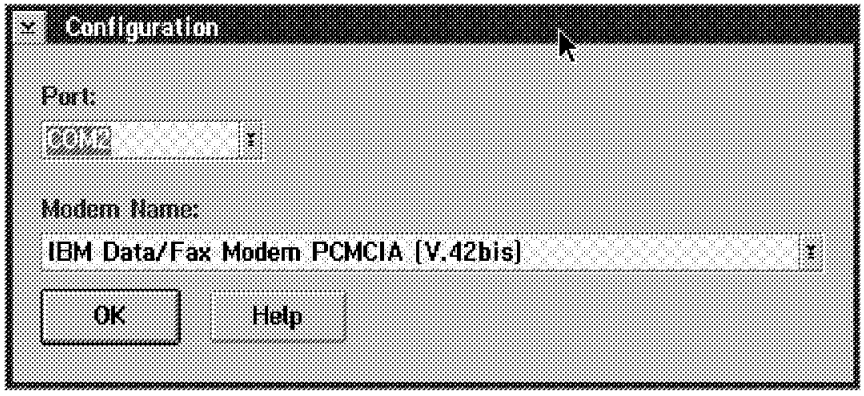


Figure 128. DIALs Configuration Window

In our example we use an IBM ThinkPad with a PCMCIA modem card that is using the virtual COM port 2.

After selecting your configuration click the **OK** button.

The following panel will indicate that the installation is complete.

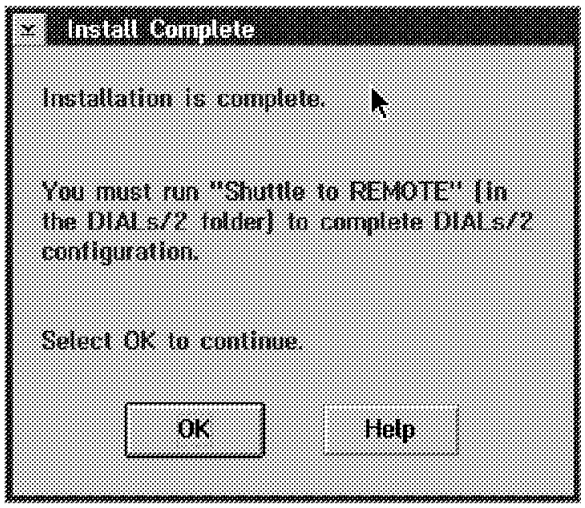


Figure 129. DIALs Install Complete Window

Select the **OK** button. You will find a new icon on your desktop for the DIALs application.

Double-click on this icon to open the folder with the program icons:

Figure 130. DIALs Icon View Window

In order to use your system with LAN and remote connections, DIALs offers two shuttle options.

In order to use your system remotely at the next boot double-click the **Shuttle to REMOTE** icon. The system will inform you that you need to reboot your system for the changes to take effect.

10.3 Installing the Mobile Client

In order to demonstrate how to use the Mobile Client with the IBM 8235 we need to install the Mobile Client for OS/2 on our system.

Refer to Part 2, “Setting Up a Mobile Client Environment” on page 41 for detailed instructions on how to do that.

We use the following response file in our example to install the Mobile Client:

```
; Target path
FILE = C:\SOFTDIST

; Work area
; It is the path for the data directory
WORK = C:\SOFTDIST\WORK

; Software Distribution components to install
COMP = Distribution Mobile Client (with GUI)
COMP = Hw/Sw Discovery Tool

DELETEBACKUP = No
SAVEBACKUP   = Yes
CFGUPDATE    = Auto
OVERWRITE    = Yes

; Software Distribution System Name. This identify the system in the network
; mandatory
SystemName   = stefanh

; Network drivers and Network addresses
Driver.TCPIP = 1
; Driver.NETBIOS = 1
; Parm1.NETBIOS = $(Parm1_NetBios)
; Driver.IPX = 1

; Distribution directories
BackupArea  = C:\SOFTDIST\BACKUP
ServiceArea = C:\SOFTDIST\SERVICE
Repository  = C:\SOFTDIST\REPOS
WorkArea    = C:\SOFTDIST\WORK

; Software Distribution Server Connection

; Distribution Server's System Name
ServerName = rs600012

; Network driver (TCP/IP for TCP/IP, NB for NETBIOS, TLI for IPX)
ServerDriver = TCP/IP

; Distribution Server's address (hostname for TCP/IP, NETBIOS/IPX address
; for NETBIOS/IPX)
ServerAddress = rs600012

; Distribution Client's hostname.
TCP.Hostname = stefanh

; Distribution Client's Target address. Warning: this field should be
; max 8 characters in length.
TargetAddress = STEFANH
```

Figure 131. Response File to Install OS/2 Mobile Client

Refer to 4.2.3, “Response File Driven Installation the OS/2 Mobile Client” on page 50 on how to use this response file to install the Mobile Client.

From the response file you can see that the name of the client we use is stefanh and the name of the TME 10 Software Distribution server we will connect to is rs600012.

10.4 Connecting to the DIALs Server

In order to establish a connection to the server, we need to establish a DIALs connection first.

Double-click the **Connect/2** icon in the DIALs/2 folder as shown in Figure 130 on page 168. The following window will appear:

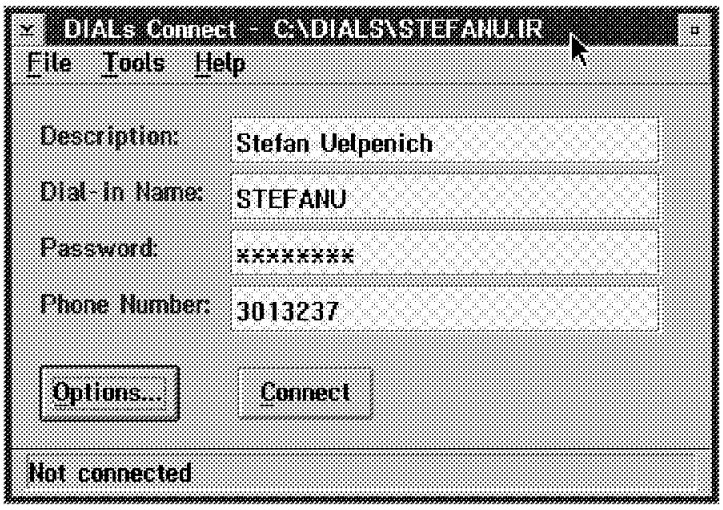


Figure 132. DIALs Connect Window

You must enter a Dial-in Name and a Phone Number to be able to dial a server. The Dial-in Name is a user ID that your local network administrator should provide; Phone Number is the number of the DIALs server you are calling.

Click the **Options** button in order to get the following window:

Figure 133. DIALs Connection File Options Window

We check the box **IP Enabled** in order to use TCP/IP and provide the IP address in the IP Address field.

Note

We use the same IP address that we also use when connected to the LAN in this example. This allows us to transparently switch between LAN and remote connections. The software distribution client will not recognize the difference.

We click the **OK** button to close the Connection File Options window and then the **Connect** button.

The following window will appear:

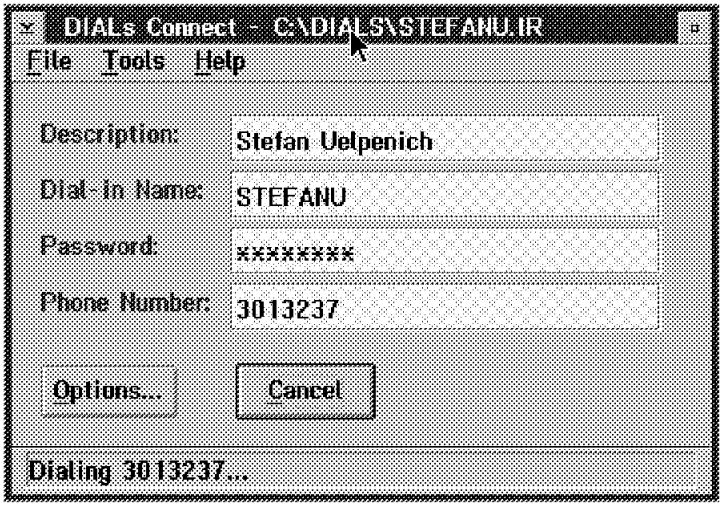


Figure 134. DIALs Connect Window (Dialling)

The DIALs client will dial and try to connect to the server. If you have entered the password in the DIALs Connect window as shown above, then you are automatically logged on to the server. Otherwise you are prompted for your password first.

After your logon has been authenticated, the following window reminds us that our connection time is limited to 2 hours:

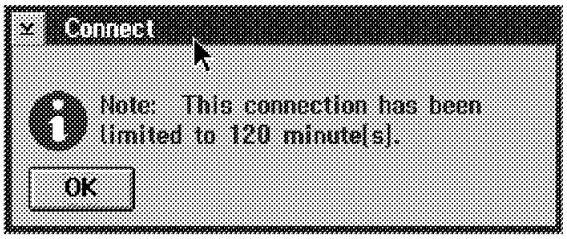


Figure 135. DIALs Connect Information

Note

You can also perform the connect from the command line using the CONNECTC command from the DIALs product directory.

This is a good idea when automating the connect process, for example, as shown in Chapter 18, “Example Front-End Application” on page 325.

Once we are connected we can check if we are able to reach the server by issuing the following command:

```
ping rs600012
```

The output should look similar to the following:

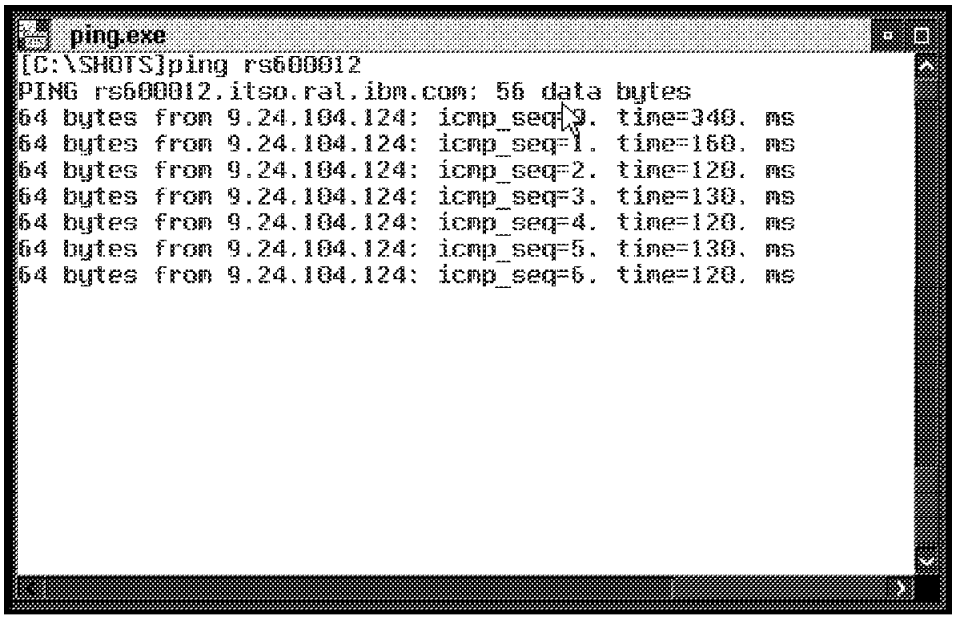


Figure 136. Result of Ping Command

Note

Since we use the same IP address and the same TCP/IP configuration that we also use when LAN-connected, we can use IP just as though we were attached to the LAN.

We have configured the name server 9.24.104.108 in our TCP/IP configuration, so we are also able to resolve hostnames. This will allow us to specify the name rs600012 in the above example.

10.5 Performing a Change Request

In order to demonstrate the Mobile Client using DIALs we perform a simple inventory request.

10.5.1 Starting the Mobile Client

If the client program is not already started, start it now, either by clicking on the corresponding icon or by typing:

```
NVDM START
```

The agent should then start. The startup messages look similar to the following:

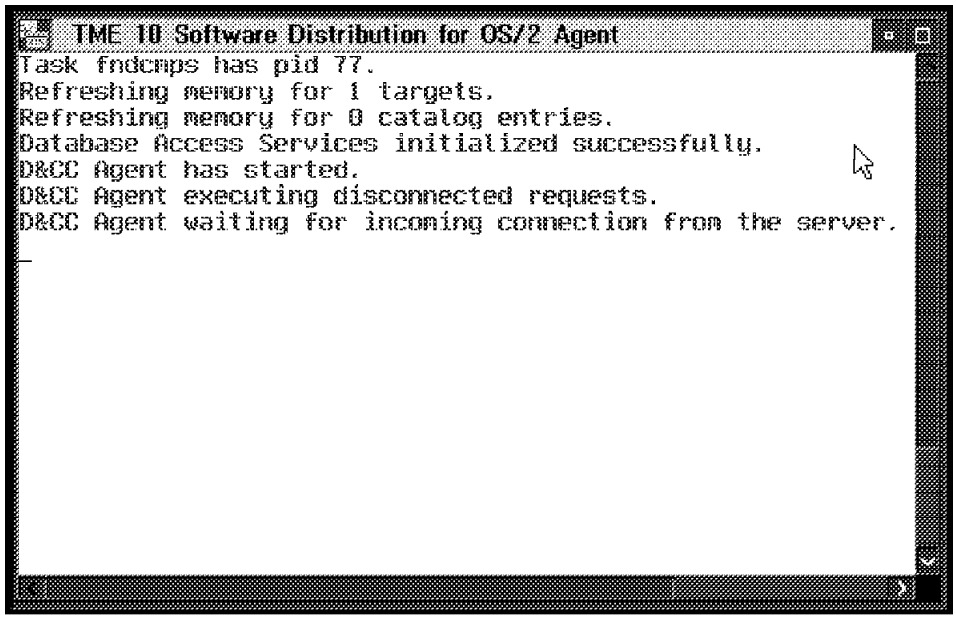


Figure 137. Agent Startup Window

10.5.2 Creating an Inventory File

We create an inventory file using the following command:

```
FNDINV
```

This creates the following hardware inventory file in C:SOFTDISTFNDHWINV:

```
#*********** Inventory Date&Time **********
Inventory.dateAcquired: 1996-11-06 13.11.14

#*********** Model and Processor Information **********
Equipment.machineModel: IBM PS/2 Thinkpad 755
PWS.busType/: ISA or AT bus
PWS.biosRevisionLevel/: 18
PWS.biosLevelDate/: 1996-02-15
PWSType.processor/: 80486DX4
PWSType.processorSpeed/: 75
PWS.memorySizeMb: 36.48
PWS.planarFRUNumber/: N/A
PWS.parallelPorts: 1
PWS.serialPorts: 2
OperatingSystem: OS/2 Operating System
OperatingSystem.version: 3.01

#*********** Video Subsystem Information **********
Display.adapterType/: Super VGA
Display.type/: Color Display
Display.memory/: 1024
Display.colors/: 256
Display.hozResolution/: 640
Display.verResolution/: 480
Display.hozSize/: 240
Display.verSize/: 180

#*********** Keyboard Information **********
Keyboard.type/: 84/85 key Enhanced Keyboard
Keyboard.countryCode/: US
Keyboard.subCountryCode/: 103
Keyboard.codePage/: 437

#*********** Printer Information **********
Printer/NULL: IBM 4039 plus
Printer.driver/NULL: PSCRIPT

#*********** Printer Information **********
Printer/NULL: QMS ColorScript 100
Printer.driver/NULL: PSCRIPT

#*********** Printer Information **********
Printer/LPT3: Fax
Printer.driver/LPT3: FXPRINT

#*********** Mouse Information **********
Mouse.type/: PS/2 Mouse
Mouse.buttons/: 2

#*********** Diskettes Information **********
DisketteDrives: 1
DisketteDrive/A: 1.44MB 3.5-inch Diskette Drive
```

Figure 138. Hardware Inventory File on OS/2 Client (Part 1)

```
#*********** Hard Disk Information **********
FixedDisks: 1
FixedDisk.capacityMb/1: 808

#*********** Logical Drive Information **********
LogicalDisk.driveLetter/C: Hard Disk 1
LogicalDisk.volumeLabel/C: CDISK
LogicalDisk.fileSystem/C: FAT
LogicalDisk.availCapacityMb/C: 163.42

#*********** Logical Drive Information **********
LogicalDisk.driveLetter/D: Hard Disk 1
LogicalDisk.volumeLabel/D: OS2
LogicalDisk.fileSystem/D: HPFS
LogicalDisk.availCapacityMb/D: 41.67
```

Figure 139. Hardware Inventory File on OS/2 Client (Part 2)

10.5.3 Connecting to the TME 10 Software Distribution Server

The following commands will schedule the inventory files to be sent to the server and then connect to the server to actually perform the request:

```
nvdm inv
nvdm connect
```

To see if the client is connected, we start the command line interface on the client and invoke the stattg command as shown below:

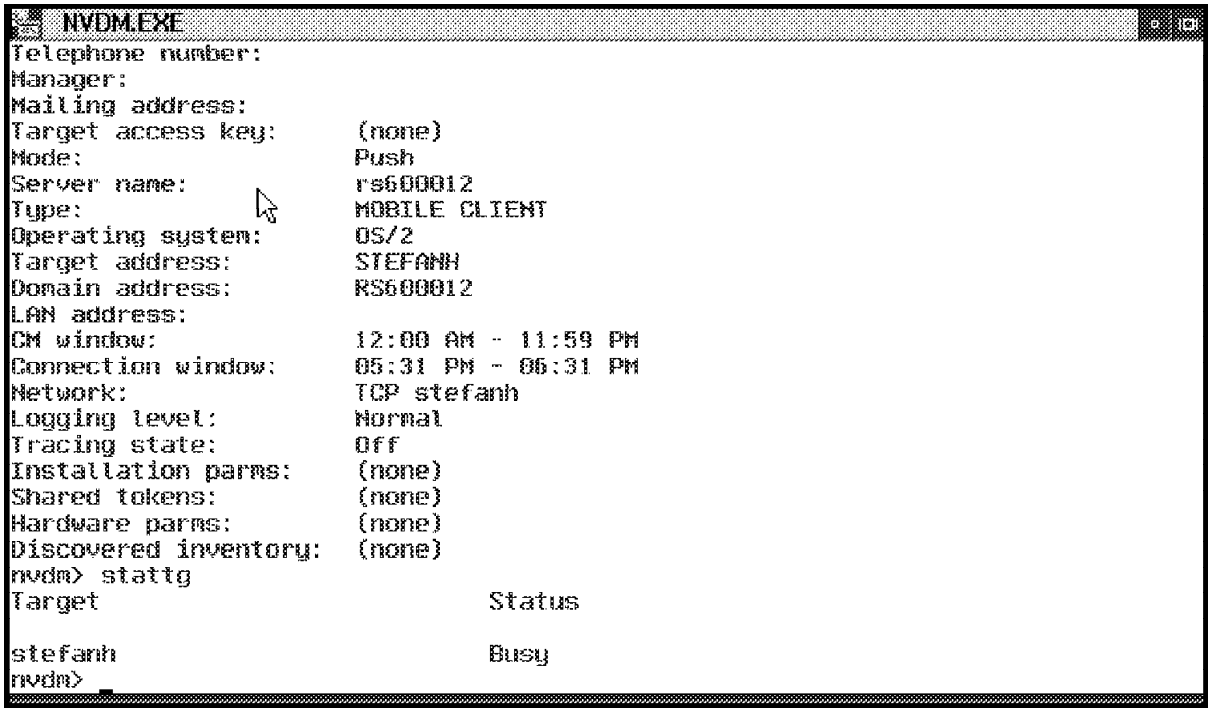


Figure 140. Querying the Target Status

In the above example the status is busy so the Mobile Client is connected to the server.

We can also type `lsrq -w stefanh` to see the requests scheduled for our target:

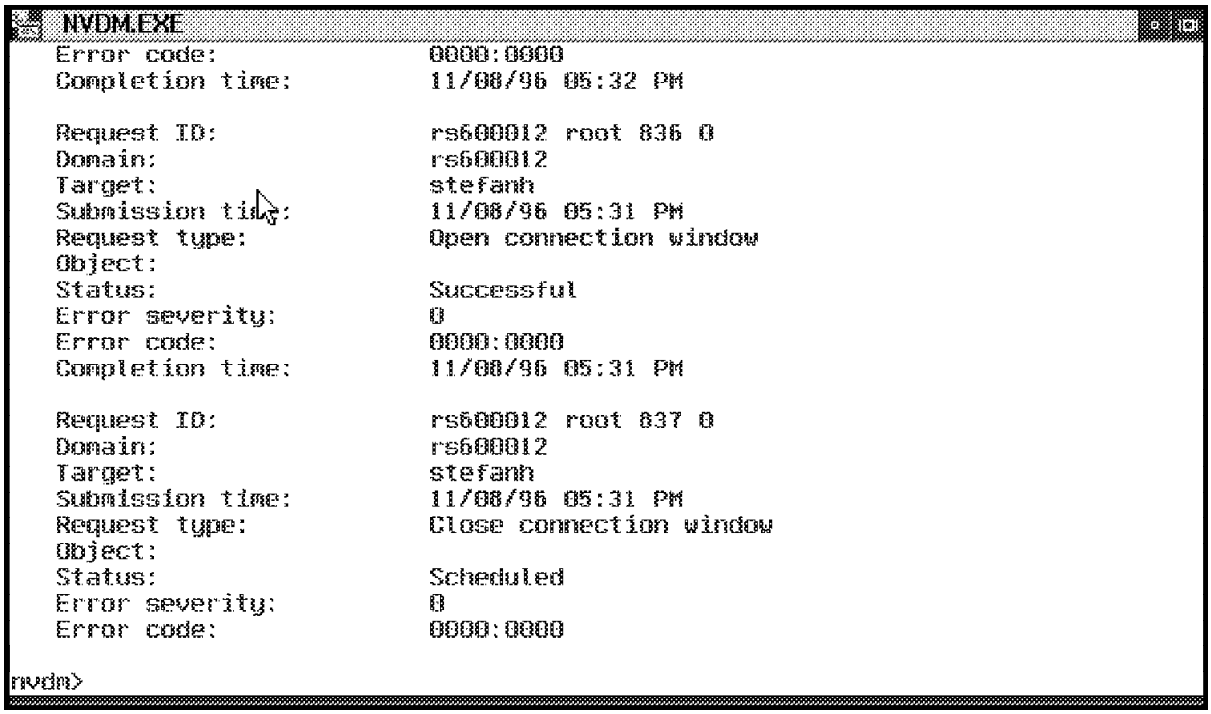


Figure 141. Querying the Scheduled Requests

After a while, we type the following command in the command line interface of the client:

`lstg -l stefanh`

In our example the output looks like the following:

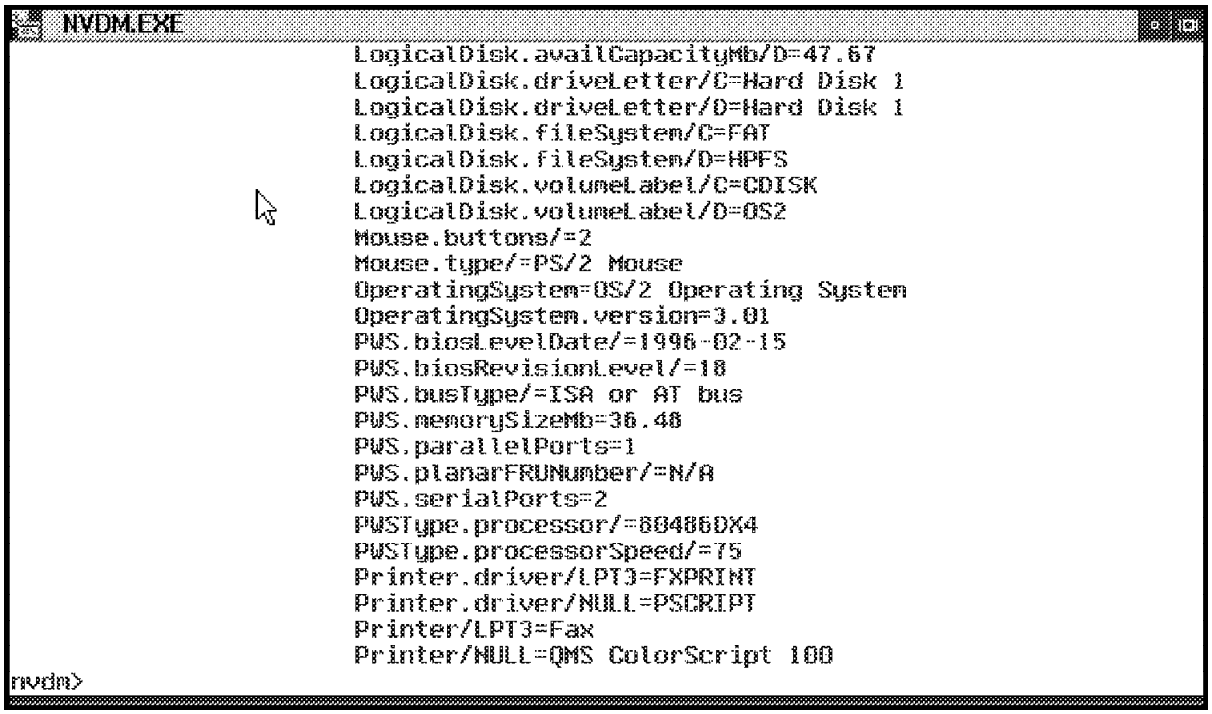


Figure 142. Output of lstg Command

The command shows the hardware inventory on the server discovered for our client, so the hardware inventory file has been successfully transferred to the server.

On the client we can see the following messages in the agent window:

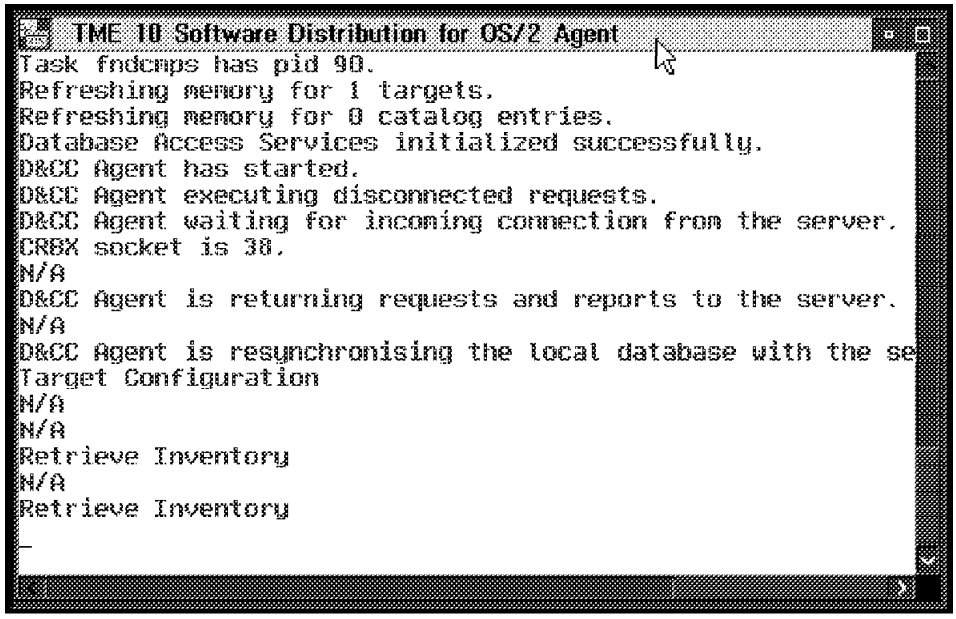


Figure 143. Agent Message Window

10.6 Disconnecting from the DIALs Server

After performing our work, we can disconnect from the DIALs server.

We use the same window that we have also used for connecting, but this time we select the **Disconnect** button as shown below:

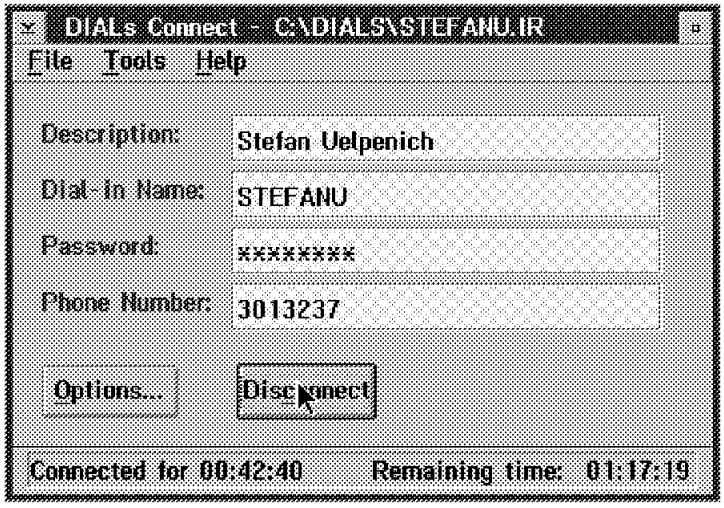


Figure 144. DIALs Connect Window

The following window will pop up:

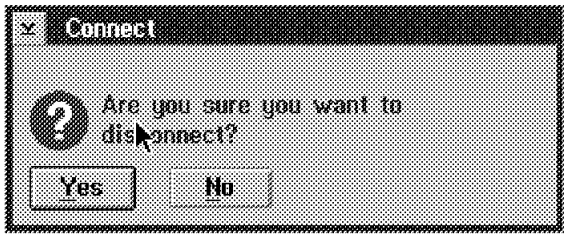


Figure 145. DIALs Disconnect Confirmation Window

Select **Yes** to disconnect from the DIALs server.

Note

To disconnect from the command line type:

```
CONNECTC /d
```

Chapter 11. Using the Asynchronous Protocol Support for Mobile Clients

The support of the native asynchronous protocol is a new feature of the Software Distribution 3.1.3 product. The support is provided in the server and in the Mobile Client product component.

The asynchronous protocol support is also called support for dialled connections and allows performing change management functions toward a Mobile Client over an asynchronous dialled communication link. The communication protocol used between the server and Mobile Clients is native asynchronous protocol.

A support has been added to the Software Distribution 3.1.3 server to be able to accept dial-in connections from Mobile Clients and make dial-out connections to Mobile Clients. For the management of multiple connections to several Mobile Clients the server is able to support simultaneous connections. Also see 11.3, “Defining Modem Pools for Multiple Simultaneous Connections” on page 198 for the usage of several modems for multiple connections.

In this chapter we show how to set up the environment for the asynchronous protocol and demonstrate the usage of it with Mobile Clients in simple change management scenarios. The environment we are using is presented in Figure 146.

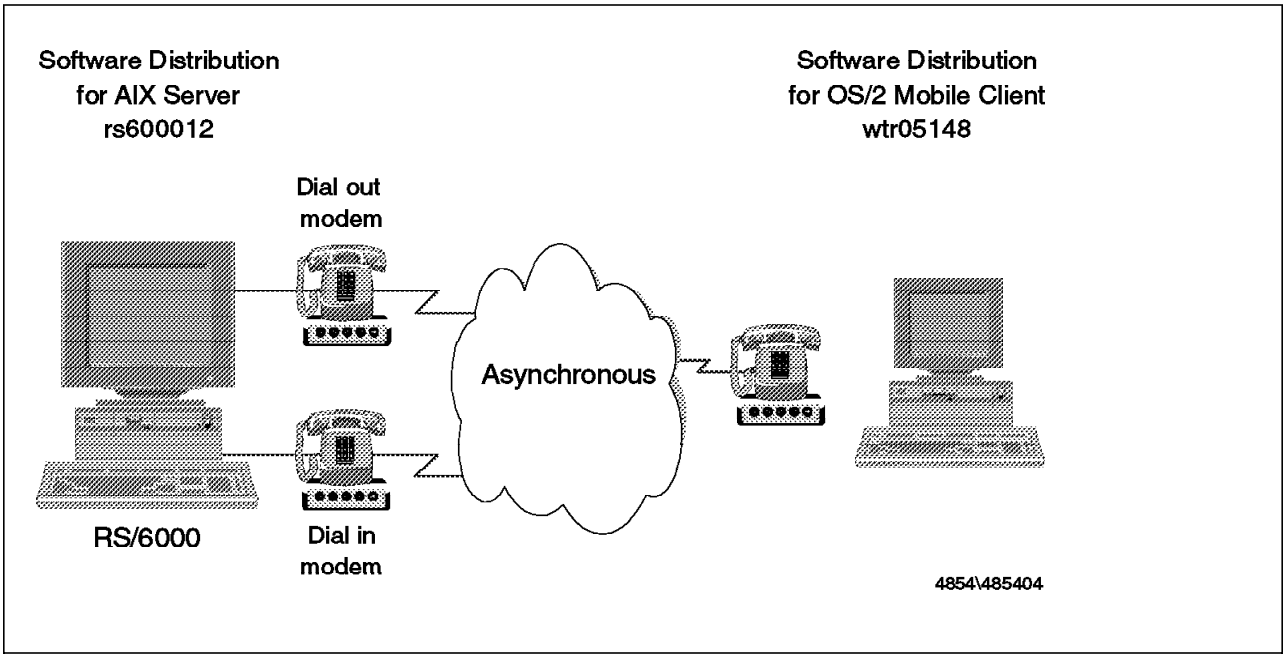


Figure 146. Configuration for the Asynchronous Environment

© Copyright IBM Corp. 1997

Note

At the time this redbook was written we had only access to the asynchronous feature on the AIX server and the OS/2 and Windows 3.1 clients.

Please check with your IBM representative to determine if asynchronous support is available in the version of TME 10 Software Distribution you run.

In this chapter we describe the complete setup for an OS/2 client connected to an AIX server. In Appendix B, “Asynchronous Support for Windows 3.1 Clients” on page 363 we explain the differences for the Windows 3.1 client.

We also give hints that help you determine if asynchronous support is available in the version of TME 10 Software Distribution you run.

You should also notice that even if your client or server does not support native asynchronous communications you can still use modem connections for the Mobile Client by using any of the various other communications subsystems shown in this redbook (PPP, DIALs, ...).

11.1 Prerequisites

To use the asynchronous protocol support with TME 10 Software Distribution you will need the following additional hardware:

- One modem for each client you want to connect
- At least two modems per TME 10 Software Distribution server

However, you will not need any communications subsystem, for example, TCP/IP because the Mobile Client can directly transfer data over the modem link without any underlying communications protocol.

11.2 Configuring the Environment for Asynchronous Protocol Support

In this section we supply the information needed to configure the Software Distribution 3.1.3 Mobile Client and server for the asynchronous protocol support.

As an example we are using the Software Distribution for OS/2 client wtr05148, which we used before as a standard client for the preparation site. The server we configure for asynchronous support is the Software Distribution for AIX server rs600012. Figure 146 on page 181 shows the configuration we are using.

First we show what needs to be done at the client workstation to set up the asynchronous support. Then we show the setup of the AIX system and describe what is required to activate the asynchronous support at the Software Distribution for AIX server.

The steps to configure an existing environment for the asynchronous protocol support covered in this section are:

1. Defining communication ports on the OS/2 workstation
2. Changing the configuration of the Mobile Client to asynchronous protocol
3. Setting up the AIX server for asynchronous communication

4. Activating the asynchronous protocol support at the Software Distribution for AIX server
5. Defining asynchronous support for the Software Distribution for OS/2 target at the server

11.2.1 Defining the Communication Port at the OS/2 Workstation

At the OS/2 desktop workstation we did not need to define a communication port for our scenario. The communication port COM1 was already available. We connected an IBM 7855 modem to this communication port for the asynchronous connection.

Note

You should be able to use any modem with the asynchronous feature that supports the Hayes standard for modems. However, we experienced some problems with the modems talking to each other when the modems on both ends were not the same.

The modem must be configured for the asynchronous support in Software Distribution 3.1.3 with the standard settings:

- 8 data bits
- No parity
- 1 stop bit

To ensure proper setting of the port before the start of the OS/2 Mobile Client we included a command in the STARTUP.CMD to set the mode of the communication port COM1:

```
mode COM1 9600,n,8,1
```

Notice that the speed is set to 9600 baud here. We set the same speed at the AIX server site as shown in 11.2.3, “Setting Up the AIX Server for Asynchronous Communication” on page 185 and started testing at this speed.

You can test if the modem is accessible from the OS/2 system by issuing the following command from an OS/2 command prompt:

```
echo "AT" >COM1
```

If the modem can be accessed, you should see the lights at the modem flashing, when the command is entered.

11.2.2 Changing the Configuration of the Mobile Client to Asynchronous Protocol

During the product installation it is not possible to define that the asynchronous protocol is used for the communication with the server. There is no option for the asynchronous protocol offered in the client configuration notebook as shown in Figure 17 on page 48. Thus the configuration of the client must be changed after the product installation to define the asynchronous protocol. Refer to 4.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 45 and 5.2.2, “Interactive Installation of the OS/2 Mobile Client” on page 74 for an installation description of the OS/2 client. The asynchronous protocol support is defined through the definition of the protocol, the communication ports used and the modem dial string in the configuration file of the Mobile Client.

Figure 147 on page 184 shows the configuration file for the OS/2 Mobile Client with the protocol set to asynchronous. Following the figure we explain the values to be defined for the asynchronous support.

```
WORKSTATION NAME:                   wtr05148
SERVER:                             rs600012 ASY 3013456
PROTOCOL:                           ASY 3013473
REPOSITORY:                         d:\SOFTDIST\REPOS
WORK AREA:                          d:\SOFTDIST\WORK
BACKUP AREA:                        d:\SOFTDIST\BACKUP
SERVICE AREA:                       d:\SOFTDIST\SERVICE
CONFIGURATION:                      CLIENT
MESSAGE LOG LEVEL:                  D
LOG FILE SIZE:                      64000
API TRACE FILE SIZE:                64000
TRACE FILE SIZE:                    64000
MAX USER INTERFACES:                20
MAX ATTEMPTS:                       5
TARGET MODE:                        PUSH
MACHINE TYPE:                       OS/2
TARGET ADDRESS:                     wtr05148
PORT:                               COM1
INIT:                               ATM1
DIAL:                               ATDT9,
```

Figure 147. OS/2 Mobile Client Configuration File for Asynchronous Support

To change the protocol being used by the client to be native asynchronous protocol specify ASY in the PROTOCOL keyword value followed by the telephone number for the client. In our example the telephone number is 3013473 and thus the keyword is specified as follows:

```
PROTOCOL:       ASY 3013473
```

The SERVER keyword must also be changed to specify the asynchronous protocol and the telephone number for the server. The value ASY followed by the telephone number of the server is appended to the server name in the SERVER keyword. In our example the telephone number of our server is 3013456 and the server name rs600012, so we specify the following keyword values:

```
SERVER:         rs600012 ASY 3013456
```

With the asynchronous protocol support new keywords have been added to the configuration file to define the communication port used for the dialled connection and the dial string for the modem. The port and modem dial string are defined at the server and the Mobile Client. For the client there is only one entry that defines the connection to the server with the keywords:

- PORT
- INIT
- DIAL

The keyword PORT specifies the port used to communicate with the server. For the OS/2 Mobile Client the value represents the OS/2 COM port where the asynchronous modem is connected to. We are using the port COM1 in our example.

The INIT keyword is used to specify the initialization string that is sent to the modem. The string defined here is a Hayes AT command string. In our example

we use ATM1 to initialize the modem, where M1 switches on the speaker of the modem on during dial operations.

The Mobile Client uses the same modem for dial-in and dial-out operations, so the DIAL keyword must be specified. It contains the prefix for the dial command which is put before the telephone number specified in the SERVER keyword. In our example we specify the keywords with the values listed below:

```
PORT:          COM1
INIT:          ATM1
DIAL:          ATDT9,
```

The configuration file NVDM.CFG is located by default in the directory C:SOFTDIST for the OS/2 client.

Now that we changed the client configuration to asynchronous protocol we can restart the Mobile Client using the nvdm start command. The OS/2 Mobile Client is now ready to communicate using the native asynchronous protocol. We show how to verify the configuration and use the Mobile Client with asynchronous protocol in 11.4, “Using the Mobile Client with the Asynchronous Protocol” on page 199.

11.2.3 Setting Up the AIX Server for Asynchronous Communication

The Software Distribution for AIX server needs at least two modems for the asynchronous communication as shown in Figure 146 on page 181. One modem is used by the server for incoming connection calls from Mobile Clients and the other modem is used for the outgoing connections to the Mobile Clients.

To set up the AIX server for the asynchronous communication you have to:

1. Connect two modems to the RS/6000 machine
2. Define the communication ports to AIX

We connected one IBM 7855 modem to the serial port 1 and another IBM 7855 modem to the serial port 2 of the RS/6000 machine where the Software Distribution for AIX server is installed.

Note

Any other modem should also work in this scenario, but was not tested here. Make sure that the modem is operational and if any settings have to be made to the modem itself, follow the instructions in the manual provided with the modem. We suggest that you use a modem that supports a speed of 9600 baud or more for the use with Software Distribution 3.1.3 to achieve satisfying performance during change management processing.

To define the communication ports to AIX for the two modems follow these steps:

1. Start the system management interface tool SMIT on the AIX machine where you connected the modems. To go directly to the TTY definition in SMIT use the command:

   ```
   smit tty
   ```

   The SMIT window for the TTY definition is displayed as presented in Figure 148 on page 186.

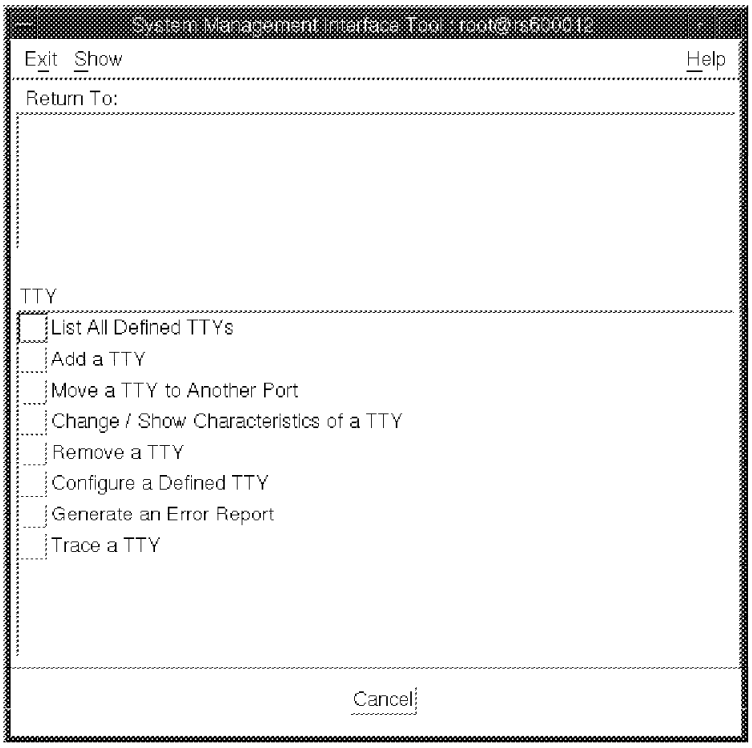


Figure 148. SMIT Window for TTY Definition

2. We assume in our example that the serial ports have not yet been defined at the AIX server. To define a port to AIX select **Add a TTY** in the SMIT window. You will see the Single Select List window for the tty terminal type selection as shown in Figure 149.

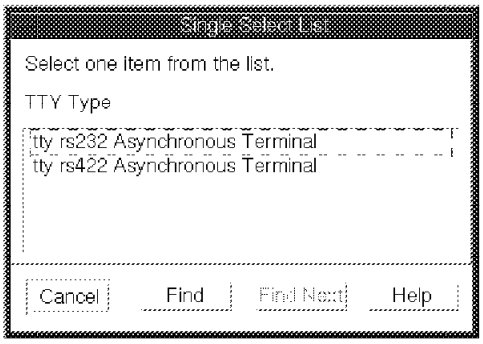


Figure 149. Single Select List Window for TTY Terminal Type Selection

3. Select **tty rs232 Asynchronous Terminal** as the tty type in that window. A second selection window is presented, such as the one shown in Figure 150 on page 187.

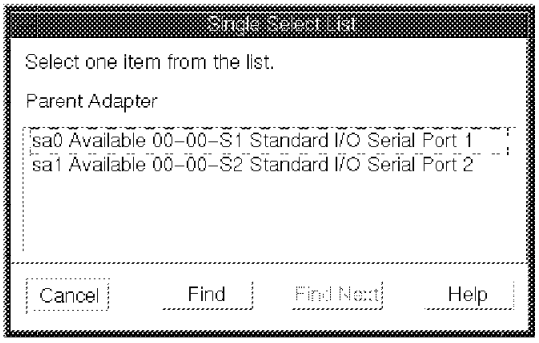


Figure 150. Single Select List Window for TTY Serial Port Selection

4. Choose the serial port the modem is connected to. Our first modem is connected to serial port number 1 in our example. Now you should see the Add a TTY window as shown in Figure 151.

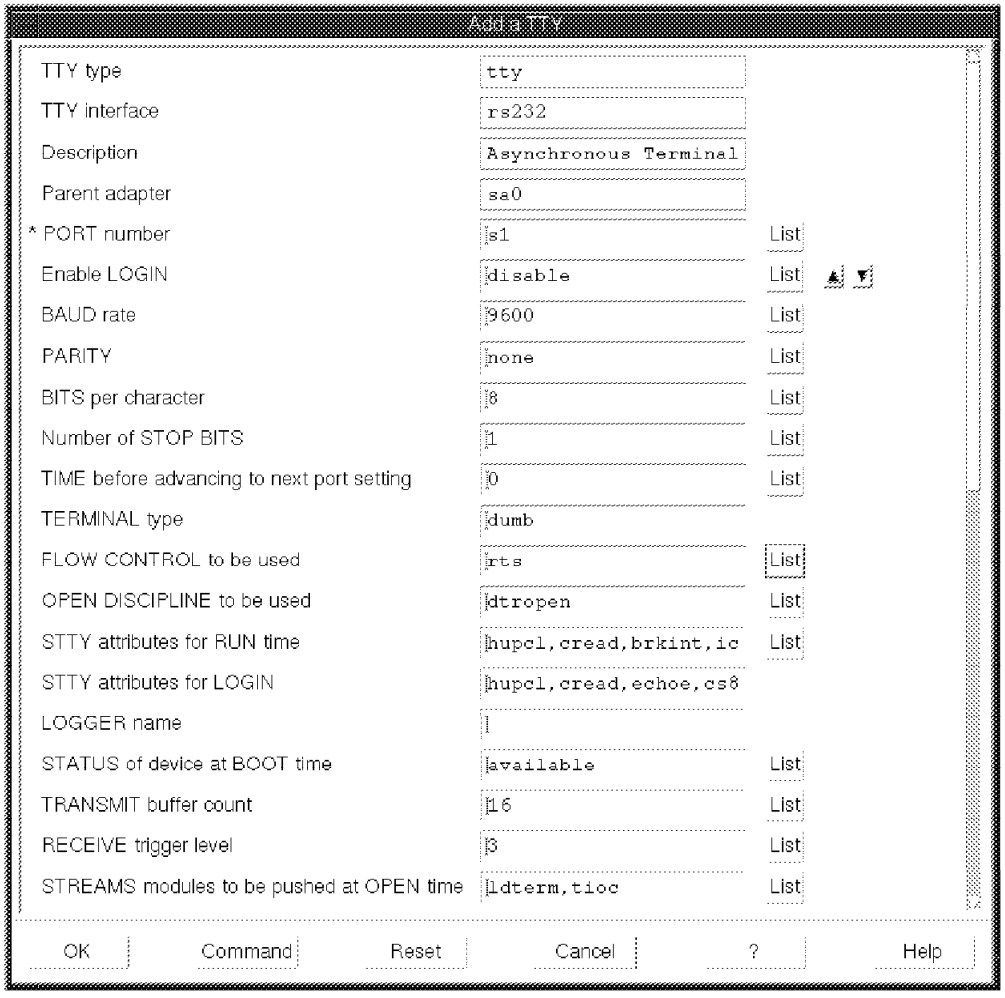


Figure 151. Add a TTY Window - tty0

5. In this window you need to define the port number for the tty port by selecting the **List** push button next to the field PORT number. In the Single Select List window, shown in Figure 152 on page 188, select **s1** for the communication port tty0.

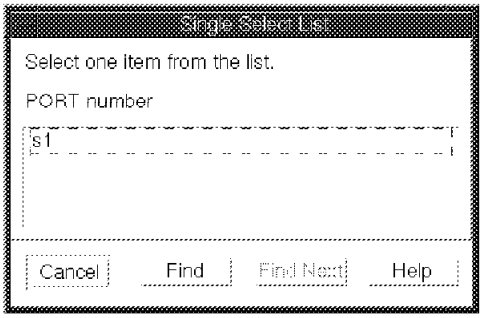


Figure 152. Single Select List Window for TTY Port Number

You also need to change the value in the field FLOW CONTROL to be used in the Add a TTY window (shown in Figure 151 on page 187). To do so, select the **List** push button next to the field and in the window that is presented, as shown in Figure 153, select rts for the flow control.

For all the other values you can accept the defaults presented in the Add a TTY window in Figure 151 on page 187.

Note

Notice that the speed here is set to 9600 baud. This must be consistent with the speed used with the modem connected to the OS/2 workstation. Also see 11.2.1, “Defining the Communication Port at the OS/2 Workstation” on page 183.

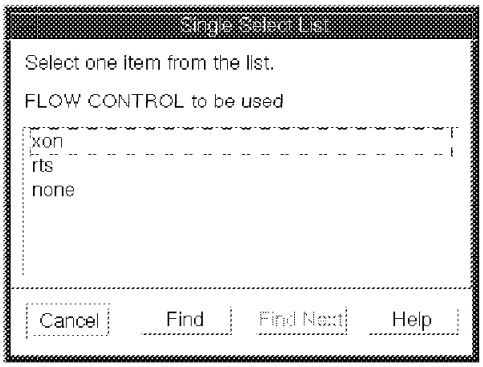


Figure 153. Single Select List Window for TTY Flow Control

6. Repeat steps 2 through 5 for the definition of the second communication port tty1. This time select serial port **2** in the window shown in Figure 150 on page 187 and **s2** for the port number in the window shown in Figure 152. (s2 should be the only option that is offered in the window.)

   The port definition for port tty1 should look somewhat like Figure 154 on page 189.

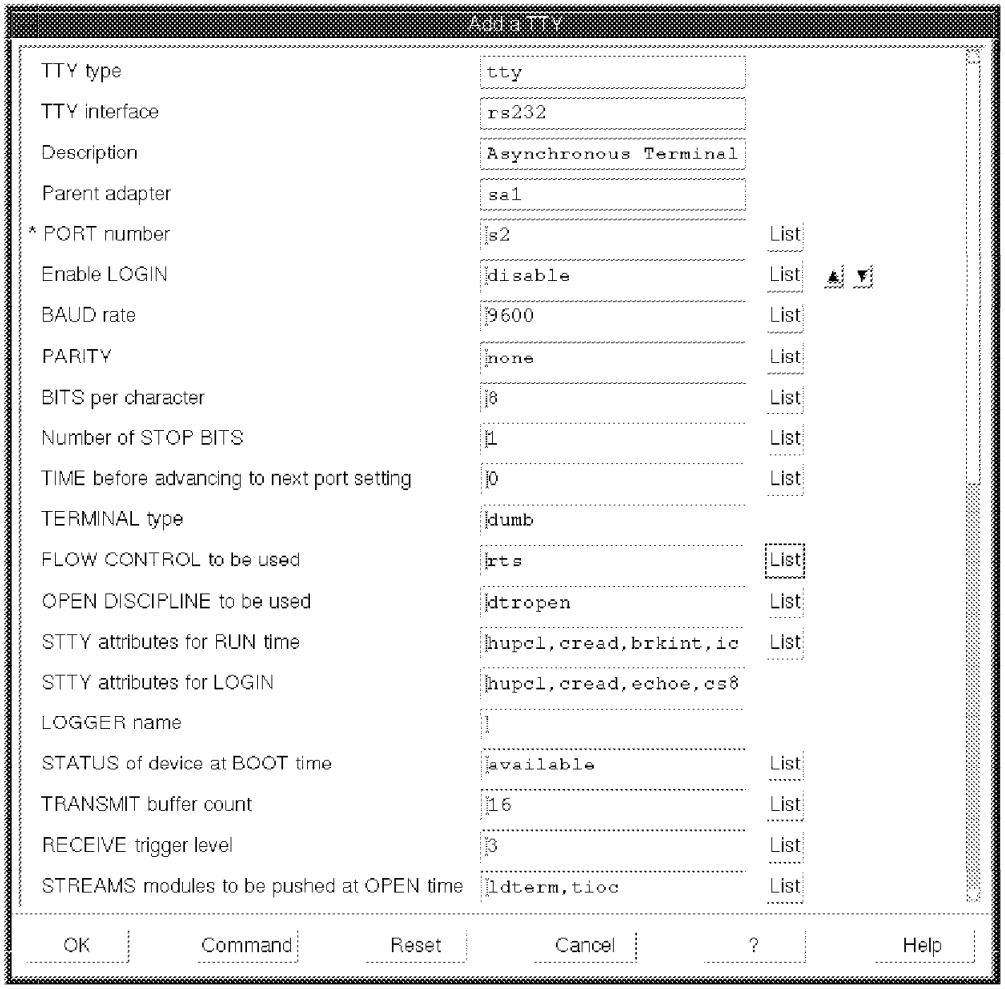


Figure 154. Add a TTY Window - tty1

The two communication ports at the AIX server are now ready for use. You can verify if the modems are accessible by the system by initializing them with the following commands:

```
echo "AT" >/dev/tty0
echo "AT" >/dev/tty1
```

The lights at the modem should be flashing for a short instance when the echo command is issued.

11.2.4 Activating the Asynchronous Protocol Support at the Software Distribution for AIX Server

In the following section we explain what actions need to be taken to activate the asynchronous protocol support at the Software Distribution for AIX server.

Note

Before trying to use the server with asynchronous support you should check for the following file in the /usr/lpp/netviewdm/bin directory of your Software Distribution for AIX server:

`fndrbasy.shr`

If this file is not there you are not able to use the asynchronous support and need to install the appropriate software level first.

Please check with your IBM representative to obtain current version information.

At the server we need to configure the asynchronous protocol support in its configuration file. As already described for the Mobile Client in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183 there are new keywords available for the configuration file at the server. These keywords define the port used to communicate and the modem dial string for the asynchronous support.

The server has definitions for all available ports. Figure 155 shows the configuration file of our Software Distribution for AIX server.

```
WORKSTATION NAME:     rs600012
PROTOCOL:             TCP rs600012 729 50
PROTOCOL:             ASY 3013456
MESSAGE LOG LEVEL:    D
LAN AUTHORIZATION:    0
LOG FILE SIZE:        500000
TRACE FILE SIZE:      1000000
API TRACE FILE SIZE:  500000
TCP/IP PORT:          729
MAX TARGETS:          600
MAX CONNECTIONS:      20
MAX USER INTERFACES:  20
CONFIGURATION:        REMOTE_ADMIN_SERVER
MACHINE TYPE:         AIX
REPOSITORY:           /usr/lpp/netviewdm/repos
SERVICE AREA:         /usr/lpp/netviewdm/service
BACKUP AREA:          /usr/lpp/netviewdm/backup
WORK AREA:            /usr/lpp/netviewdm/work
SERVER:               rs600012
MAX SERVER TARGETS:   2048
PLAN FEATURE:         Y
AUTOMATIC TARGET REGISTRATION:   Y
MAX DIAL RETRIES:     5
DIAL RETRY PERIOD:    180
PORT:                 /dev/tty0
INIT:                 AT
PORT:                 /dev/tty1
DIAL:                 ATDT9,
INIT:                 AT
```

Figure 155. Software Distribution for AIX Server Configuration File for Asynchronous Support

Two types of ports are supported by the Software Distribution for AIX server:

- Answer ports

The answer ports are used for incoming connections from Mobile Clients.

- Dial ports

  The dial ports are used for outgoing connections to Mobile Clients.

In our example we have two tty communication ports that we use for the asynchronous communication in Software Distribution 3.1.3, one for incoming calls and one for outgoing calls. We defined these ports to be tty0 and tty1 in 11.2.3, “Setting Up the AIX Server for Asynchronous Communication” on page 185. So now we add two entries for the ports in the servers configuration file as follows:

```
PORT:           /dev/tty0
INIT:           AT

PORT:           /dev/tty1
DIAL:           ATDT9,
INIT:           AT
```

The first entry references the communication port tty0. This port is used by the Software Distribution for AIX server for incoming connections and is initialized with the Hayes AT command set string AT, as specified by the keyword INIT.

The second entry specifies the port tty1 as a dial-out port. This is indicated by the presence of the keyword DIAL after the PORT keyword. The DIAL keyword contains the prefix for the dial string, which is put before the telephone number to dial. In our example ATDT9 is the prefix, where 9 is required to get an outside connection from within the building.

The telephone number to dial is either specified in the targets protocol definition at the server as shown in 11.2.5, “Defining Asynchronous Support for the Target at the Server” on page 192 or set by the phone number parameter in the connect command as shown in 3.2.1, “General Information about the Mobile Client” on page 18 and later in 11.4, “Using the Mobile Client with the Asynchronous Protocol” on page 199.

The server supports several protocols to communicate with the clients in its domain simultaneously; therefore we added a second PROTOCOL keyword in the servers configuration to define the asynchronous protocol. The value ASY is followed by the telephone number of the server. Notice that this is the same number we added in the configuration file of the OS/2 Mobile Client in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183.

Additional parameters for the server have been added with the asynchronous support to define the number of re-dials and pause between re-dials:

- MAX DIAL RETRIES

  This keyword specifies the number of dial attempts for each dial period.

- DIAL RETRY PERIOD

  This keyword specifies the time in seconds the server waits between following re-dials.

The configuration file nvdm.cfg for the AIX server is located by default in the path /usr/lpp/netviewdm/db.

To activate the new definitions in the configuration file we now have to stop nvdm and restart it. We do this by using the command:

```
nvdm stop -k -x
nvdm start
```

The parameter -x shuts down the base and server component and -k terminates all software distribution tasks, but not running transmission processes.

Note

You must make sure that no graphical user interface component of the server is running at the server itself or on any connected machine to activate the changes in nvdm.cfg. If any component of the Software Distribution for AIX server is still running after stopping nvdm the changes will not be activated.

11.2.5 Defining Asynchronous Support for the Target at the Server

We assume here that the target for the OS/2 Mobile Client was already defined at the AIX server. In our example this is the case, because the client was used as a preparation site in other scenarios.

If you have not yet defined the target either manually or through auto-registration, then do so now. Refer to 5.4.1, “Defining a Target Using the Graphical User Interface or the Command Line Interface” on page 83 and 5.4.2, “Using Automatic Client Registration” on page 86 for information about target definition.

To change the protocol configuration of the target to asynchronous, the protocol definition for that target must be updated at the server. This can be done using two different approaches:

- Changing the target configuration manually
- Deleting and redefining the target using auto-registration

We show both approaches in the following sections.

11.2.5.1 Changing the Target Configuration Manually

We show how to change the target configuration to use asynchronous protocol manually at the server using the graphical user interface and give an example of the same function using the command line interface.

From the Catalog window at the server select the option **Windows** and **Targets** from the pull-down menu. The Target window is opened as shown in Figure 156 on page 193.

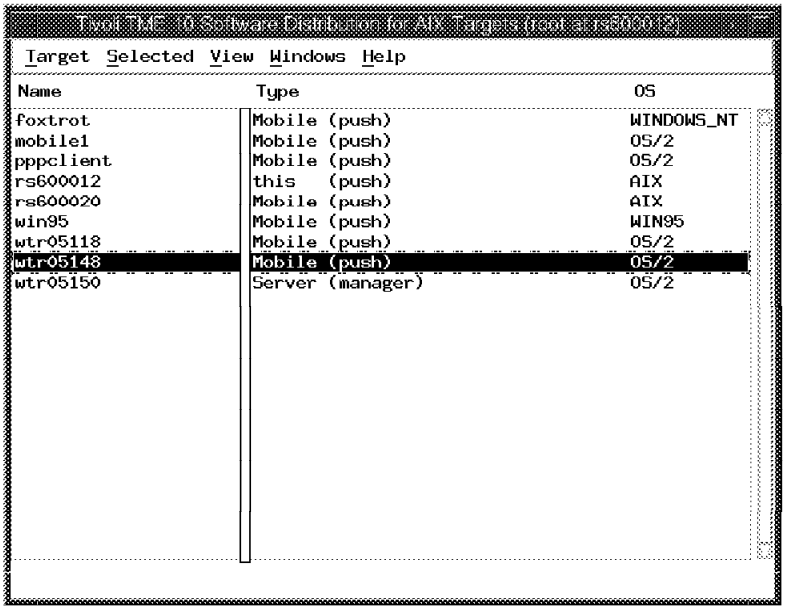


Figure 156. Target Window at AIX Server

From the Target window position the cursor on the target that is defined for asynchronous support. In our example this is the target wtr05148. Select the option **Selected** from the Target window with the target of your choice being the one highlighted and select **Update** from the pull-down menu as shown in Figure 157.

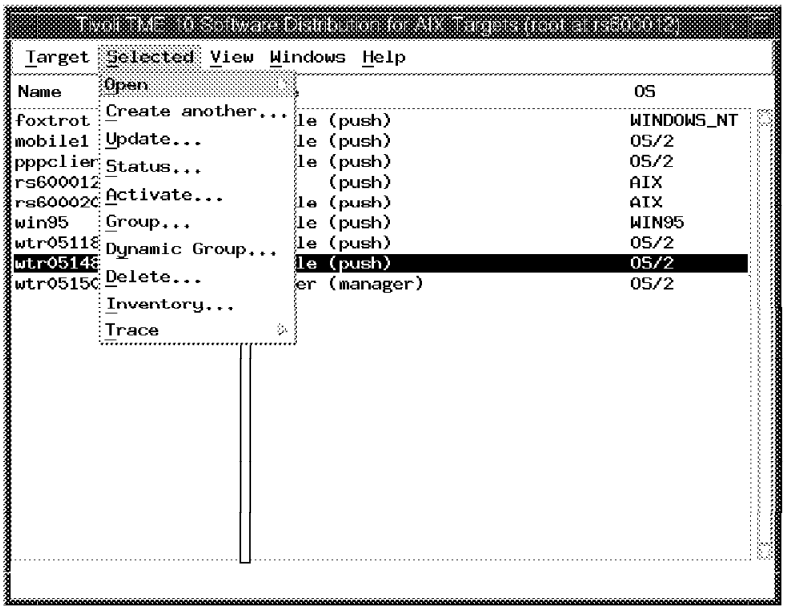


Figure 157. Selected Option in Target Window at AIX Server

The Update Target window is displayed with the definitions for the selected target at the server, as shown in Figure 158 on page 194.

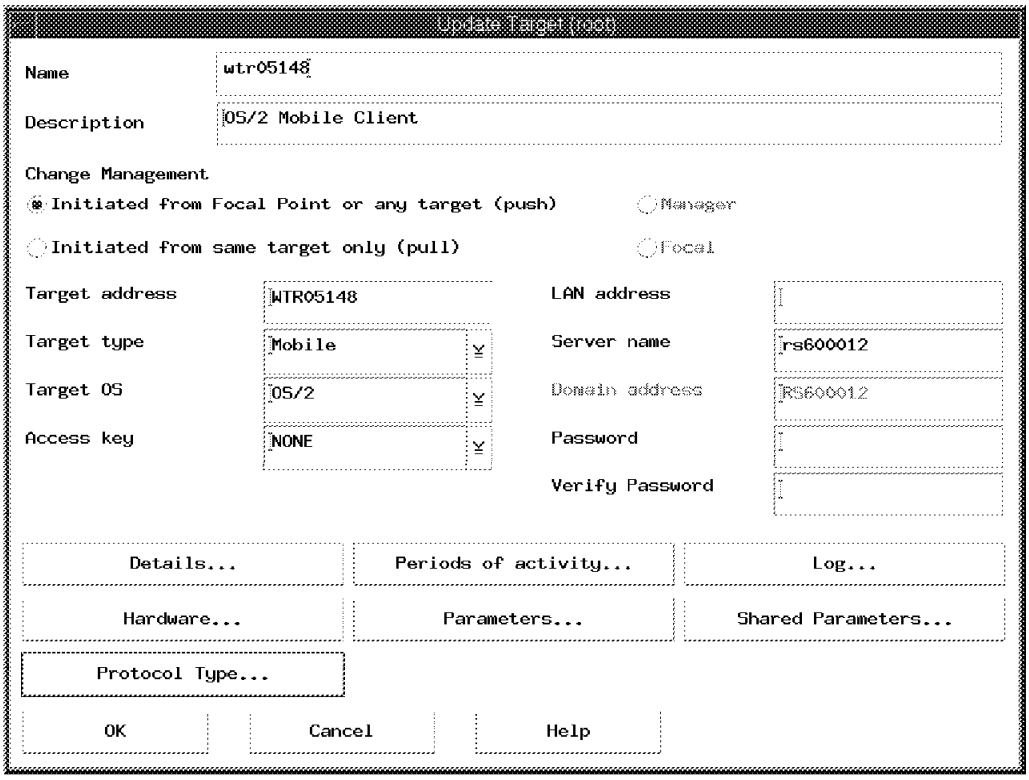


Figure 158. Update Target Window at AIX Server

In the Update Target window select the selection button **Protocol Type...** and the Protocol Type window will appear for the target wtr05148 as seen in Figure 159.

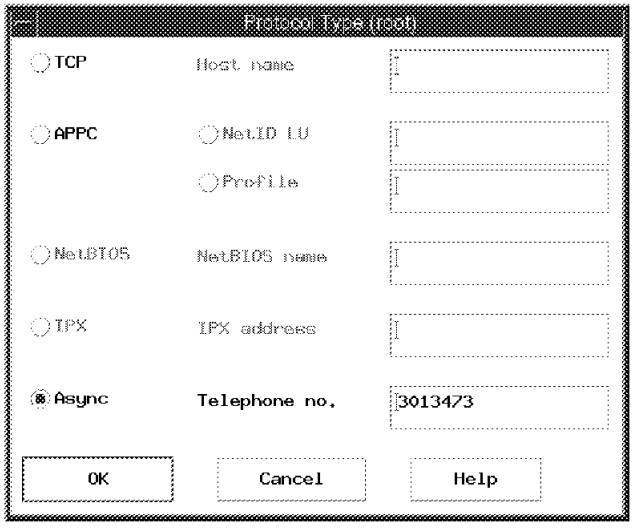


Figure 159. Protocol Type Window for Targets at AIX Server

Select the **Async** radio button to indicate that the protocol type for this target is asynchronous. Enter the valid telephone number under which the target can be reached in the Telephone no. field. In Figure 159 we entered 3013473 as the telephone number for our target. The telephone number defined here is the same number we specified in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183.

Note

Notice that the number defined at the server can be changed any time after defining the target. This can be done by using the command nvdm updtg to update the target, as shown later in this chapter, or by specifying a new number in the connect command. See 3.2.1, “General Information about the Mobile Client” on page 18 and 11.4, “Using the Mobile Client with the Asynchronous Protocol” on page 199 for more information about the connect command.

Select **OK** to accept the value entered in the Protocol Type window and **OK** in the Update Target window to change the target configuration for the Mobile Client.

Note

The complete telephone number consists of the number entered here, which is prefixed with the entry made for the dial-out port in the server configuration file as shown in 11.2.4, “Activating the Asynchronous Protocol Support at the Software Distribution for AIX Server” on page 189.

In our example this prefix is ATDT9, so the complete dial string used to connect the client wtr05148 is:

```
ATDT9,3013473.
```

Command Line Example

To perform the update operation described above using the command line interface enter the following command at the server or any target connected to the AIX server:

```
nvdm updtg wtr05148 -tp async:3013473
```

Where the -tp parameter is used to define the communication protocol and async:3013473 specifies the asynchronous protocol and the telephone number of the client wtr05148.

Verify the correct settings after using the command by entering the following:

```
nvdm lstg wtr05148 -l
```

The output for this command should look somewhat like this:

```
Target:                 wtr05148
Description:            OS/2 Mobile Client
Customer name:
Contact name:
Telephone number:
Manager:
Mailing address:
Target access key:      (none)
Mode:                   Push
Server name:            rs600012
Type:                   MOBILE CLIENT
Operating system:       OS/2
Target address:         WTR05148
Domain address:         RS600012
LAN address:
CM window:              00:00:00 - 23:59:59
Connection window:      Not defined
Network:                async 3013473
Logging level:          Normal
Tracing state:          Off
Installation parms:     (none)
Shared tokens:          (none)
Hardware parms:         (none)
Discovered inventory:   (none)
```

We have now completed the steps to configure our environment for the asynchronous protocol support in Software Distribution 3.1.3. In 11.4, “Using the Mobile Client with the Asynchronous Protocol” on page 199 we provide the information for how you can verify the configuration and show the usage of the Mobile Client with asynchronous protocol in simple change management scenarios.

11.2.5.2 Deleting and Redefining the Target Using Auto-Registration

In this section we explain how you can redefine your target for the asynchronous support using the auto-registration. This step can only be taken after completing the steps described in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183. Before using the auto-registration to redefine the target, the target must be deleted at the server. Do the following steps to delete the target:

1. Check for pending requests for the target wtr05148 at the server:

To delete the target configuration at the server we have to ensure that no requests are outstanding for the target. Pending requests are requests that were scheduled by the server for the target, but did not get executed yet. If requests are pending the system will not allow the deletion of the target. You will receive the following message when trying to delete the target wtr05148 with pending requests:

```
FNDCLB83E: The target wtr05148 cannot be deleted because there is at
least one request pending for it.
```

To obtain if any requests are pending for the target, you can issue the following command at the server or any other target:

```
nvdm lsrq -w wtr05148 -cp
```

Where -cp lists only the requests that have the status scheduled. The output shows the request information including the request identification and the target name, similar to the following:

```
Request ID:             rs600012 root 628 0
Domain:                 rs600012
Target:                 wtr05148
Submission time:        11/06/96 08:41:54
Request type:           Ret inv
Object:
Status:                 Scheduled
Error severity:         0
Error code:             0000:0000

Request ID:             rs600012 root 653 0
Domain:                 rs600012
Target:                 wtr05148
Submission time:        11/06/96 11:07:46
Request type:           Open connection window
Object:
Status:                 Scheduled
Error severity:         0
Error code:             0000:0000

Request ID:             rs600012 root 654 0
Domain:                 rs600012
Target:                 wtr05148
Submission time:        11/06/96 11:07:46
Request type:           Close connection window
Object:
Status:                 Scheduled
Error severity:         0
Error code:             0000:0000
```

In our example we have three pending requests, the first one for the retrieve inventory function and the other two for the open and close connection window.

2. Delete the pending requests:

   Use the request number to delete the pending requests for the target. The request number is the third value in the request ID as shown above. To delete the three pending requests in our example enter the commands:

   ```
   nvdm delrq 628
   nvdm delrq 653
   nvdm delrq 654
   ```

Enter y to the system reply `Delete the request 628 [y/n?]`

Attention

Before deleting any pending request you must ensure that the request processing is obsolete. In our example the requests were obsolete and can be repeated any later time.

3. Delete the target at the server rs600012:

   When no more requests are pending for the target you can delete the target definition at the server with the following command:

   ```
   nvdm deltg wtr05148
   ```

   Enter y to the system reply `Delete the target wtr05148[y/n?]`

The target can now be redefined using the auto-registration. The auto-registration is done automatically when you start the OS/2 Mobile Client with the new configuration file, as shown in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183. Now that we deleted the target definition the target is configured automatically with the asynchronous support as soon as the Mobile Client connects the server. See 3.2.3.3, “Data Flow during Automatic Client Registration” on page 36 and 5.4.2, “Using Automatic Client Registration” on page 86 for more information about the automatic client registration.

The configuration of our environment for the asynchronous protocol support in Software Distribution 3.1.3 is now completed. We show how to verify the correct configuration and give examples for the usage of the asynchronous protocol in 11.4, “Using the Mobile Client with the Asynchronous Protocol” on page 199.

11.3 Defining Modem Pools for Multiple Simultaneous Connections

The Software Distribution 3.1.3 server is able to support multiple simultaneous connections to several Mobile Clients with the asynchronous protocol support. The connection of several modems to the server is required for the multiple simultaneous connection support.

This can be achieved using multi-port adapter cards.

Note

Since the RS/6000 machine normally only has two serial ports the usage of a multi-port asynch card is required for the Software Distribution for AIX server to communicate simultaneously with several Mobile Clients.

If multiple modems or communication ports are available, these can be split into two pools at the server:

- One modem pool for listening
- One modem pool for sending

You do not need an equal amount of modems in the listening pool and sending pool. When running Software Distribution 3.1.3 change management processing for Mobile Clients in a productive environment the incoming session will be

much shorter than the outgoing session. This is because the server always calls the client back when a connection is opened by the client.

Since there can only be one asynchronous session up at a time for each Mobile Client the server never needs two modems simultaneously for the same client.

Important Note

Consider the fact that the Software Distribution 3.1.3 server calls the Mobile Client when:

- The connection is initiated by the server.
- The connection is initiated by the Mobile Client.

  After issuing the nvdm connect command at the client, the connection to the server is opened for a short time and then broken. The server calls back the client to start the synchronization and any change management processing.

11.4 Using the Mobile Client with the Asynchronous Protocol

In this section we will show you how to verify the successful configuration performed in 11.2, “Configuring the Environment for Asynchronous Protocol Support” on page 182 at the server and the Mobile Client. Then we demonstrate the behavior of the asynchronous support during simple change management scenarios.

11.4.1 Verifying the Configuration of the Asynchronous Protocol Support

We show the steps you can take to verify the configuration of your asynchronous environment by using the Mobile Client. Thus by taking these steps from the client we also verify that the connection from the server to the client works properly.

The first step to verify if the OS/2 Mobile Client is configured correctly and has access to the modem is to check the startup messages in the TME 10 Software Distribution for OS/2 Agent window, after starting the Mobile Client with the nvdm start command. There will be an entry in the OS/2 task list for this window.

In the agent window you should see that the D&CC agent started and is waiting for incoming requests from the server, as shown in Figure 160 on page 200. If any errors occurred when the D&CC agent tried to initialize the modem, the error message is displayed after the start messages in the window. The modem is initialized by the D&CC agent with the string that is defined in the INIT keyword for the port in the clients configuration file, as described in 11.2.2, “Changing the Configuration of the Mobile Client to Asynchronous Protocol” on page 183.

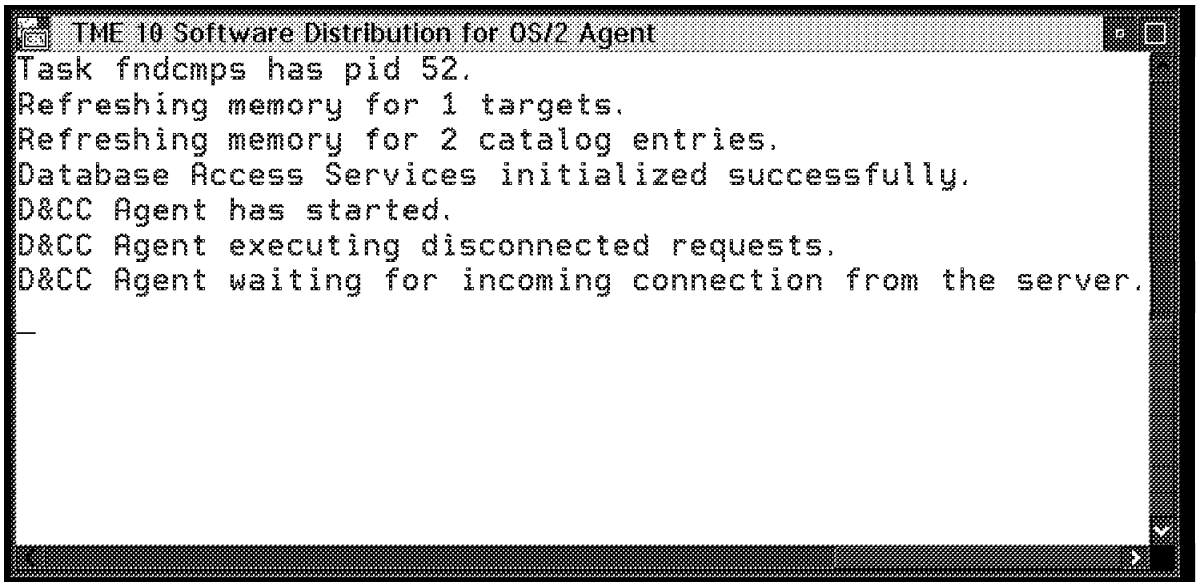


Figure 160. OS/2 Mobile Client Agent Window Showing Successful Start

Experiencing Problems

We received the following error message when the modem was not switched on during the start of the D&CC agent:

```
Async communications error. Error message type 1 returned after modem
initialization string ATM1 was written to the modem.
```

Where ATM1 was the string we defined in the INIT keyword in our example.

The problem was solved simply by switching on the modem before the start of the Mobile Client.

The next verification step is to connect the server from the Mobile Client. As a simple connection setup we issued the command to check the status of the client at the server with the following result:

```
nvdm stattg

Target                  Status

wtr05148                Not available
```

If you receive the output for the command the connection to dial in from the client is working fine. Notice that the status Not available is correct, because we only connected using the command line interface and no synchronization of the databases is done between the server and the client when connecting with the command line interface. See 3.2.1, “General Information about the Mobile Client” on page 18 for more information.

To verify if the server can dial back to the client and that the synchronization is performed correctly issue the connect command:

```
nvdm connect -d 2
```

Note

As a reminder, these are the parameters that can be specified with the connect command:

```
nvdm connect target -s start -d duration -t phone number
```

target Specifies the target name for which the connection window is set. If not specified the target of the request originator is used.

-s Specifies the start time for the connection window. If omitted the current time is used.

-d Specifies the duration for the connection in minutes. If this parameter is not specified the default duration of 60 minutes is used.

-t Is the telephone number under which the client can be reached. The telephone number is used only with the asynchronous connection.

The Mobile Client will dial out to connect the server and place a connect request at the server. The connection is broken after the request is successfully placed. Shortly after breaking the connection the Mobile Client will receive a dial-in connection from the server. After establishing the dial-in connection the synchronization is executed. Figure 161 shows the messages at the OS/2 Mobile Client during synchronization with the asynchronous protocol. The message New Async connection established on port COM1 indicates that the protocol used to communicate with the server is asynchronous.

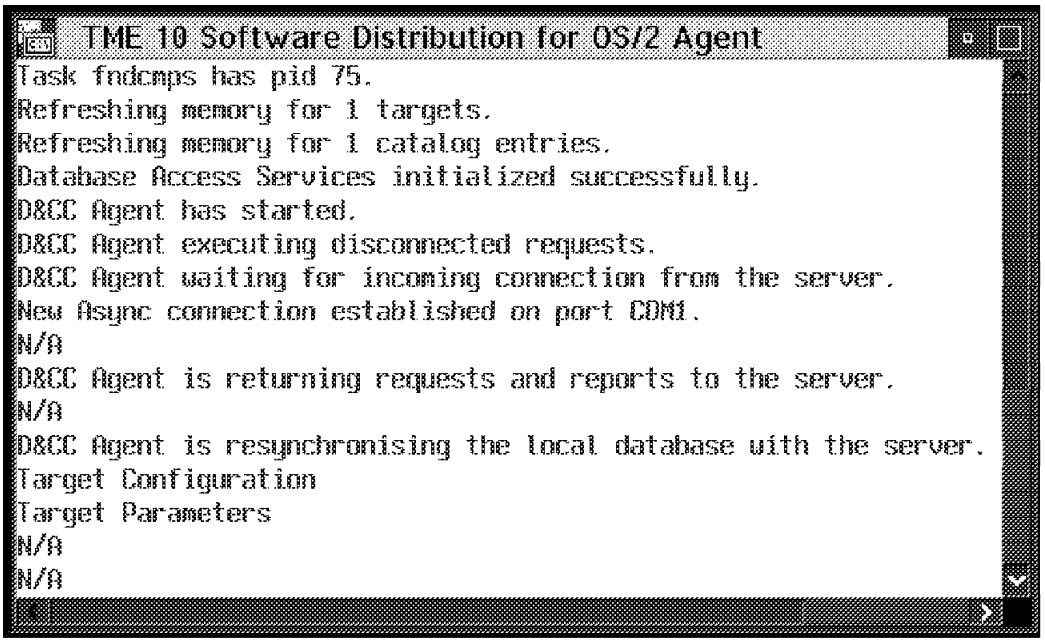


Figure 161. OS/2 Agent Window Showing Synchronization with Asynchronous Protocol

Experiencing Problems

We occasionally received problems that seemed to be related to the modems we were using.

For example, we received the following message in the fndlog of the server while processing a nvdm connect request from the client:

```
FNDRX139E: Failed to receive data packet with sequence number 0. Sending
NAK and retrying.
FNDRX132E: Async communications error when sending data. Packet with
sequence number 0 was not acknowledged: error message type returned
was 64.
```

And in the agent window of the Mobile Client we see the message:

```
Async communications error waiting for an incoming connection on port
COM1. Message type 8 was received.
```

This error occurred after issuing the nvdm connect command from the client. We were able to connect to the server, but then further processing failed with the above errors.

If you discover the same or similar errors do the following to try to recover from the error:

- Stop the Mobile Client with the command nvdm stop
- Switch the modem at the Mobile Client off and back on again
- Restart the Mobile Client with the command nvdm start
- Retry the connect request with the command nvdm connect

In most cases this should help when problems are related to the modem.

If the above steps worked in your environment the configuration for the asynchronous protocol support was successful and the Mobile Client is now ready for change management processing using native asynchronous protocol.

Note

When doing the verification steps described before you should also see the lights of the modem at the client workstation flashing, the lights for transmitting data and receiving data should come on.

If you set the initialization string to ATM1 as shown in our example, you will also be able to hear the dial tone when dialling out to the server or receiving incoming calls from the server.

11.4.2 Connecting the Mobile Client with Changing Telephone Numbers

One major aspect of the Mobile Client is providing travelling users with the software and data they need, whenever and wherever they need it. A travelling user is considered to be a person that travels across the country for his/her business and stays in different locations. Yet this user requires access with his/her laptop PC to the change management server using the telephone connections provided in his/her current location. This could be, for example, a hotel room.

Since the telephone costs are usually higher in hotels it is required that the user is called back from the server in the central department. Thus avoiding long connection times and costs at the users site.

This requires that the user can provide the current telephone number to the server, which uses this number to call back. With the asynchronous connection support the user can specify the telephone number where he/she can be reached in the nvdm connect command. See 11.4.1, “Verifying the Configuration of the Asynchronous Protocol Support” on page 199 for the syntax of the connect command.

The number currently defined at the server for the target is replaced with the number provided in the connect command. If no number is specified in the connect command, the last number used is called during a connect process.

In the following we give an example of the usage of changing the telephone number. Remember that we defined the number 3013473 for our client wtr05148 at the server, when we changed the targets definition to asynchronous in 11.2.5, “Defining Asynchronous Support for the Target at the Server” on page 192. Now let′s assume we moved the workstation to a different location with a different telephone number. To connect from the server to the new telephone number we specify the number in the connect command. The following happens:

- Connect from the Mobile Client to the Software Distribution for AIX server with the command:

  ```
  nvdm connect -t 3313131 -d 2
  ```

  Where 3313131 is the telephone number at the new location of the Mobile Client.
- The Mobile Client dials in to the server and places the connect request together with the new telephone number.
- The server changes the target definition and schedules the open connection request and close connection request. The protocol definition for the target is updated at the server and now holds the new telephone number.
- When the connect requests are scheduled, the connection established by the Mobile Client is broken.
- The open connection request is processed and the connection to the client is established by the server using the new telephone number.
- After establishing the connection the database synchronization is executed and change management processing can be started.

Experiencing Problems

We experienced some problems during the connection process as described here.

The connect command was processed from the Mobile Client with a duration of 2 minutes. The open connection request was executed by the server but the connection could not be immediately opened. This seemed to be because the modem we were using at the client was still holding the connection. We received the message at the server:

```
FNDRX129E: Async communications error. Error message type 1 returned
after modem initialization string AT was written to the modem.
```

The server did manage to establish the connection to the client and started the synchronization process. We see the message at the client:

```
D&CC Agent is returning requests and reports to the server.
```

But the synchronization process is interrupted by the server, because the second open connection request scheduled for internal processing does not get executed anymore since the request is expired. We see the message in the fndlog at the server:

```
FNDSH261E: Open connection window request / report expired.
FNDRQ110W: Execution window for Open connection window request has
expired. Sense data is 0383:0009.
```

To get around this problem specify a connection duration of at least 3 minutes.

Notice there is a known problem when specifying a duration of 1 minute, where the problem behavior is similar to the above description.

You should also consider that the above behavior occurred only when using a modem with a slow baud rate and should not occur with a fast modem.

11.4.3 Inventory Discovery Using Asynchronous Protocol Support

The inventory discovery process for the Mobile Client using the asynchronous protocol does not differ from the inventory process using a Mobile Client with a standard protocol such as TCP/IP. Also see 3.2.3.3, “Data Flow during Automatic Client Registration” on page 36.

For example, the following steps are performed when executing the inventory discovery for the Mobile Client wtr05148 at the server and the connect request from the client:

1. The inventory request is issued at the server by entering:

   ```
   nvdm inv -w wtr05148
   ```

2. The inventory request is scheduled at the server for later processing.
3. A connect to the server with a duration of 5 minutes is executed from the Mobile Client by entering:

   ```
   nvdm connect -d 5
   ```

4. The Mobile Client dials into the server and places the connect request.
5. The connection from the client to the server is broken.

6. Two requests are scheduled at the server:
   - Open connection window
   - Close connection window
7. The open connection request is executed and the server dials out to the Mobile Client using port tty1.
8. The connection from the server to the Mobile Client is established and the database synchronization executed.
9. After the database synchronization the inventory request scheduled earlier gets executed.
10. The FNDSWINV file is read at the client and the history database at the server is updated.

Figure 162 shows the messages displayed at the OS/2 Mobile Client in the agent window during processing of the steps described before.

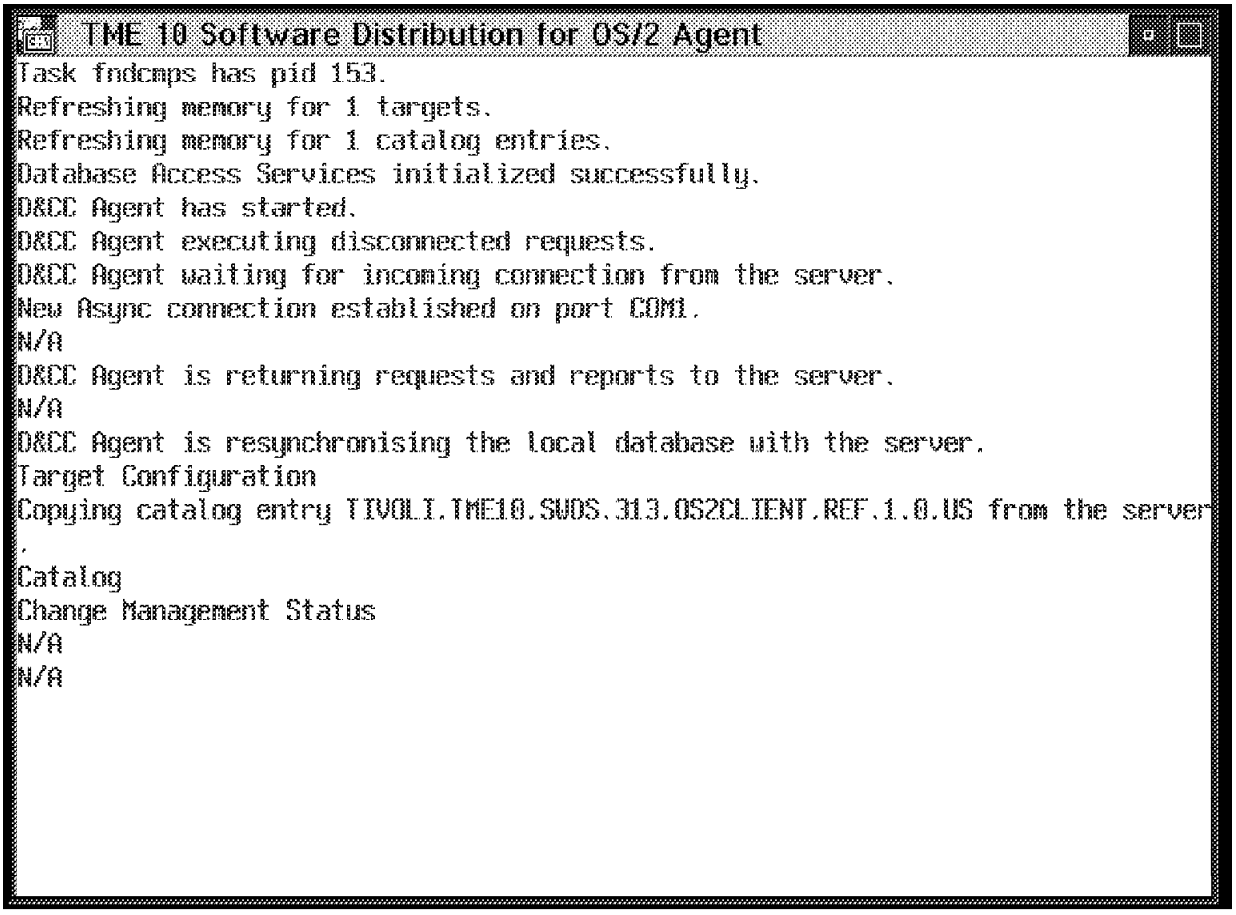


Figure 162. OS/2 Agent Window Showing Inventory Discovery with Asynchronous Protocol

11.4.4 Change Management Processing Using Asynchronous Protocol Support

In this section we show how change management processing works for the Mobile Client with the asynchronous protocol.

The Mobile Client with asynchronous support has the same functionality as a Mobile Client with, for example, TCP/IP protocol. So the change management processing in disconnected and connected mode that we described for the Mobile Client in 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18 and Part 3, “Simple Scenarios Using the Mobile Client” on page 87 is also valid for the Mobile Client with asynchronous protocol.

Since the behavior during processing does not differ much depending on the complexity of the process, we will explain processing using a simple generic change file example. We show the processing for the following functions:

1. Creating a simple generic change file using the command line interface.
2. Installing the change file immediately in connected mode.
3. Removing the change file in local disconnected mode.
4. Installing the change file in deferred disconnected mode.

11.4.4.1 Creating a Simple Generic Change File Using the Command Line Interface

We create a generic change file for the installation of the OS/2 Presentation Manager Screen Capture Program Version 2.12, in the following called PMCAMERA.

We use an ASCII editor on an OS/2 workstation to create a change file profile with the name PMCAM.PRO:

```
GLOBAL NAME:                  OS2.PMCAMERA.REF.1.2.12
DESCRIPTION:                  PM Camera
CHANGE FILE TYPE:             GEN
COMPRESSION TYPE:             LZW
REBOOT REQUIRED:              NO
REMOVABLE:                    YES
ACTIVABLE:                    YES
INTERACTIVE:                  NO
AUTHORIZE:                    NONE
SW HISTORY RESET:             NO
INSTALLATION DURATION:        00:00
COST:                         0
PACK FILES:                   NO
SECURE PACKAGE:               NO
POST-INSTALL:                 D:\PMCAM\PMCAM200.EXE
OBJECT:
    SOURCE NAME:              C:\OS2UTILS\PMCAM200.EXE
    TARGET NAME:              D:\PMCAM\PMCAM200.EXE
    TYPE:                     FILE
    ACTION:                   COPY
    INCLUDE SUBDIRS:          NO
    GENERAL ATTR:             -A----
    UNIX ATTR:                ----------
    NETWARE ATTR:             -------
OBJECT:
    SOURCE NAME:              C:\OS2UTILS\PMCAMERA.INI
    TARGET NAME:              D:\PMCAM\PMCAMERA.INI
    TYPE:                     FILE
    ACTION:                   COPY
    INCLUDE SUBDIRS:          NO
    GENERAL ATTR:             -A----
    UNIX ATTR:                ----------
    NETWARE ATTR:             -------
```

Figure 163. Generic Change File Profile for OS/2 PMCAMERA

Then we use an OS/2 client connected to the AIX server to build the change file and catalog it at the AIX server, with the command:

```
nvdm bld pmcam.pro
```

The change file is now created and cataloged at the Software Distribution for AIX server and we can continue with the installation of the change file.

11.4.4.2 Installing the Change File Immediately in Connected Mode

After the change file was successfully built at the server we start the installation of the change file on the Mobile Client wtr05148 by entering the command at the server:

```
nvdm inst OS2.PMCAMERA.REF.1.2.12 -w wtr05148
```

Since we did not specify a time and the disconnected option the installation request is processed immediately and in connected mode . The immediate installation request is scheduled at the server and waits for a connection to be established to or from the Mobile Client.

In this example we establish the connection from the server after we scheduled the installation request. To initiate the connection setup from the server we enter the command:

```
nvdm connect wtr05148
```

The server dials out to the client with the asynchronous protocol support and establishes the connection. As soon as the Mobile Client is connected the synchronization of the databases is executed.

The immediate installation request is executed after the synchronization is completed. The communication link and the session between the server and the Mobile Client is up during the transfer of the installation files and the installation itself. When the installation completes at the client the report is sent immediately back to the server. The connection between the Mobile Client and the server is now broken.

We check the status of the change file after the installation with the command:

```
nvdm lscm OS2.PMCAMERA.REF.1.2.12 -w wtr05148

Global file name:      OS2.PMCAMERA.REF.1.2.12
       Target:         wtr05148
       Status:         Installed, removable, active
```

11.4.4.3 Removing the Change File in Local Disconnected Mode

To demonstrate the different change management functions of Software Distribution 3.1.3 and the different processing mode for the Mobile Client with asynchronous protocol we now remove the change file we have just installed.

The remove is done as an example in local disconnected mode on the Mobile Client, so no server connection is required to remove the change file. Notice that the change file itself was not stored in the local repository of the client, because we performed an immediate connected installation in the previous step. Refer to 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18 for more information.

Note

For a local disconnected remove or uninstall no change file needs to be stored in the local repository of the Mobile Client. The catalog entry and the change management status is sufficient.

To perform the remove function in local disconnected mode we first need to change the command line interface mode of the asynchronous Mobile Client to disconnected mode. We do this by entering the command:

```
nvdm svr myself
```

You will immediately receive the message that indicates the Mobile Client is now working in disconnected mode:

```
Mobile client now disconnected from server.
```

The remove of the change file OS2.PMCAMERA.REF.1.2.12 is initiated by the command:

```
nvdm rem OS2.PMCAMERA.REF.1.2.12
```

After entering the command a local request is scheduled at the Mobile Client and stored in the local request queue. You should receive an output like the following:

```
Server:              rs600012
Originator:          root-mobile
Sequence number:     1
Recursion:           0
```

The server rs600012 is the default server of the Mobile Client we are using and even though the request is performed disconnected the server name is shown here. The originator indicates that we used the user ID root on a Mobile Client.

As described in 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18, any locally issued request does not get processed on the Mobile Client unless the Mobile Client is restarted or a new connection with the server is established.

You can check that the request is stored in the local request queue with the disconnected command:

```
nvdm lsq
```

The output you receive for the list queue command shows all the requests that are queued locally. In our example we only have one local request queued, so the output looks like this:

```
Command:             Remove
Sequence number:     1
Originator:          wtr05148
Submitted on:        11/07/96
Global file name:    OS2.PMCAMERA.REF.1.2.12
Status:              Ready
Request ID:          rs600012 root-mobile 1 0
```

We stop and restart the Mobile Client in our example to see the request processing happening, with the commands:

```
nvdm stop
nvdm start
```

At startup we can see that the local remove operation is executed as shown in the OS/2 agent window in Figure 164 on page 210.

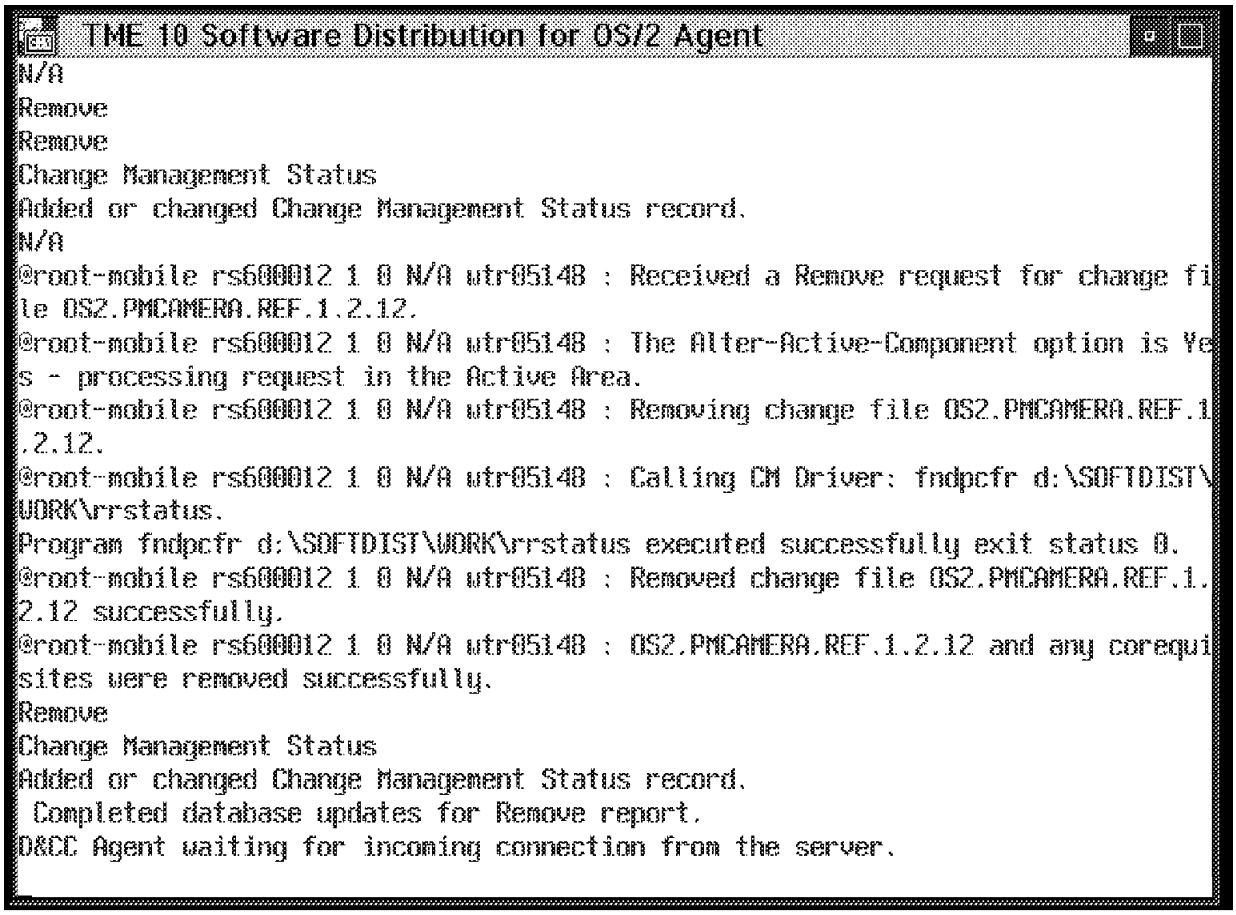


Figure 164. OS/2 Agent Window Showing Execution of Local Remove Operation

To return the status of the operation back to the server we connect the server this time from the Mobile Client with the commands:

```
nvdm svr rs600012
nvdm connect
```

The first command changes the operation mode of the command line interface to server connected. This is required to process the connect command. Refer to 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18 for more information on this.

Notice that we used the default value for the duration here, so the connection window will be open for 60 minutes. This allows us to schedule the disconnected deferred installation described in 11.4.4.4, “Installing the Change File in Deferred Disconnected Mode” on page 211 within the same connection window.

Note

A connection to the server is established with the command nvdm svr rs600012 to change the command line interface mode. A new connection is also established for the nvdm connect command, which then causes the server to dial back the client.

The Mobile Client receives a dial-in connection request by the server and the asynchronous connection is established. The synchronization is executed and the status of the remove operation performed while being disconnected is sent from the Mobile Client to the server. The database at the server is updated accordingly. The change file is now available for the target wtr05148 as seen in the output of the following command:

```
nvdm lscm OS2.PMCAMERA.REF.1.2.12 -w wtr05148

Global file name:      OS2.PMCAMERA.REF.1.2.12
       Target:         wtr05148
       Status:         Available
```

The connection window stays open, but the physical connection is broken. You can check the status of the Mobile Client at the server with the command

```
nvdm stattg wtr05148
```

During the synchronization the command output should look like:

```
Target          Status

wtr05148        Busy
```

And after synchronization is completed the output should show:

```
Target          Status

wtr05148        Available
```

We have now removed the installation performed earlier and show the installation of the same object in disconnected deferred mode in the next section.

Note

When the server and the client finished the synchronization of their databases and any change management processing, the physical communication link is broken.

One indication is the status of the Mobile Client at the server. We can also see at the modem connected to the client that no connection is up.

11.4.4.4 Installing the Change File in Deferred Disconnected Mode

For simplicity we install the same generic change file used in 11.4.4.2, “Installing the Change File Immediately in Connected Mode” on page 207 to demonstrate the deferred disconnected installation using a Mobile Client with asynchronous communication.

We execute the following commands for the disconnected deferred installation of the change file OS2.PMCAMERA.REF.1.2.12 at the server:

```
nvdm inst OS2.PMCAMERA.REF.1.2.12 -w wtr05148 -dc -db 12/1/96
nvdm send OS2.PMCAMERA.REF.1.2.12 wtr05148
```

We send the change file with the installation request since it must be stored locally at the Mobile Client to process the request disconnected from the server. Refer to 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18 for more information. Figure 165 on page 212 shows the messages at the client during request forwarding and storage of the change file.

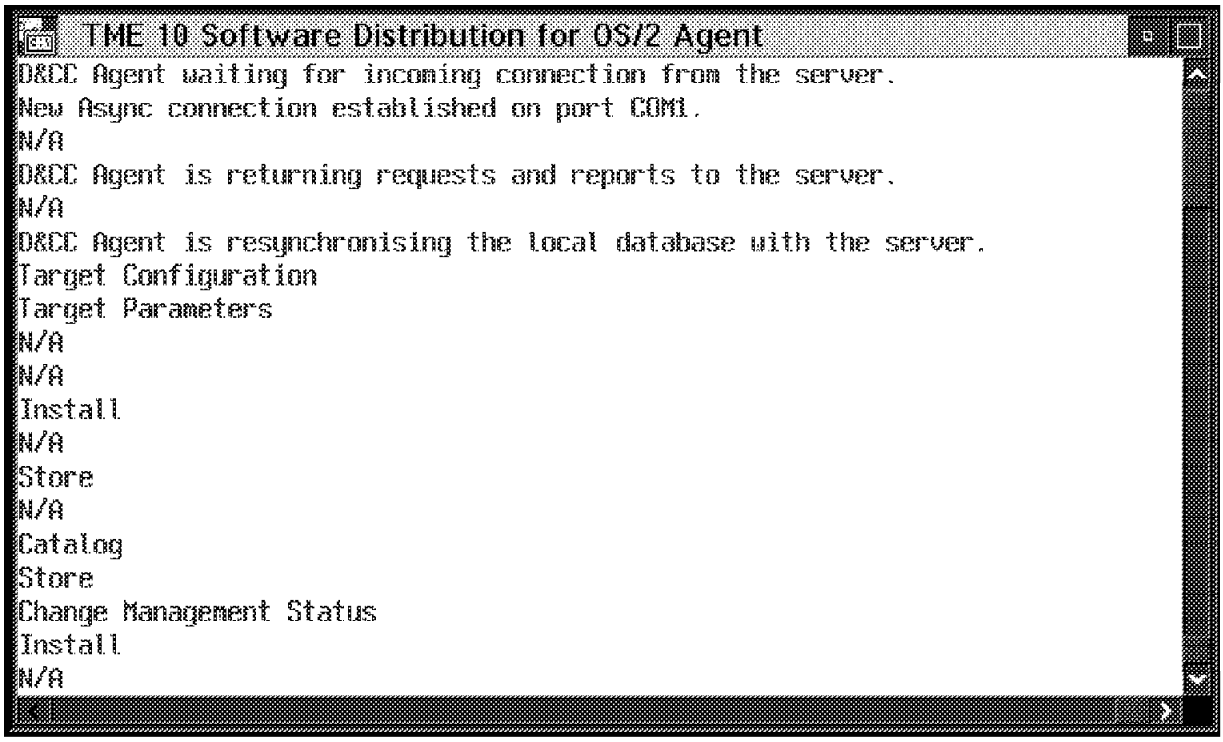


Figure 165. OS/2 Agent Window Showing Forwarding of Disconnected Request

Note

Depending on the size of the change file allow some time to pass for the transfer of the change file to the local repository of the Mobile Client.

Since we defined a connection window with a default duration of 60 minutes in the previous section, the window is still open and the requests are immediately sent to the Mobile Client. The request execution though is delayed until the date specified.

Note

The physical communication link was broken after the synchronization in the previous section was completed. The server automatically dials out to the Mobile Client since the connection window is still open.

No new connect request is required to send the installation request and send the change file if a connection window is still open.

We can see by the lights at the modem and the sound of the dial tone, that a new dial-in connection is received from the server.

To achieve processing of the disconnected request we change the date on the OS/2 Mobile Client to be December 1, 1996 using the DATE command. Then we stop and restart the Mobile Client with the commands:

```
nvdm stop
nvdm start
```

When the D&CC agent of the Mobile Client starts it checks if any requests are pending in its local request queue that are due for processing. In our example the D&CC agent finds the installation request for the object OS2.PMCAMERA.REF.1.2.12. The request is executed immediately by the D&CC agent without a connection to the server. Figure 166 shows the messages of the OS/2 agent when processing the installation.

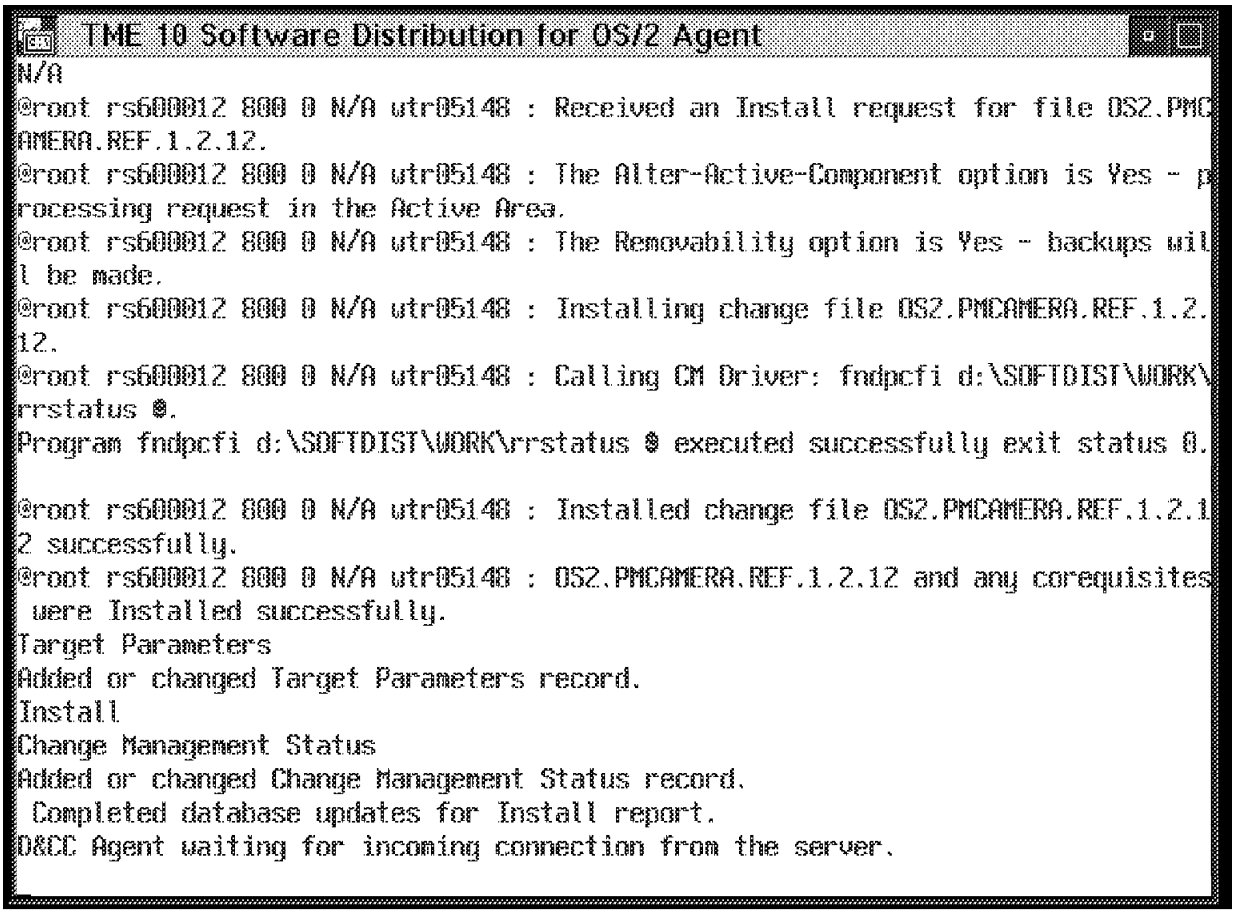


Figure 166. OS/2 Agent Window Showing Processing of Disconnected Request

When a new connection is established between the server and the Mobile Client the status of the installation request is reported to the server. Refer to 3.2, “Data Flow between Software Distribution 3.1.3 Server and Mobile Client” on page 18 for more information.

Chapter 12. Using Change Objects with Mobile Clients

In this section we look at issues relating to change objects and Mobile Clients. We examine how to modify CID change objects to make them more suitable for use with Mobile Clients (for example by removing the use of redirected drives) and also how to use a plan to deal with Mobile Clients as a special case when performing software distribution to a group of machines.

In this section we assume that you are familiar with CID and with the basic operation of TME 10 Software Distribution.

12.1 A Standard Install Scenario

First we create a change object to perform a CID install on an OS/2 client. At this stage we deal with standard Distribution Clients and not Mobile Clients.

We assume the following environment:

- A community of OS/2 PCs with NFS and the Software Distribution for OS/2 Distribution Client installed.
- A Software Distribution for AIX server running NFS. (We used rs600012 as described in other chapters.)
- Drive letters X: and W: reserved on every PC for use by software distribution.

The descriptions that follow should work in other environments just as well, but for this section we take this as being our configuration.

To install our new software the following steps must be performed on each client:

1. CIDMOUNT is executed as a pre-installation step to map X: to directory /nfs/share_a and W: to directory /nfs/share_b. Both directories are on our Software Distribution for AIX server. X: contains our installation program, its source files and response files and is mounted as read-only. W: is the destination for the log files and is mounted as read-write.
2. The installation program is run with the relevant parameters and issues a return code to the Software Distribution for AIX server upon completion. The installation program here is a REXX program that copies files from its source directory to a directory specified in the response file.
3. After installation has finished CIDUNMNT is run as a post-installation step to unmount X: and W:

However, before we can install the change object there are several things to be done.

12.1.1 Preparation

Ensure that you have some form of redirected drive available. We use NFS but any supported form of redirection is acceptable. Create the following directory structure on your NFS server:

© Copyright IBM Corp. 1997

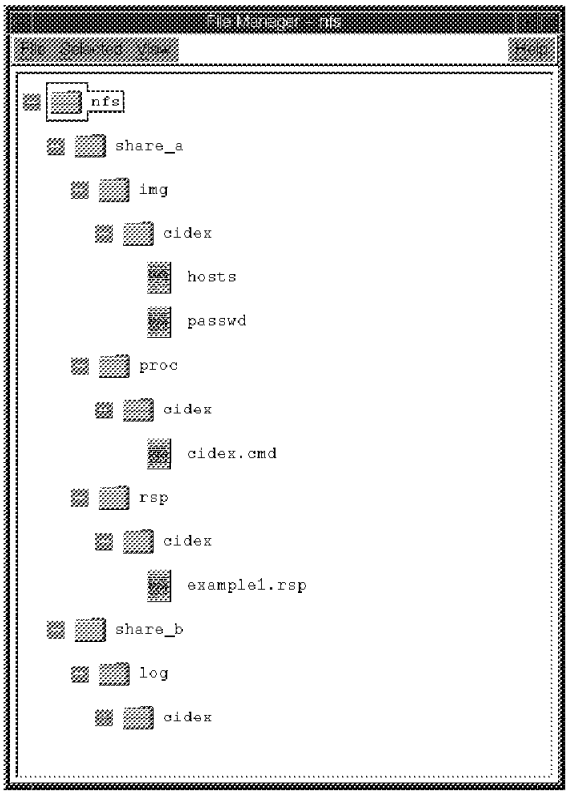


Figure 167. File Server Directory Structure

Create the file /nfs/share_a/proc/cidex/cidex.cmd on the AIX server as shown in Figure 168 on page 217. This will be the installation program.

```
/*********************************************************/
/* CID Installation example                              */
/*                                                       */
/* Mark Guthrie 4th November 1996                        */
/*********************************************************/
/*********************************************************/
/* This program is an example of a CID installation      */
/* program.                                              */
/*                                                       */
/* It expects 2 parameters to be passed.                 */
/*  1) Response file                                     */
/*  2) Logfile directory                                 */
/*                                                       */
/*********************************************************/
/* Function                                              */
/* --------                                              */
/*                                                       */
/* Copies files from the current directory to a          */
/* destination directory specified in the response file  */
/*                                                       */
/*********************************************************/
/* Comments                                              */
/* --------                                              */
/*                                                       */
/* To simplify the program, all error checking has been  */
/* removed.                                              */
/*                                                       */
/* The current directory (Working Directory) is taken    */
/* as the source directory to copy from.                 */
/*********************************************************/

/*********************************/
/* Receive parameters            */
/*********************************/
arg response log

parse value response with '/R:'response_file
parse value log      with '/L:'log_file

/*********************************/
/* Log that we have started      */
/*********************************/
call lineout log_file,'Installation Started at 'time()

/*********************************/
/* Read Response file            */
/*********************************/
line=linein(response_file)
parse value line with 'DESTINATION=' dest
dest=strip(dest)

/*********************************/
/* Copy Files                    */
/*********************************/
'xcopy *.* 'dest'\'

/*********************************/
/* Log that we have finished     */
/*********************************/
call lineout log_file,'Installation Finished at 'time()
exit 0
```

Figure 168. /nfs/share_a/proc/cidex/cidex.cmd

Create the file /nfs/share_a/rsp/cidex/example1.rsp as shown in Figure 169. This is a simple response file.

```
DESTINATION=C:\CIDEX
```

Figure 169. /nfs/share_a/rsp/cidex/example1.rsp

Copy any number of files into the directory /nfs/share_a/img/cidex. Whatever is placed in here is copied to the destination directory at install time. Remember that this is just an example to mimic the behaviors of a real CID program. We created sample files by issuing the following commands from AIX:

```
cp /etc/hosts /nfs/share_a/img/cidex/hosts
cp /etc/passwd /nfs/share_a/img/cidex/passwd
```

If NFS is not already started, then start it by issuing the following command on the AIX server:

```
/usr/sbin/mknfs '-B'
```

Now export the share_a and share_b directories using the commands:

```
/usr/sbin/mknfsexp -d '/nfs/share_a' -t 'ro' '-B'
```

and

```
/usr/sbin/mknfsexp -d '/nfs/share_b' -t 'rw' '-B'
```

You are now ready to create the change object.

12.1.2 Setting Up for Redirected Installs

If this is the first time that you have used CID installs, then you may have to make changes to your client environment. The main requirement is to ensure that CIDMOUNT and CIDUNMNT both work. These are OS/2 CMD files that reside by default in the directory C:\SOFTDIST\BIN. You can test CIDMOUNT for NFS by opening an OS/2 window and typing:

```
CIDMOUNT NFS X: W: rs600012 nfs/share_a nfs/share_b
```

Replace rs600012 with the name of your NFS server. If this command is successful then you should find X: mapped to the directory nfs/share_a on your server and W: similarly mapped to nfs/share_b.

Test the unmount option by entering:

```
CIDUNMNT NFS X: W:
```

We modified CIDMOUNT.CMD to include the user ID and password for the AIX server. Our version is shown in Figure 170 on page 219.

```
@echo off
IF "%1"=="SRVIFS" goto SRVIFS
IF "%1"=="NFS" goto NFS
IF "%1"=="IBMLS" goto IBMLS
IF "%1"=="NWREQ" goto NWREQ
ECHO "BAD COMMAND INVOCATION"
exit 1
:SRVIFS
REM 1=SRVIFS 2=freedrive1 3=freedrive2 4=server 5=cidalias 6=logalias
SRVATTCH %2 \\%4\%5
SRVATTCH %3 \\%4\%6
goto :end
:NFS
REM 1=NFS 2=freedrive1 3=freedrive2 4=server 5=exported_cid 6=exported_log
REM *********************************************
REM ** We modified the following lines to add -lswuser -pswuser
REM **  This causes NFS to use the user ID swuser with a password
REM **  of swuser.
REM *********************************************
MOUNT -lswuser -pswuser %2 %4:%5
MOUNT -lswuser -pswuser %3 %4:%6
REM *********************************************
REM ** End of modifications
REM *********************************************
goto :end
:IBMLS
REM 1=IBMLS 2=freedrive1 3=freedrive2 4=server 5=cidalias 6=logalias
NET USE %2 \\%4\%5
NET USE %3 \\%4\%6
goto :end
:NWREQ
REM 1=NWREQ 2=freedrive1 3=freedrive2 4=server 5=cidalias 6=logalias
MAP %2=%5
MAP %3=%6
:end
```

Figure 170. Modified C:\SOFTDIST\BIN\CIDMOUNT.CMD

Note

Ensure that you have started NFS on OS/2 by issuing the command:

```
NFSSTART
```

12.1.3 Creating the Change Object

Now that you have your CID directory structure, your installation program, image files and response file, and you have checked that NFS will work, you need to build a change object for this installation.

Use a text editor to create the file CIDEX.PRO as shown in Figure 171 on page 220. You can do this on the Software Distribution for AIX server or on a connected client. You will notice that the PREREQ and POSTREQ commands are the same commands that we used to test NFS.

```
GLOBAL NAME:                  CID.EXAMPLE.REF.1.0
DESCRIPTION:                  Test CID Install - Example
LOCAL NAME:                   $(REPOSITORY)\CID.EXAMPLE.REF.1.0
CHANGE FILE TYPE:             OS2CID
COMPRESSION TYPE:             LZW
PREREQ COMMAND:               cidmount NFS X: W: rs600012 nfs/share_a nfs/share_b
POSTREQ COMMAND:              cidunmnt NFS X: W:
REBOOT REQUIRED:              NO
REMOVABLE:                    NO
ACTIVABLE:                    YES
INTERACTIVE:                  NO
AUTHORIZE:                    NONE
SW HISTORY RESET:             NO
INSTALLATION DURATION:        00:00:00
COST:                         0
INSTALL PROGRAM:
    PROGRAM NAME:             x:\proc\cidex\cidex.cmd
    PARAMETERS:               /R:x:\rsp\cidex\example1.rsp /L:w:\log\cidex\test.log
    WORKING DIRECTORY:        x:\img\cidex
```

Figure 171. Example Profile CIDEX.PRO

From the command prompt enter:

```
NVDM BLD CIDEX.PRO
```

(Note that on AIX this must be entered in lowercase.)

Tip

In general it is a good idea to keep all of your profiles in a common directory, possibly on the TME 10 Software Distribution server. This means you can quickly find examples to copy to create new profiles.

You have now built and cataloged your change file. To test it enter:

```
NVDM INST CID.EXAMPLE.REF.1.0 -w testpc -n
```

replacing testpc with the name of your OS/2 client.

Note

The parameter -n is required to install it as non-removable. We did not create a section in our profile to cover removing the object, the installation will fail if removable is specified. Remember that removable is the default.

Now that we have our basic environment in place, we can introduce Mobile Clients.

12.2 Installing on a Mobile Client

If you install the object that you created in 12.1, “A Standard Install Scenario” on page 215 onto a LAN-connected Mobile Client then you will see the behavior described in Part 3, “Simple Scenarios Using the Mobile Client” on page 87, with one exception: if the client is not connected to the LAN when the installation takes place, as for example in the case of deadline activation, then the install will fail because it will not be possible to mount the redirected drives. If the

client is connected over PPP as described in Chapter 9, “Using the Mobile Client with PPP” on page 131, then the install *may* work. The reason it may fail is that access to the redirected files is now over a slow link. The installation program may timeout while copying files since it will expect fast access to them. (In our simple example this would not be a problem but in general it might.)

A further issue is that installation programs, which assume that the files are local, do not worry about efficient use of network bandwidth and often read files at the start of an install and at the end. For example, DLLs that the install program uses itself and also copies to the installation directory, are copied over the modem link twice.

So in general we want to re-engineer our installation to effectively target Mobile Clients. How we tackle this depends on factors such as:

- Speed and cost of connections
- Amount of notice given for deadline activations
- Whether the client connects to the LAN or is always modem connected
- Ratio of Mobile Clients to standard clients

When the Mobile Client is LAN-connected before the installation, and we know in advance that a change object must be installed, then we can move the required files to the client over a LAN connection. On the other hand, if the client will only ever connect over a modem, then we should look to create a package that has the bare minimum of required files and transfer this in a timely fashion across the modem link.

The general approach to installing CID software onto a Mobile Client is to create an additional change object that contains the code images and response file and to install this onto the Mobile Client before installing the CID software in a CID manner using the local drives rather than redirected drives.[1] There are several issues to bear in mind when doing this.

Software packages often contain several components of which only a subset are installed. Which packages are installed is usually controlled by the response file or by user input. If it is known in advance that only certain packages are required, then only the images required for those packages should be copied to the Mobile Client. Even if the connection speed is not a problem, laptops rarely have abundant disk space and so minimizing the space taken by software images is a good idea.

In a similar vein, a mechanism needs to be put in place to remove the code images once a package has been successfully installed. It may be required to recover the log files from the laptop later.

This would be an ideal time to remove the image files also.

Before we look at a concrete example based upon the CID installation that was introduced in 12.1, “A Standard Install Scenario” on page 215, we try to bring these points together into a chart.

---

[1] The reader with experience of fan-out, or focal-point connected servers, will see several similarities here. In both cases the central problem is that TME 10 Software Distribution cannot know what a CID program will do when it executes, hence it cannot provide the resources locally at execution time that the CID installation needs. It is because the Mobile Client has a catalog the same as a server that it experiences similar problems.

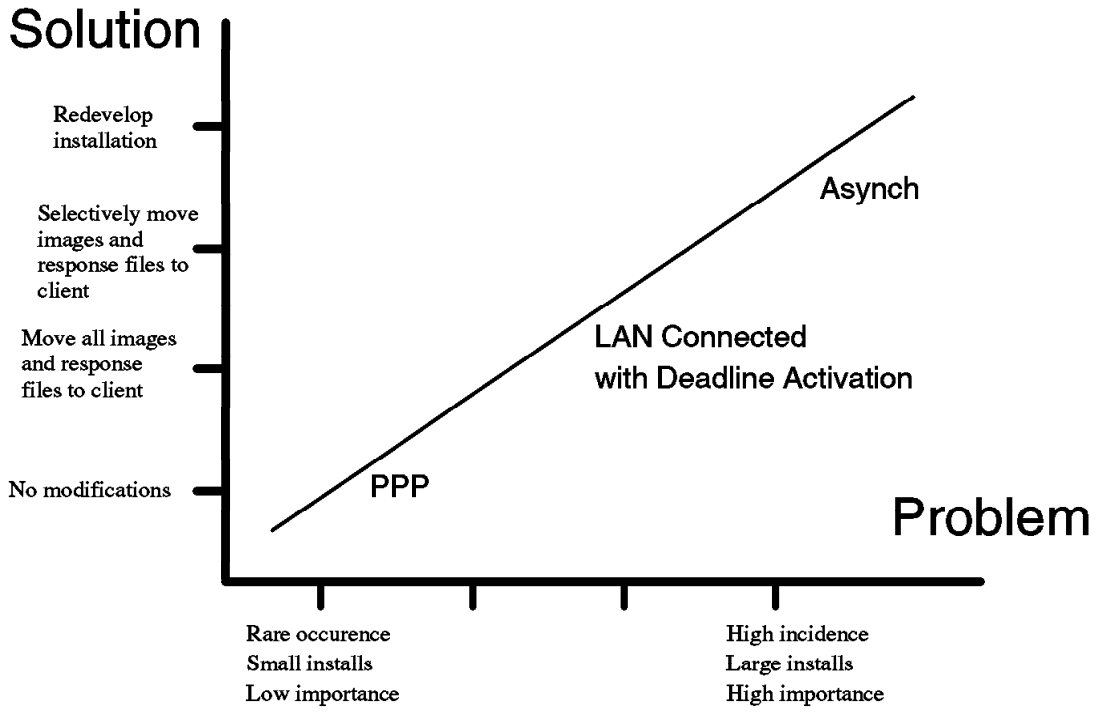


Figure 172. CID Installations with Mobile Clients

Considering the mixed case, where we have desktop and laptop users, we want to minimize the amount of additional work that has to be done to support adding the Mobile Clients to the desktop community. Figure 172 has two axes. Along the X-axis we have the problem which increases from minor as defined to be low business impact, small number of laptops involved and small size images, up to major which is typified by any case where the business impact is great, the software is large or a significant number of users are involved.

We match this on the Y-axis by the solution which can range from doing nothing where it causes a minor impact to a few users, up through moving the code images to the laptop, then moving only those image files which are actually required, until finally we end with a customized installation technique. The latter option should not be ruled out, since even for CID-enabled products it may be more efficient to use Disk Camera or a similar approach to create a new installation than to understand everything that the CID installation is doing. Along the line in the chart we move from PPP through the typical deadline activation approach and end at asynchronous connections.

This chart is only a discussion point; the actual solutions that you use will have to be based on what is the best option to solve your particular problem.

12.2.1 Example Installation

Now we look at one solution for the example that was introduced in 12.1, “A Standard Install Scenario” on page 215. To implement this for Mobile Clients we want to minimize the amount of changes that are made. In this case the installation program is simple, but in general this will not be the case. We also developed the installation program here as an example but for most CID installs, this program is delivered to us as an executable and cannot be changed. We therefore leave the installation program untouched and make our changes in the change object.

The changes we need to make are fairly easy, but will result in us having two copies of the change object that effectively do the same thing. Both objects (the one for desktop machines and the one for Mobile Clients) must have different names since they reside in the same catalog. We modify only the version field of the change object name so that both entries can be seen to be the same thing. We discuss later whether this is a good idea or not. The original object was called:

```
CID.EXAMPLE.REF.1.0
```

We will now create a new object called:

```
CID.EXAMPLE.REF.1.MOB
```

which will perform the installation, and an object called:

```
CID.EXAMPLE.REF.1.PRE_MOB
```

that will copy the required files to the Mobile Client before installation. To ensure that the files are installed before we try to install our CID object we use pre-requisite checking.

Create a profile as shown in Figure 173. This is a modified version of the profile that we used above.

```
GLOBAL NAME:                  CID.EXAMPLE.REF.1.MOB
DESCRIPTION:                  Test CID Install - Mobile Version
LOCAL NAME:                   $(REPOSITORY)\CID.EXAMPLE.REF.1.MOB
CHANGE FILE TYPE:             OS2CID
COMPRESSION TYPE:             LZW
REBOOT REQUIRED:              NO
SOFTWARE PREREQUISITE:        CID.EXAMPLE.REF.1.PRE_MOB
REMOVABLE:                    NO
ACTIVABLE:                    YES
INTERACTIVE:                  NO
AUTHORIZE:                    NONE
SW HISTORY RESET:             NO
INSTALLATION DURATION:        00:00:00
COST:                         0
INSTALL PROGRAM:
 PROGRAM NAME:                c:\admin\share_a\proc\cidex\cidex.cmd
 PARAMETERS: /R:c:\admin\share_a\rsp\cidex\example1.rsp /L:c:admin\share_b\log\cidex\test.log
 WORKING DIRECTORY:           c:\admin\share_a\img\cidex:exmp.
```

Figure 173. Example Profile MOBCIDEX.PRO

You will notice that now we are using local drives rather than network drives.

From the command prompt, build this by entering:

```
NVDM BLD MOBCIDEX.PRO
```

Now create the profile shown in Figure 174. This will build the change object that is specified as a pre-requisite for the mobile version of CIDEX that you have just created. Notice that we are creating a directory on the Mobile Client for administrative purposes. In this directory we create SHARE_A and SHARE_B directories to mimic the structure on the server. We copy everything under the CIDEX directories on SHARE_A to the the Mobile Client but only create the directory for SHARE_B to hold the log file. By using the INCLUDE SUBDIRS: YES parameter we will pick up the files that are present when we build this object. We do not need to specify which files to include. Note that this will copy the files that are present when this object was created and it will need to be re-built if we change any of the files in the directory. For the original CID installation, it picked up the files that were there when it executed.

```
GLOBAL NAME:                CID.EXAMPLE.REF.1.PRE_MOB
DESCRIPTION:                Pre-requisite copy step
CHANGE FILE TYPE:           GEN
COMPRESSION TYPE:           LZW
REBOOT REQUIRED:            NO
REMOVABLE:                  NO
ACTIVABLE:                  YES
INTERACTIVE:                NO
AUTHORIZE:                  NONE
SW HISTORY RESET:           NO
COST:                       0
PACK FILES:                 NO
SECURE PACKAGE:             NO

OBJECT:
    SOURCE NAME:            x:\proc\cidex\*.*
    TARGET NAME:            c:\admin\share_a\proc\cidex\*.*
    TYPE:                   FILE
    ACTION:                 COPY
    INCLUDE SUBDIRS:        YES

OBJECT:
    SOURCE NAME:            x:\img\cidex\*.*
    TARGET NAME:            c:\admin\share_a\img\cidex\*.*
    TYPE:                   FILE
    ACTION:                 COPY
    INCLUDE SUBDIRS:        YES

OBJECT:
    SOURCE NAME:            x:\rsp\cidex\*.*
    TARGET NAME:            c:\admin\share_a\rsp\cidex\*.*
    TYPE:                   FILE
    ACTION:                 COPY
    INCLUDE SUBDIRS:        YES

OBJECT:
    TARGET NAME:            c:\admin\share_b\log\cidex
    TYPE:                   DIRECTORY
    ACTION:                 CREATE
    INCLUDE SUBDIRS:        YES
```

Figure 174. Example Profile PREMOB.PRO

If you build this object from a PC, then ensure that X: is mounted as rs600012:/nfs/share_a (change rs600012 to the name of your Software Distribution for AIX server). If you build it from the AIX server then change X:\????\cidex*.* to /nfs/share_a/????/cidex, where ???? represents the directories rsp, img and proc.

Now from the command prompt build this by entering: NVDM BLD PREMOB.PRO There are several ways to install these objects onto your Mobile Client. We do this by using deadline activation with a LAN connection as an example. To do this we need to:

1. Schedule the immediate installation of CID.EXAMPLE.REF.1.PRE_MOB.
2. Send CID.EXAMPLE.REF.1.MOB to the Mobile Client.
3. Schedule the delayed installation of CID.EXAMPLE.REF.1.MOB.
4. Open a connection window to the client to allow the requests to flow.
5. Wait until the specified installation date.

To do this enter the following commands from the server or a connected client:

```
NVDM INST CID.EXAMPLE.REF.1.PRE_MOB -w mobile1 -n

NVDM SEND CID.EXAMPLE.REF.1.MOB mobile1 -w rs600012

NVDM INST CID.EXAMPLE.REF.1.MOB -w mobile1 -n -i -dc -db "01-01-97"

NVDM CONNECT mobile1 -d 10
```

Replace mobile1 with the name of your Mobile Client, and rs600012 with the name of your server.

Note that we have to specify -i for the installation of CID.EXAMPLE.REF.1.MOB. This option tells TME 10 Software Distribution to ignore the current status of the object and to check at installation time. Remember that we specified that this was not to be installed until after CID.EXAMPLE.REF.1.PRE_MOB, which at the time of issuing the installation request has only been scheduled and not yet processed.

After the requests have been processed you should see the following on your Mobile Client:

```
C:\>date
Current date is: Wed  6-11-1996
Enter the new date: (dd-mm-yy)

C:\>dir c:\admin

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of C:\admin

.            <DIR>      6-11-96  10:03
..           <DIR>      6-11-96  10:03
SHARE_A      <DIR>      6-11-96  10:03
SHARE_B      <DIR>      6-11-96  10:03
        4 file(s)          0 bytes used
                       58740736 bytes free

C:\>dir c:\cidex

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of C:\

SYS0002: The system cannot find the file specified.


C:\>dir c:\softdist\repos

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of c:\softdist\repos

24-10-96  10:26      <DIR>           0  .
24-10-96  10:26      <DIR>           0  ..
 6-11-96  10:06      18076           0  CID.EXAMPLE.REF.1.MOB
        3 file(s)      18076 bytes used
                       58740736 bytes free
```

Figure 175. Status Mobile Client before Activation Date

If you now set the system date on the Mobile Client to January 1, 1997 and reboot the system, this will simulate the machine being switched on at the activation date for this request.[2]

The Mobile Client will now look like:

[2] The reboot is not strictly required but adds greatly to the effect. If you wish you can simply enter:
NVDM STOP & NVDM START

```
C:\>date
Current date is: Mon  1-01-1996
Enter the new date: (dd-mm-yy)

C:\>dir c:\admin

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of C:\admin

.            <DIR>      6-11-96  10:03
..           <DIR>      6-11-96  10:03
SHARE_A      <DIR>      6-11-96  10:03
SHARE_B      <DIR>      6-11-96  10:03
        4 file(s)          0 bytes used
                       58744132 bytes free

C:\>dir c:\cidex

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of C:\cidex

.            <DIR>      6-11-96  10:07
..           <DIR>      6-11-96  10:07
HOSTS            1459   5-11-96   9:19
PASSWD           2881   5-11-96   9:19
        4 file(s)       4340 bytes used
                       58744132 bytes free



C:\>dir c:\softdist\repos

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of c:\softdist\repos

24-10-96  10:26      <DIR>           0  .
24-10-96  10:26      <DIR>           0  ..
 6-11-96  10:06      18076           0  CID.EXAMPLE.REF.1.MOB
        3 file(s)      18076 bytes used
                       58744132 bytes free
```

Figure 176. Status Mobile Client after Activation Date

The software has been successfully installed on the date requested. There are still some points to consider, however. We are left with the change object itself and all of the files that we installed to allow the change to happen. These are occupying valuable space on the laptop. The log file is also sitting on the Mobile Client and cannot be seen from the server.

Our use of the version field in the change object name has a side effect. If you look in the FNDLOG in the directory C:\SOFTDIST, you will see the messages:

```
Program 'cidex.cmd' terminated successfully with return code '0000'.
Secondary effect of Install request:
  change file CID.EXAMPLE.REF.1.PRE_MOB no longer installed on target.
```

Figure 177. Abbreviated FNDLOG Entries

This means that TME 10 Software Distribution thinks that we have installed a different version of the same software and has dropped the catalog entry for the old version. Whether you like this or not will depend on how you use your catalog. The other option is to add a part to the name and create two new catalog entries for the mobile object and its pre-requisite. You could for example use the names:

```
CID.EXAMPLE.DESKTOP.REF.1.0
CID.EXAMPLE.PREMOBILE.REF.1.0
CID.EXAMPLE.MOBILE.REF.1.0
```

The advantage of this approach is that you can then automatically remove the pre-requisite object by installing it without the -n flag and then removing it after the installation of the CID program. However, if you are also interested in the log file and wish to remove the repository file from the Mobile Client, then you will have to use a post-installation step anyway which can delete the SHARE_A files.

If you do create a post-installation step, then you can choose to run this as a change object in its own right or as part of the installation by using the POSTREQ COMMAND value of the profile. Figure 178 on page 229 shows a REXX program which can take care of the tidy up for you. In Figure 179 on page 230 you will see the modified profile to include this step.

```
/*********************************************************/
/*     Example REXX Routine to remove image files and    */
/*     repository entry                                  */
/*                                                       */
/*     Note that because it written in REXX this file    */
/*     can delete itself                                 */
/*                                                       */
/*     There is not enough error checking to             */
/*     use this in a production environment.             */
/*     This is only an example.                          */
/*                                                       */
/*                                                       */
/*                                                       */
/*********************************************************/
/*                                                       */
/* Mark Guthrie 4th November 1996                        */
/*********************************************************/

/***********************************/
/* set up variables                */
/***********************************/
signature='c:\cidex\hosts'
repos='CID.EXAMPLE.REF.1.MOB'
remove1='C:\ADMIN\SHARE_A\PROC\CIDEX'
remove2='C:\ADMIN\SHARE_A\IMG\CIDEX'
remove3='C:\ADMIN\SHARE_A\RSP\CIDEX'

file=stream(signature,'c','Query exists')
if file='' then do
 say 'Product not installed.'
 say 'Not deleting installation files'
 exit -1
end

'echo y> c:\y'

'del 'remove1'<c:\y'
'del 'remove2'<c:\y'
'del 'remove3'<c:\y'
'rd 'remove1
'rd 'remove2
'rd 'remove3

'del c:\softdist\repos\'repos

exit 0
```

Figure 178. Example Post-Installation File POSTCIDX.CMD

```
GLOBAL NAME:                  CID.EXAMPLE.REF.1.MOB
DESCRIPTION:                  Test CID Install - Mobile Version
LOCAL NAME:                   $(REPOSITORY)\CID.EXAMPLE.REF.1.MOB
CHANGE FILE TYPE:             OS2CID
COMPRESSION TYPE:             LZW
POSTREQ COMMAND:              C:\ADMIN\SHARE_A\PROC\CIDEX\POSTCIDX.CMD
REBOOT REQUIRED:              NO
SOFTWARE PREREQUISITE:        CID.EXAMPLE.REF.1.PRE_MOB
REMOVABLE:                    NO
ACTIVABLE:                    YES
INTERACTIVE:                  NO
AUTHORIZE:                    NONE
SW HISTORY RESET:             NO
INSTALLATION DURATION:        00:00:00
COST:                         0
INSTALL PROGRAM:
    PROGRAM NAME:             c:\admin\share_a\proc\cidex\cidex.cmd
    PARAMETERS: /R:c:\admin\share_a\rsp\cidex\example1.rsp /L:c:admin\share_b\log\cidex\test.log
    WORKING DIRECTORY:        c:\admin\share_a\img\cidex:exmp.
```

Figure 179. Modified MOBCIDEX.PRO

Note

If you use this profile, you must first rebuild CID.EXAMPLE.REF.1.PRE_MOB and install it so that the file POSTCIDX.CMD is installed onto drive C.

12.3 Dynamic Change Files

So far our discussions have revolved around CID change objects. Much of the software that an administrator is called upon to install consists of simple files to be copied to a directory. Here a generic change file is simpler to create and use than a CID change file. It is also more portable since no operating system specific installation file has to be invoked. (Remember that currently CID installations are only possible on OS/2.)

The Mobile Client has been developed primarily for laptops. These have different hardware from desktop PCs and often software is installed differently onto laptops. It is not uncommon to be presented with the option of a standard install, a custom install and a minimal install, with the comment that the minimal install suits laptop users.

It is therefore apparent that we need a way to install the same software differently if the target is a Mobile Client. This can be achieved using dynamic change files.

A dynamic change file is a change file that at installation time can check certain values and perform alternative installations dependant on those values.

We will demonstrate this with an example.

Imagine that you have desktop PCs and laptops. On the laptops you run the Mobile Client. You are going to build a change object to install your company's security guidelines package onto every PC. The security guidelines consists of

three text files. One file contains the common guidelines and is to be installed on all PCs, one file is specific to desktop security, and the last file covers laptops.

You have these files on your AIX system in a directory called /sec, as:

GENERAL.TXT The general security file. For all PCs

DESKTOP.TXT The desktop security file. For all desktop PCs only

LAPTOP.TXT The laptop security file. For all laptop PCs only

To install these files you need to build a change object using either the graphical user interface as described in 12.3.1, “Building the Dynamic Change Object from AIX,” or the command line as described in 12.3.2, “Building the Dynamic Change File from a Profile” on page 235.

12.3.1 Building the Dynamic Change Object from AIX

Start the TME 10 Software Distribution graphical user interface on AIX and display the catalog window. From the menu bar choose **Catalog**, **Change File**, **Create New** and **Refresh...**. This will display the Change File Type window.

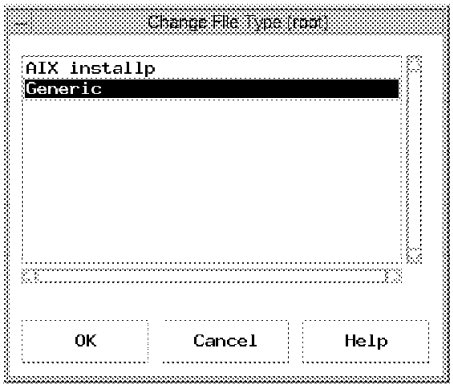


Figure 180. Change File Type Window

Select **Generic** and click on **OK**. You will now see the Change File window as shown in Figure 181 on page 232.

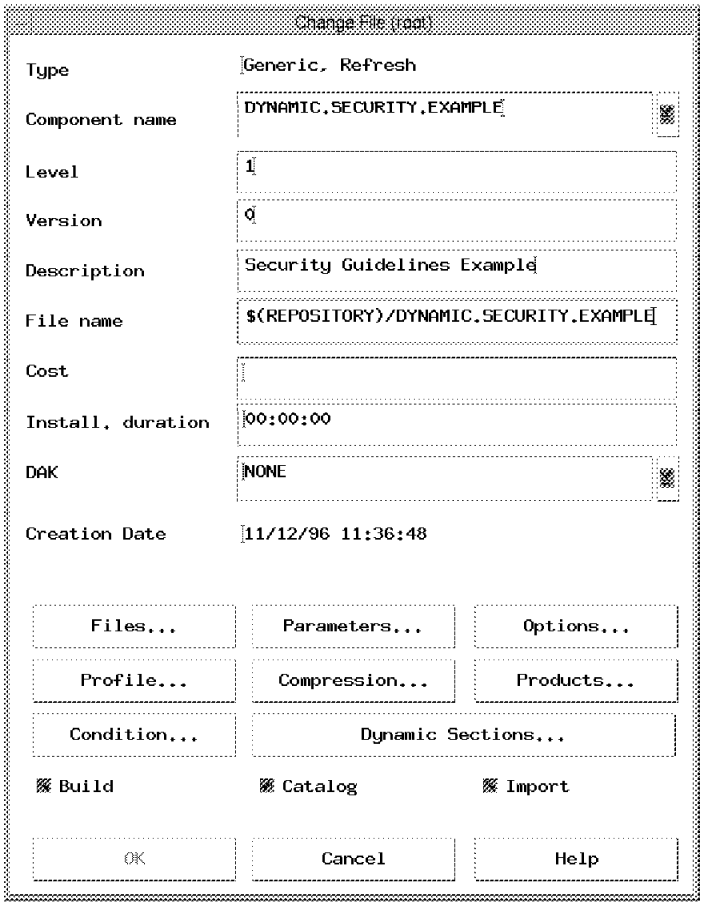


Figure 181. Change File Window

Enter the component name as DYNAMIC.SECURITY.EXAMPLE. Fill in the other fields as appropriate. Click **Dynamic Sections...** to display the Dynamic Sections window as shown in Figure 182.

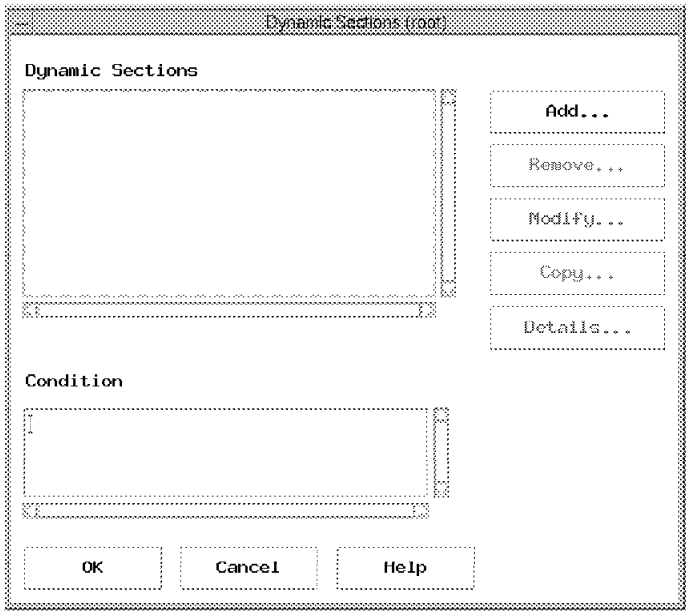


Figure 182. Dynamic Sections Window

Click on the **Add...** push button to add an entry. This will bring up the Add Dynamic Section window.

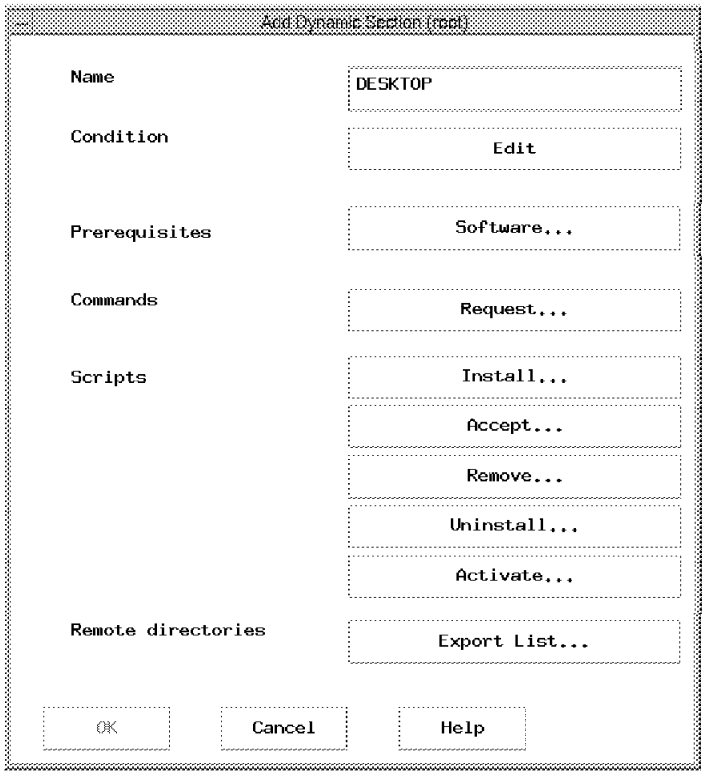


Figure 183. Add Dynamic Section Window

Enter DESKTOP as the name for this section and click **Edit** to create a condition for execution of this section. This will display the Edit CF Condition window.

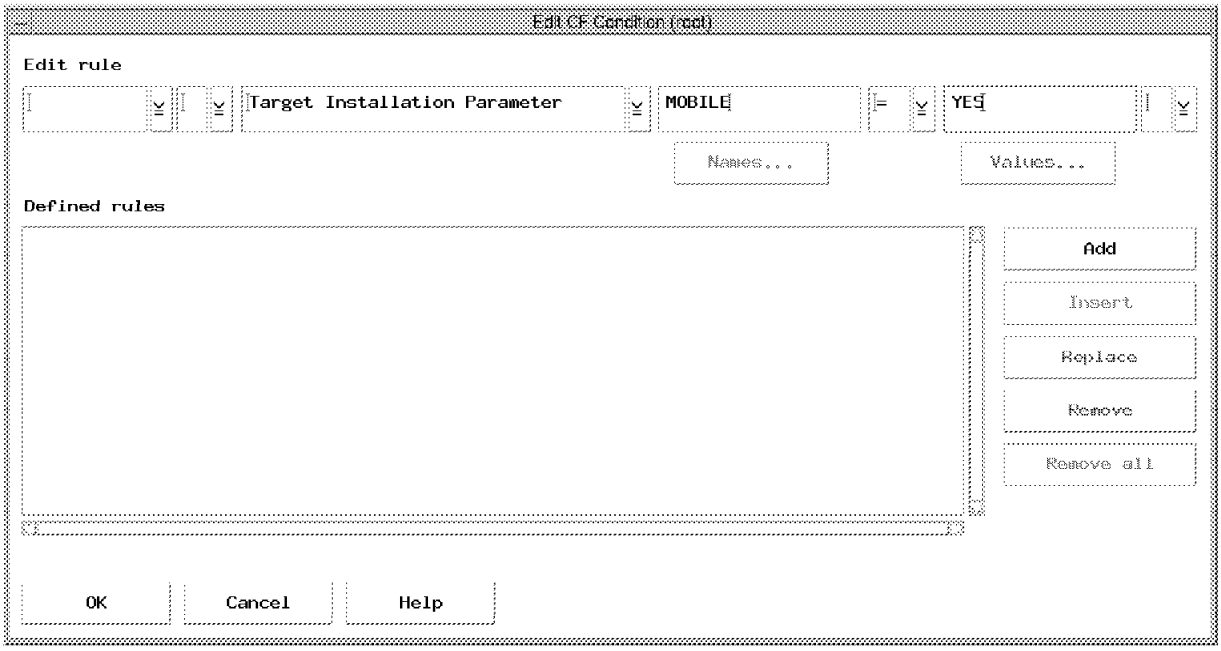


Figure 184. Edit CF Condition Window

Fill in the fields as shown and click **Add** to add this rule. Now click **OK** to close this window, and then **OK** again to close the Add Dynamic Section window.

Repeat this to create a dynamic section called LAPTOP with the condition that the installation target MOBILE does not equal YES. Once you have done this click **OK** to close the Dynamic Sections

You will now be back at the Change File window as you saw in Figure 181 on page 232. Click on **Files...** to bring up the File and Directories in the Change File window.

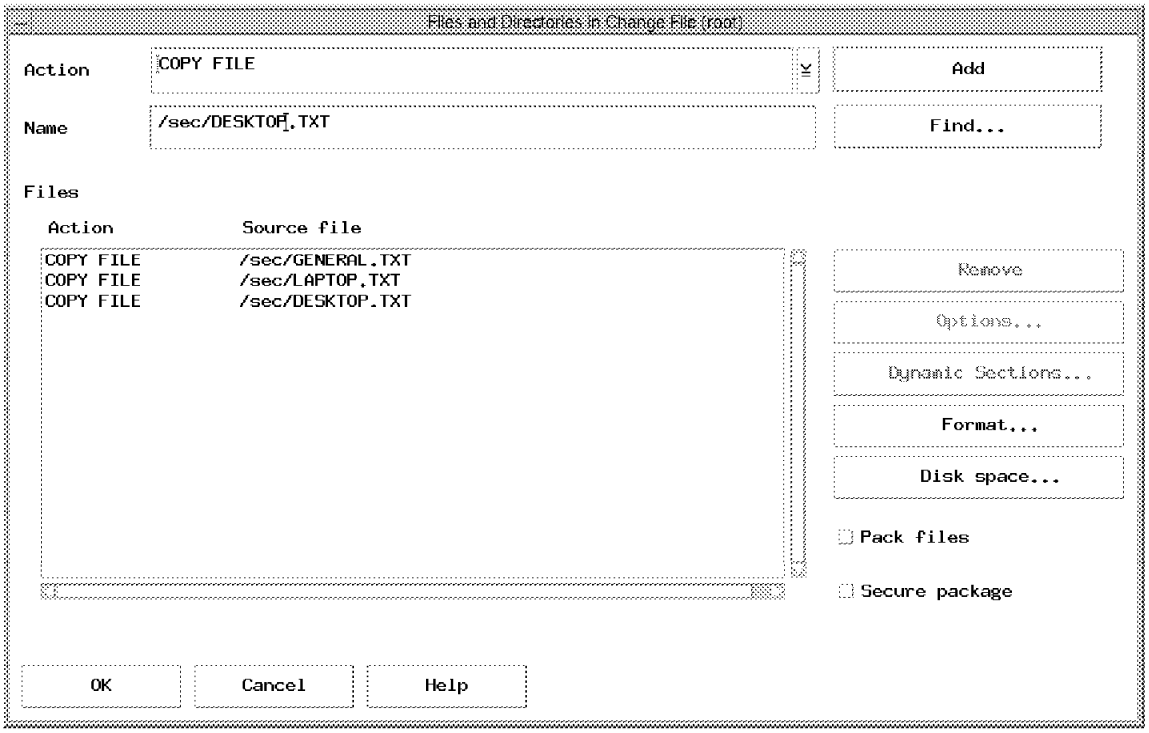


Figure 185. File and Directories in Change File Window

Enter the three file names as shown into the name field and click **Add** to add them to the Files panel. Now select **/sec/DESKTOP.TXT** and click the **Dynamic Sections...** push button.

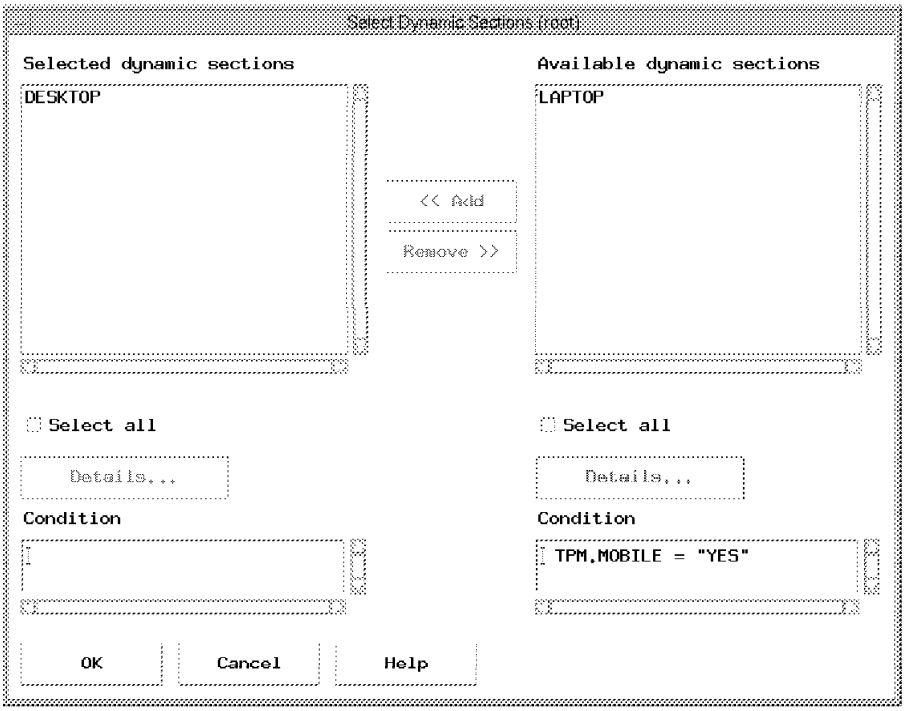


Figure 186. Select Dynamic Sections Window

Choose the **DESKTOP** section and click **OK** to make the change.

Repeat this to link the installation of /sec/LAPTOP.TXT to the dynamic section LAPTOP. Leave /sec/GENERAL.TXT as it is.

Now click **OK** to return to the Change File window and then click **OK** again to build the change object.

12.3.2 Building the Dynamic Change File from a Profile

Warning

At the time this redbook was written building the change file from a profile would not work. If you attempt to build the profile then you will receive the message ″MEMORY ERROR″. You must do this using the graphical user interface as described in 12.3.1, “Building the Dynamic Change Object from AIX” on page 231, above. Also note that if you export the profile from the graphical user interface the file that is created is invalid and you cannot use it without modification.

To build the same change object from a profile, create the file shown in Figure 187 on page 236.

```
GLOBAL NAME:                    DYNAMIC.EXAMPLE.REF.1.0
DESCRIPTION:                    xxxx
LOCAL NAME:                     $(REPOSITORY)\DYNAMIC.EXAMPLE.REF.1.0
CHANGE FILE TYPE:               GEN
COMPRESSION TYPE:               LZW
DEFAULT TOKEN:                  THW.MOBILE(ALL) = "NO"
REBOOT REQUIRED:                NO
REMOVABLE:                      YES
ACTIVABLE:                      YES
INTERACTIVE:                    NO
AUTHORIZE:                      NONE
SW HISTORY RESET:               NO
INSTALLATION DURATION:          00:00:00
COST:                           0

# This is the dynamic part for desktop PCs
# It sets the conditions for execution of the
# section DESKTOP below.
DYNAMIC SECTION:                DESKTOP
    CONDITION:
 TPM.MOBILE != "YES"

# This is the dynamic part for Laptops
# It sets the conditions for execution of the
# section LAPTOP below.
DYNAMIC SECTION:                LAPTOP
    CONDITION:
 TPM.MOBILE = "YES

# This is installed for every machine
OBJECT:
    SOURCE NAME:                /sec/GENERAL.TXT
    TARGET NAME:                C:\SEC\GENERAL.TXT
    TYPE:                       FILE
    ACTION:                     COPY

# This is for desktops only
OBJECT:
    DYNAMIC SECTION:            DESKTOP
    SOURCE NAME:                /sec/DESKTOP.TXT
    TARGET NAME:                C:\SEC\DESKTOP.TXT
    TYPE:                       FILE
    ACTION:                     COPY

# This is laptops only
OBJECT:
    DYNAMIC SECTION:            LAPTOP
    SOURCE NAME:                /sec/LAPTOP.TXT
    TARGET NAME:                C:\SEC\LAPTOP.TXT
    TYPE:                       FILE
    ACTION:                     COPY
```

Figure 187. Example Profile DYNAMIC.PRO

This should be built from your AIX system since you are including files from the /sec directory. Build this change file using the command:

```
nvdm bld DYNAMIC.PRO
```

12.3.3 Using the Dynamic Change File

When you install this change object, it will check to see if the token value MOBILE is set to YES. If it is then the laptop security guidelines are installed, otherwise the desktop guidelines are used.

This value is not set automatically. You must set it yourself for all of the Mobile Clients by issuing the command:

```
NVDM ADDPM mobile1 -i MOBILE=YES
```

Here you replace mobile1 with the name of your Mobile Client. Alternatively you can use a group name, which we discuss later in 12.4, “Using Plans and Groups” on page 238. You can see what tokens are defined by entering:

```
NVDM LSTG mobile1 -l
```

This will display something similar to Figure 188.

```
Target:                 mobile1
Description:            AUTO REGISTERED
Customer name:
Contact name:
Telephone number:
Manager:
Mailing address:
Target access key:      (none)
Mode:                   Push
Server name:            rs600012
Type:                   MOBILE CLIENT
Operating system:       OS/2
Target address:         92410455
Domain address:         RS600012
LAN address:
CM window:              00:00:00 - 23:59:59
Connection window:      Not defined
Network:                TCP mobile1
Logging level:          Normal
Tracing state:          Off
Installation parms:     BOOTDRIVE=C:
                        FREEDRIVE1=D:
                        FREEDRIVE2=E:
                        FREEDRIVE3=F:
                        FREEDRIVE4=G:
                        FREEDRIVE5=H:
                        LOG1=EXTLOG1
                        LOG2=EXTLOG2
                        LOG3=EXTLOG3
                        LOG4=EXTLOG4
                        LOG5=EXTLOG5
                        RSPFILE=c:\SOFTDIST\WORK\RSPFILE
                        SYSTEMNAME=mobile1
                        MOBILE=YES
Shared tokens:          (none)
Hardware parms:         (none)
Discovered inventory:   (none)
```

Figure 188. Output from NVDM LSTG mobile1 -l

If you now install this onto your Mobile Client with the command:

```
NVDM INST DYNAMIC.EXAMPLE.REF.1.0 -w mobile1
```

then you will find that after the next connection is made, you have the following files on the PC:

```
C:\>dir c:\sec

The volume label in drive C is C_DRIVE.
The Volume Serial Number is 6261:0414.
Directory of C:\sec

.            <DIR>      6-11-96  14:03
..           <DIR>      6-11-96  14:03
GENERAL  TXT     1529   7-11-96  14:19
LAPTOP   TXT     1529   7-11-96  14:19
        4 file(s)       3058 bytes used
                     58699776 bytes free
```

Figure 189. Mobile Client after Installation of Dynamic Change Object

If you install the same object onto a machine that does not have MOBILE set to YES then you will see:

```
C:\>dir c:\sec

The volume label in drive C is OS2V300.
The Volume Serial Number is 1231:9234.
Directory of C:\sec

.            <DIR>      6-11-96  14:03
..           <DIR>      6-11-96  14:03
GENERAL  TXT     1529   7-11-96  14:19
DESKTOP  TXT     1529   7-11-96  14:19
        4 file(s)       3058 bytes used
                     43458216 bytes free
```

Figure 190. Non-Mobile Client after Installation

12.4 Using Plans and Groups

In this part we look at a method of handling Mobile Clients from standard clients. We use TME 10 Software Distribution plans to place the control outside the change objects themselves. We also use groups to handle all Mobile Clients together.

Availability

Currently this is only possible with Software Distribution for AIX servers.

12.4.1 Creating Groups

We can use groups to allow us to handle several clients as though they were one. This is especially useful if we make the group dynamic since then we do not have to manually add clients to the group.

12.4.1.1 Creating Static Groups

Often one particular set of users will use the same software. For example, users in an accounts department may use Lotus 1-2-3, WordPro and Lotus Notes, while users in the marketing department may use Freelance Graphics and Lotus Notes. By creating groups that match job function and hence software, we can simplify the job of applying updates and fixes when these are required.

In this example we create a static group of managers who all use the same software.

At the Software Distribution for AIX server start the graphical user interface and display the targets window as shown in Figure 191.

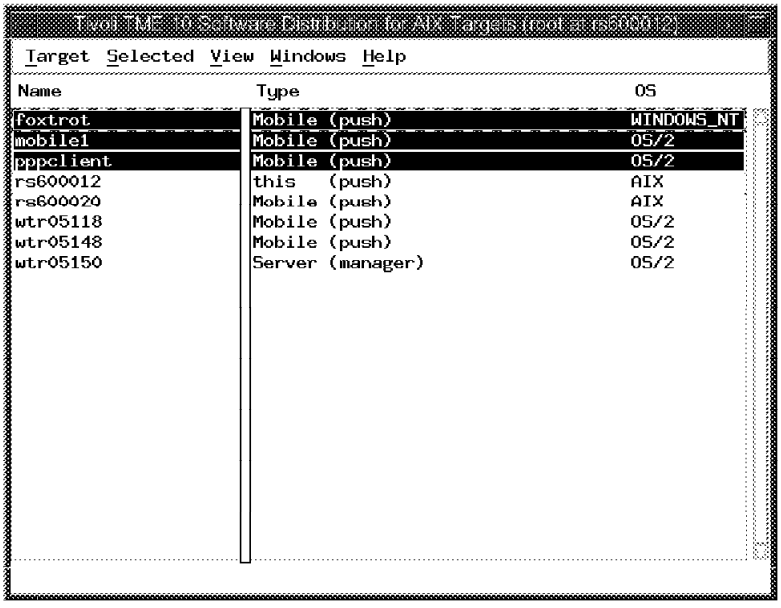


Figure 191. Software Distribution for AIX Targets Window

To create the group, select the machines that are to be in the group and from the menu bar select **Selected** and then **Group...**. This will display the window shown in Figure 192 on page 240.

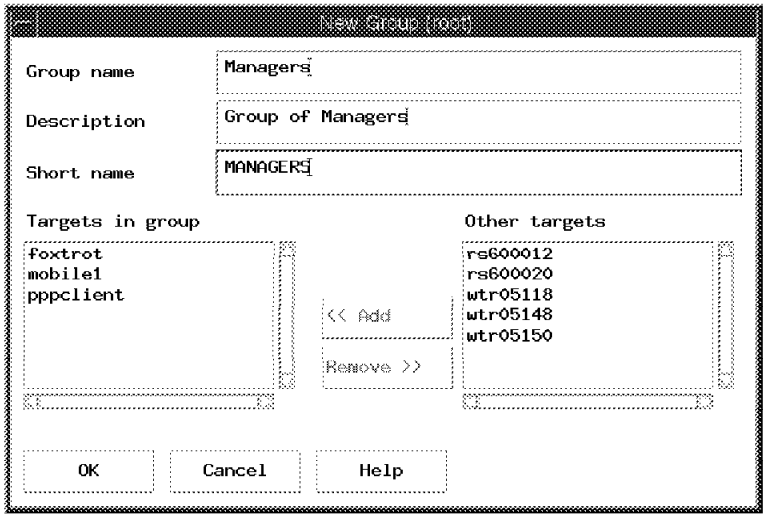


Figure 192. Software Distribution for AIX New Group Window

Enter a name, description and short name for the group and click **OK** to create the group.

You can now install objects to this group as though it were a single client.

12.4.1.2 Creating Dynamic Groups

Using static groups provides a level of administrative control since every new machine must be manually added to the group. Sometimes this is desirable as it enforces discipline when creating new targets. Often, however, this is an unnecessary overhead, for example with auto-registration it would be nice if machines were automatically added to groups for us. Dynamic groups address this requirement.

We can use dynamic groups to specify a group whose characteristic is that all the members are Mobile Clients. We can then install common functions to the group and all members will receive the same installation. If you like, this can be taken further so that you define a group consisting of, for example, Mobile Clients who run on Windows NT and are managers.

To create a dynamic group return to the targets window as shown in Figure 191 on page 239. From the menu bar select **Selected** and then **Group...** This will display the window shown in Figure 193 on page 241.

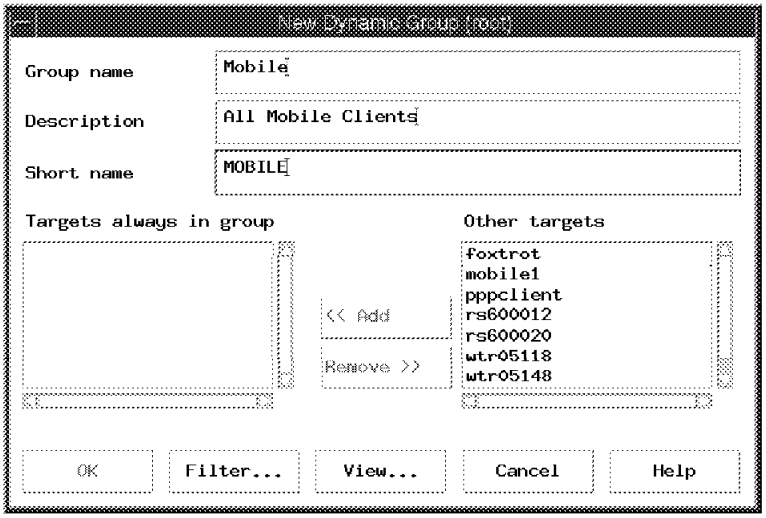


Figure 193. Software Distribution for AIX New Dynamic Group Window

As before, enter a name, description and short name for the group. You can choose to have some clients always in this group if you like. Click **Filter...** to set the entrance requirements for membership of this group (Figure 194).

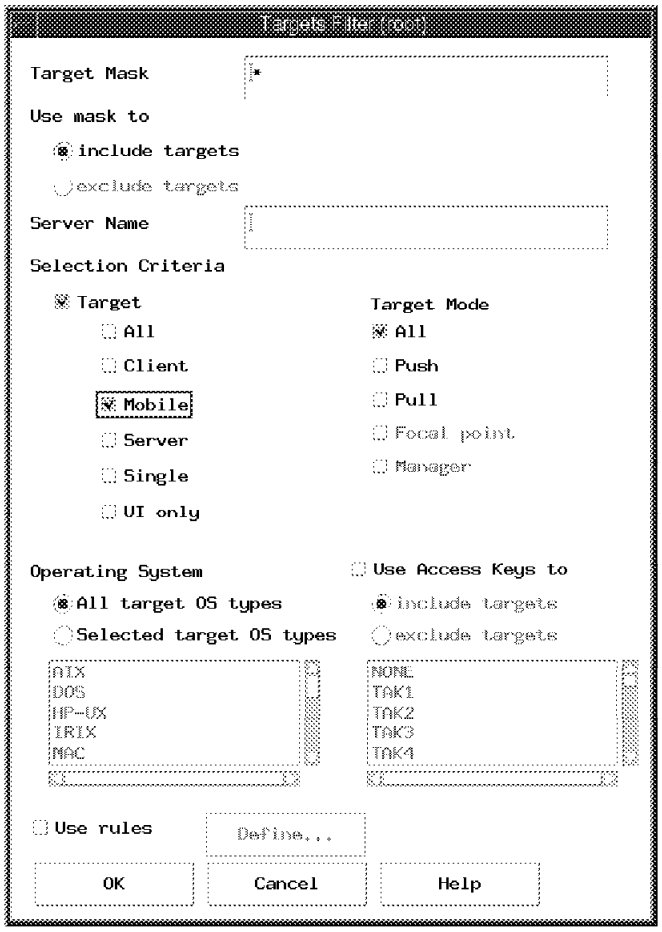


Figure 194. Software Distribution for AIX Target Filters Window

Select **Mobile** to specify that this group is to contain targets that are Mobile Clients. Click **OK** to return to the previous window and then **OK** to create the group.

You can now install software to this group as though it were a single client and Software Distribution for AIX will dynamically resolve the group membership and install to all Mobile Clients.

Note that creating groups does nothing to change objects. The objects themselves will install in exactly the same way. It does, however, make it more obvious to you when you perform an installation that these machines are different. For example, if you create a group for Mobile Clients and one for standard clients, then at installation time you must consciously install an object to both groups. If the object is not appropriate for one of the groups, then a new object can be built and installed to those machines.

12.4.2 Creating Plans

We now create a plan to perform a deadline activation of a generic change object for a mixed group of desktop and mobile targets. All of the steps here are performed from the Software Distribution for AIX server.

To do this, the plan must execute the following steps in order:

1. Send the change object to the Mobile Clients
2. Schedule to install the change object on *all* clients at a set date
3. Connect to the Mobile Clients to allow the requests to flow

First we need a change object that will do the installation for us. Create a simple change object by using the graphical user interface or at the command line enter:

```
nvdm bld /plan_deadline_object.pro
```

after creating the file shown in Figure 195 on page 243.

```
GLOBAL NAME:                PLAN.DEADLINE.EXAMPLE.REF.1.0
DESCRIPTION:                Deadline Activation Object
LOCAL NAME:                 $(REPOSITORY)/PLAN.DEADLINE.EXAMPLE.REF.1.0
CHANGE FILE TYPE:           GEN
COMPRESSION TYPE:           LZW
REBOOT REQUIRED:            NO
REMOVABLE:                  YES
ACTIVABLE:                  NO
INTERACTIVE:                NO
AUTHORIZE:                  NONE
SW HISTORY RESET:           NO
INSTALLATION DURATION:      00:00:00
COST:                       0
PACK FILES:                 NO
SECURE PACKAGE:             NO
OBJECT:
   SOURCE NAME:             /etc/hosts
   TARGET NAME:             c:\Deadline.txt
   TYPE:                    FILE
   ACTION:                  COPY
   INCLUDE SUBDIRS:         NO
```

Figure 195. Profile /plan_deadline_object.pro

Now start the graphical user interface and display the catalog window. From the menu bar select **Catalog** then **Plan** and **Create new...**. This will display the Define Plan window as shown in Figure 196.

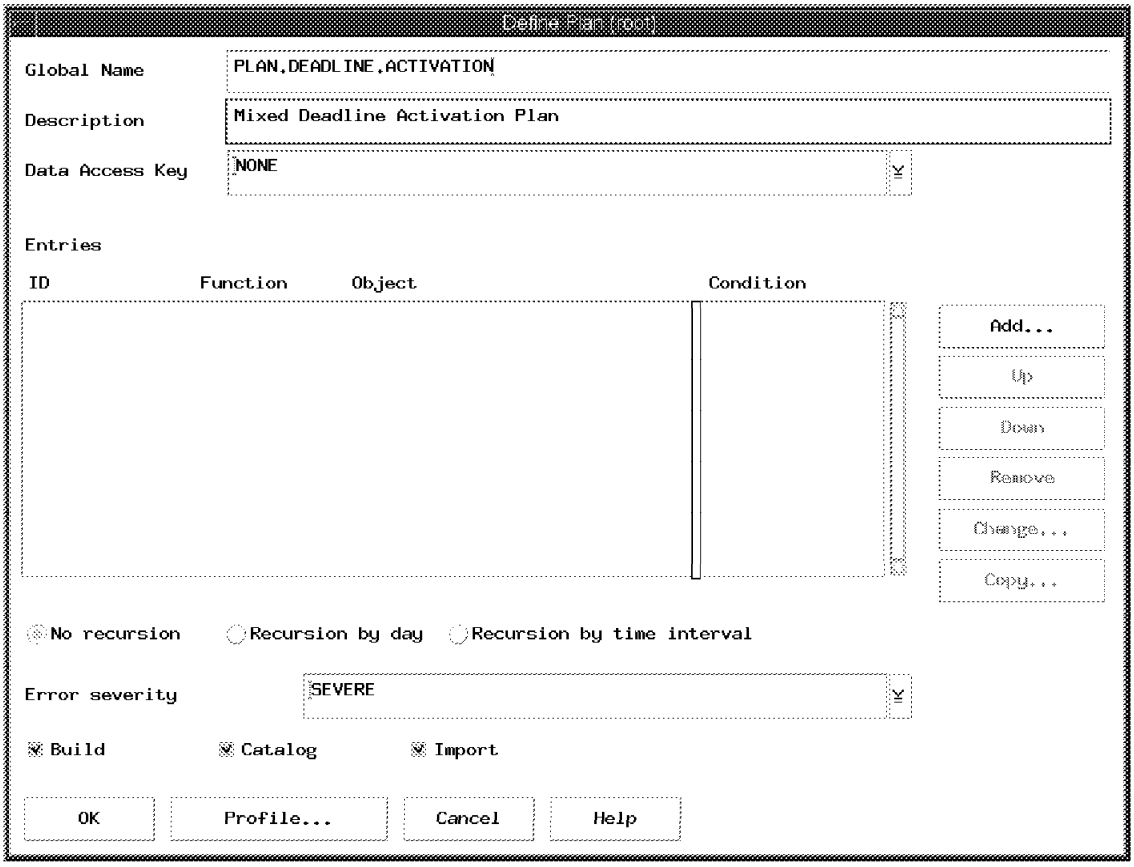


Figure 196. Software Distribution for AIX Define Plan Window

Enter a name and description for the plan and click **Add...** to add an activity to the plan. This will display the Add Plan Entry window as shown in Figure 197 on page 244.

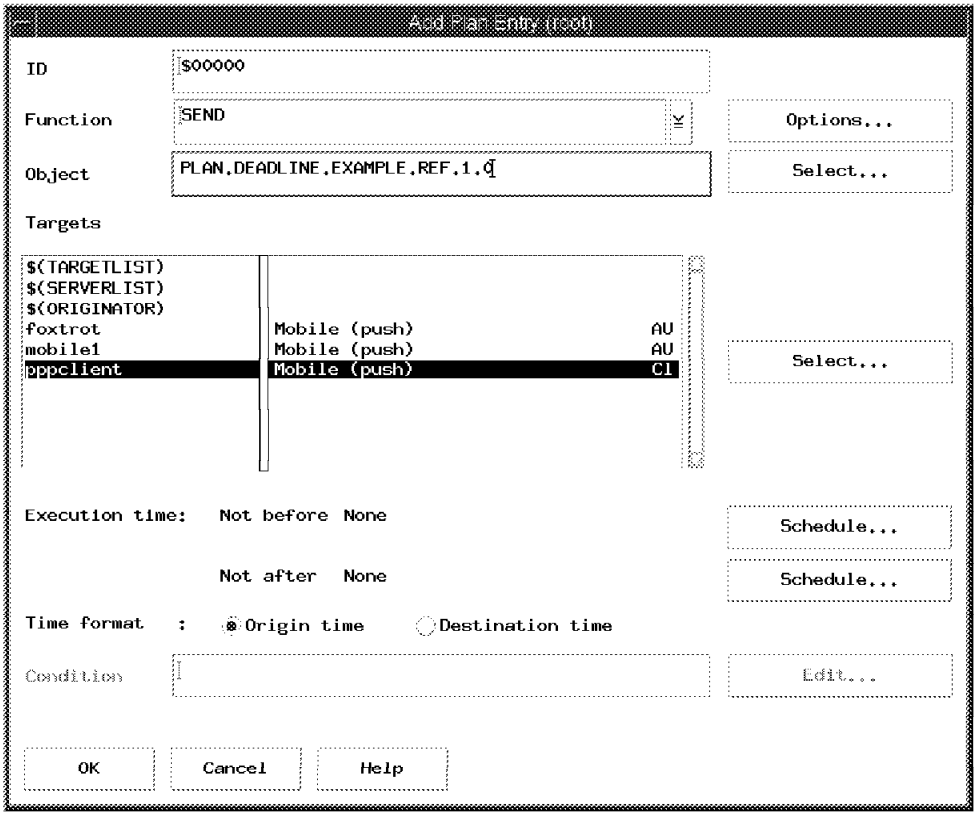


Figure 197. Software Distribution for AIX Add Plan Entry Window

Set the function to SEND and enter the name of the deadline activation object in the Object field. (Alternatively you can search for your object using the **Select...** push button next to the object entry field.) Click the **Select...** push button next to the targets box to select the targets to add. This will display the Select Targets with Filter window as in Figure 198 on page 245.

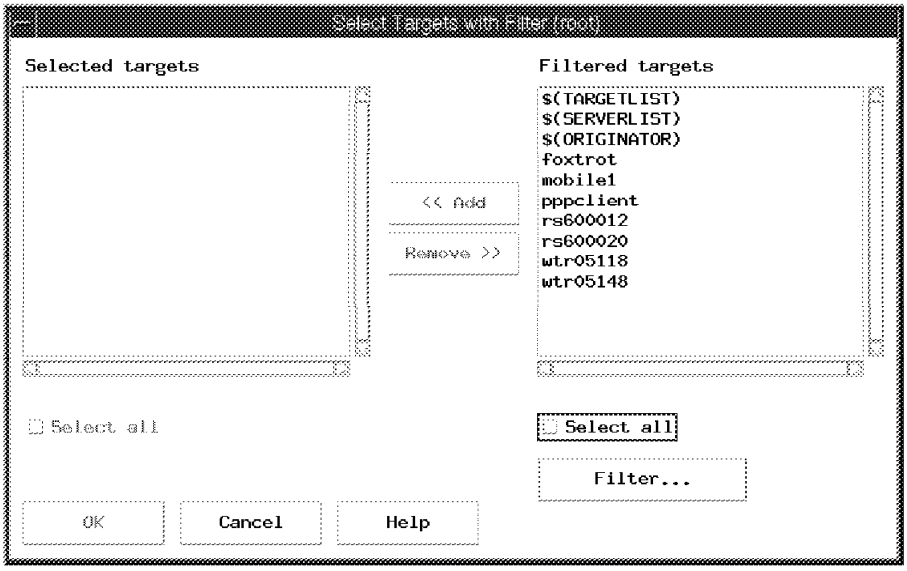


Figure 198. Select Targets with Filter Window

From here click **Filter...** to set the filter criterion. This will bring up a dialog like Figure 199.

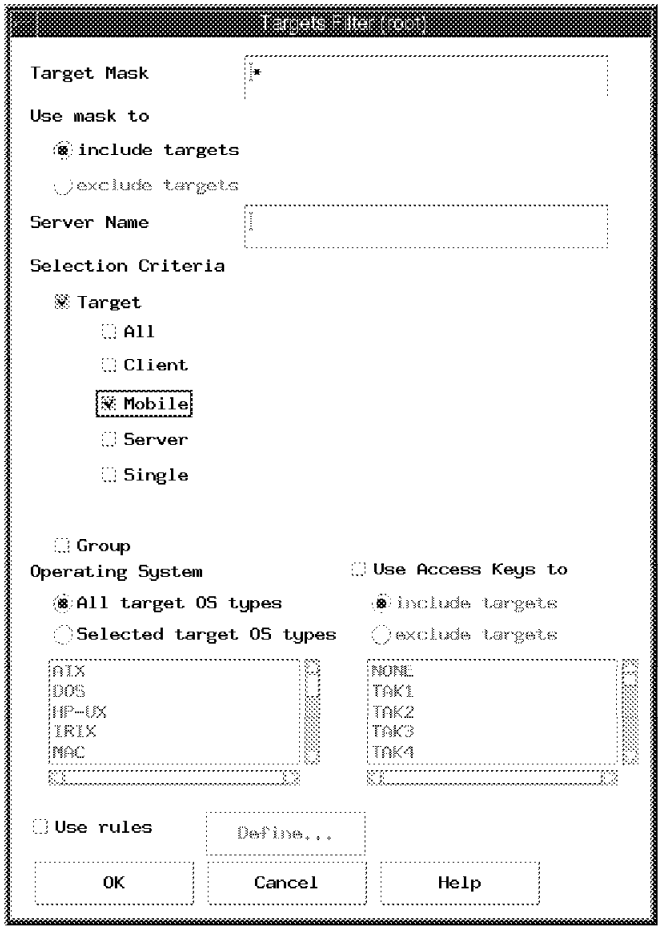


Figure 199. Targets Filter Window

Select the **Mobile** check box and click **OK**. You have now chosen to see all of the Mobile Clients only. You can now mark the **Select All** check box to select all of these targets and click **Add** to add them to the list of selected targets. Do not include the targets called $(TARGETLIST), $(SERVERLIST) and $(ORIGINATOR). We will discuss these later in this section. Now click **OK** to leave this screen and return to the Add Plan Entry screen. You have now selected the change object to send and the targets so click **OK** to create this plan entry.

You now need to repeat this to install the object.

From the Define Plan window click **Add...** to add an activity to the plan. This will display the Add Plan Entry window again as you saw in Figure 197 on page 244.

Set the function to INSTALL and enter the name of the deadline activation object in the Object field. Click the **Select...** push button next to the targets box to select the targets to add. This will display the Select Targets with Filter window as in Figure 198 on page 245. From here click **Filter...** to set the filter criterion. This will bring up a window we saw in Figure 199 on page 245. This time select the **All** check box and click **OK** to return to Select targets with Filter. You can again mark the **Select All** check box to select all of these targets and click **Add** to add them to the list of selected targets.

Click **OK** to close this window.

We want this installation to be a deadline activation so click the top **Schedule...** button to set the date (see Figure 197 on page 244). This will show the schedule window as below.

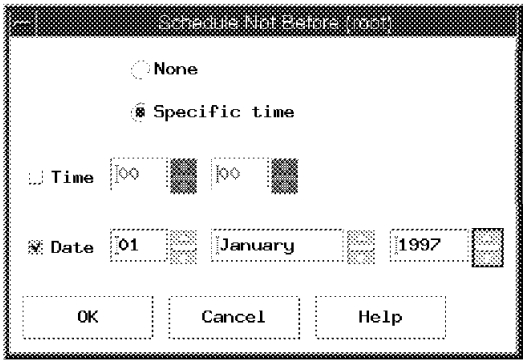


Figure 200. Schedule Not before Window

Enter a date in the future and click **OK**.

From the Add Plan Entry window we have one more thing to do. Click the **Options...** push button to show the window in Figure 201 on page 247.

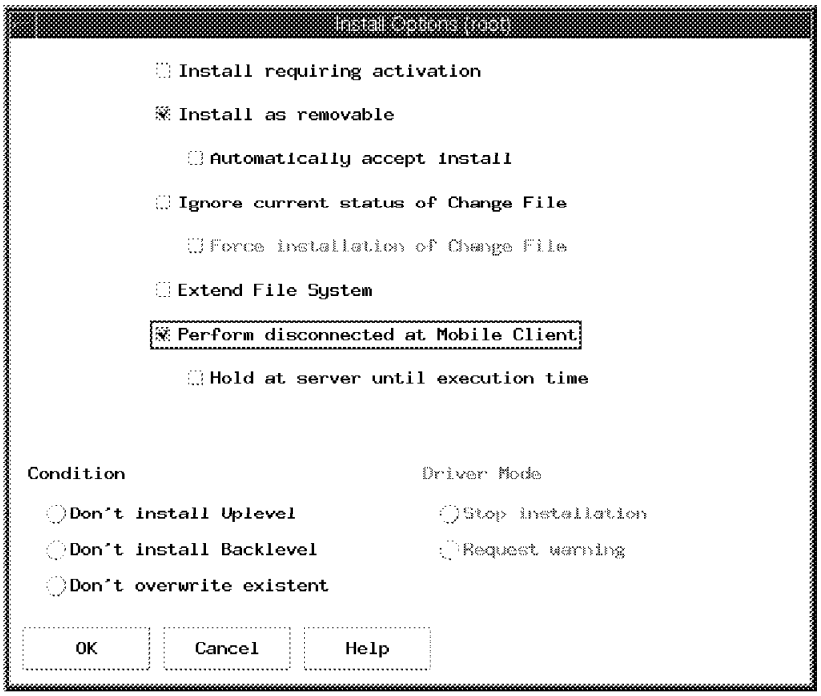


Figure 201. Install Options Window

Select the check box marked **Perform disconnected at Mobile Client** and click **OK** to leave this screen.

This option is ignored for non-mobile clients.

You should now be at the Add Plan Entry window. Click on **OK** again to add this entry.

The last thing we want to do is an open connection window to the Mobile Clients. Unfortunately this cannot be done directly from a plan, so we can either do this manually using groups, or create a small shell script. As an example, we show the shell script approach here. Click **OK** to create the plan. Later we modify it to include the shell script.

Create the file /usr/lpp/netviewdm/bin/mobile_connect as shown in Figure 202 on page 248.

Make sure that this can be executed by the user ID that is submitting the plan. In the worst case make it globally accessible using the command:

```
chmod 733 /usr/lpp/netviewdm/bin/mobile_connect
```

```
#########################################################
# Example Shell Script for Software Distribution for AIX #
# ----------------------------------------------------- #
# This script opens a connection window for 10 minutes   #
# to all mobile clients.                                 #
#########################################################
# A better way to do this is to see which mobile clients #
# have requests queued for the object we created:        #
# PLAN.DEADLINE.EXAMPLE.REF.1.0                          #
# and to open windows to these clients only.             #
#########################################################
# Mark Guthrie 7th November 1996                         #
#########################################################
# Any UNIX fans offended by the use of comments may
# ignore them  :)
#########################################################
# First we will get NVDM to tell us about all of the     #
# clients that it has and pipe this through GREP to      #
# limit this to the two lines we are interested in:      #
#                                                        #
#  Target:               targetname                      #
#  Type:                 targettype                      #
#                                                        #
# We push this to a temporary file                       #
#########################################################
nvdm lstg '*' | grep -E "Target|Type"  > /tempfile

#########################################################
# Now change the input stream to come from our file      #
#########################################################
exec </tempfile

#########################################################
# Read the file until empty and parse the first line     #
# into $title (always "Target:") and $target which is    #
# the client name.                                       #
##                                                      ##
# Then read the second line and parse it into $title     #
# (always "Type:") and $type.                            #
##                                                      ##
# If it is a Mobile Client then schedule a connect.      #
#########################################################
while read title target
do
   read title type
   if [ "$type" = "MOBILE CLIENT" ]
   then
      nvdm connect $target -d 10
   fi
done

#########################################################
# Finally, delete the temporary file                     #
#########################################################

rm /tempfile
```

Figure 202. /usr/lpp/netviewdm/bin/mobile_connect

Now we need to catalog this file to use it from the plan. From the menu bar on the catalog window, select **Catalog** and **Data file...** to display the Catalog Data File window as seen in Figure 203 on page 249.

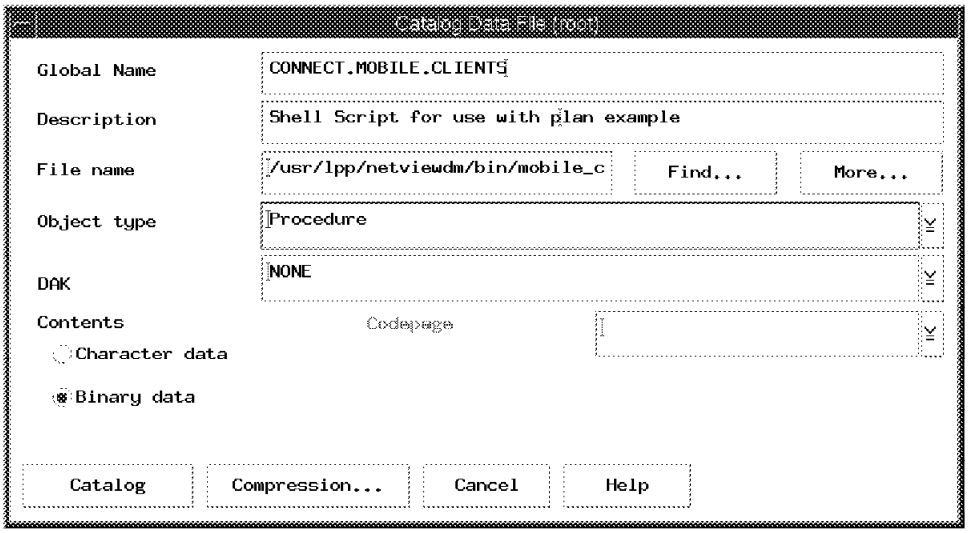


Figure 203. Catalog Data File Window

Enter CONNECT.MOBILE.CLIENTS as the global name and key in a description. In the file name field put /usr/lpp/netviewdm/bin/mobile_connect and set the object type to procedure.. Now click **Catalog** to create the entry.

We now need to modify our plan. From the catalog window select your plan, for example:

PLAN.DEADLINE.ACTIVATION

and choose **Selected** from the menu bar. From the drop-down menu choose **Create another** and **As is...**. This will bring up the Define Plan window as you saw before. Click **Add...** to add a new entry to the plan. This will display the Add Plan Entry window.

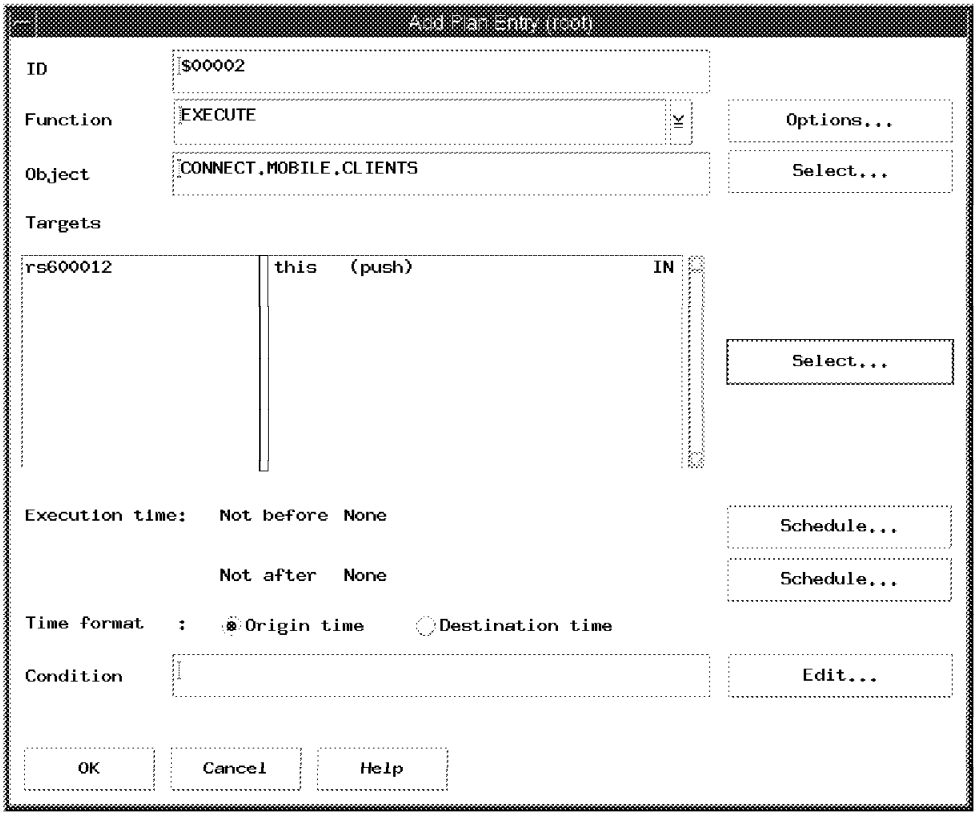


Figure 204. Add Plan Entry Window

Set the function to EXECUTE and enter the object name as CONNECT.MOBILE.CLIENTS, which is the object that we have just created. Click the **Select...** push button next to the targets panel and enter the name of your Software Distribution for AIX server. When we execute this plan we want to run this shell script on the server. The server will then schedule the connection requests to the Mobile Clients.

Click **OK** to add this entry. You should now be at the Define Plan window. You will need to change the name of the plan to save it since there is already a plan of this name. Change the name to PLAN.DEADLINE.ACTIVATE and then click **OK** again to save the modified plan. (You may wish to delete the temporary plan that we created before.)

You can now execute this plan by selecting it from the catalog window and choosing **Selected** and then **Execute Plan...** from the menu bar. Before execution you are presented with a dialog allowing you to specify targets. You may have noticed in Figure 204 that the targets window also contained $(TARGETLIST) and $(SERVERLIST). If you had used these tokens then by selecting targets from this dialog the targets that you choose would be substituted everywhere that you used $(TARGETLIST), and the corresponding servers for those targets would be substituted everywhere that you used $(SERVERLIST). We did not use this feature.

Remember that this installation will not take place until the scheduled date.

12.5 Summary

Managing Mobile Clients requires thought. You need to consider how the Mobile Client is connected and you need to understand how your change objects are built and what they will do when installed.

In this chapter we have provided some general discussion on modifying change objects for use with Mobile Clients. We have worked through a specific example of changing a CID change object to use with a Mobile Client for deadline activation. We have also covered dynamic and static groups and using plans to cope with Mobile Clients.

In Chapter 13, “Mobile Clients in a Focal Point Connected Environment” on page 253 we look at using plans to manage Mobile Clients in a more complex environment.

Chapter 13. Mobile Clients in a Focal Point Connected Environment

In this chapter we look at examples of what happens when Mobile Clients are introduced into a TME 10 Software Distribution configuration, which includes a central focal point server.

We assume that the reader is familiar with this type of environment. For information on setting up and using a focal point connected environment refer to the redbook *The NetView Distribution Manager/6000 Cookbook*, GG24-4246, or the books *Tivoli TME 10 Software Distribution for AIX Up and Running!*, SH19-4333 and *Tivoli TME 10 Software Distribution for OS/2 Up and Running!*, SH19-4334.

13.1 Introduction

In a typical network environment, multiple LANs are connected over a wide area network (WAN). Workstations and servers are connected over fast LAN connections for optimum performance. Links from LANs to central systems and other LANs are over slower WAN connections.

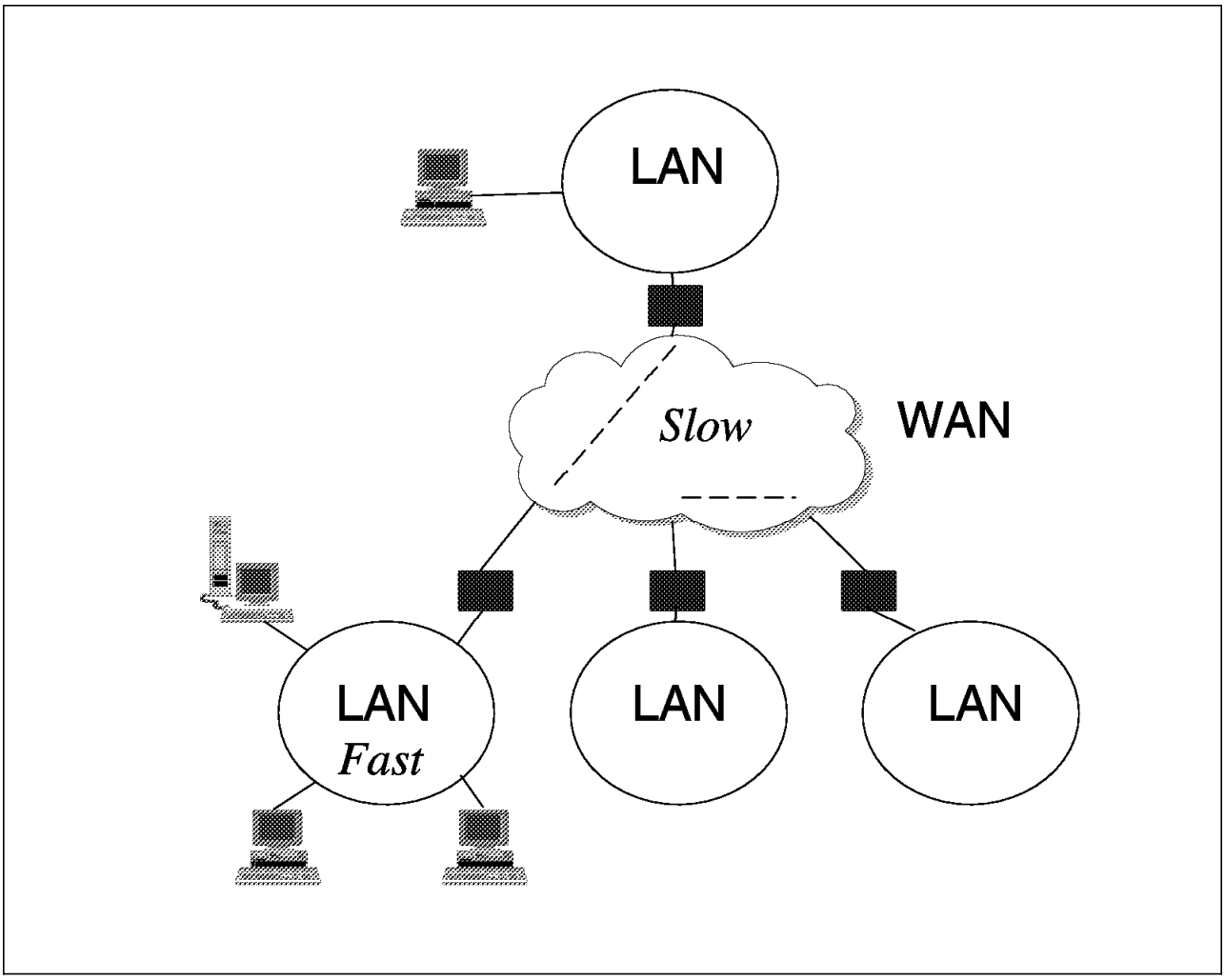


Figure 205. A Typical WAN Network

© Copyright IBM Corp. 1997

For software distribution to target machines located on remote LANs, you will probably install the TME 10 Software Distribution server onto the LAN itself and manage it from a central location over the WAN links. This is because traffic from the central site to the server is less plentiful than traffic from the server to its targets. For example, to install 5 MBs of data to 30 machines directly from a central server would use 150 MBs of valuable WAN bandwidth. To install this using a local server would only use 5 MBs of WAN resource and 150 MBs of cheap LAN bandwidth.

Remote servers are managed directly from a central site exactly as described in this redbook. You can use the graphical user interface over a WAN connection or use X-Windows or telnet to control a server.

As the number of servers in the network increases, so does the administrative effort of looking after them. TME 10 Software Distribution provides a way to control all of the remote servers from a central focal point server, so for most operations the targets of the remote servers can be treated as though they belonged to the central server. This focal point server can be AIX or NetView DM/MVS. In our examples we use AIX as the focal point server.

If you wish, you can define a hierarchy of servers with intermediate ″fan-out″ servers passing requests from the central server to the servers at the end points.

13.2 Test Environment

We installed a focal point connected server as shown in Figure 206 on page 255.

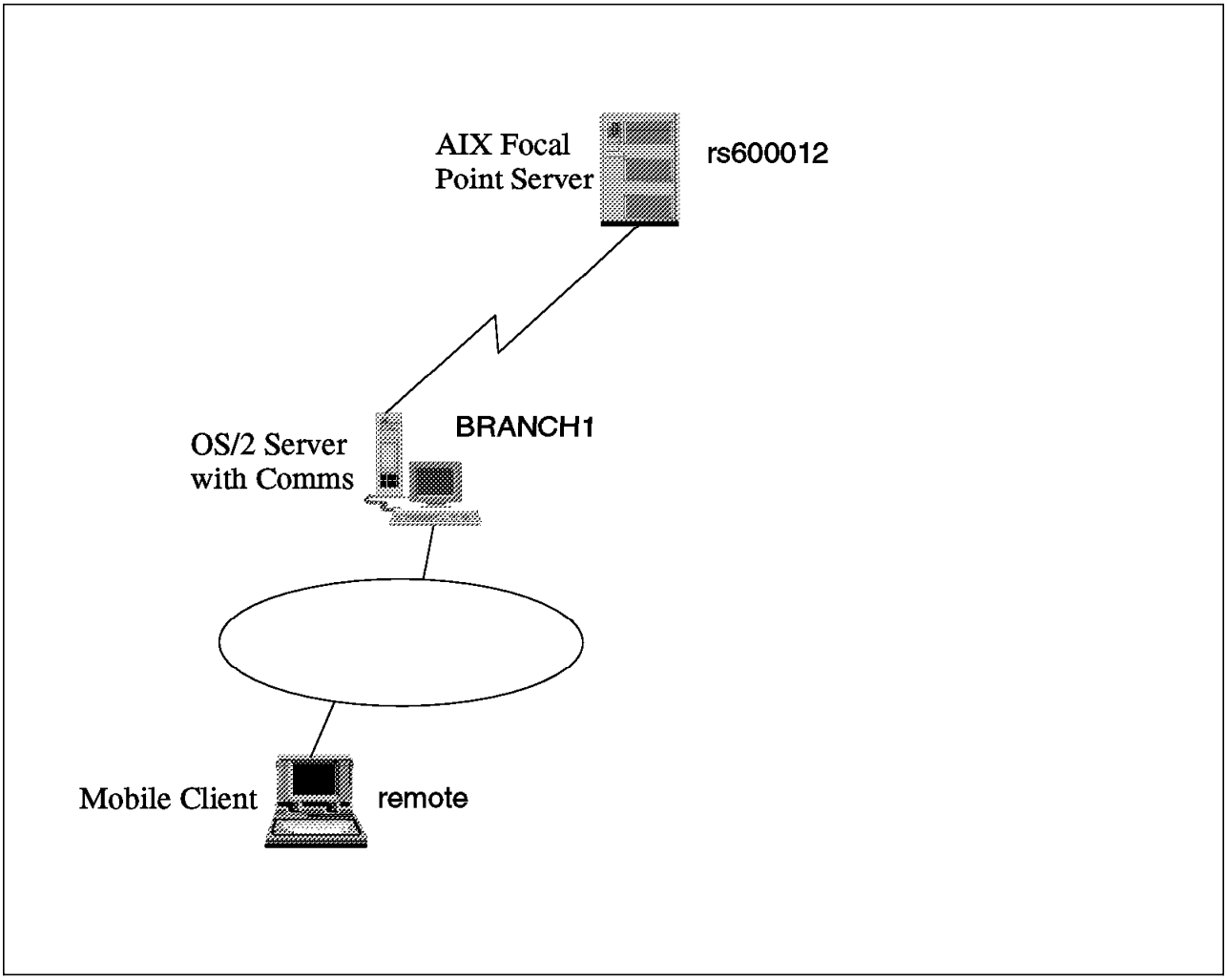


Figure 206. Focal Point Connection Test Environment

We used the same Software Distribution for AIX server that was used in the rest of this book, rs600012, and modified it to become our focal point server. We installed a Software Distribution for OS/2 server called BRANCH1 as a focal point connected server and connected a Mobile Client to this server.

Starting from a basic installation, these are the steps to follow to build this environment:

1. Define your OS/2 server to the Software Distribution for AIX server using the following command from AIX:

   ```
   nvdm addtg BRANCH1 -b server -y os/2 -s BRANCH1 -tp tcp:branch1
   ```

   Replace branch1 with the IP hostname that you will use for your OS/2 server.

2. Stop the Software Distribution for AIX server using the command:

   ```
   nvdm stop -x -k
   ```

3. On the Software Distribution for AIX server, edit the file /usr/lpp/netviewdm/db/routetab to contain the following lines:

```
NETWORK PROTOCOL:   TCP/IP

# Destination     Connection File     Hop Count

BRANCH1.BRANCH1     BRANCH1              10
```

Figure 207. /usr/lpp/netviewdm/db/routetab

This tells Software Distribution for AIX that to get to the server BRANCH1 in domain BRANCH1, it should use the configuration file BRANCH1.

4. Now on the AIX server, create the file /usr/lpp/netviewdm/db/snadscon/BRANCH1 as follows:

```
TYPE:                   STS
PROTOCOL:               TCP/IP
REMOTE SERVER NAME:     BRANCH1
TCP/IP TIME-OUT:        0
NEXT DSU:               .
TRANSMISSION TIME-OUT:  0
RETRY LIMIT:            0
SEND MU_ID TIME-OUT:    0
RECEIVE MU_ID TIME-OUT: 0
```

Figure 208. /usr/lpp/netviewdm/db/snadscon/BRANCH1

5. Now restart the Software Distribution for AIX server with the command:

   ```
   nvdm start
   ```

6. You now need to obtain the name and domain of your Software Distribution for AIX server. You can do this by entering the command:

   ```
   nvdm lstg -l
   ```

   This will produce output similar to the following:

   ```
   Target:                 rs600012
   Description:            INITIAL TARGET CONFIGURATION RECORD
   Mode:                   Push
   Type:                   SERVER
   Operating system:       AIX
   Target address:         SERVER
   Domain address:         SERVER
   Network:                TCP rs600012
   ```

   In our case the server is called SERVER and the domain is also called SERVER.

7. Install the Software Distribution for OS/2 server with a server name of BRANCH1, a target address of BRANCH1 and a domain name of BRANCH1. We installed our server manually. The configuration panels are shown below.

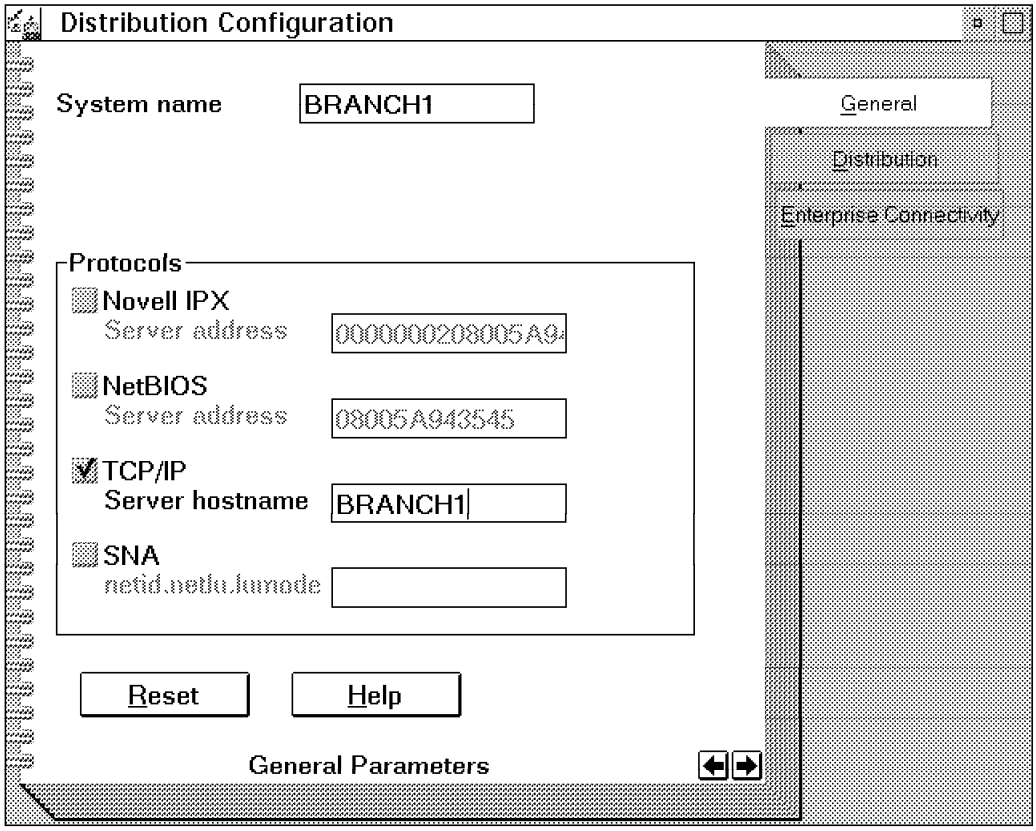


Figure 209. Distribution Configuration - General

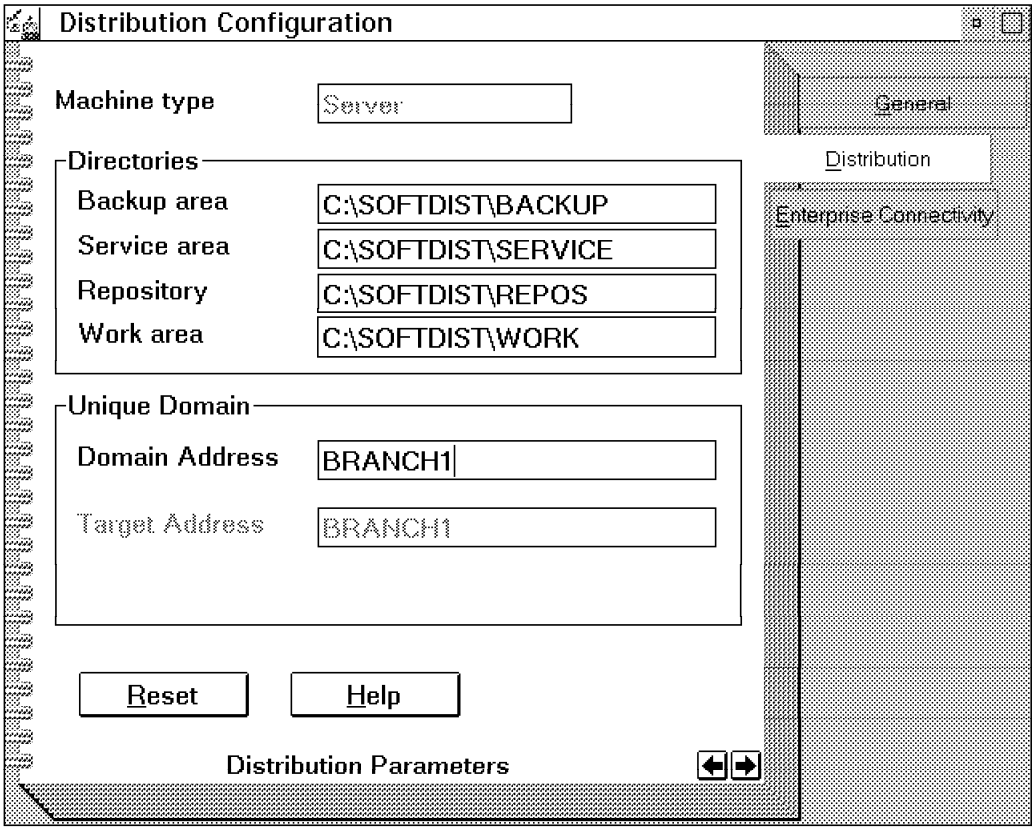


Figure 210. Distribution Configuration - Distribution

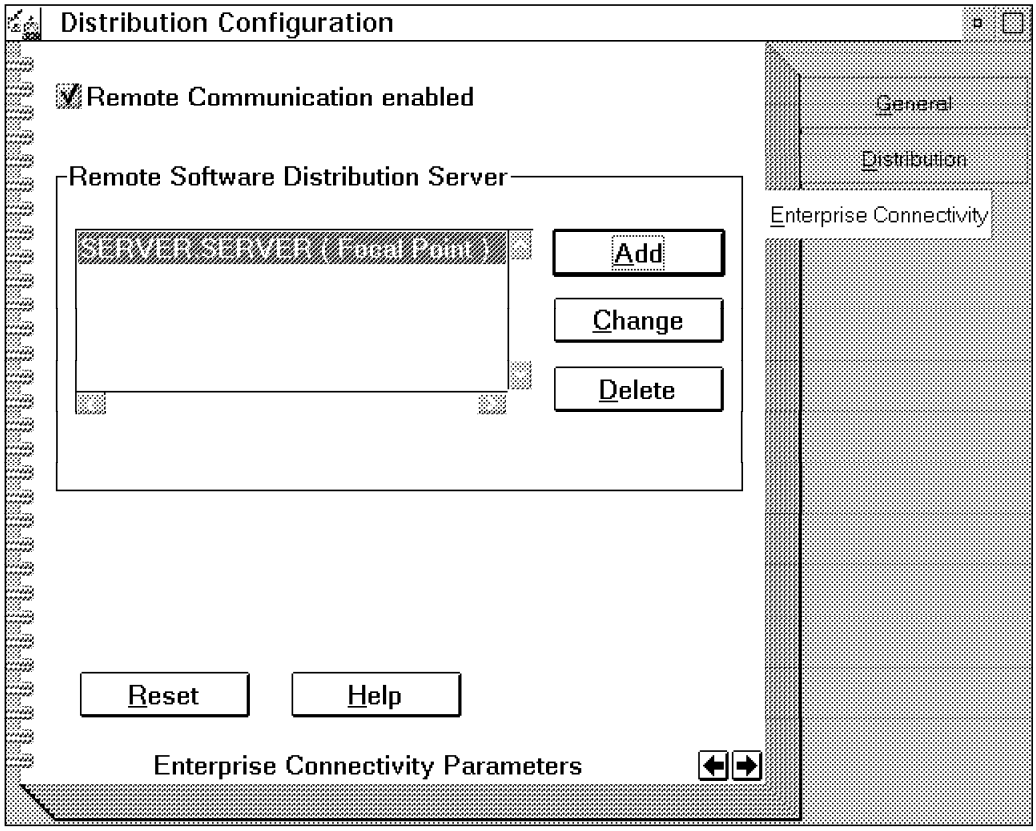


Figure 211. Distribution Configuration - Enterprise Connectivity

In the Enterprise Connectivity panel (Figure 211), click **Add** to add a connection to the Software Distribution for AIX server. This will display the following panel.

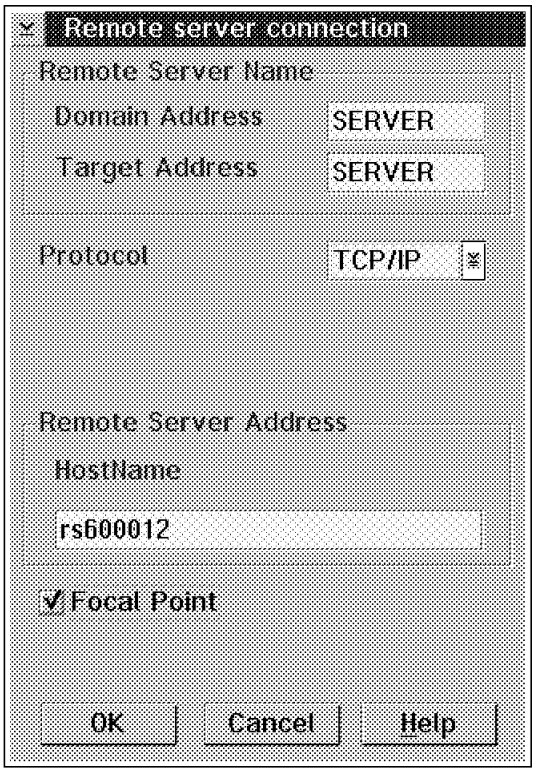


Figure 212. Remote Server Connection Window

Click **OK** to create this connection and continue with the installation of the server as normal.

The process that you have just followed will create a definition for the server rs600012 in the target database. It will also create the following files on the Software Distribution for OS/2 server:

```
WORKSTATION NAME:                       BRANCH1
SERVER:                                 BRANCH1
PROTOCOL:                               TCP branch1 729 50
REPOSITORY:                             C:\SOFTDIST\REPOS
WORK AREA:                              C:\SOFTDIST\WORK
BACKUP AREA:                            C:\SOFTDIST\BACKUP
SERVICE AREA:                           C:\SOFTDIST\SERVICE
CONFIGURATION:                          SERVER_WITH_COMMS
MESSAGE LOG LEVEL:                      D
LOG FILE SIZE:                          2000000
API TRACE FILE SIZE:                    2000000
TRACE FILE SIZE:                        2000000
MAX USER INTERFACES:                    20
MAX ATTEMPTS:                           5
MAX TARGETS:                            600
TARGET PASSWORD AUTHENTICATION:         NO
AUTHORIZE:                              NONE
AUTOMATIC CLIENT UPGRADE:               NO
LAN AUTHORIZATION:                      0
AUTOMATIC TARGET REGISTRATION:          YES
DACA RETRY TIME:                        300
MACHINE TYPE:                           OS/2
```

Figure 213. C:\SOFTDIST\NVDM.CFG

```
DOMAIN NAME BRANCH1
SERVER IDENTIFIER BRANCH1
```

Figure 214. C:\SOFTDIST\DOMAIN.CFG

```
NETWORK PROTOCOL:                               TCP/IP
SERVER.SERVER SERVER0 5
```

Figure 215. C:\SOFTDIST\DB\ROUTETAB

```
PROTOCOL:                       TCP/IP
TYPE:                           STS
REMOTE SERVER NAME:             rs600012
```

Figure 216. C:\SOFTDIST\DB\SNADSCON\SERVER0

8. We modified the NVDM.CFG to change `authorize` to ALL and rebooted the OS/2 server.

The two servers should now be able to communicate using a server-to-server (STS) connection over TCP/IP.

13.3 Automatic Target Registration

If both the OS/2 server and the AIX server are set to allow automatic target registration by setting the value AUTOMATIC TARGET REGISTRATION: to YES in the NVDM.CFG, then Mobile Clients can connect to the network without being predefined.

As an example, consider the network shown in Figure 206 on page 255. If a new Mobile Client connects to server BRANCH1 then it is registered with the server and the server will update the focal point server with the information.

We created a client called REMOTE using the configuration file shown in Figure 217.

```
WORKSTATION NAME:                   remote
SERVER:                             BRANCH1 TCP branch1
PROTOCOL:                           TCP remote 729 50
REPOSITORY:                         d:\SOFTDIST\REPOS
WORK AREA:                          d:\SOFTDIST\WORK
BACKUP AREA:                        d:\SOFTDIST\BACKUP
SERVICE AREA:                       d:\SOFTDIST\SERVICE
CONFIGURATION:                      CLIENT
MESSAGE LOG LEVEL:                  D
LOG FILE SIZE:                      64000
API TRACE FILE SIZE:                64000
TRACE FILE SIZE:                    64000
MAX USER INTERFACES:                20
MAX ATTEMPTS:                       5
TARGET MODE:                        PUSH
MACHINE TYPE:                       OS/2
TARGET ADDRESS:                     remote
```

Figure 217. C:\SOFTDIST\NVDM.CFG for Mobile Client

When the Mobile Client is started, the following data flow takes place:

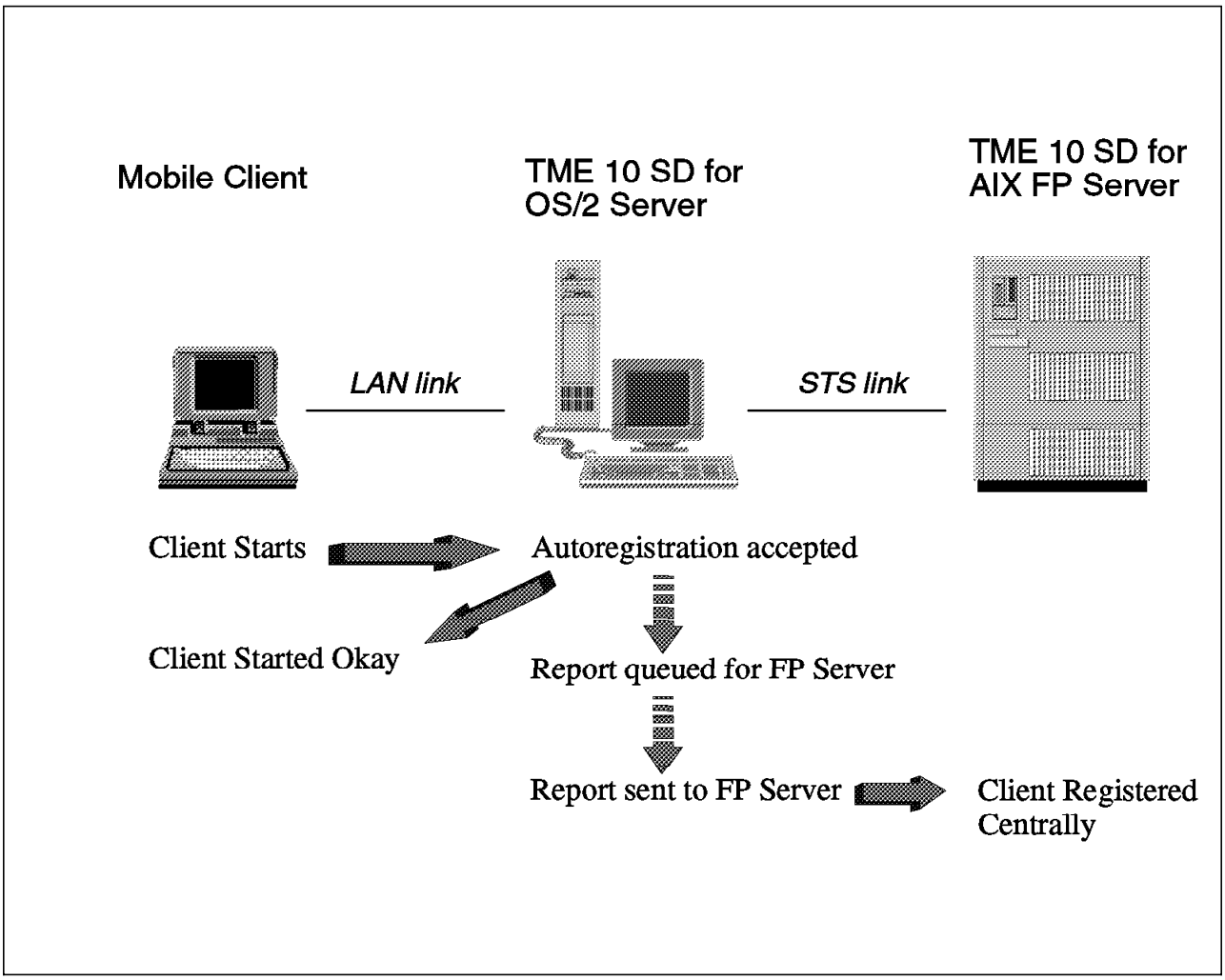


Figure 218. Data Flow

We can see this by looking at the FNDLOG files on all three machines.

On the Mobile Client:

```
FNDRX020I: Attempting to connect to server BRANCH1 on port 729.
FNDRX001I: New connection 14 from client remote agent 3.
FNDRX021I: CRBX socket is 12.
FNDCM662I: D&CC Agent is returning requests and reports to the server.
FNDCM663I: D&CC Agent is resynchronising the local database with the server.
```

On the OS/2 Server:

```
FNDRX005I: Processing a connection request:
           Application shadow task started for agent CLI on client remote.

FNDDB001I: Added or changed Target Configuration record.

FNDRB052I: Target remote has been successfully added or updated.

FNDRB111I:  Auto registration request BRANCH1 root 3 0 was successfully
            placed on the Request Handler Input Queue.

FNDRQ108I:  @root BRANCH1 3 0 N/A BRANCH1 :
            Received successful Auto registration report.
```

```
FNDRQ044I:  @root BRANCH1 3 0 N/A BRANCH1 : Auto registration report queued
             to the Scheduler for remote targets on connection SERVER0.

FNDSH267I:  @root BRANCH1 3 0 N/A BRANCH1 : Auto registration request/report
             queued for transmission on STS connection SERVER0.

FNDSV001I:   STS task started.

FNDRX020I:    Attempting to connect to server rs600012 on port 729.

FNDRX001I:     New connection 276 from client BRANCH1 agent 4.

FNDDB001I:     Added or changed Target Configuration record.

FNDSV008I:     Connection SERVER0 is being processed by STS.

FNDSV005I:     @root BRANCH1 3 0 N/A BRANCH1 : Request completed transfer via STS.

FNDSV009I:    Connection SERVER0 is no longer being processed by STS.

FNDRX023I:    Connection 276 is closing.

FNDSV002I:   STS task ended.
```

On the AIX Server:

```
FNDRQ108I: @root BRANCH1 3 0 N/A BRANCH1 : Received
             successful Auto registration report.

FNDRQ163I: @root BRANCH1 3 0 N/A BRANCH1 : auto registration report:
             add target configuration record required.

FNDRQ108I: @root BRANCH1 4 0 N/A remote : Received successful Install report.

FNDRQ108I: @root BRANCH1 4 0 N/A remote : Received successful
Retrieve Inventory report.
```

Note

This is the only time that information regarding the new client can flow between the servers. If you wish to add an installation token or modify the record in any way, then this information does not get successfully passed between the servers and they are out of step. However, you can define a token for this client on the focal point server and on the OS/2 server and use the same value each time.

13.4 Installing to Mobile Clients

When administering Mobile Clients from a server or administrator's workstation there are two main points to remember:

1. You need to open a connection window before anything will happen.
2. For disconnected installs, you need to send the change object to the Mobile Client's catalog before the install can take place.

In early chapters we have looked at how to deal with these points. In Chapter 7, "LAN-Connected Mobile Client" on page 103, we showed simple examples of

installing change objects for deadline activation. In Chapter 12, “Using Change Objects with Mobile Clients” on page 215, we showed how to manage a mixed environment of targets that includes Mobile Clients. In a focal point connected environment we can build on these scenarios.

Restrictions

You can open a connection window to a Mobile Client only from its own server. You cannot schedule a connection from the focal point server. The focal point server does not own the Mobile Client and cannot control requests to connect and disconnect such a target.

In Chapter 12, “Using Change Objects with Mobile Clients” on page 215 we found a similar restriction when using plans. From a plan you cannot schedule a connection or a disconnection to a Mobile Client. We got around this by using a simple shell script (Figure 202 on page 248) to open a connection to all of the Mobile Clients. We cataloged this on the Software Distribution for AIX server and included this in our plan. We can take the same approach in a focal point connected environment.

We will create a plan to install a change file to a Mobile Client. This time we will use the variable $(SERVERLIST) to initiate our procedure on the server.

Create the text file shown in Figure 219 on page 266 and Figure 220 on page 267. Store this on the Software Distribution for AIX server.

```
/*******************************************/
/* OS/2 Rexx script to connect Mobile      */
/* Clients.                                */
/*                                         */
/* To be run from Focal Point Connected    */
/* TME 10 Software Distribution for OS/2   */
/* servers.                                */
/*                                         */
/* Mark Guthrie December 5th 1996          */
/*                                         */
/*******************************************/

/***********************************/
/* Get args. If we are passed a    */
/* target name then we will only   */
/* connect this target.            */
/* If we are called with no args   */
/* then we will connect all Mobile */
/* Clients.                        */
/***********************************/
parse arg target .

if target<>"" then               /* Only 1 target */
   call DO_ONE
else                             /* All targets   */
   call DO_ALL

exit

/***********************************/
/* Routine to connect a single     */
/* target.                         */
/***********************************/
DO_ONE:
say 'nvdm connect 'target' -d 10'
return

/***********************************/
/* Routine to connect all targets  */
/***********************************/
DO_ALL:
say 'Connecting all Mobile Clients'

/*******************************/
/* clear rexx queue            */
/*******************************/
do while queued()<>0
   pull .
end /* do */

/*******************************/
/* find all targets            */
/*******************************/
say 'Listing all targets'
'nvdm lstg * |rxqueue'
```

Figure 219. /usr/lpp/netviewdm/bin/mobcon.cmd Part 1 of 2

```
/*****************************/
/* Read output and find      */
/* Mobile Clients            */
/*****************************/
do while queued()<>0
   parse pull line                          /* case sensitive */
   parse value line with "Target:" target
   if target<>"" then do

/*****************************/
/* Found a block we are      */
/* interested in. 3 lines down*/
/* is the type               */
/*****************************/
      pull .  /* skip line */
      pull .  /* skip line */
      parse pull line                       /* case sensitive */
      parse value line with "Type:" type
      if type="MOBILE CLIENT" then do

/*****************************/
/* Mobile Client so connect it */
/*****************************/
         target=strip(target)             /* take his clothes off */
         say 'Connecting: 'target
         'nvdm connect 'target' -d 10'
      end
   end
end

return
```

Figure 220. /usr/lpp/netviewdm/bin/mobcon.cmd Part 2 of 2

Now start the Software Distribution for AIX graphical user interface and display the catalog window. From the menu bar select **Catalog** and **Data file...** to bring up the Catalog Data File window as seen in Figure 221.

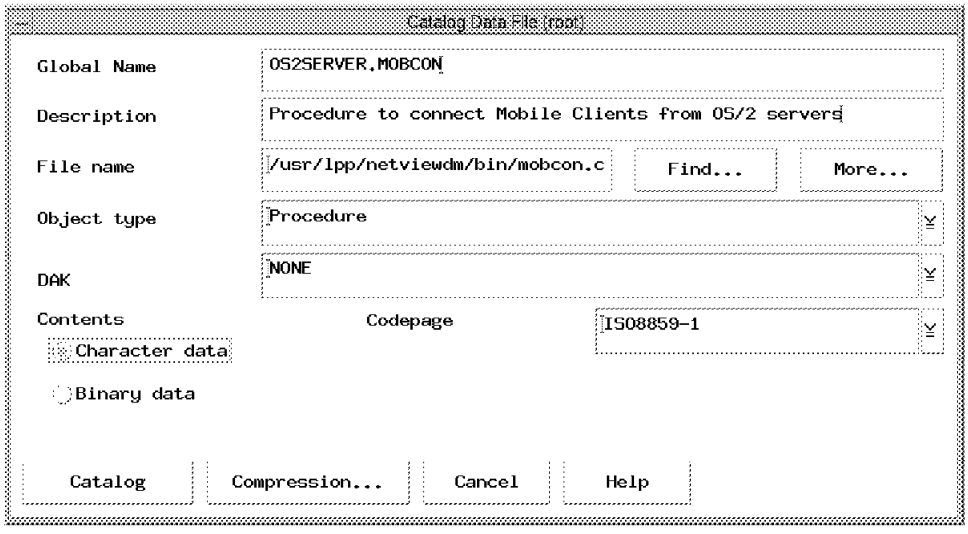


Figure 221. Catalog Data File Window

Enter OS2SERVER.MOBCON as the global name and key in a description. In the File name field, enter /usr/lpp/netviewdm/bin/mobcon.cmd and set the object type to Procedure. Set the contents to character data and click **Catalog** to create the entry.

Note

We will be executing this procedure on the Software Distribution for OS/2 server itself and not on any of its targets. However, we must still send it to the server in order to use it, because the client agent that runs on the server gets its change objects from the OS/2 server and not the AIX server.

From the graphical user interface choose this object and select **Selected** and **Send...** from the menu bar. Enter BRANCH1 as the destination and send this object.

Repeat this procedure to send the change object if you have not already done so.

Now create a simple change object to install. For example, use PLAN.DEADLINE.EXAMPLE as created in Chapter 12, “Using Change Objects with Mobile Clients” on page 215.

Once you have your change object, start the graphical user interface and display the catalog window. From the menu bar select **Catalog** then **Plan** and **Create New...** This will display the Define Plan window as shown in Figure 222 on page 269.

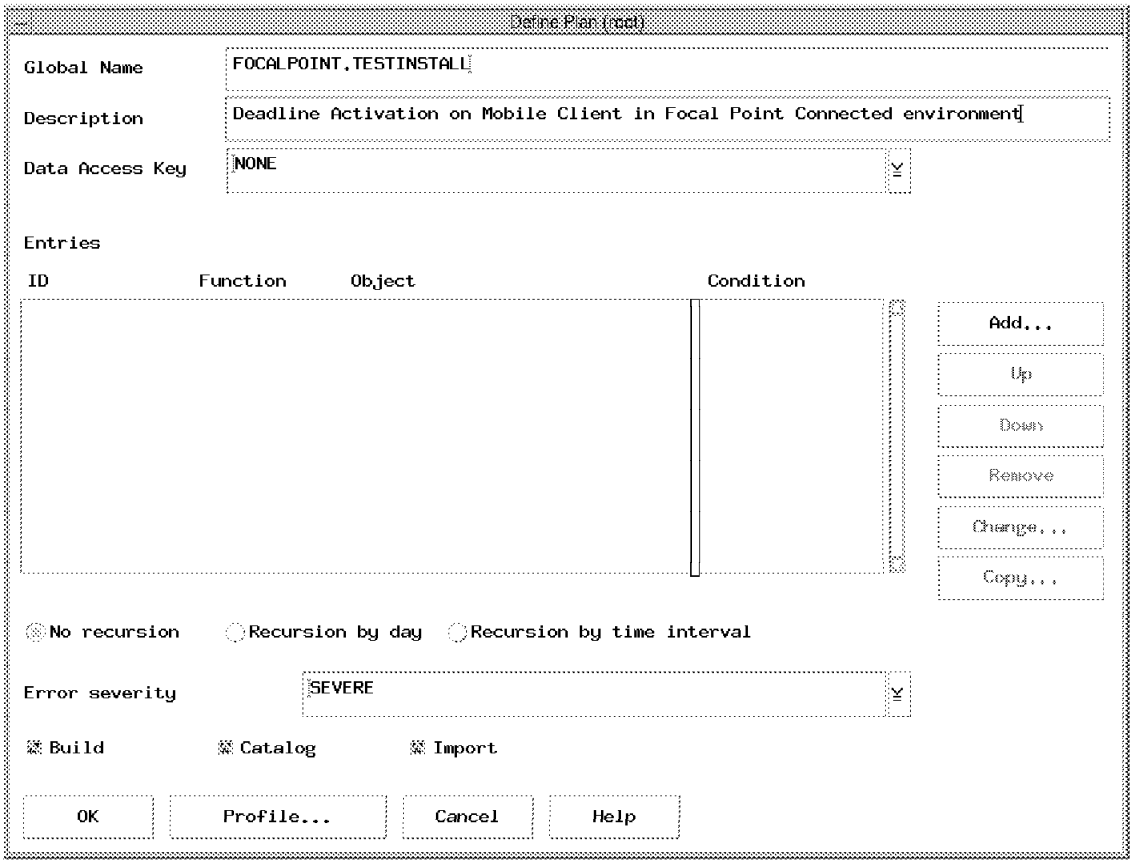


Figure 222. Software Distribution for AIX Define Plan Window

Enter a name and description for the plan and click **Add...** to add an activity to the plan. This will display the Add Plan Entry window as shown in Figure 223 on page 270.

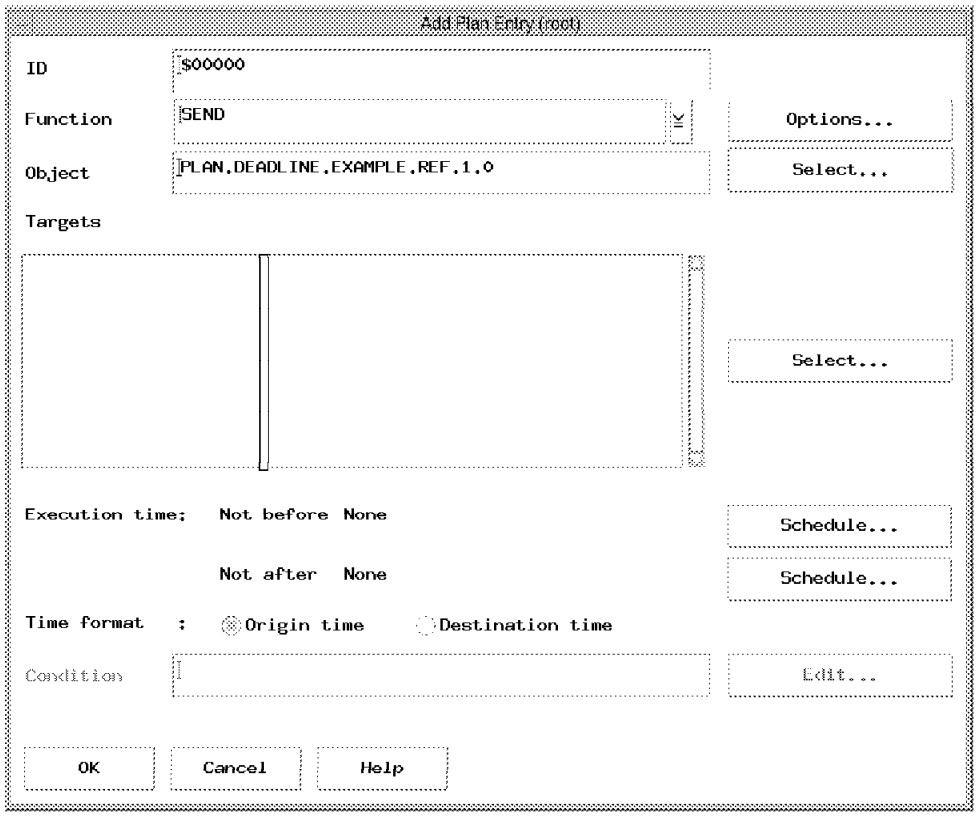


Figure 223. Software Distribution for AIX Add Plan Entry Window

Set the function to SEND and enter the name of the deadline activation object in the Object field. (Alternatively you can search for your object using the **Select...** push button next to the object entry field.)

Click the **Select** push button next to the targets window to bring up the following window:

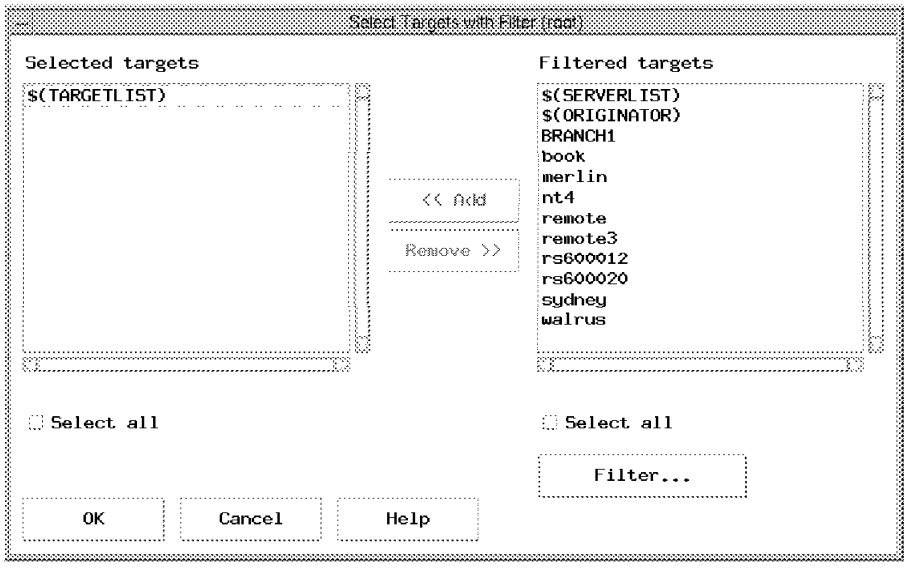


Figure 224. Select Targets with Filters Window

Rather than entering specific targets, this time we will choose the dynamic entry $(TARGETLIST). Select this from the list and click **Add** to add it to the selected targets window. Now click **OK** to return to the add plan entry window and **OK** again to create this plan entry.

You now need to add an entry to the plan to install the object.

From the Define Plan window click **Add...** to add an activity to the plan. This will display the Add Plan Entry window again as you saw in Figure 223 on page 270.

Set the function to INSTALL and enter the name of your change object in the Object field. Again choose the dynamic entry $(TARGETLIST).

We want this installation to be a deadline activation so click the top **Schedule...** button to set the date. This will show the schedule window as below.

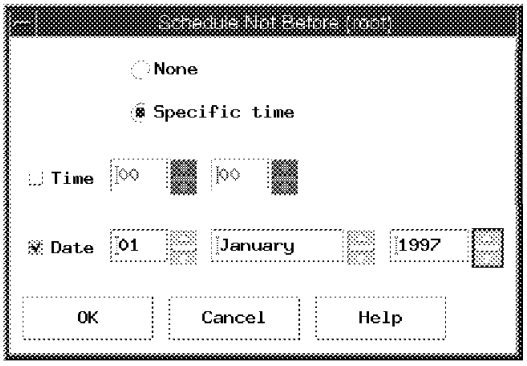


Figure 225. Schedule Not before Window

Enter a date in the future and click **OK**.

You should now be at the Add Plan Entry window. Click the **Options...** push button next to function to show the window in Figure 226 on page 272.

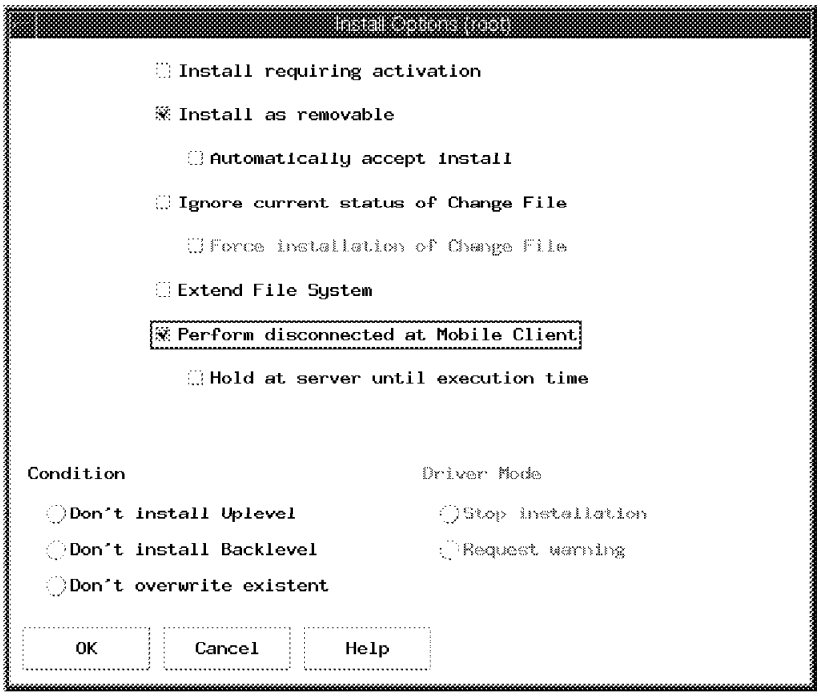


Figure 226. Install Options Window

Select the check box **Perform disconnected at Mobile Client** and click **OK** to leave this screen.

You should now be at the Add Plan Entry window and you can click **OK** again to add this entry.

The final thing to add to this plan is the procedure to open the connection window. Click **Add...** to add a new entry to the plan. This will display the Add Plan Entry window.

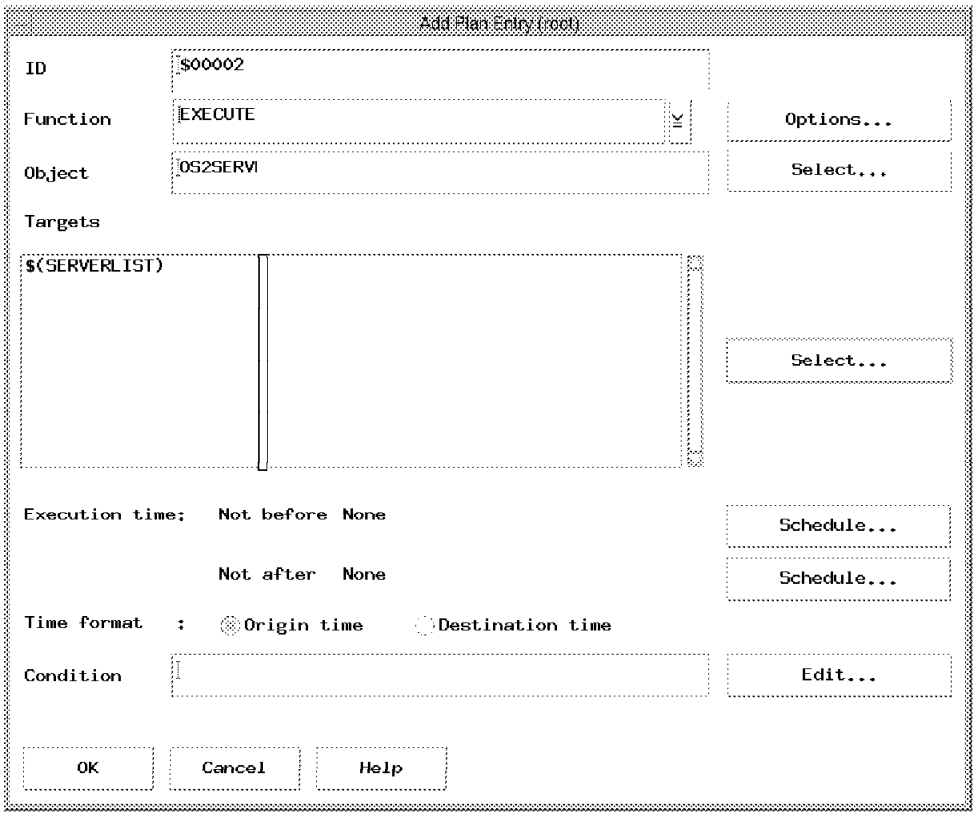


Figure 227. Software Distribution for AIX Add Plan Entry Window

Set the function to EXECUTE and enter the object name as OS2SERVER.MOBCON, which is the object that you have just created. Click the **Select...** push button next to the targets panel and this time select **$SERVERLIST** to be the target. When we execute this plan we want to run this procedure on the server to which the Mobile Client is defined. The server will then schedule the connection requests to the Mobile Client.

Click **OK** to add this entry. You should now be at the Define Plan window. Click **OK** again to create the plan.

You can now execute this plan by selecting it from the catalog window and choosing **Selected** and then **Execute Plan...** from the menu bar. You are presented with a dialog allowing you to specify targets.

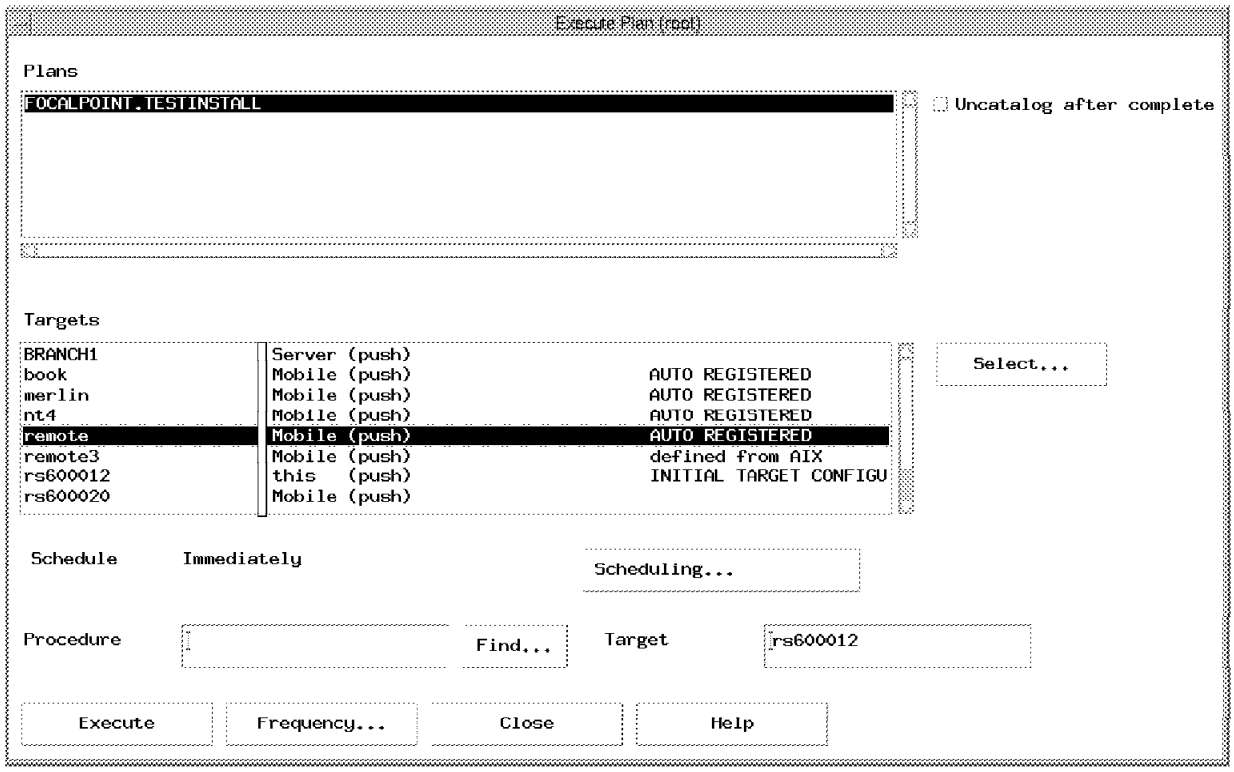


Figure 228. Execute Plan Window

Select **remote** and click **OK** to run this plan. The change object is sent to the OS/2 server first since the product is smart enough to know that it is not already in the catalog of the server. Next it is scheduled to be sent to the Mobile Client and the installation request will also be scheduled. Finally the connection is opened by executing the procedure on the server.

Remember to set the date on the Mobile Client past the scheduled date and restart the agent for the install to take place.

13.5 Design Considerations

Chapter 16, “Designing a Network” on page 307 discusses designing a network to support Mobile Clients. If you have a large network already in place with focal point connections and several distributed servers then you have additional decisions to make regarding how you implement Mobile Clients.

You may choose to add Mobile Client support to all or some of your remote servers.[3] Alternatively you may choose to use a central server for all Mobile Clients. You may even choose to separate the management of Mobile Clients entirely from the rest of your network. You should consult Chapter 16, “Designing a Network” on page 307 for more information on planning for Mobile Clients.

[3] There is no additional software to install; however, as we have seen there are several different things to bear in mind with Mobile Clients and they will inevitably complicate how you run your servers.

Chapter 14. Using the Mobile Client with Dynamic IP

In this chapter we look at how to use the Mobile Client in an environment where Dynamic IP is deployed. We briefly discuss Dynamic IP and show how to set up a small test environment. Then we include TME 10 Software Distribution into the picture and see how it interacts in a dynamic environment.

14.1 Introduction to Dynamic IP

Before Dynamic IP, TCP/IP configuration information was held in multiple locations. Any IP node such as a workstation, server or router would configure its own basic information. This included IP address, subnet mask, default router, name server address and local hostname. A Domain Name Services (DNS) server, or hierarchy of such servers, would hold the mapping of hostnames to IP addresses. If a user moved to another location or a new workstation was added, then the DNS server and the workstation must both be changed. For many applications, including TME 10 Software Distribution, this could require configuration changes at the application level, too, since nodes in the network are known by IP address or hostname.

For users, this required a set of configuration parameters to be supplied by an administrator every time they moved to a new location. For administrators, this meant changing DNS configurations, which was a laborious manual task.

The solution to this is to have the parameters that vary supplied by a local server and to provide a mechanism to dynamically update the DNS server with the new values. Dynamic Host Configuration Protocol (DHCP) provides the first of these services, allowing a potential IP host to request information from a DHCP server. The DHCP client can then configure itself for the local environment without any user involvement.

DHCP is an extension to the Bootstrap Protocol (BootP), which has been used since 1985 to boot diskless workstations from network servers. In simple terms a DHCP client connects to the network using an IP address of 0.0.0.0 and issues a broadcast to the BootP well-known port (UDP port 67).

A DHCP server replies and provides the client with an IP address and other local information, such as default router address. The server can provide an IP address from a pool of available addresses on a leased basis.

The client renews this lease periodically. Alternatively, the server can provide an address permanently from a pool or it can use a manual configuration to provide certain clients with fixed addresses. The server recognizes clients in several ways, such as by physical network address. In our examples, we will use the leased approach.

Once the client has an address and is established on the network, it needs to update the lookup tables so that it can be found using the new address. This can be done using Dynamic Domain Name Services (DDNS). DDNS is a superset of DNS and so a DDNS server can operate as a direct replacement for a DNS server. We will use this feature in our example.

A client can register with a DDNS server the first time it connects to the network. It specifies the hostname that it wants to use and if this is available then the

© Copyright IBM Corp. 1997

DDNS server will allocate it to this host. Subsequently, if a client is allocated a different IP address on the network, it can inform the DDNS server of the change. In this way once a laptop is registered on the network as hugo.sales.banana, it can be reached at this name regardless of the physical location or IP address.

Although there is some interaction between DHCP and DDNS (for example, if a lease expires on an IP address and a new address is issued, the DHCP server can inform the DDNS server of the change), they can be used independently.

For more information on Dynamic IP the reader is referred to the redbook *TCP/IP Implementation in an OS/2 Warp Environment*, SG24-4730.

14.2 Setting Up a Dynamic IP Network

Dynamic IP is not an institution to be entered lightly. Before using DHCP and DDNS, you should do extensive research to understand fully how they work.

Most companies choosing to implement a Dynamic IP network will have a team of network engineers dedicated to supporting the environment. The most probable scenario that will face a TME 10 Software Distribution administrator is having to run software distribution on a network that is already running Dynamic IP.

To this end it is useful to have an understanding of how Dynamic IP operates and so in this section we will describe how to set up a simplified Dynamic IP network consisting of three machines. This will provide a basis for the discussion that follows, and demonstrates all of the problems that will face you in a large network.

14.2.1 Overview

In this section we describe the steps to follow to build the Dynamic IP environment shown in Figure 229 on page 277.

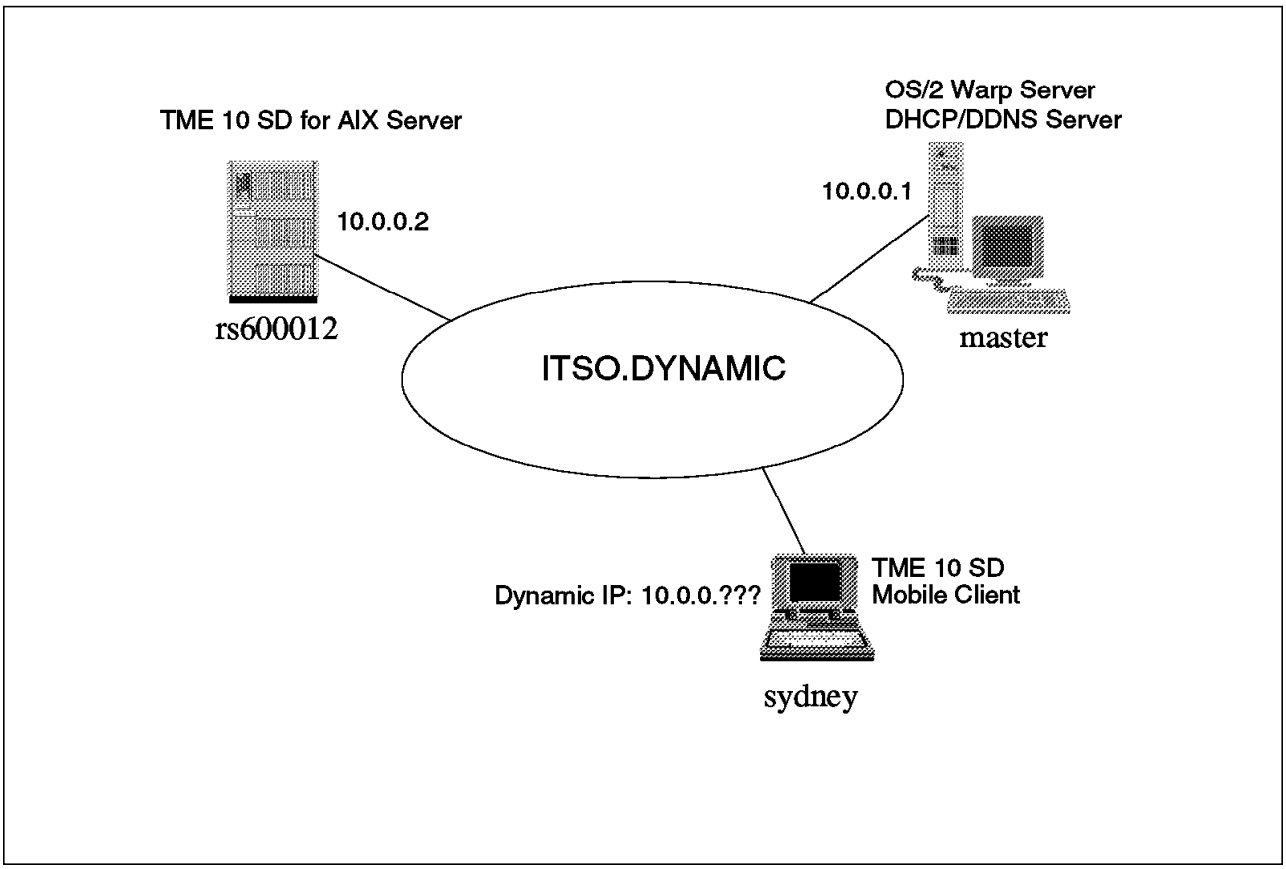


Figure 229. Dynamic IP Test Environment

To avoid any conflict with other IP users, we will use a network address that is not already in use. In our environment we used 10.0.0.0. You should find an address that applies to your environment and use this.

We will create a network called ITSO.DYNAMIC. This will have a combined DHCP and DDNS OS/2 server with an IP address of 10.0.0.1 and a hostname of MASTER. On this network we have an AIX system called rs600012. This is our Software Distribution for AIX server and we want it to have a static IP address. We could use Dynamic IP; however, for fixed resources such as main servers and routers, there is very little benefit. To make our server use the IP address 10.0.0.2, we have two options:

1. Configure the DHCP server to always give rs600012 the same IP address.
2. Configure a fixed IP address on rs600012 and tell it not to use DHCP.

We will use option 2 for simplicity.

Also on our network is a Mobile Client called sydney.[4] We want sydney to be able to move around our network and receive Dynamic IP addresses. This means that we can connect anywhere in our company without needing to reconfigure the laptop for local routers, subnet masks, IP addresses and proxy servers. We also want to be able to use TME 10 Software Distribution from anywhere, which requires that we can always connect to and from rs600012.

[4] You can name this after your home city if you also run out of names.

In a real environment we would have multiple DHCP servers distributed about the physical locations. We would have one or more DDNS servers and the laptop would connect to the nearest DHCP server to be provided with an IP address and local information. Once on the network, the laptop would connect to the DDNS server to tell it where to find the laptop. Figure 230 shows an example of this with a single DDNS server.

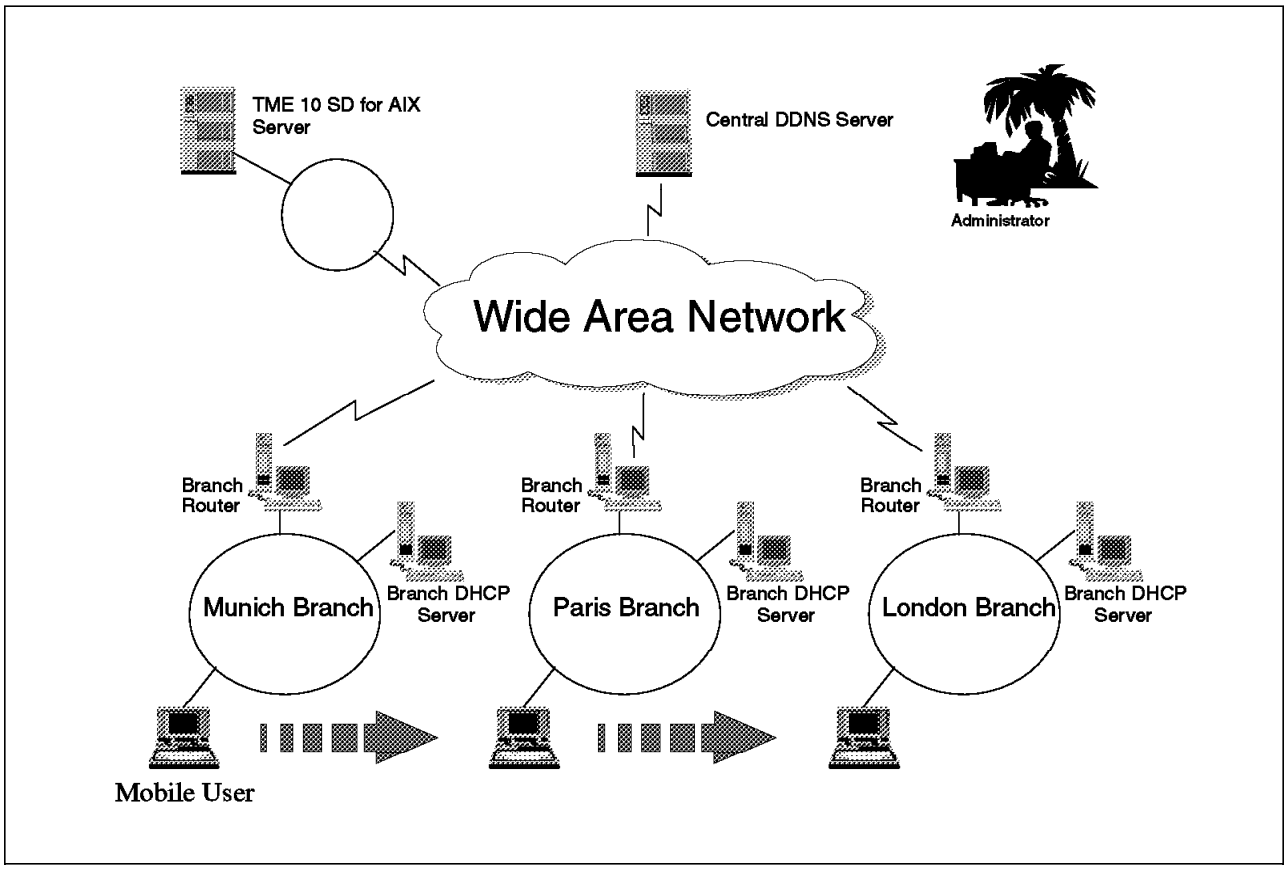


Figure 230. Larger Scale Dynamic IP Environment

14.2.2 Configuring the AIX Server

If you already have a working Software Distribution for AIX server, the only thing to change is the TCP/IP configuration. We will configure this to use the IP address 10.0.0.2 and the DNS server 10.0.0.1. Remember that even though this is a DDNS server, it can still service clients that believe it to be a traditional DNS server.

From AIX, start the systems management interface tool by entering:

```
smit tcpip
```

This will display the following window:

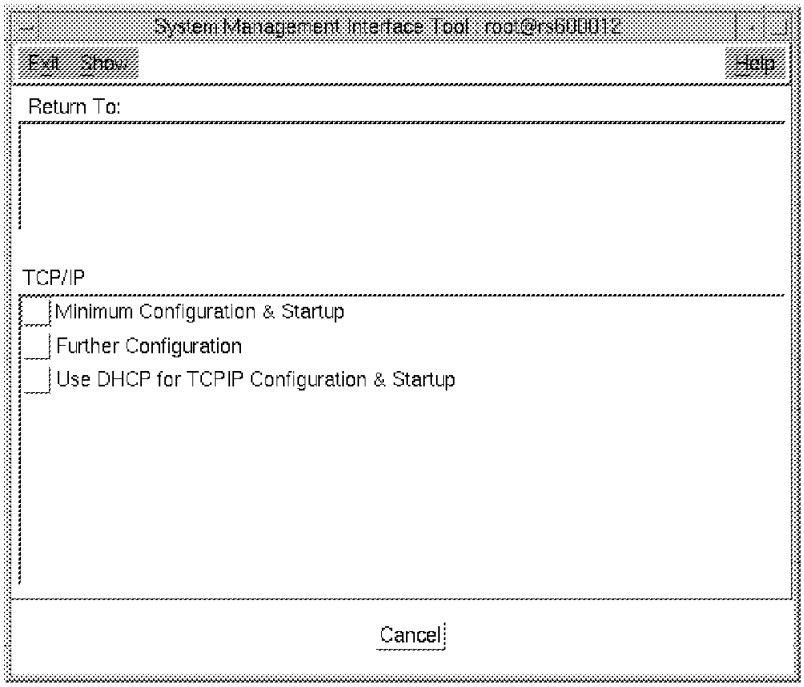


Figure 231. AIX SMIT TCP/IP Panel

Select **Minimum Configuration and Startup**. Now select the interface you use to connect to the LAN from the following panel:

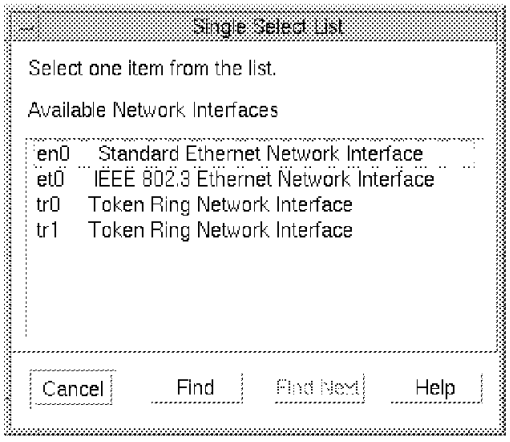


Figure 232. SMIT Interface Selection Panel

This will display the Minimum Configuration & Startup window as shown in Figure 233 on page 280.

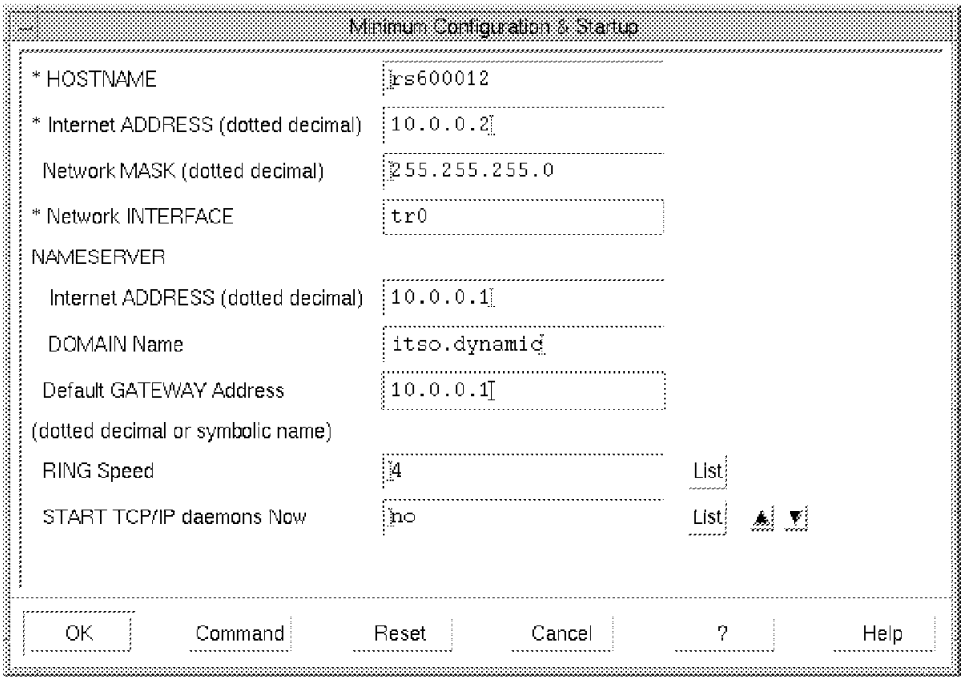


Figure 233. SMIT Minimum Configuration and Startup Panel

In this window enter the new IP address as 10.0.0.2 and the new name server address as 10.0.0.1. Specify the domain name and set the default router address to be 10.0.0.1. We will not configure this host to be a router but, by specifying an address, we will remove the old address from the configuration.

Now click **OK** to make this change. The command will execute and you should receive a successful output message.

If you have defined multiple DNS servers, then you may have to remove these as well using SMIT.

14.2.3 Setting Up Basic IP on the Server

We used OS/2 Warp Server on both the server and the client for this testing. We installed master with the option to operate as a DHCP and DDNS server. We installed sydney as a client.

For OS/2, IBM TCP/IP for OS.2 Version 3.1 is required for Dynamic IP support. This is supplied with OS/2 Warp Server. We assume that you have already installed TCP/IP 3.1 onto both OS/2 systems, and that everything has been installed into the default directories on drive C.

Before you can do anything else, you need to define the hostname and IP address of the server. From the OS/2 desktop on master, open the TCP/IP folder as shown in Figure 234 on page 281.

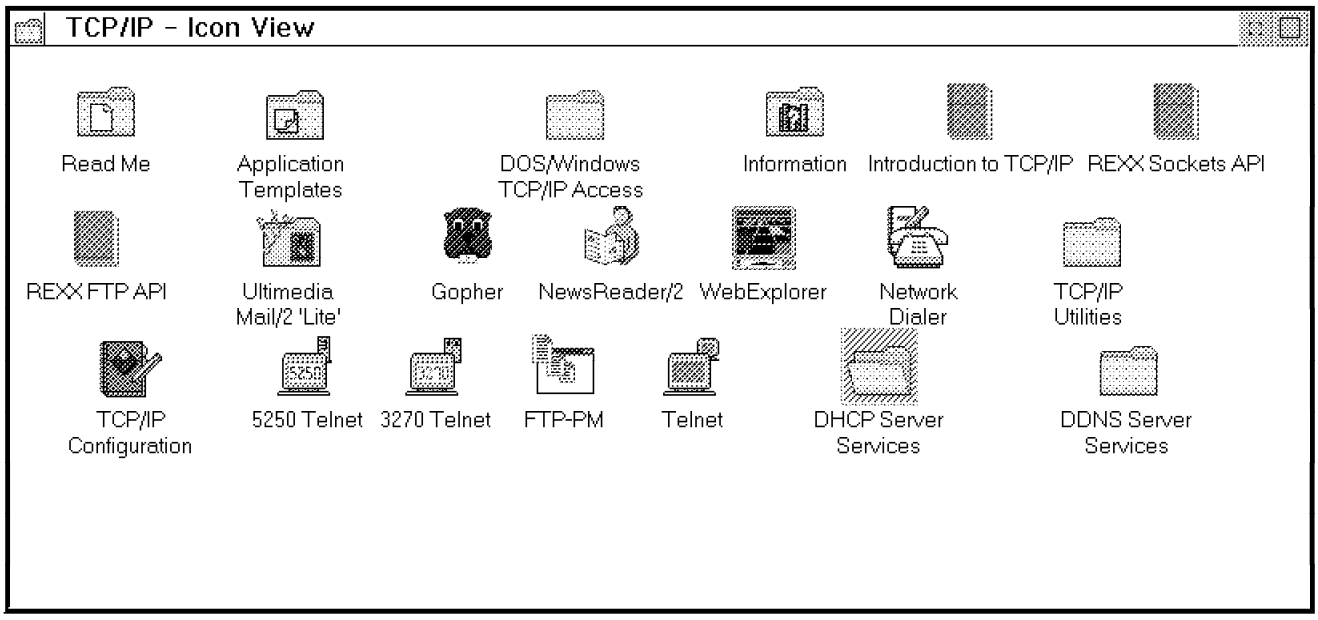


Figure 234. TCP/IP Folder

Double click the **TCP/IP Configuration** icons to bring up the notebook shown in Figure 235.

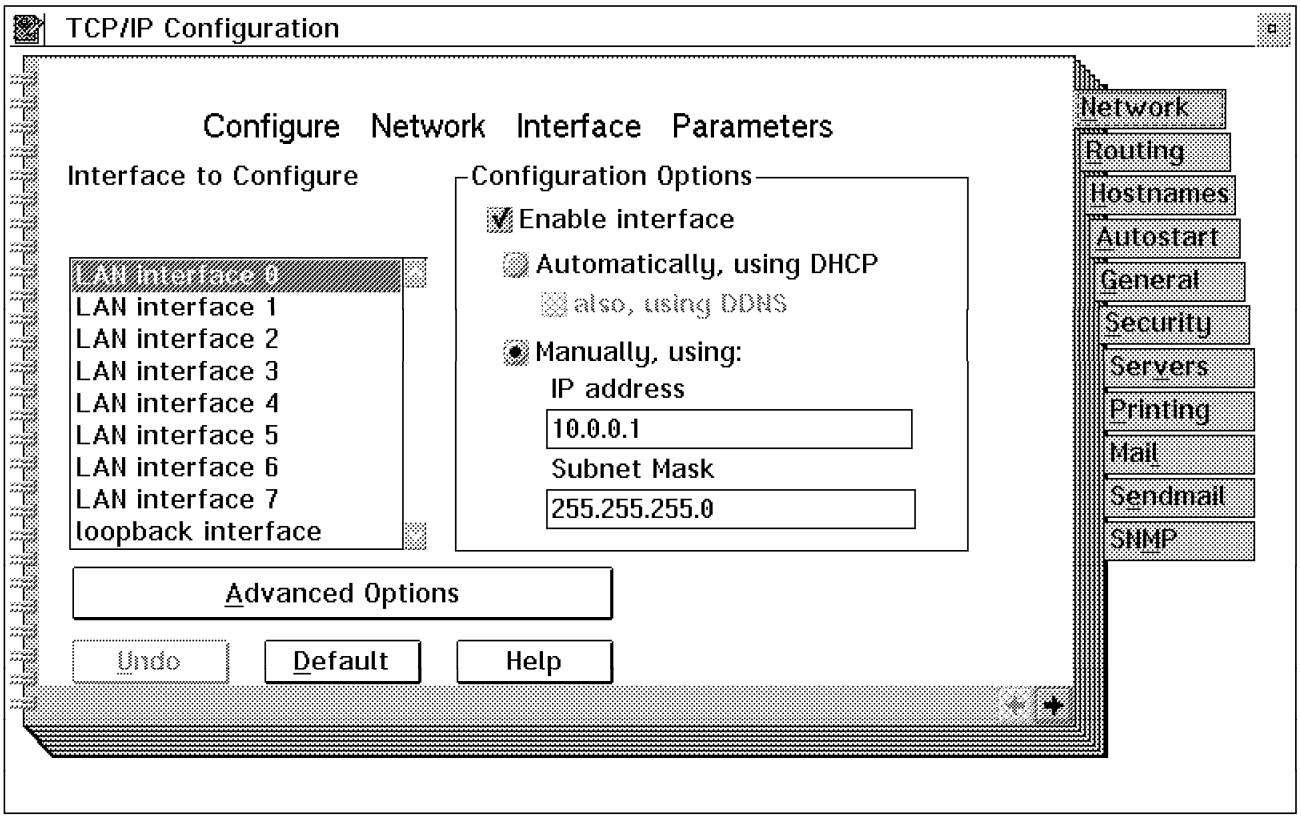


Figure 235. TCP/IP Configuration Notebook - Network Settings

Enter the IP address manually as 10.0.0.1 and the subnet as 255.255.255.0.

Click the **Hostnames** tab to move to the Hostnames page and enter the values shown below.

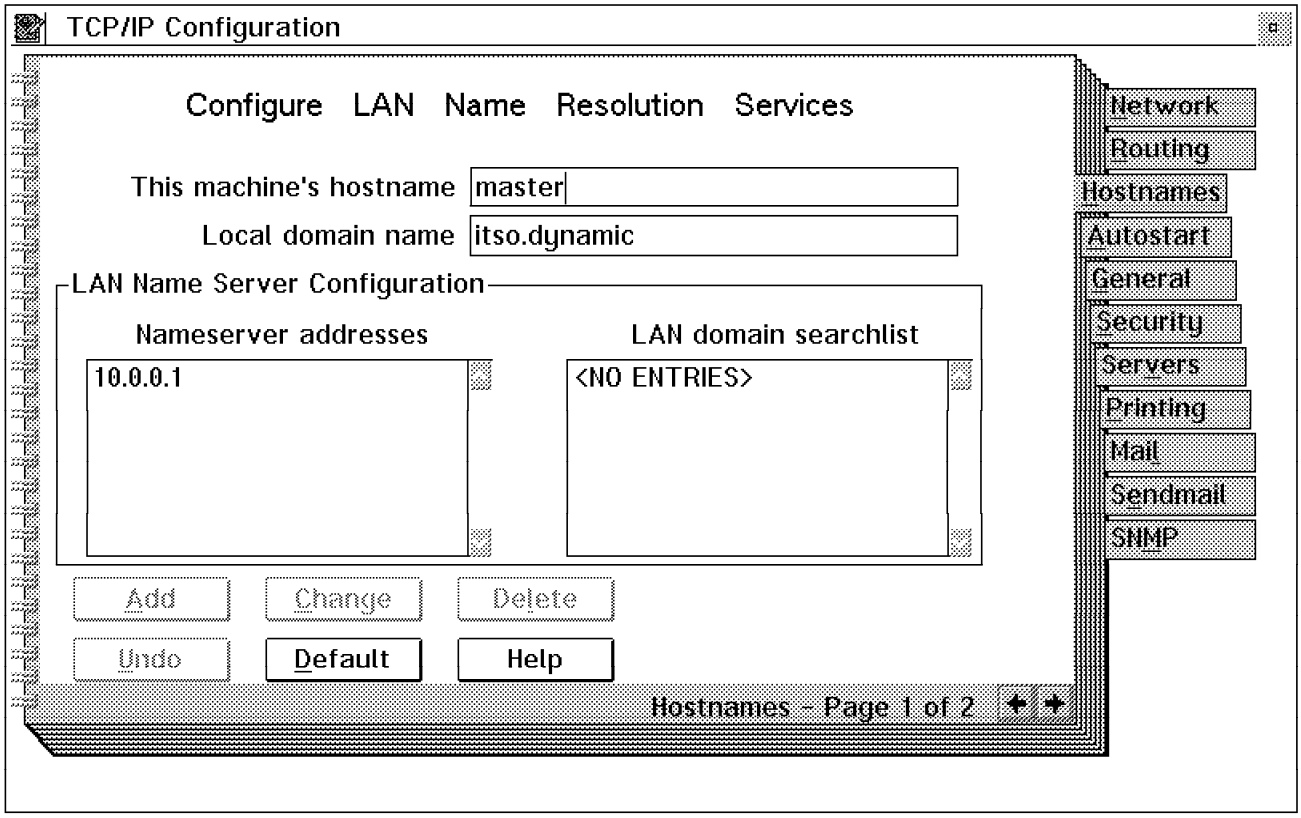


Figure 236. TCP/IP Configuration Notebook - Hostname Settings

Close the notebook and reboot the PC for the changes to take effect.

14.2.4 Setting Up the DHCP Server

Once you have installed TCP/IP onto the server and configured its IP address, you need to configure it for DHCP and DDNS.

Inside the TCP/IP folder on the OS/2 desktop, you will find a folder called DHCP Server Service. Double click this to open the folder as shown in Figure 237.

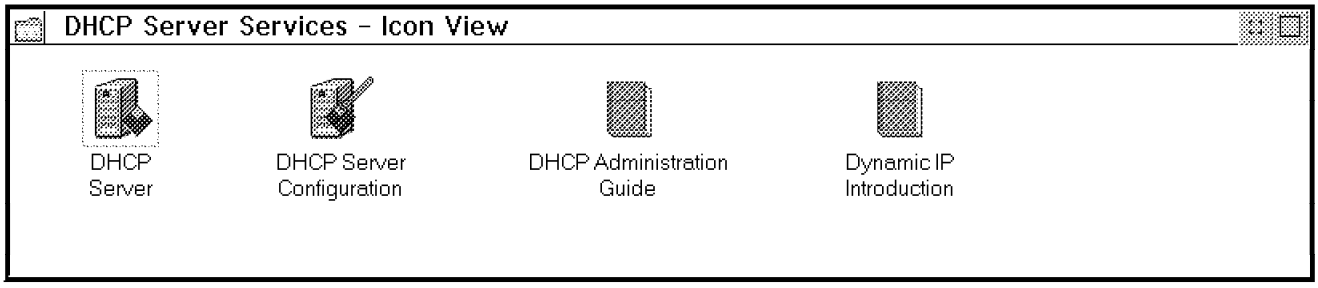


Figure 237. DHCP Server Service Folder

Double click the **DHCP Server Configuration** icon to start the DHCP configuration tool as shown in Figure 238 on page 283.

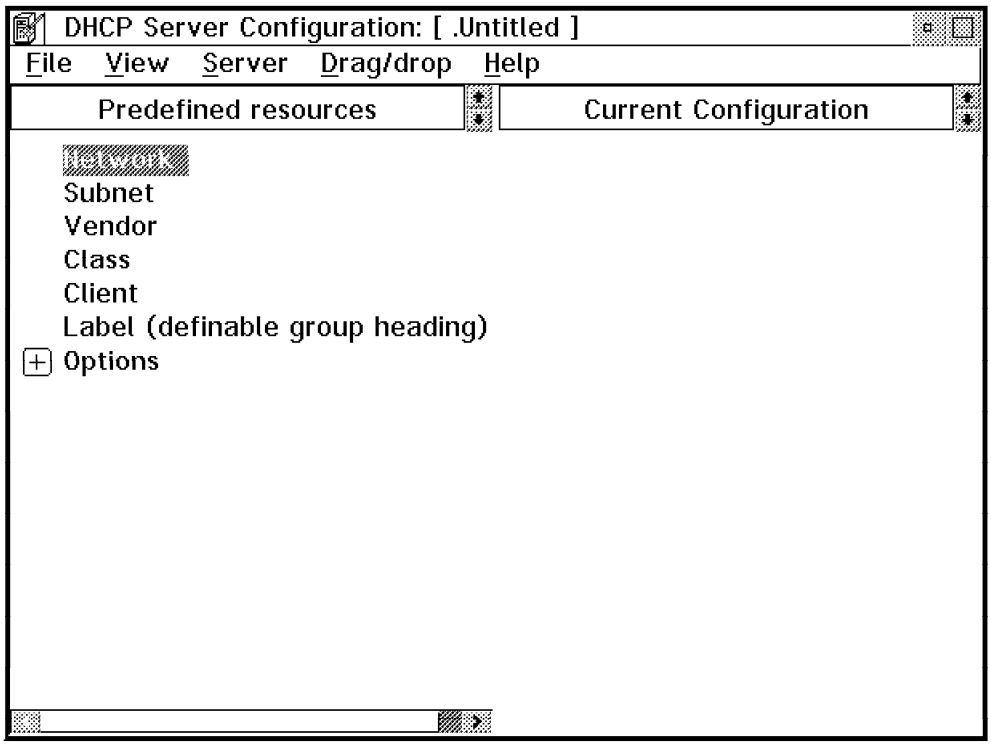


Figure 238. DHCP Server Configuration Tool

This tool operates using drag-and-drop. Create a new network definition by clicking on **Network** in the left window with the right mouse button and drag this to the right window and release. You can now double click this in the right window to edit it. This will show the Network window as in Figure 239 on page 284.

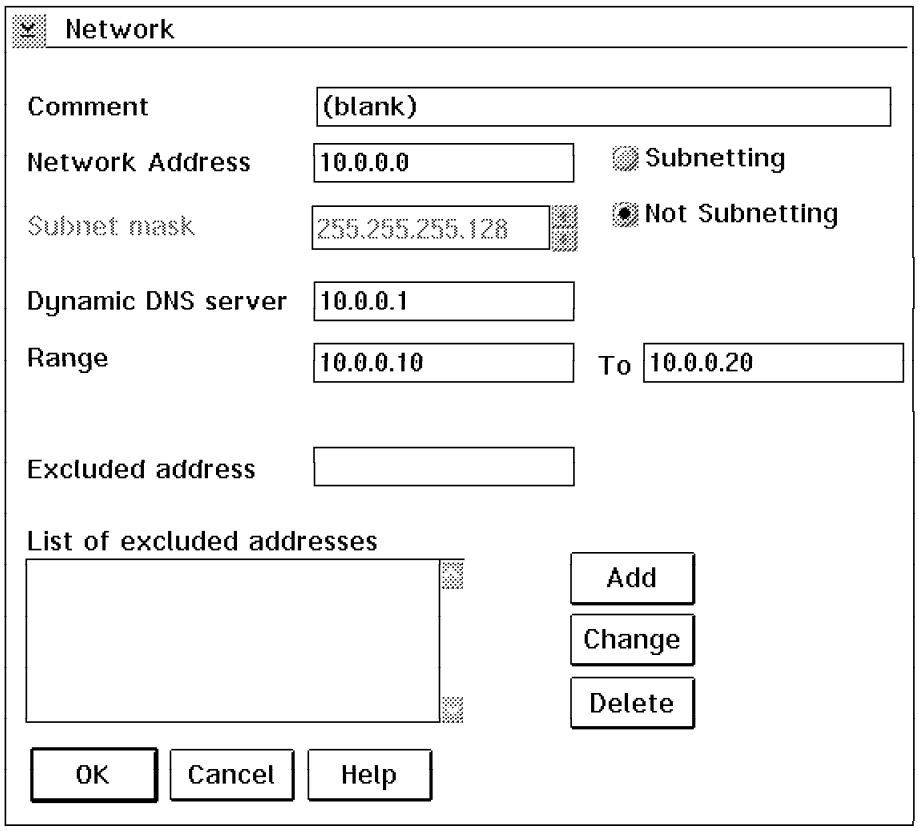


Figure 239. DHCP Server - Network Window

Enter 10.0.0.0 as the network address and select the **Not Subnetting** radio button. Enter 10.0.0.1 as the Dynamic DNS server and enter a range of addresses to control from 10.0.0.10 to 10.0.0.20. Click **OK** to close this window and return to the main window as you saw in Figure 238 on page 283.

Now click on the plus sign next to Options to expand the list and then click again to expand the list called Base Options. Select **Option 6 Domain Name Server** and drag this to the right window. Drop it onto Network. You can now expand Network and double click on **Option 6 Domain Name Server** to edit it. This will display the window shown in Figure 240 on page 285.

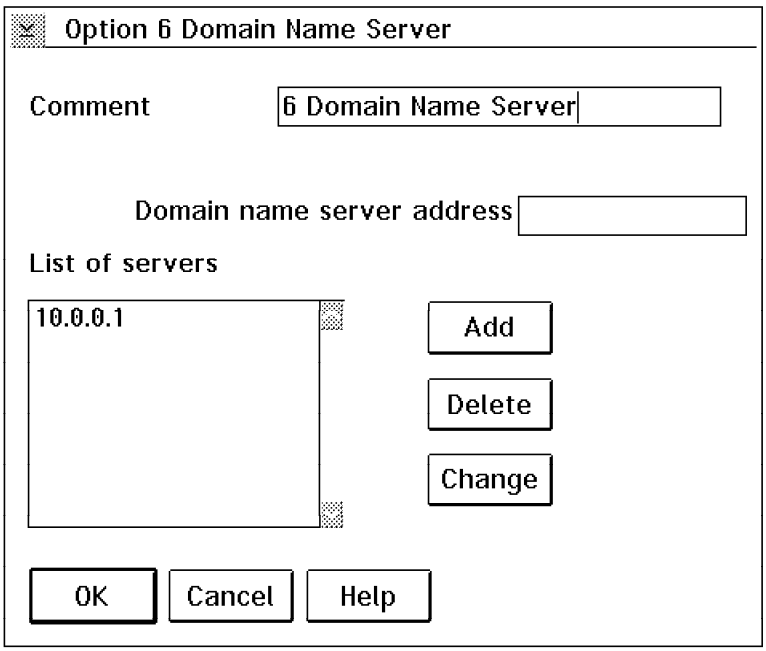


Figure 240. DHCP Server - Domain Name Server Window

Enter the address 10.0.0.1 and click **Add** to add this server to the list. Click **OK** to make this change.

Now add a domain name in a similar way by dragging **Option 15 Domain Name** to the right window and double clicking it. This will allow you to define the domain name of the network as shown in Figure 241.

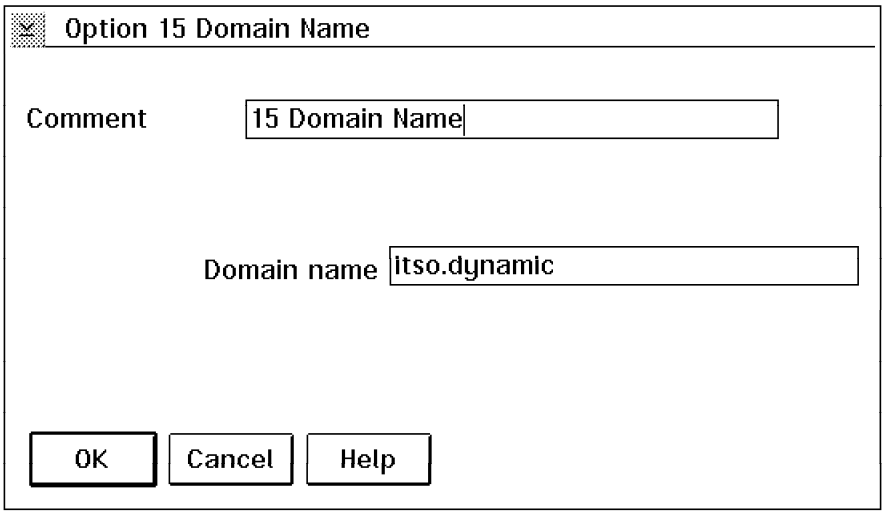


Figure 241. DHCP Server - Domain Name Window

Enter the domain name and click **OK**.

Your configuration should now look like Figure 242 on page 286.

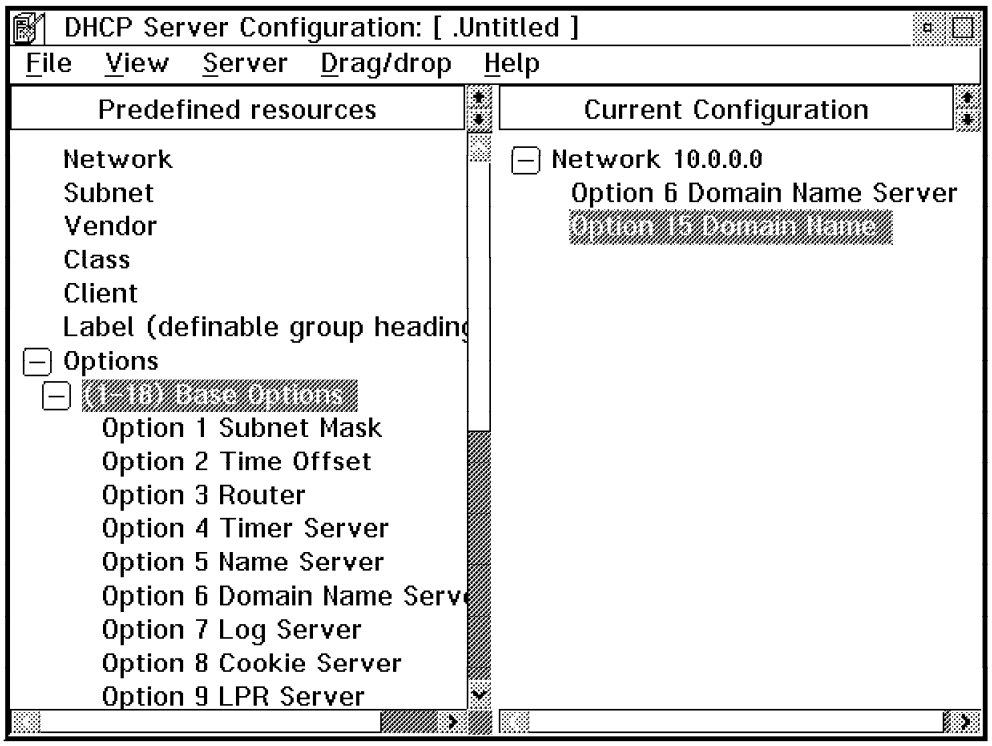


Figure 242. DHCP Server Configuration Tool - Complete

Now we need to tell the DHCP server to inform the DDNS server when it issues IP addresses, so that we can perform reverse name resolution. This will allow us to translate an IP address into a host name. To do this, select **Server** from the menu bar and then select **View/change server parameters**. This will display the following window.

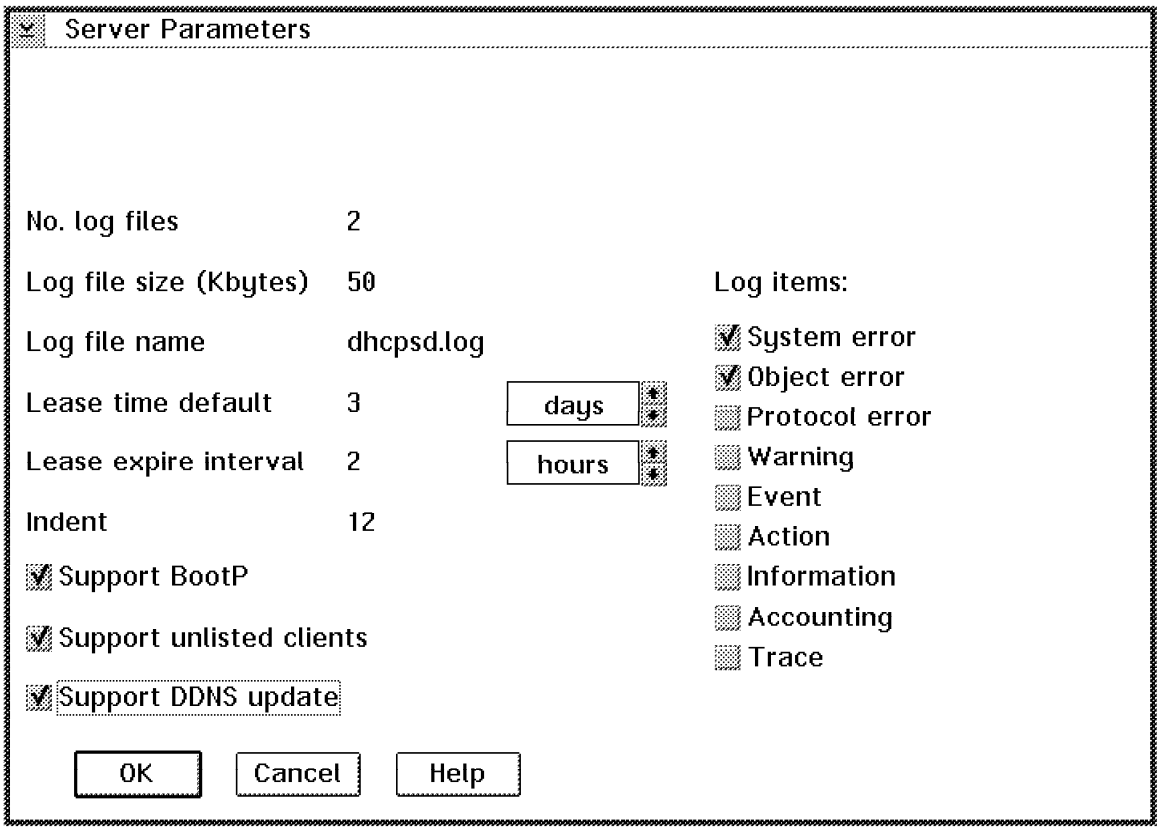


Figure 243. DHCP Server Parameters Window

Select the **Support DDNS update** check box and click **OK** to save these options.

Now from the menu bar select **File**, and from the pull-down menu select **Save** and enter `dhcpsd.cfg` as the file name. This is the name of the default configuration file used by the DHCP server.

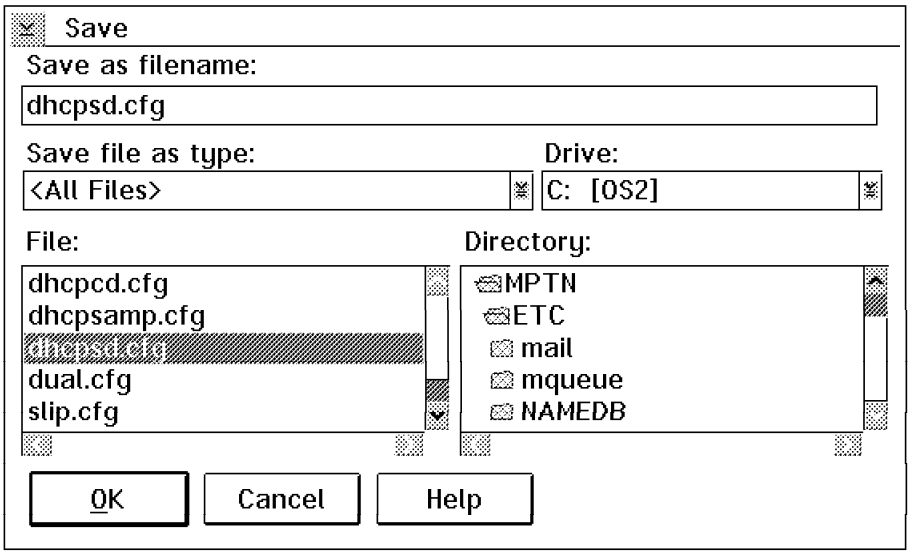


Figure 244. Save Configuration Window

Click **OK** and then click **Yes** to overwrite the old file.

Now select **File** from the menu bar and from the pull-down menu select **Update DDNS data file**. This will create the file C:\MPTN\ETC\DHCPSD.DAT. Ensure that you do this after saving the configuration file.

You can now exit this window.

Command Line Option

You can do most of this from the command line by creating a text file called c:\mptn\etc\dhcpsd.cfg with the following contents:

```
numLogFiles          2
logFileSize          50
logFileName          dhcpsd.log
leaseTimeDefault     3 days
leaseExpireInterval  2 hours
supportBOOTP  yes
supportUnlistedClients  yes
logItem     SYSERR
logItem     OBJERR
#.indent 12

updateDNS "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q""

network     10.0.0.0  10.0.0.15-10.0.0.20    #.name (blank)
{
    #.ddns 10.0.0.1
    option 6       10.0.0.1    #.name 6 Domain Name Server
    option 15       itso.dynamic    #.name 15 Domain Name
}
```

You must still go into the configuration tool described above and load this file. Then create the DHCPSD.DAT file by selecting **File** from the menu bar and from the pull-down menu selecting **Update DDNS data file**.

14.2.5 Setting Up the DDNS Server

To configure the DDNS server you need to create three files in the directory C:\MPTN\ETC\NAMEDB. The exact format of these files is a secret handed down through generations of network administrators. However, as outsiders who have partially cracked the mystique surrounding these ancient rituals, we can show you examples in the three figures that follow without fear of persecution.

```
;
; NAMED.BT file for name server configuration.
;
; type       domain                      source file or host
;
; the next line defines us as the primary name server
; for itso.dynamic. It says to use the file NAMED.DOM
primary  itso.dynamic              c:\\mptn\\etc\\namedb\\named.dom dynamic
;
; the next line says we are the primary server to
; resolve host names from 10.0.0.xxx
primary  0.0.10.in-addr.arpa       c:\\mptn\\etc\\namedb\\named.rev dynamic nokeytosec
```

Figure 245. C:\MPTN\ETC\NAMEDB\NAMED.BT

```
; define that everything gets the domain name added by default
$ORIGIN itso.dynamic.

; define some magic numbers for this domain
; specify the domain twice incase we aren't heard the first time
@ IN SOA master.itso.dynamic. master.itso.dynamic. (
                      48 86400 300 864000 3600 300 )

; specify who is the name server
@ IN NS master.itso.dynamic.

; hardcode how to find the name server
master      IN      A       10.0.0.1

; hardcode how to find rs600012
rs600012    IN      A       10.0.0.2
```

Figure 246. C:\MPTN\ETC\NAMEDB\NAMED.DOM

```
; everything is backwards to resolve hostnames
; specify that we are using 10.0.0.???
$ORIGIN 0.0.10.in-addr.arpa.

; magic numbers
@ IN SOA master.itso.dynamic. master.itso.dynamic. (
                      47 86400 300 864000 3600 300 )

; define name server
@ 0.0.10.in-addr.arpa.  IN  NS   master.itso.dynamic.

; hardcode nameserver's address
1          IN  PTR    master.itso.dynamic.

; hardcode rs600012's address
2          IN  PTR    rs600012.itso.dynamic.
```

Figure 247. C:\MPTN\ETC\NAMEDB\NAMED.REV

These files need to be pre-processed before they can be used. From an OS/2 window enter the following the commands (the output is also shown):

```
C:\> set SACRIFICE=3 virgins and a goat
C:\> start named
C:\> ddnszone
*********************************************************************
Make sure the DDNS nameserver is down before continuing.  When you
are ready, press 'enter' to continue.

<--ignore this and press Enter -->

*********************************************************************
Boot file is                              :  C:\MPTN\ETC\NAMEDB\NAMED.BT

*********************************************************************
Primary nameserver name (used in keyfile):

     If this is the correct primary name, press 'enter' to continue.
     Otherwise please enter the fully-qualified primary name.

master.itso.dynamic           <---------------------- you enter this

New primary nameserver name (for keyfile) : master.itso.dynamic

*********************************************************************
Primary dynamic domain found         :  itso.dynamic
The associated configuration file is :  c:\mptn\etc\namedb\named.dom

Do you wish to add or modify the KEY RR in this configuration file?
If not, enter "N" to skip this file.  Otherwise, just press
'enter' to continue.

     It may take a few seconds to create a key ...

     >>>> SUCCESSFULLY completed update <<<<.

          Configuration file updated : c:\mptn\etc\namedb\named.dom
          Backup of original file in : C:\MPTN\ETC\NAMEDB\dom.BK1

*********************************************************************
Primary dynamic domain found         :  0.0.10.in-addr.arpa
The associated configuration file is :  c:\mptn\etc\namedb\named.rev

Do you wish to add or modify the KEY RR in this configuration file?
If not, enter "N" to skip this file.  Otherwise, just press
'enter' to continue.

     It may take a few seconds to create a key ...

     >>>> SUCCESSFULLY completed update <<<<.

          Configuration file updated : c:\mptn\etc\namedb\named.rev
          Backup of original file in : C:\MPTN\ETC\NAMEDB\rev.BK2

*********************************************************************
Program is complete.
All primary, dynamic domains have been processed.
```

```
C:\> nsupdate -f -g -h *.0.0.10.in-addr.arpa -p master.itso.dynamic
--- NSUPDATE Utility ---
---
Key Gen ...... succeeded ...
```

Note that during this you started the name server before it had been configured and then ignored the warning to stop it. This is because the script tries to find the name for your server from a name server and waits for a very long time for an answer. By starting the name server you quickly get an answer that it doesn't know you.

14.2.6 Starting the Server Tasks

Now you must stop NAMED.EXE by locating it from the OS/2 Window List and pressing Ctrl+C. Start DHCP and DDNS either by using the icons on the OS/2 desktop or from the command line by entering:

```
START DHCPSD -v
START NAMED
```

The option -v causes verbose output which is useful if you want to see the server in action.

14.2.7 Setting Up the Client

If you selected to use Dynamic IP during the installation, then the first time the client starts it will look for a DHCP server.

Warning

Your client may not find your server. It will broadcast a request for a DHCP server and use the one that answers first. If another server replies before your server, then you will not be given the IP address that you want. If this happens then you will have to use a physically isolated LAN for your testing.

If the server is found, then it will pass configuration parameters to your client. To check the address that is being used, you can use the following command from an OS/2 prompt:

```
IFCONFIG LAN0
```

Once the OS/2 desktop has come up, you will see a panel similar to Figure 248 on page 292.

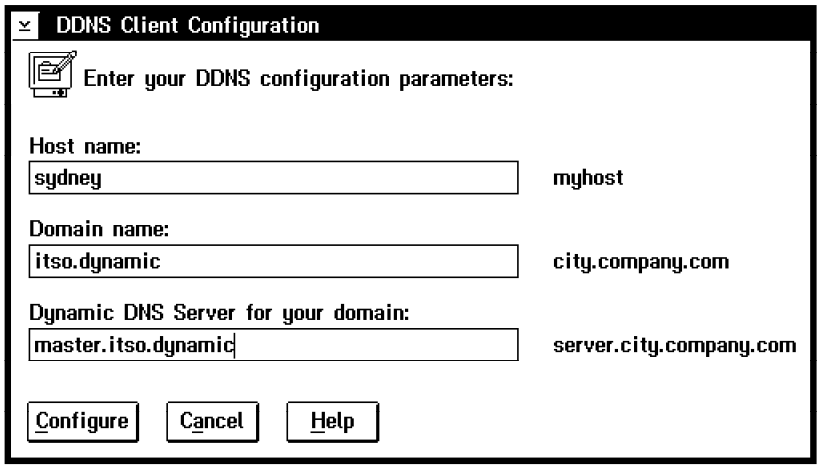
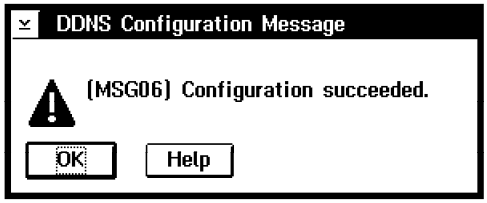


Figure 248. DDNS Client Configuration

Enter the values as shown and click **Configure** to search for the DDNS server. If this is successful then you will see the following confirmation:



Testing the Client

You can check that the client is configured properly by using the command:

```
HOSTNAME
```

This should return the value:

```
sydney.itso.dynamic
```

To check the IP address, use the command:

```
IFCONFIG LAN0
```

You should see:

```
lan0: flags=3c63<UP,BROADCAST,NOTRAILERS,RUNNING,BRIDGE,SNAP>
        inet 10.0.0.10 netmask ff00000x broadcast 10.255.255.255
```

You are now ready to introduce TME 10 Software Distribution.

14.3 Mobile Client

Install the Mobile Client code onto your DHCP client machine (sydney) as described in Part 2, “Setting Up a Mobile Client Environment” on page 41. Configure this client with an NVDM.CFG as shown in Figure 249 on page 293.

```
WORKSTATION NAME:                   sydney
SERVER:                             rs600012 TCP rs600012
PROTOCOL:                           TCP sydney 729 50
REPOSITORY:                         d:\SOFTDIST\REPOS
WORK AREA:                          d:\SOFTDIST\WORK
BACKUP AREA:                        d:\SOFTDIST\BACKUP
SERVICE AREA:                       d:\SOFTDIST\SERVICE
CONFIGURATION:                      CLIENT
MESSAGE LOG LEVEL:                  D
LOG FILE SIZE:                      64000
API TRACE FILE SIZE:                64000
TRACE FILE SIZE:                    64000
MAX USER INTERFACES:                20
MAX ATTEMPTS:                       5
TARGET MODE:                        PUSH
MACHINE TYPE:                       OS/2
TARGET ADDRESS:                     sydney
```

Figure 249. C:\SOFTDIST\NVDM.CFG for Dynamic IP

Now start your client and register with the server using the command:

```
NVDM START
```

If you have automatic target registration enabled on your server then you should find that the client connects and registers with the server. You will find the following appended to the file /usr/lpp/netviewdm/fndlog on the AIX server:

```
1996/11/28 18:09:06 rs600012  20116 FNDRB052I: Target sydney has been
successfully added or updated.

1996/11/28 18:09:17 rs600012  18320 FNDRQ108I: @root rs600012 134 0 N/A
rs600012 : Received successful Auto registration report.
```

If you now issue the command `nvdm lstg sydney -l` on your AIX server, then you will see output that includes the following:

```
Target:               sydney
Description:          AUTO REGISTERED
Server name:          rs600012
Type:                 MOBILE CLIENT
Operating system:     OS/2
Target address:       SYDNEY
Domain address:       SERVER
LAN address:
Network:              TCP sydney
```

The important point to notice is that the value `Network:` has come directly from the NVDM.CFG on the client. This is how the server will contact the client to perform change management activities. It will use the TCP/IP hostname sydney.

This means that if sydney is allocated a different IP address from 10.0.0.10, then the server will still be able to find it. We can demonstrate this by changing the range of addresses administered by master to force it to allocate sydney a different address.

To do this, start the DHCP Server Configuration tool on master, as described in 14.2.4, “Setting Up the DHCP Server” on page 282. This will display the window shown in Figure 250 on page 294.

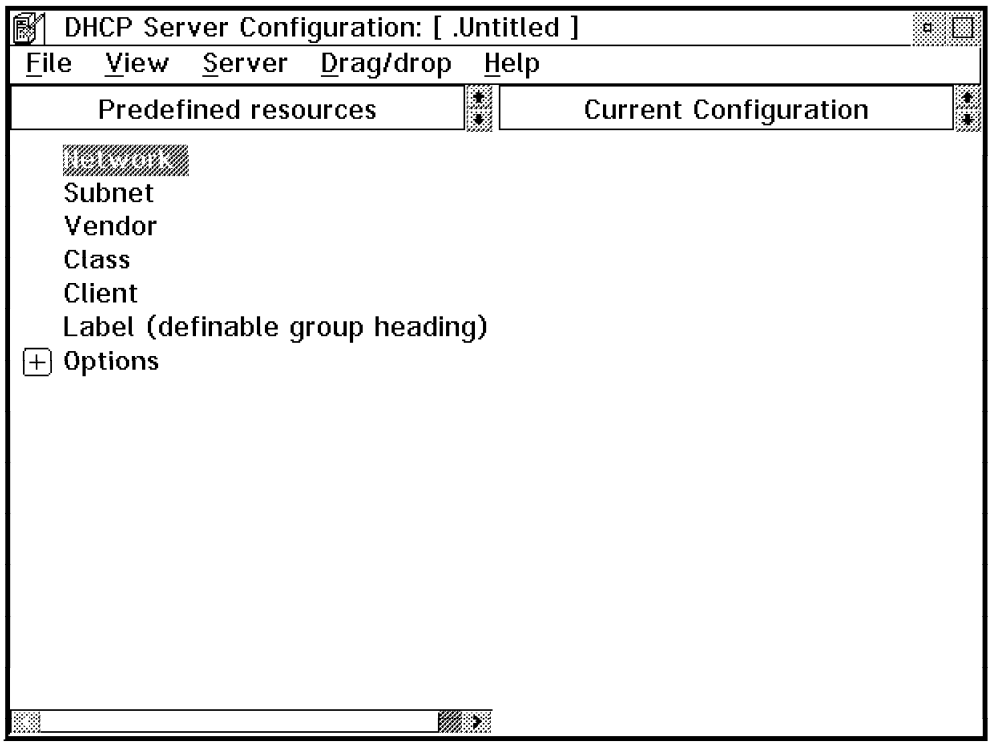


Figure 250. DHCP Server Configuration Tool

Select **File** from the menu bar and choose the option **Open**. Select the file C:\MPTN\ETC\DHCPSD.CFG. Now double click Network 10.0.0.0 in the right window to bring up the Network window seen below.

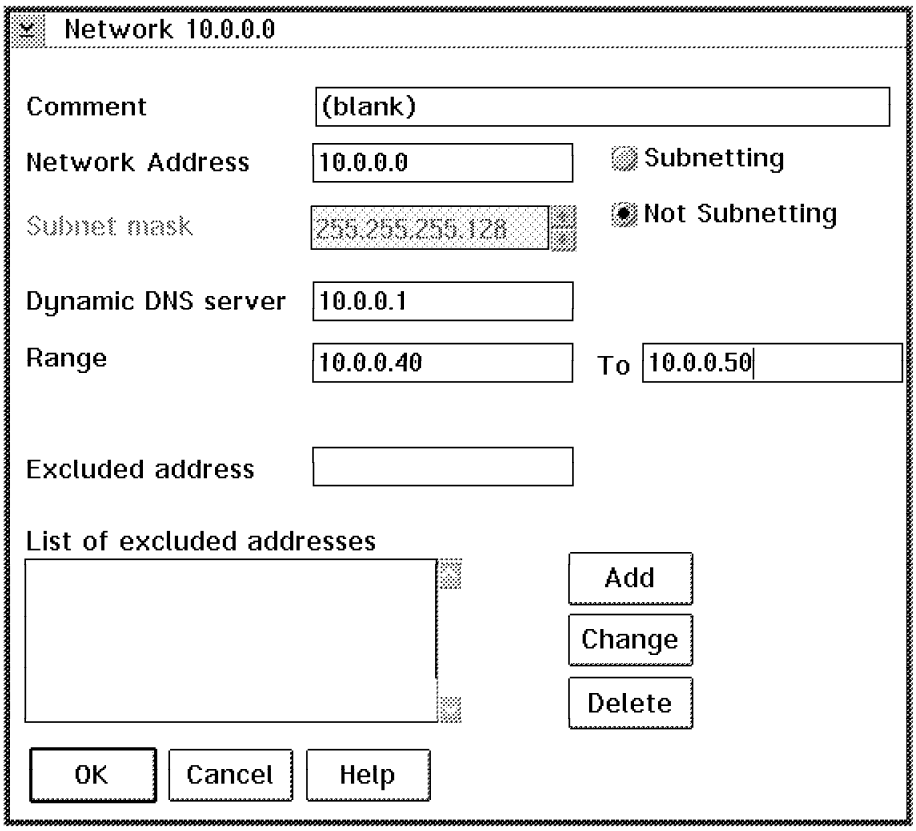


Figure 251. DHCP Server - Network Window

Change the range to be from 10.0.0.40 to 10.0.0.50 and click **OK**. Save this file and exit.

Now restart the DHCP server by closing it with Ctrl+C and starting it with the command:

```
DHCPSD -v
```

Reboot the DHCP client, sydney, to force it to reconnect to the DHCP server. The server will ignore its initial request to use 10.0.0.10, since this is no longer in its pool of addresses, and will issue a new address of 10.0.0.40 once the client gives up waiting for a DHCP server to issue it with the old address. The DDNS server will be updated with the new address.

Once the client has rebooted, enter the following from an OS/2 window on sydney:

```
NVDM CONNECT
```

The connection will be successful, because the server used the hostname SYDNEY to contact the Mobile Client and did not use the IP address which has now changed.

If you repeat this using the NVDM.CFG file in Figure 252 on page 296, then changing the IP address will cause the connection to fail.

```
WORKSTATION NAME:                brisbane
SERVER:                          rs600012 TCP rs600012
PROTOCOL:                        TCP 10.0.0.10 729 50
REPOSITORY:                      d:\SOFTDIST\REPOS
WORK AREA:                       d:\SOFTDIST\WORK
BACKUP AREA:                     d:\SOFTDIST\BACKUP
SERVICE AREA:                    d:\SOFTDIST\SERVICE
CONFIGURATION:                   CLIENT
MESSAGE LOG LEVEL:               D
LOG FILE SIZE:                   64000
API TRACE FILE SIZE:             64000
TRACE FILE SIZE:                 64000
MAX USER INTERFACES:             20
MAX ATTEMPTS:                    5
TARGET MODE:                     PUSH
MACHINE TYPE:                    OS/2
TARGET ADDRESS:                  brisbane
```

Figure 252. Wrong C:\SOFTDIST\NVDM.CFG for Dynamic IP

In this case you will be able to issue the NVDM CONNECT command successfully from the client, after changing the IP address. This will add an open and a close connection window request to the server's queue. When the server tries to open the connection window it will look for a client with an IP address of 10.0.0.10 and will not find it. The following messages can be seen in the file /usr/lpp/netviewdm/fndlog on the AIX server:

```
1996/11/29 14:01:30 rs600012  21924 FNDRB052I: Target brisbane has been
successfully added or updated.

1996/11/29 14:01:41 rs600012  18320 FNDRQ108I: @root rs600012 141 0 N/A
rs600012 : Received successful Auto registration report.

1996/11/29 14:04:30 rs600012  18320 FNDRQ034I: @root rs600012 143 0 N/A :
Open connection window request completed store in the
local database.

1996/11/29 14:04:34 rs600012  18320 FNDRQ034I: @root rs600012 1
44 0 N/A N/A : Close connection window request completed store in the
local database.

1996/11/29 14:04:45 rs600012  20796 FNDRX100W: Failed to connect to DACA
at client brisbane with outstanding request.
The system will retry the connection later.
```

You can see that this request will never be successful by issuing the command `nvdm lstg brisbane -l` on your AIX server. This will produce the following output:

```
Target:               brisbane
Description:          AUTO REGISTERED
Server name:          rs600012
Type:                 MOBILE CLIENT
Operating system:     OS/2
Target address:       BRISBANE
Domain address:       SERVER
LAN address:
Network:              TCP 10.0.0.10
```

Both of the NVDM.CFG files presented here are valid but only the values shown in Figure 249 on page 293 will work with Dynamic IP.

14.4 Using Other Operating Systems

Throughout this chapter we have used OS/2 as the Mobile Client operating system. With TCP/IP for OS/2 Version 3.1 we can use DHCP and DDNS which we have seen is a required combination to run TME 10 Software Distribution in a Dynamic IP environment.

AIX, Windows 95 and Windows NT all support DHCP and can therefore be used as a basis for running TME 10 Software Distribution in a Dynamic IP environment. At this time none of these platforms support DDNS. Further testing would be needed to see whether this is likely to cause problems or not.

14.5 Using the Internet

Usually you will use a private internet, often termed an *intranet*, to run IP. Many companies have firewall links to the real Internet for mail and World Wide Web access. Usually these firewalls are configured to allow those on the inside to view external sites via proxy or socks servers, but to prevent the general population of the Internet from gaining access to resources on the Intranet. Sometimes access to specific internal hosts is allowed to access HTTP servers, although it is more common to place these outside the firewall.

Users on the Internet often gain access through Internet Service Providers (ISPs) who hold a range of IP addresses that are valid Internet addresses. These are allocated to modem-connected customers from a pool using protocols such as PPP. This is similar in principle to DHCP, although different protocols are used, in that an IP address is dynamically allocated to a user when required.

As we mentioned above, DDNS can be used independently from DHCP, which opens the possibility of using a private DDNS server with IP addresses on the public Internet. For example, users with no means of accessing their company′s network could access the Internet through an ISP. They could then use DDNS to connect to the company′s DDNS server and register their IP address.

Subsequently they could access their Software Distribution for AIX server to receive software updates. This depends, of course, on both the DDNS server and the TME 10 Software Distribution server being accessible from the Internet, something which usually you should be very cautious about implementing.

DDNS provides extensive security features, allowing only authorized clients to change the IP addresses of registered names. Also, TME 10 Software Distribution uses its own socket and its own format for data exchange, which limits the chances of anyone who intercepts the transmission understanding it. Even so, this would only be a sensible option for unsecured data in specialized circumstances.

14.6 Conclusions

The Mobile Client features of TME 10 Software Distribution and Dynamic IP are both intended to benefit mobile computer users. In this chapter we have seen how the two work together to provide users with the ability to move around a network and still access all of the change management services of their home Software Distribution for AIX server.

To summarize:

- Using Dynamic IP simplifies configuration problems for mobile computers.
- Using DDNS alongside DHCP allows us to keep the same hostname when we move in our network and change IP address.
- TME 10 Software Distribution Mobile Clients can work unaffected in a Dynamic IP environment, provided we always use the hostname and avoid using IP addresses to specify targets.

Part 5. Implementing Mobile Clients

In this part of the redbook we provide some input to the planning process that should occur before you start implementing Mobile Clients within your organization. The preceding sections of this book have looked at specific technical examples involving Mobile Clients. In this section we look at general issues, such as factors to consider when selecting how to connect Mobile Clients remotely.

We also look at security and audit problems, and provide a sample menu application that can be used as a front end to the different remote connection types.

© Copyright IBM Corp. 1997

Chapter 15. Mobile Clients in the Enterprise

Why you have laptop computers in your organization?

Once you understand what you use mobile computers for, you can define the systems management needs and hence your requirements for software distribution.

15.1 Introduction

Before we begin, let us define some of the terms that we will use in this chapter:

Laptop	An Intel-based mobile PC such as an IBM ThinkPad, running OS/2 or Windows.
Personal Digital Assistant	A hand-held computer such as an Apple Newton.
Cost of Ownership	The total cost to your company of owning a computer system. This includes purchase cost of hardware and software, proportion of network costs and all support costs, including hidden costs such as lost productivity.
Tight Systems Management	A systems management solution where all of the control and accountability lies with the system administrators. The user has little or no control over the environment. This kind of solution suits users, such as tele-sales staff who use specific systems and have no requirement to use their PCs for anything else.
Loose Systems Management	A systems management solution where some of the control is given to the user, for example, choices over what software to install and where to install it to are made by the user. This kind of solution is best for knowledge workers and IT staff who have a range of requirements that cannot be met by a tight solution.

15.2 Business Function

Companies use laptop computers in a variety of ways:

- As executive toys
- As status symbols
- To allow/encourage employees to work from home
- For staff who travel
- To run IT systems while mobile

Your company's justification for using laptops could be anything from no justification whatsoever to being the only way you can do business.

It is very important that you understand what purpose the laptops in your company serve before you design a software distribution system for them. If you

© Copyright IBM Corp. 1997

implement a tight systems management solution for laptops that are used as executive toys (3270 terminals with Microsoft Golf), then your solution is a bad match for the requirements. Equally if you implement a loose systems management solution for your company's key operational systems, then you will have also misunderstood the requirements.[5]

The key things to understand about how your company uses laptops are:

1. What business function do they serve?
2. How critical is this business function?
3. What is the effect of downtime on your business?
4. What level of IT skill do the users have?
5. Who is responsible for the availability of these systems?
6. (What version of Microsoft Golf should you install?)

15.3 Alternative Solutions

If you are being asked to provide software distribution for laptops then it is already too late to consider alternative solutions. If you are designing a total system and are looking at software distribution as an aspect of this system, then you should consider the alternatives.

The following table lists some of the alternatives that you could consider.

Table 4. Alternatives to Using Laptops

Business Problem	**Laptop Solution**	**Alternative Solution**
Status	Give all users above a certain level laptops.	Indicate importance with larger desks for important people.
Work from home and office	Carry laptop from home to office every day.	Use a computer at home and one at the office. Move data between them using diskettes or LAN Distance. If necessary use external hard disks.
Mobile e-mail and document writing	Use laptops on trains and planes.	Use a Personal Digital Assistant for e-mail and writing while travelling.
Occasionally mobile users	Issue laptops to all.	Provide a pool of laptops for use when mobile.
Users who move within your organization	Issue laptops.	Provide standard workstations at all locations to allow any user to work from any location.
Often mobile users	Laptops.	Few alternatives.

[5] Both of these mistakes are bad for your company, although the former is more likely to get you fired.

15.4 The Cost of Laptops

If you are still reading, then the arguments above have failed to dissuade you. In that case you should be aware of what laptops are likely to cost your company.

The Gartner Group have estimated that the cost of ownership of PCs is about $11,900 per seat per year. 73% of ownership expense comes from personnel costs (*The Cost of LAN Computing: A Working Model*, Gartner Group Inc., Feb 7, 1994). This was based on desktop PCs. Now add to this the fact that laptops cost about three times as much as desktop PCs and have a much shorter lifetime since they cannot be upgraded to any real extent. Also remember that one of the hidden costs of ownership is peer support, where people turn to their colleagues for help rather than using the official support section. Truly mobile users do not always have this option since they may be in a hotel or customer site when they need help.

To understand why PCs are so expensive, we need to look at how they are used and contrast this with traditional host-based systems. In the good old days users arrived at work, switched on their terminals, logged onto one or more systems and started work. They worked with specific applications that were designed to support the business functions performed by a user or group of users. Application updates were installed onto a single host system and operating system functions were undertaken by a central group of highly trained experts.

With PCs, each machine has its own operating system. This is looked after by the users themselves with little input from the experts. The user is required to boot the hardware, load the operating system, log on to multiple remote operating systems and navigate through many applications to perform his/her business functions. While this is, usually, more time consuming than using a terminal-based approach, much of it is performed automatically and the total process is within the ability of the average user.

The expense comes in when things do not work as expected or upgrades are required. In a host-based environment, problems cannot be resolved by individual users and become the responsibility of a centralized team. This team is employed for this task and approaches it in a professional manner. As a specialized team, they develop detailed experience of how to solve these problems and can do their jobs in an efficient manner.

With distributed systems, fixing problems and installing new software must involve the computers which are, in a branch environment, located at remote sites. If users have a problem, then the first reaction is to try to fix it. If they can′t fix it then they ask their colleagues and finally they turn to the support section. This is a very inefficient way to solve the problem and when duplicated over multiple sites results in hundreds of people learning from scratch how to fix something that members of a central support team already know how to fix. The people who are fixing the problems were never employed for this task and neglect their real duties while they ″play″ with the PCs.

Software installation is an even worse problem with PCs. How many of your users know what type of video card they have in their PCs, or what address to use for their default IP router? Hopefully very few. Probably quite a lot!

Centralized electronic software distribution provides standard installation of software which reduces support costs, and removes the need for every underwriting expert, bank teller and policeman to be an IT expert as well.

Laptops are worse. Laptops often have multiple configurations to support different LAN configurations. They have different hardware requiring a new set of installation choices to be learned, and they are shinier and brighter toys that attract more spectators when they have problems.

It becomes very obvious that if you implement laptop solutions for your business, then you need systems management in place to control the costs.

15.5 Systems Management Requirements

Systems management is the discipline of building computer systems to manage other computer systems. When things become too large to do them by hand, or the task is so repetitive that it becomes a chore, we turn to computers to do the task for us. When that task is software installation or running backups, then the computer system that we develop to do this for us is termed a systems management solution.

In all other respects, building a systems management solution is identical to building any other IT system. We start by defining the requirements, then we design the solution and finally we build and support it, using a little of our experience of what is possible along the way.

While you do this, you should consider the following:

- What are the real requirements for software distribution?
- Which of these are essential, and which are nice to have?
- Can I reuse an existing solution for desktop software distribution?

15.6 Laptops Vs Desktops

Software distribution requirements for laptops are not the same as those for desktops. At the simplest level, even if you install the same software onto desktops and laptops, you will need to use different options. For example, if you install OS/2 and basic communications software such as MPTS, then you will probably need to install power management, PCMCIA support, different token-ring drivers, and a different video driver for your laptops. So straight away laptops introduce a new set of problems for software distribution.

It is very likely that you will need to install extra software onto laptops. After all, there must be some reason they are used rather than desktops. So you may be required to install LAN Distance or DIALs for laptops, but not for desktops.

So already by introducing laptops we can see that your software distribution catalog is going to have modified versions of existing objects and some new objects.

The real differences come with *how* you install software to laptops. Laptops may connect using different options, they may appear over modems or ISDN links, they may use different IP addresses and they may require software distribution when there is no connection available. Often the connection is too slow for

traditional installation programs to work, or the user may need to collect a change object over a modem link but not install it until much later. So the Mobile Client has been developed and the purpose of this redbook is to deal with the problems that laptops introduce to software distribution by virtue of their differences from desktop PCs.

15.7 Building a Support Team

Once you decide to provide centralized support for laptop users, you will need to build a team capable of carrying out this task. Sometimes you will be augmenting an existing software distribution team to handle laptops. In other cases you will be building a team from scratch.

In either case you will need to have people with the following skills:

- TME 10 Software Distribution skills
- Communications skills (TCP/IP, LAN protocols, modem configuration skills)
- Production system experience
- Possibly REXX, C, and UNIX skills if you wish to build your own utilities and exits

You will have to consider the hours that support is provided, since for many users who are travelling, the main times that they will need help are evenings and weekends. This is different from most desk-based users.

You may also want to consider issuing your users with cellular phones so they can talk to the support areas while the laptop uses the land-based phone for its modem.

Chapter 16. Designing a Network

In this chapter we give some guidance on the various network connections you can use with the Mobile Client and explain the differences.

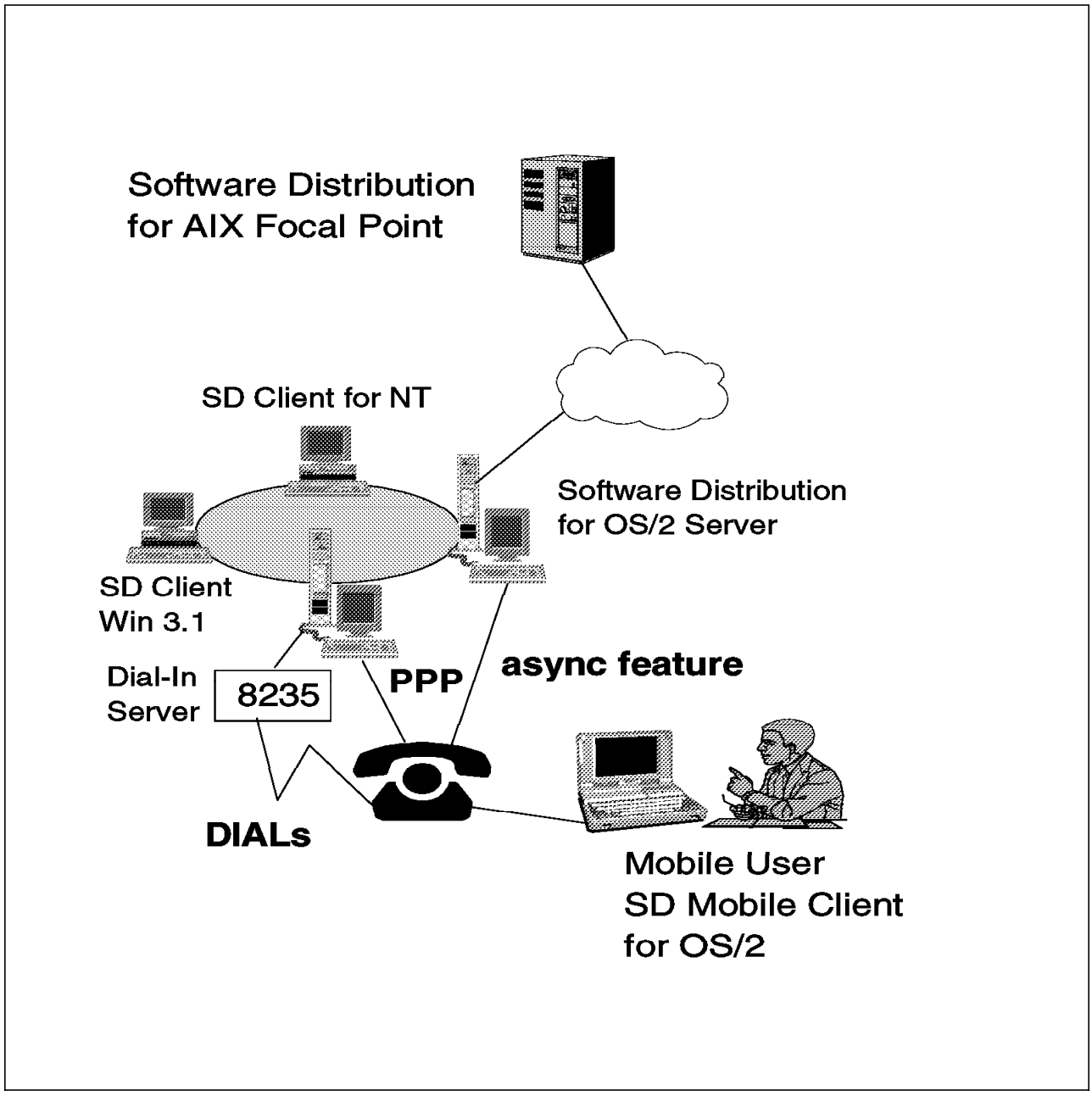


Figure 253. Network Connections

The figure above shows some of the communications options you have that are covered in this redbook.

For example, you can use:

- A standard LAN connection, for example, token-ring

© Copyright IBM Corp. 1997

- A direct link between the Mobile Client and the server using the asynchronous support in TME 10 Software Distribution
- A PPP link
- A DIALs 8235 connection

Also, there are other options, for example:

- Using LAN Distance
- Using ISDN

Which of the above options is best for you depends mainly on your environment. However, we explain some relevant differences in the following sections.

16.1 Using the Asynchronous Support

Using the TME 10 Software Distribution asynchronous support feature you will have TME 10 Software Distribution directly control the modem link, thus eliminating the need for an additional network protocol stack, for example, TCP/IP.

The major advantages of this approach are:

- TME 10 Software Distribution directly controls the link, so it can ensure that the link is usable before transmitting data. TME 10 Software Distribution does not rely on other components to establish the link.
- Since there are no additional protocols used, the overhead is minimal. So the performance is likely to be better.
- The connection is made directly between the server and the remote client, so there are no other security issues to consider, such as access to other LAN resources.

The major disadvantage with this approach is that, since no intermediate protocol layer is used, the communication part is not as interchangeable as with, for example, a PPP or IBM 8235 connection.

Therefore, this approach seems to be most feasible for users who work most of their time remotely, that is away from their office. In that case you don't have to deal with transparently switching between WAN and LAN.

16.2 Using a PPP Link

In Chapter 9, "Using the Mobile Client with PPP" on page 131 we describe how to set up and use PPP with TME 10 Software Distribution. This can be used to enable a link directly between the client and the server, or to provide access from the client to the IP LAN via a PPP server. This is neither direct access, such as asynchronous support, nor is it transparent access as with DIALs and LAN Distance. If you operate on the LAN and then use PPP externally, you will be allocated a different IP address and the server will not recognize you.

So PPP is best suited to use when the connection is always remote. PPP is available on all operating systems and can be used for those platforms on which TME 10 Software Distribution does not support asynchronous connections.

16.3 Using a DIALs 8235 Connection

In Chapter 10, “Using the Mobile Client with IBM 8235 LAN Dialer” on page 165, we describe how to use the Mobile Client with a DIALs connection.

The advantages of this solution are:

- You can transparently switch between your standard LAN connection in the office and your dial-in connection. The network protocol layer (for example, TCP/IP and therefore TME 10 Software Distribution) will not see a difference and you can switch just by using the Shuttle option in DIALs/2.
- You will not need additional hardware per TME 10 Software Distribution server, for example, a modem or two per server. You just dial-in to the DIALs server and then have transparent access to all stations on the LAN.

You will, however, need an IBM 8235 unit and a control workstation to be able to use this option.

16.4 Using LAN Distance

Using LAN Distance with TME 10 Software Distribution is similar to using a DIALs 8235 connection. In both cases the communications product is first used to establish the connection before TME 10 Software Distribution can establish its connection.

To use LAN Distance with TME 10 Software Distribution you have to perform the following steps:

1. Install the LAN Distance client on your laptop and define a connection to the LAN Distance server.
2. Create a user account on the LAN Distance server.
3. Connect to the server using the connection icon or the command line.
4. Pass the user ID and password, if required, to the server. This can be done with the previous step, if the command line is used.
5. Optionally, the LAN Distance server can drop the telephone connection and issue a call-back to your Mobile Client.
6. Now you can connect to the TME 10 Software Distribution server, for example with the NVDM CONNECT command.

Some advantages of LAN Distance are:

- Authentication and call-back feature (security)
- Large number of supported communication protocols (TCP/IP, NetBIOS, SNA, IPX)

You can also use different communication protocols in parallel; for example, you can use TCP/IP for software distribution and SNA for 3270 emulation.

We do not have examples of using LAN Distance in this redbook, since it is very similar to DIALs as far as TME 10 Software Distribution is concerned.

16.5 Using ISDN

We do not show examples of using Integrated Services Digital Network (ISDN) with TME 10 Software Distribution in this redbook.

However, ISDN is an important factor in network communications so we include some points to consider here:

- When you intend to use ISDN for your mobile software distribution clients you will need an ISDN card for each of your clients. These cards are normally slightly more expensive than a standard analog modem. However, the network speed is considerably higher.
- The standard network protocol to be used with ISDN is TCP/IP. In order to use it you will normally also need a TCP/IP software that supports your ISDN card.
- Once you have configured TCP/IP properly to run on ISDN you should be able to use TME 10 Software Distribution just as with the other communication features we show in this redbook.

16.6 Using a Standard LAN Connection

When using the Mobile Client in a standard LAN (for example, a token-ring LAN with TCP/IP) the Mobile Client is similar to a standard client. However, the behavior is slightly different since the Mobile Client employs a different architecture from that of the standard client. These differences are described in Chapter 3, “Architecture and Data Flow of the Mobile Client” on page 13. In Part 3, “Simple Scenarios Using the Mobile Client” on page 87 we walk through some basic examples of using the Mobile Client on a LAN to show the architectural differences in practice.

There are two main ways to use a Mobile Client with a LAN connection:

- LAN only
- In association with remote access

We will discuss these in turn.

16.6.1 LAN Only

In some circumstances you may wish to use a Mobile Client that is either connected to a LAN or is stand-alone (for example, a user who works from an office for part of the time and is mobile for the rest.). This user may need to have data tables updated on his/her PC while he/she is mobile. This can be achieved by using the Mobile Client along with Deadline Activation. There is no need for the laptop to have any connection to the server while away from the office for this to happen.

16.6.2 LAN and Remote Access

If your users also need to connect to the server while away from the office, then you must install a solution capable of allowing this. Here you must combine LAN access with one of the options identified above. This is potentially a more complex environment, since you must allow the switching between local and remote operation to occur as shamelessly as possible.

Some points to consider when performing this switch are:

- When I work remotely, can I use the same communication protocols that I use when LAN connected?
- Can I use the same TME 10 Software Distribution configuration in both cases?

For example, some remote communication options support only TCP/IP (such as PPP). Therefore, if you work with NetBIOS while LAN connected you need to adjust your configuration when working remotely. You will also need to update the server entry for this target to use the new protocol.

Communication methods that provide transparent remote LAN access, such as LAN Distance or DIALs, are very well suited for switching between LAN and remote connections, because for such applications as TME 10 Software Distribution the switch is transparent and therefore the configuration does not need to be changed.

16.7 Summary

The following table summarizes some of the network connection options and their features:

Table 5 (Page 1 of 2). Comparison of Network Connections

Function	LAN	Asynch	PPP	DIALs	LAN Distance
Protocols supported	TCP/IP, NetBIOS, SNA, IPX	n/a	TCP/IP	TCP/IP, IPX, NetBEUI	TCP/IP, NetBIOS, SNA, IPX
Transparent to TME 10 Software Distribution	yes	no	no (different IP address)	yes	yes
Transparent switch between LAN and remote	n/a	no	no (peer-to-peer)	yes	yes
HW Prerequisites	n/a	1 modem per client, at least 2 modems per server	1 modem per client, 1 modem per server	IBM 8235, 1 modem per client	1 modem per dial-in server, 1 modem per client
SW Prerequisites	MPTS or equiv.	n/a	MTPS or equiv., TCP/IP	MTPS or equiv., DIALs Client	MTPS or equiv., LAN Distance Client
Supported TME 10 Software Distribution Server Platforms	all	AIX	all	all	all

Table 5 (Page 2 of 2). Comparison of Network Connections					
Function	**LAN**	**Asynch**	**PPP**	**DIALs**	**LAN Distance**
Supported TME 10 Software Distribution Client Platforms	all	OS/2, Windows 3.1	all	OS/2, Windows 3.1, Windows 95, Windows NT (using RAS), UNIX (using SLIP)	OS/2, Windows
Protocol Overhead	n/a	low	high	medium	medium
Ability to isolate errors related to the network	high	low	medium	medium	medium
Configuration effort (server)	medium	medium	high	high (8235 server)	medium
Configuration effort (client)	medium	low	high	low	medium

The following table provides a very brief summary of our recommendations for selecting a connection.

Table 6. Recommendations		
Problem	**Solution**	**Comment**
Laptop users on LAN	Use standard distribution client.	Treat all PCs the same and minimize unnecessary configuration.
Laptop users on LAN requiring deadline activation while away from office	Mobile Client with LAN Connection.	Use local catalog of Mobile Client to effect deadline activation even when away from the office.
File server or workstation in isolated branch with no network connection.	Use asynchronous Mobile Client on file server with modem.	Connection only established when needed.
File server or workstation in isolated branch with no network connection and no telephone.	Use fully disconnected client.	Best check if they have electricity too.
Laptop users, always mobile.	Mobile Client with asynch	Could also use PPP or ISDN.
Laptop users sometimes mobile.	Mobile Client with DIALs or LAN Distance.	Allows transparent remote LAN access.

Note that for normal laptops with the same requirements as desktop users, we recommend using the standard Distribution Client on all machines. You could choose to use the distribution client along with PPP or DIALs if you wish and to avoid using the Mobile Client altogether. This would be a valid solution if the modem links were very fast, or the change objects very small and you could guarantee to connect when needed. However, if you use the Mobile Client then you can exploit its local catalog to transfer compressed change objects over the modem link and make the best use of available bandwidth. The Mobile Client can then install the object locally whenever required.

Remember that the benefits of the Mobile Client to laptops are not purely in its ability to use a modem connection.

Chapter 17. Security Considerations

In this chapter we take a brief look at security issues relating to use of the Mobile Client. We will discuss some of the security and audit factors that need to be taken into consideration when using Mobile Clients. We will also show an example audit log created by using the user exits available with Software Distribution for AIX.

17.1 Introduction

Laptops pose a unique set of problems to those involved with computer security. Laptops are mobile resources that have the capacity to hold a large amount of important corporate data. Laptops are the only component of the IT infrastructure likely to be found outside the office environment and away from the usual physical security of the office building.

Being portable, laptops are prone to theft. Measures such as power-on passwords are no defense to a laptop that has been switched on and left unattended. Because they do operate away from the office, laptops are also required to hold sensitive corporate data locally such as budget plans, customer details and information on business partners. Many organizations have security policies stating that no volatile information should be contained on PCs and that only servers are backed up. For laptops this is not practical since without local data the laptop cannot function usefully away from the office.

An additional problem caused by laptops is the requirement for dial-up access to corporate resources such as e-mail, file servers, mainframe systems, and in our case, software distribution. It is important when designing a software distribution solution involving laptops to ensure that:

1. The security of the Data Center is not compromised by the links required for dial-up software distribution.
2. The security of the laptop is not compromised, since as discussed, the laptop may contain sensitive information that is not otherwise protected.

How you choose to control security depends very much on the data involved, the attitude of the company to security and the architecture of the proposed solution. In this section we discuss some of the relevant points about security of the Mobile Client. We do not address security of laptops in general, since this is too large a subject to cover adequately here.

17.2 Using PPP

If you use PPP with TME 10 Software Distribution as described in Chapter 9, “Using the Mobile Client with PPP” on page 131, then you need to be aware of the security issues that this introduces. In the examples we created a normal user account called pppuser. This had normal user access for our system and a password that was set the same as the user ID. We modified the profile to start PPP automatically on logon.

In a production environment you have two main options regarding PPP. You can assign each user a personal account on the server and make PPP startup optional, for example, by adding it to the end of the attachment script, or you can

© Copyright IBM Corp. 1997

use a shared sign-on with a commonly known password. If you choose to use a shared sign-on then you should limit the access of that user account. In our example, the only thing that pppuser needs to be able to do is to start PPP. Remember that the same account can be used for ftp, rexec and other secure services, so if you try to close the security loopholes, make sure you cover all of them.

If you can adequately protect resources, so the shared PPP logon account can run PPP and nothing else, then the security exposure is the same as allowing someone physical access to your IP network. An intruder can do no more than they could by connecting a machine to your LAN and running TCP/IP.

Many organizations will not allow the use of shared sign-ons. Here you must provide each user with a personal account. This can be a large administrative overhead if many users are defined purely to use PPP and have no other need for an account on the server. Remember also that if the password is set to expire, you must give the users a way to change their passwords. This is a facility not provided with basic PPP.

If your PPP server runs on an operating system other than AIX, then you should ensure that you are familiar with the security arrangements for that operating system before implementing PPP.

You may also wish to consider modifying the PPP login script to audit the times of connections.

Other Dial-up Options

In this redbook we also discuss other ways to achieve remote dial-up access, such as IBM LAN Distance and IBM 8235 LAN Dialer. These products provide their own security features and you should review these before implementation.

17.3 Controlling Access to the Catalog

The catalog within TME 10 Software Distribution typically contains a mixture of standard commercial software, in-house developed packages and data. Access to the catalog needs to be controlled to ensure that these are adequately protected.

Licensing of commercial products can be enforced by using the catalog to see for example, how many copies of Lotus Organizer have been installed. The emphasis today is very much on the network administrator to prove that they have adequate licenses to cover the installed base of a product; the catalog can help with this.

For in-house-developed applications, access control can be just as important if the application provides business facilities that the company requires be limited to certain individuals.

However, data is probably the most critical asset, since items such as rating tables and preferred intermediary lists make fascinating reading for competitors. With Mobile Clients this data can quickly leave the company over modem connections and tracing it can be very hard. In 17.7, “An Audit Example” on page 317, we show how to produce a simple audit trail for Mobile Clients. This

provides some assistance in tracing the movement of change objects but will probably require modifications to meet the requirements of most organizations.

TME 10 Software Distribution offers ways to control access to the catalog. If you wish to implement security, then you should create a user ID within TME 10 Software Distribution for everyone who will use the system. You should assign each of these user IDs with a different password in line with your company's security policy.

When you create a change object in your catalog, you can assign it a Data Access Key (DAK). A DAK is a token to control access to change objects. You can think of DAKs as groups, if you like. When an administrator tries to access a change object in your catalog, the system checks whether that object is associated with a DAK. If it is, then the system checks whether this user has authority to use this DAK (effectively, whether this user is in this DAKs group). If not, then the action is not permitted.

Note

The default user, root, has no password and has access to all of the predefined DAKs.

Additional user security is available with Authorization Profiles. Every user is associated with one Authorization Profile which specifies the tasks that a user can perform within TME 10 Software Distribution. This includes control over management of queues, building of change objects and creation of other users.

If you choose to implement them, TME 10 Software Distribution contains extensive security features.

17.4 Dynamic IP

Chapter 14, "Using the Mobile Client with Dynamic IP" on page 275 discusses Dynamic Host Configuration Protocol (DHCP) and Dynamic Domain Name Services (DDNS), which are referred to collectively as dynamic IP. In a nutshell, these allow TCP/IP hosts, such as laptop PCs, to connect anywhere in your network and be automatically configured without user input. Dynamic IP also allows a machine to keep the same hostname as it moves within your network.

Being relatively new, Dynamic IP includes many security features and if you choose to implement them, your IP network will probably be more secure with dynamic IP than it was before.

The two main security problems with TCP/IP are spoofing and sniffing. Spoofing occurs when a malicious host machine pretends to be one of the machines on your network by using its IP address and mimicking its logon prompt in order to capture user IDs and passwords. Network sniffing can be achieved using monitoring tools such as IBM DatagLANce to capture network packets that are not intended for your machine, and to view the contents. The format of TCP/IP data streams are public knowledge and deciphering simple IP packets is straightforward.

Neither of these issues are peculiar to TME 10 Software Distribution Mobile Clients or even laptops, and solutions do exist.

17.5 Controlling Access to Laptops

As we discussed above, laptops are very likely to have sensitive data held on their local disk. This data is protected physically because the laptop owner keeps the laptop with them or secured at all times. If the laptop is not configured as a server of any kind and that there are no other logical holes in the security process, then the laptop can be considered to be adequately protected by power-on passwords and automatic screen locks. The only other security exposure comes through file servers, where logon profiles can be used for other purposes and Trojan Horse programs can be created on shared drives (for example, XCOPY.BAT on a common drive). Usually these are not problems, because the perpetrator can be identified easily from audit files or file owner information.

If we now introduce TME 10 Software Distribution Mobile Clients then without security we open all of the Mobile Clients to all of the administrators. An administrator can create a procedure and execute it on a Mobile Client to perform any activity that they wish. They can send files from the Mobile Client to the server or directly to their own workstation, and they can delete the log files and remove all evidence of the activity. They can also return modified files to the Mobile Client if they wish.

As we discussed in 17.3, “Controlling Access to the Catalog” on page 314, you can limit the actions of TME 10 Software Distribution users by assigning them Authorization Profiles. You can also use DAKs to control their access to catalog entries. TME 10 Software Distribution also has Target Access Keys (TAKs), which work in a similar way to DAKs but control access to targets. TAKs can be used to create groups of targets that only certain administrators are allowed to control.

As with all security and audit measures, you must weigh the potential risks against the cost of implementation.

17.6 Auto-Registration

A new feature with TME 10 Software Distribution Version 3.1.3 is Automatic Target Registration. This is an optional feature that allows a new target to register itself with a server, relieving the administrator of the chore of pre-registering new machines on the network.

If you choose to implement this option, then you should be aware that anyone with the TME 10 Software Distribution target code (Mobile Client or Distribution Client) can gain the default level of access to your catalog. Without other security measures, this would mean that all of your software would become available to anyone on your network.

If you use the asynchronous support available with TME 10 Software Distribution, then this extends access to people outside your network who have knowledge of your configuration.

17.7 An Audit Example

Source Code Example

This section contains some sample C code and an AIX shell script. To fully appreciate this section, the reader should have some knowledge of C and AIX.

TME 10 Software Distribution has user exits that can be used to provide audit information. As an example, we have developed a user exit and an AIX shell script to produce a report detailing what change objects have been installed on Mobile Clients. You might choose to do this if you are concerned about software that is potentially leaving the company over modem connections. For standard distribution clients you at least know that the software was installed onto a machine within your company.[6] With a Mobile Client it could have gone anywhere.

Our example produces a report that lists which Mobile Clients had software sent to them, when it happened and who submitted the commands.

We do this by modifying the sample user exits for Software Distribution for AIX so that a file is produced that contains details of all installation requests. To produce the audit report, we run a shell script that checks which of the targets identified in the file are Mobile Clients.

All of the steps that follow are performed on the AIX server.

17.7.1 Installing the User Exit

Software Distribution for AIX is provided with a sample user exit that we will modify. It is advisable to back up the file `/usr/lpp/netviewdm/src/fndcx.c` before performing the steps described here.

Replace the supplied user exit sample with the file shown in 17.7.3, “Source Code Listing” on page 321.

Compile and link this file by entering the following commands:

```
cd /usr/lpp/netviewdm/src
make -f fndex.mak
```

You should see the following messages returned:

```
cc -c -w -D_XOPEN_SOURCE -I/usr/lpp/netviewdm/src /usr/lpp/netviewdm/src/
fndcx.c
# Linking CX library
cc -o libfndcx.a fndcx.o fndcxcm.o fndcxmo.o -lc -lrts -ls  -bM:SRE -bE:
/usr/lpp/netviewdm/src/fndcxlib.exp -e _nostart
ld: 0711-327 WARNING: Entry point not found: _nostart
        chmod 660 fndcx.o fndcxcm.o fndcxmo.o
        chmod 550 libfndcx.a
        chgrp FNDADMN  libfndcx.a
        chown root libfndcx.a
Target "all" is up to date.
```

[6] Some of the connectivity options that we have discussed extend your network out over modems (for example, PPP and LAN Distance). These can also be used with Distribution Clients.

You can safely ignore the warning about the entry point.

To install the new user exit module, issue the following commands:

```
nvdm stop -k -x
cp /usr/lpp/netviewdm/bin/libfndcx.a /usr/lpp/netviewdm/bin/libfndcx.safe
rm /usr/lpp/netviewdm/bin/libfndcx.a
cp /usr/lpp/netviewdm/src/libfndcx.a /usr/lpp/netviewdm/bin/libfndcx.a
nvdm start
```

If you have any problems then copy `libfndcx.safe` back to `libfndcx.a` and restart Software Distribution for AIX.

Finally create the file `nvdm_audit` in a suitable directory, such as `/usr/lpp/netviewdm/bin` as shown in Figure 254 on page 319.

```
################################################
# Sample Audit report shell script.            #
#                                              #
# Reads the output file produced by the user exit  #
# and prints only those entries that are for   #
# Mobile Clients.                              #
#                                              #
# This could be extended to only show those    #
# Mobile Clients which are connected over asynch   #
# if you do not use any other remote access method.#
#                                              #
# Mark Guthrie November 1996                   #
################################################

##########################################
# Print Title                            #
##########################################
echo Audit Report for Mobile Clients
echo ------------------------------

##########################################
# Open the audit file created by the user  #
# exit.                                  #
##########################################
exec 3</tmp/audit.out

##########################################
# Read until done                        #
# splitting first line into targetname and #
# the rest                               #
##########################################
while read -u3 target rest
do

##########################################
# Now ask NVDM about this target and use   #
# grep to strip out only the machine type. #
##########################################
  nvdm lstg $target | grep -E Type > /tempfile

##########################################
# Open this temporary file for read      #
##########################################
  exec 4< /tempfile

##########################################
# read to find the machine type          #
##########################################
  read -u4 title type

##########################################
# Close the temporary file               #
##########################################
  exec 4<&-
```

Figure 254. /usr/lpp/netviewdm/bin/nvdm_audit - Part 1 of 2

```
########################################
# Read the rest of the block from the      #
# audit file                               #
########################################
  read -u3 line2
  read -u3 line3
  read -u3 line4
  read -u3 line5

########################################
# If it is a Mobile Client then print entry#
########################################
  if [ "$type" = "MOBILE CLIENT" ]
  then
    echo $target $rest
    echo "   " $line2
    echo "   " $line3
    echo "   " $line4
    echo "   " $line5
  fi

########################################
# Finished                               #
########################################
done
```

Figure 255. /usr/lpp/netviewdm/bin/nvdm_audit - Part 2 of 2

For more information on the user exits refer to the redbook *Netview Distribution Manager/6000 Agents and Advanced Scenarios*, GG24-4490.

17.7.2 Producing the Audit Report

Install some change objects to Mobile Clients to create some audit data. Once this is done you should find a file is created called /tmp/audit.out. You can now run nvdm_audit, which will produce output similar to Figure 256.

```
Audit Report for Mobile Clients
-------------------------------
mobile1 is scheduled to have AUDIT.TEST.REF.1.0 installed/sent/removed.
    This will happen after 27/11/1996 at 15:33:47.
    This piece of treachery was carried out by user root@rs600012.
    The crime took place on 27/11/1996 at 15:33:48.

spy is scheduled to have COMPANY.SECRETS.REF.1.0 installed/sent/removed.
    This will happen after 27/11/1996 at 15:33:57.
    This piece of treachery was carried out by user dodgy@rs600012.
    The crime took place on 27/11/1996 at 15:33:57.

spy reports that it has done something with COMPANY.SECRETS.REF.1.0!
    To find out what, check the catalog.
    This piece of treachery was planned by user dodgy@rs600012.
    The original crime took place on 27/11/1996 at 15:33:57.
```

Figure 256. Output from nvdm_audit

17.7.3 Source Code Listing

```
/*****************************************************************/
/* Sample audit program based on the example user exit         */
/* provided with TME 10 Software Distribution for AIX 3.1.3    */
/*                                                               */
/* We only handle two of the exits.                              */
/*                                                               */
/* sx_server_report: is called when a report is returned to    */
/*                   the server. We print an entry in our      */
/*                   audit file to say what happened.          */
/*                                                               */
/* sx_server_request: is called when an adminstrator issues    */
/*                    a request. If it is an install type      */
/*                    of request then we print the details     */
/*                    in our audit file.                       */
/*                                                               */
/* Note: If we find a global name in the data structure then   */
/*       we print the output and claim it was an install.      */
/*       In fact it could have been a remove or a send.        */
/*                                                               */
/* Mark Guthrie November 1996                                    */
/*****************************************************************/
#include <fndhdr.h>
#include <fndshrr.h>
#include <fndcx.h>
#include <stdio.h>
#define OUTPUTFILE "/tmp/audit.out"

 DC_VOID  glob_to_dot(DC_CHAR *dname, DC_GLOB *gname);

/*****************************************************************/
/*                                                               */
/* This is the exit called when a report is returned to the    */
/* server                                                        */
/*                                                               */
/*****************************************************************/
DC_VOID sx_server_report(RR_INFO *ptr)
{
  FILE * trcfile ;
  char dotname[DVA_LOCNAME_LEN];
  memset(dotname, '\0', sizeof(dotname));

/************************************/
/* open output file for append      */
/************************************/
  trcfile = fopen(OUTPUTFILE, "a" ) ;
  if (trcfile != NULL)
  {

/************************************/
/* translate global name to string  */
/************************************/
    glob_to_dot(dotname, &ptr->global_name);

/************************************/
/* If global name is empty then we  */
/* can ignore this report. It is    */
/* not an install                   */
/************************************/
```

```
    if (strcmp(&ptr->global_name.name,"")!=0)
    {

/************************************/
/* Print our stuff                  */
/************************************/
      fprintf(trcfile, "%s reports that it has done something with ",
              ptr->destination);
      fprintf(trcfile, "%s!\n",          dotname);
      fprintf(trcfile, "To find out what, check the catalog.\n");
      fprintf(trcfile, "  This piece of treachery was planned by user %s",
                        ptr->req_id.user);
      fprintf(trcfile, "@%s.\n",ptr->originator);
      fprintf(trcfile, "%s%d/%d/%u at %d:%d:%d.\n\n",
                       "  The original crime took place on ",
                        ptr->sched_info.submit_time.date.day,
                        ptr->sched_info.submit_time.date.month,
                        ptr->sched_info.submit_time.date.year,
                        ptr->sched_info.submit_time.time.hour,
                        ptr->sched_info.submit_time.time.minute,
                        ptr->sched_info.submit_time.time.second);

/************************************/
/* Close file                       */
/************************************/
      fclose(trcfile);
    }/* end of if strcmp...*/
  }/* end of if trcfile...*/
}/* end of sx_server_report */

/*****************************************************************/
/*                                                               */
/* This is the exit called when a request is scheduled at the  */
/* server                                                        */
/*                                                               */
/*****************************************************************/
DC_VOID sx_server_request(RR_INFO           *ptr,
                          CX_USER_RESPONSE  *puserrsp)
{
  FILE * trcfile ;
  char dotname[DVA_LOCNAME_LEN];
  memset(dotname, '\0', sizeof(dotname));

/************************************/
/* open output file for append      */
/************************************/
  trcfile = fopen(OUTPUTFILE, "a" ) ;
  if (trcfile != NULL)
  {

/************************************/
/* translate global name to string  */
/************************************/
    glob_to_dot(dotname, &ptr->global_name);

/************************************/
/* If global name is empty then we  */
/* can ignore this request. It is   */
```

```
/* not an install                    */
/************************************/
    if (strcmp(&ptr->global_name.name,"")!=0)
    {

/************************************/
/* Print our stuff                  */
/************************************/
      fprintf(trcfile, "%s is scheduled to have ", ptr->destination);
      fprintf(trcfile, "%s installed/sent/removed.\n",          dotname);
      fprintf(trcfile, "%s%d/%d/%u at %d:%d:%d.\n",
                       "  This will happen after ",
                        ptr->sched_info.sched_time.date.day,
                        ptr->sched_info.sched_time.date.month,
                        ptr->sched_info.sched_time.date.year,
                        ptr->sched_info.sched_time.time.hour,
                        ptr->sched_info.sched_time.time.minute,
                        ptr->sched_info.sched_time.time.second);
      fprintf(trcfile, "  This piece of treachery was carried out by user %s",
                        ptr->req_id.user);
      fprintf(trcfile, "@%s.\n",ptr->originator);
      fprintf(trcfile, "%s%d/%d/%u at %d:%d:%d.\n\n",
                       "  The crime took place on ",
                        ptr->sched_info.submit_time.date.day,
                        ptr->sched_info.submit_time.date.month,
                        ptr->sched_info.submit_time.date.year,
                        ptr->sched_info.submit_time.time.hour,
                        ptr->sched_info.submit_time.time.minute,
                        ptr->sched_info.submit_time.time.second);

/************************************/
/* Close file                       */
/************************************/
      fclose(trcfile) ;
    }/* end of if strcmp...*/
  }/* end of if trcfile...*/
}/* end of sx_server_request */


/*****************************************************************/
/*                                                               */
/* This is the routine to translate global name to a printable */
/* string. Copied exactly from sample user exit.                 */
/*                                                               */
/*****************************************************************/
DC_VOID glob_to_dot(DC_CHAR *dname, DC_GLOB *gname)
{
   DC_USHORT  ii;                       /* Loop counter                    */
   memcpy(dname, gname->name, DVA_GLOB_LEN);
   if (gname->tokens[0] != DVA_END_TKN)
   {
     for (ii = 1;
          ii < DVA_NUM_GN_COMP && gname->tokens[ii] != DVA_END_TKN;
          ii++)
     {
       dname[gname->tokens[ii] - 1] = '.';
     }
   }
```

```
   return;
} /* co_glob_to_dot */


/************************************************************/
/* We must include these, even if they do nothing or nvdm   */
/* will not start                                           */
/************************************************************/
DC_VOID cx_daca_filename(DC_GLOB *globname,
                         DC_CHAR  locname[DVA_LOCNAME_LEN]){}


DC_VOID cx_daca_report(RR_INFO *report_ptr){}


DC_VOID cx_daca_request(RR_INFO          *request_ptr,
                        CX_USER_RESPONSE *puserrsp){}
```

17.8 Summary

In this chapter we presented you with some of the security and audit issues that you need to consider when implementing Mobile Clients.

We have discussed specific security issues such as PPP and access controls within TME 10 Software Distribution, and we have mentioned some of the general things that you need to consider.

We have also presented a sample user exit that can be used for auditing Mobile Clients.

Chapter 18. Example Front-End Application

In this chapter we look at one approach to using the Mobile Client in a mixed environment. We present an imaginary company that makes extensive use of the Mobile Client and have developed a front-end menu system for use by their staff.

You can use the front-end menu system as a start to creating your own, for example, by modifying the Java examples that we supply.

You should have some knowledge of a programming language, such as C or C++, to benefit fully from this chapter. Some knowledge of Java is helpful but not mandatory.

18.1 Background

Imagine the following scenario. XYZ Company is an IT company that employs a group of field engineers to go around the country supporting customers and looking for new business. Every engineer has a mobile computer and they operate in different market areas. The operating system on the mobile computer matches the dominant operating system for that engineer's customers, so there is a mixture of OS/2, Windows 95, Windows NT and AIX. Figure 257 shows the organization of XYZ Company.

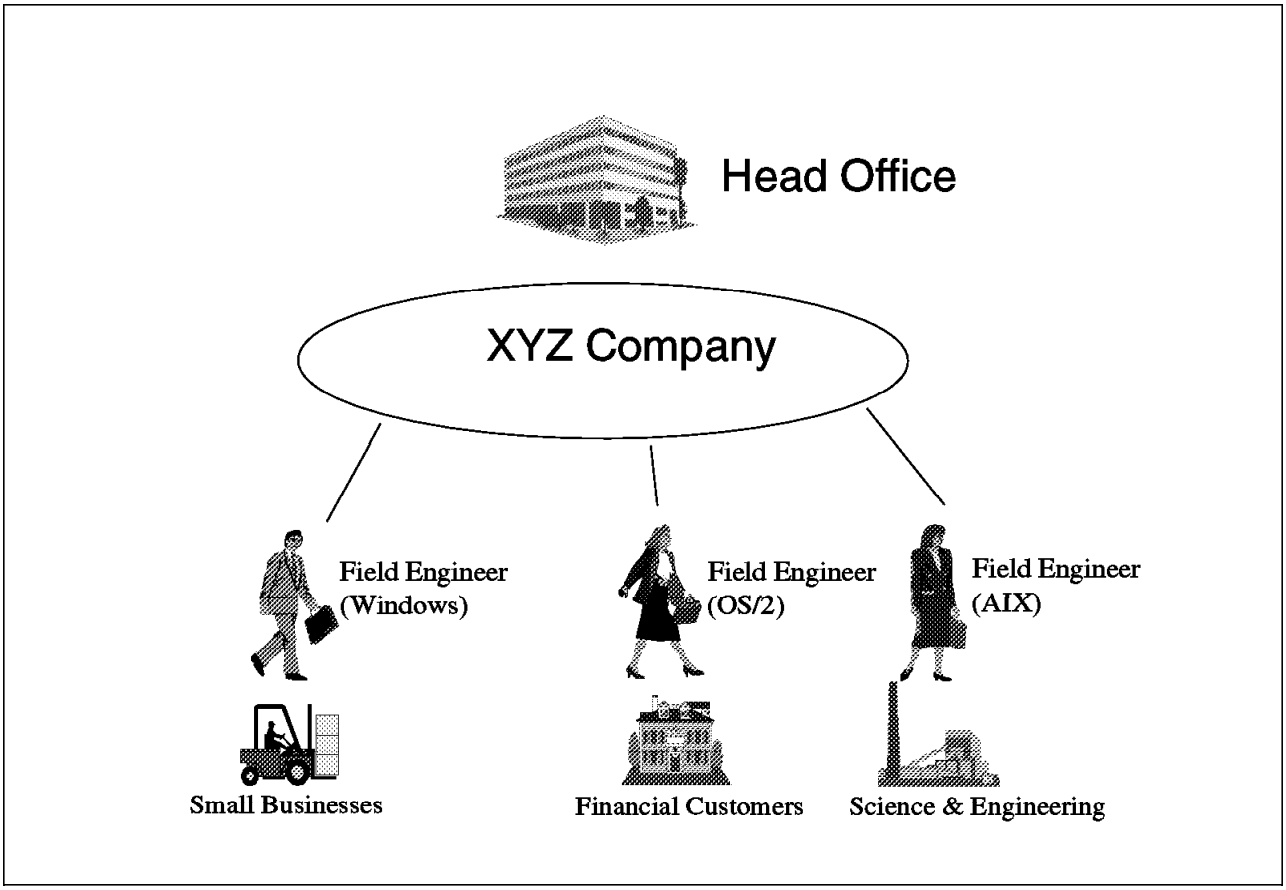


Figure 257. XYZ Company Field Engineers

© Copyright IBM Corp. 1997

All engineers must have the current price list for the company's products and the latest marketing information available on their machines. This is achieved through regular updates that are distributed to the field engineers through TME 10 Software Distribution as shown in Figure 258 on page 326.

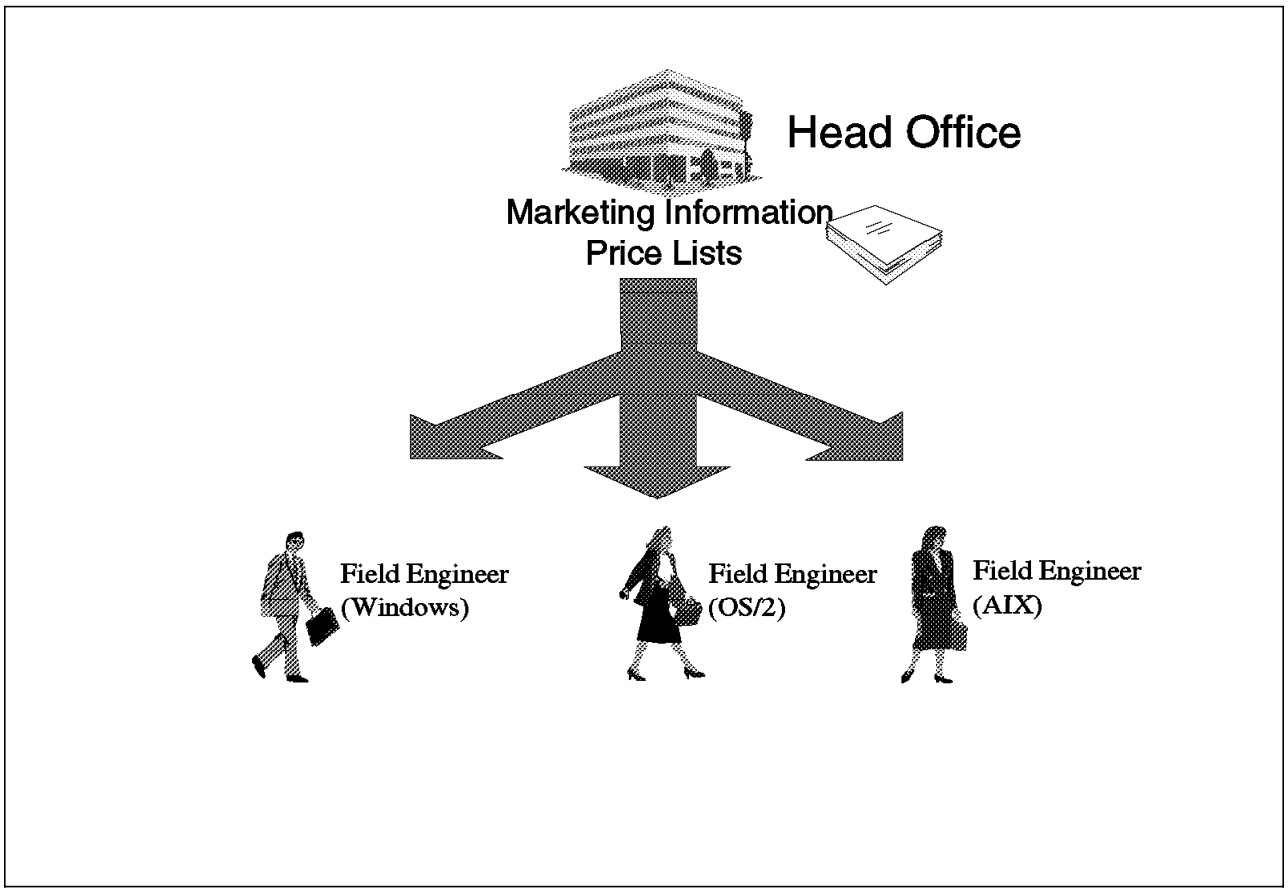


Figure 258. XYZ Company Software Distribution - Central Push

Samples of the company's latest products are made available to the engineers through TME 10 Software Distribution. Engineers can request demonstration packages or even have them made to order at the head office and distributed to them while still at the customer site. This is shown in Figure 259 on page 327.

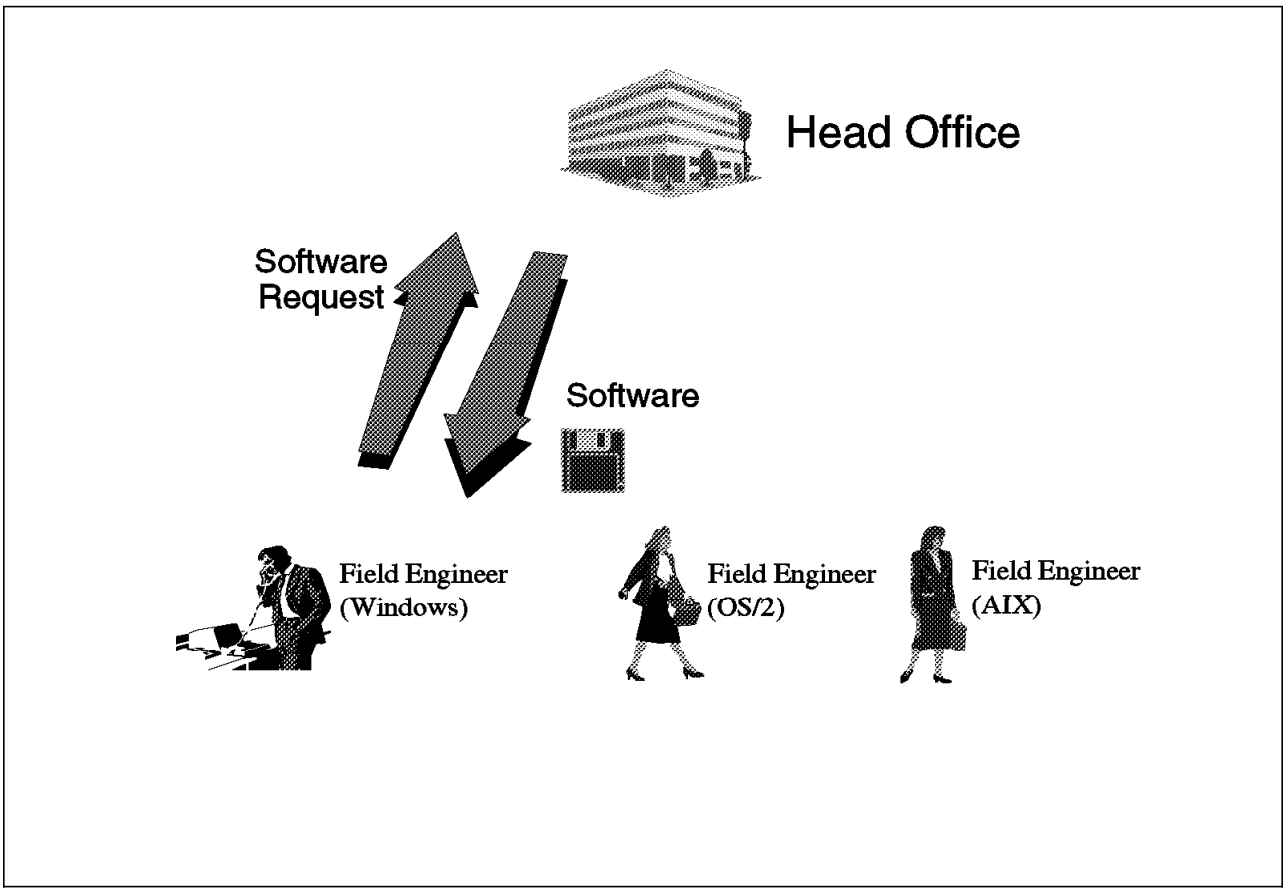


Figure 259. XYZ Company Software Distribution - by User Request

While TME 10 Software Distribution provides a command line interface and a graphical user interface that suit administrators very well, some companies may choose to implement a menu system specific to their site for end users and in particular, for users of the Mobile Client who may also have to issue connect commands to use the server. XYZ Company has chosen to develop a simple menu for their engineers.

One very strong point of the TME 10 Software Distribution family of products is the consistency of the command line interface. Most of the command line options will work on any of the Mobile Clients. To ensure portability of their Hotel Application, XYZ Company has chosen to write the program in the Java programming language, which is portable over all of the platforms that their engineers use. Since the command line directives that the application produces are the same over all of the platforms, this means that one front-end menu program can cover all of the operating systems and there is no need to write a different menu system for each one. To handle the differences in the operating system commands for underlying communications such as PPP, the application uses a configuration file that changes for each operating system.

18.2 Scenario

Whenever possible the engineers connect their computers through modems and into a local phone socket. Since they have no way of knowing when they will be able to connect, or where they will be when they connect, control over the connection must come from the Mobile Client and not the server. They run the Hotel Application and can choose to connect to the head office and receive anything that is waiting for them, or to install any of the applications that have been sent to their catalog in the past. When a connection to the head office is required, the application first starts the underlying communications, if required, and then issues an NVDM CONNECT command. Once the activities are finished, the link is closed.

18.3 The Hotel Application

Note

The scenarios and source code provided here are for demonstration only. If you wish to implement a solution along these lines, you must be aware that there are several areas of weakness in the Hotel Application architecture that must be addressed before using it in a production environment.

The Hotel Application is written in Java. It has been developed and demonstrated on OS/2 and should work on other operating systems with Java support. Discussion of Java itself is beyond the scope of this book. If you require information on Java, then you should consult one of the many books which are available. Brief installation instructions are provided in 18.5, “Installing the Application” on page 333.

The Hotel Application is started from an icon on the desktop by the field engineer. This is the case for all operating systems. A window appears as shown in Figure 260.

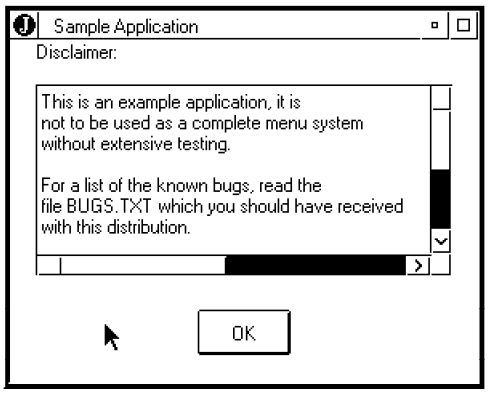


Figure 260. Hotel Application Splash Screen

Once the user clicks **OK**, the initial window is displayed as shown in Figure 261 on page 329 and the engineer chooses between connecting to the head office or using the local catalog of the Mobile Client.

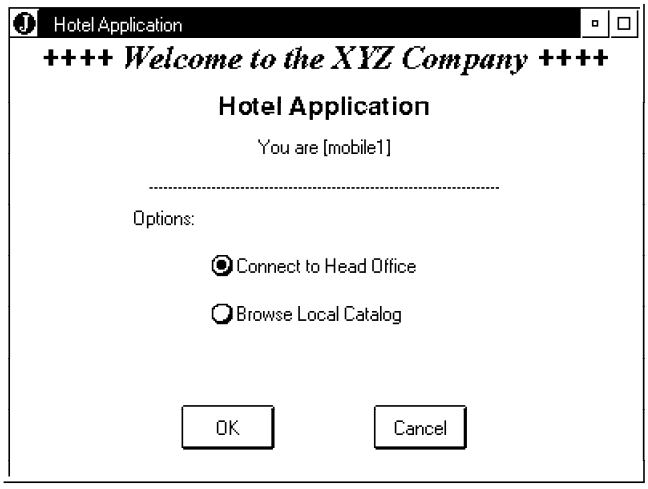


Figure 261. Hotel Application First Screen

The subsequent path taken depends on this initial choice. The following sections describe each route through the application.

18.3.1 Connecting to the Head Office

If a connection to the head office is required, then the Connection Options window is shown as in Figure 262.

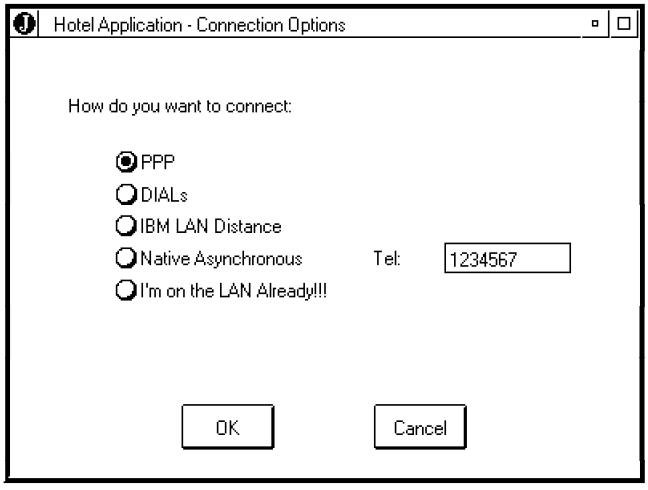


Figure 262. Hotel Application - Connection Options

The user has a choice of the different connection protocols that are in use within XYZ Company. Currently these are:

- PPP
- IBM 8235 LAN Dialer
- IBM LAN Distance
- TME 10 Software Distribution asynchronous support

The same menu can be used when the laptop is connected to the head office LAN, so this is also provided as an option.

Once a connection type has been chosen, an operating system call is made if the connection is PPP, DIALs or LAN Distance. The exact syntax of the command varies between platforms and so these commands are held in a configuration file in the same directory as the Hotel Application. This file is called hotel.cfg and examples are shown in 18.6, " The Source Code" on page 334. This file is the only thing that varies between platforms.

Once a connection has been established, the Connect to Server window is displayed and the user can select how long they wish the connection window to be opened for (Figure 263). This is the logical connection window and may not match the duration of the physical connection.

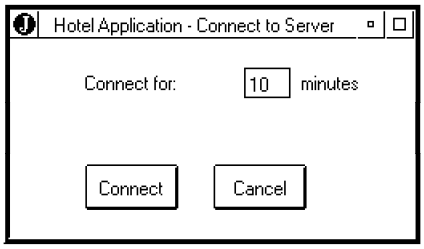


Figure 263. Hotel Application - Connect to Server Window

Once the connection has been established, a window is displayed as shown in Figure 264 to inform the user. This window may be minimized and left running. If the user wishes to close the connection, they can click **Disconnect**.

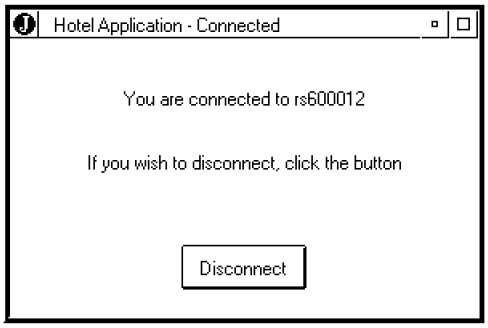


Figure 264. Hotel Application - Connected Window

Now that the connection has been opened, the TME 10 Software Distribution server can send any pending requests to the Mobile Client and the Mobile Client can return any reports that are pending to the server.

18.3.2 Browsing the Local Catalog

The Hotel Application also serves as a menu system to install and remove software locally. The XYZ Company price lists and marketing information are installed under the control of the central administrators. The engineers are unaware of the installation taking place; the software is installed by the administrator issuing commands such as:

```
NVDM SEND PRICES.MARCH97.REF.1.0 mobile1 -w rs600012
NVDM INST PRICES.MARCH97.REF.1.0 -w mobile1 -dc
```

The March 97 edition of the price list is then automatically installed when the Mobile Client next connects to the server.

However, the demonstration applications that are sent to the engineers are stored in the catalog available for use as required. These can be installed and removed by the engineers themselves as needed.

In order to do this the engineers select **Browse Local Catalog** from the first screen of the Hotel Application as shown in Figure 261 on page 329.

This displays the Hotel Application - Browser window as you see in Figure 265.

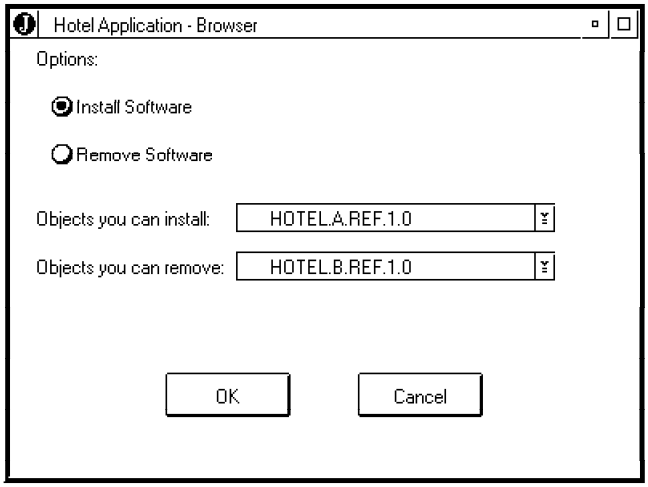


Figure 265. Hotel Application - Browser Window

From this window the engineer can choose to install software from the local catalog or to remove software that is already installed.

Note

The XYZ Company installs all of its software as removable.

18.4 Portability

To show the portability of the application, here are three variations of the same window from OS/2, Windows 95 and AIX.

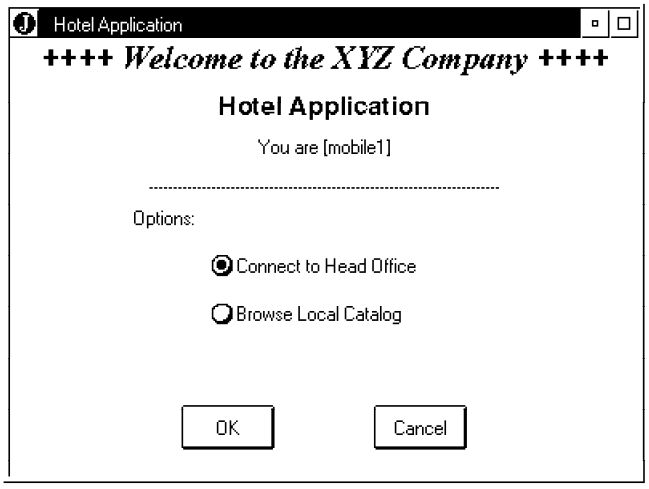


Figure 266. Hotel Application - OS/2

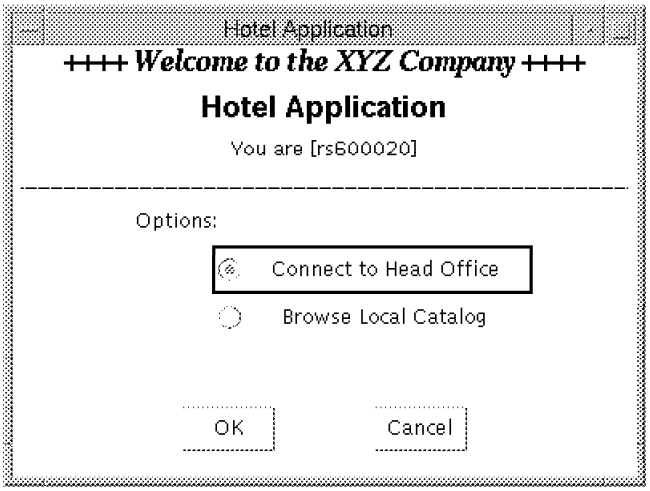


Figure 267. Hotel Application - AIX

Figure 268. Hotel Application - Windows 95

18.5 Installing the Application

To install and use the Hotel Application you must first install Java for the operating system that you are using.

18.5.1 Obtaining Java

Java code can be obtained from the following Internet locations:

Windows 95/NT Via ftp from ftp.javasoft.com. Log in as anonymous and transfer the code from the pub directory as binary. The Java Development Kit is contained in a compressed file such as JDK-1_0-win32-x86.exe.

OS/2 and AIX To obtain the Java Development Kit for OS/2 or AIX you will need to register with IBM. This can be done by following the links from http://ncc.hursley.ibm.com/javainfo/hurindex.html. IBM staff can get the code from the IBM internal site at http://hurwww.hursley.ibm.com/java/codedemos/quickdl.html. Users of OS/2 Warp Version 4 already have Java support available. However, you may wish to upgrade to the most recent version. If you use the version that comes with Warp 4, then you need to use JAVAPM rather than JAVA to start the application.

18.5.2 Obtaining the Hotel Application

The source code and compiled classes for the Hotel Application are available as a ZIP file for OS/2 or Windows, and as a tar file for AIX. Both compressed files contain the same code. The only differences are:

- The text files in the TAR archive have been formatted for AIX (lines are terminated with line feed characters only).
- The default hotel.cfg file in the TAR archive is for AIX, while in the ZIP file it is for OS/2.

Obtaining the Code

To download the code do the following:

- IBM internal: You can find the code at `http://w3.itso.ral.ibm.com/~stefanu`.
- External: send a note to `uelpenich@vnet.ibm.com`.

18.5.2.1 Installing the ZIP File

To extract the ZIP file, create a directory called, for example, Hotel. From within this directory unzip the file HOTEL.ZIP using an unzip program that can handle long file names. You must use a file system such as HPFS or NTFS that can also support long file names.

You must rename the file `disc.class` to `Disconnect.class`, before running the application.

18.5.2.2 Installing the TAR File

To extract the tar file create a directory called, for example, Hotel. From within this directory issue the command:

```
tar -xvf hotel.tar
```

You will need to provide the full path to the tar file if it is not in the current directory.

18.5.3 Running the Application

The Hotel Application can be run from the Hotel directory by entering[7] :

```
java Hotel
```

You should review the accompanying text files and modify the hotel.cfg as required. Samples are provided for OS/2, Windows 95 and AIX.

If you wish to change the code, you should modify the .java files and compile the application by typing:

```
javac Hotel.java
```

It is sometimes a good idea to delete the class files before compiling, since the compiler can be a little temperamental at times.

18.6 The Source Code

This section contains all of the source code for the Hotel Application. Each Java class is built from a single text file.

18.6.1 Hotel.java

This class corresponds to the first window of the application. If this class is called from the command line, then the main() method is invoked. The code for this is listed following this section. This creates an instance of Hotel that invokes the constructor() method. This is a method with the same name as the class. Inside the constructor, all of the window elements are created. Then main() makes this instance of Hotel visible and control passes to Java.

User input comes in through the methods action(Event evt, Object arg) or handleEvent(Event evt). If the user wants to exit then the application closes. Otherwise, if the input was OK then we check the state of the radio buttons and create an instance of either Connect or Browse. If the user clicks **Cancel**, then the application quits.

This structure is followed in all of the dialog classes.

Although we do not explicitly call the class Global, by using one of its variables we cause the Java run time to call it that.

---

[7] This takes you directly into the Hotel Application. If you wish to see the splash screen as described in Figure 260 on page 328, then you should run Java Splash.

```
/*******************************************************/
/* XYZ Company's Hotel Application                     */
/* ------------------------------                      */
/*                                                     */
/* This application is an example of a front-end       */
/* Menu for TME 10 Software Distribution Mobile        */
/* Clients.                                            */
/*                                                     */
/* For more information see the text files that        */
/* accompany this distribution.                        */
/*                                                     */
/* Mark Guthrie  - November 1996                       */
/*                                                     */
/*******************************************************/
import java.awt.*;
import java.applet.*;

public class Hotel extends Frame {
/*******************************************************/
/* This class is the first real panel                 */
/*******************************************************/

/***********************************/
/* Define the objects for the panel */
/***********************************/
  CheckboxGroup actionChoice = new CheckboxGroup();
  Checkbox      checkHO      = new Checkbox
   ("Connect to Head Office", actionChoice, true);
  Checkbox      checkCat     = new Checkbox
   ("Browse Local Catalog", actionChoice, false);
  Button        buttonOK     = new Button("OK");
  Button        buttonCancel = new Button("Cancel");
  Label         title        = new Label
   ("++++ Welcome to the XYZ Company ++++",Label.CENTER);
  Label         title2       = new Label
   ("Hotel Application",Label.CENTER);

//Note: next line causes static class Global to be initiated
  Label         title3       = new Label
   ("You are ["+Global.target+"]",Label.CENTER);
  Label         title4       = new Label
   ("-----------------------------------",Label.CENTER);
  Label         title5       = new Label
   ("Options:",Label.LEFT);

  public Hotel() {  //Constructor
    super("Hotel Application");

// Define window objects and size them
    setLayout(null);  // no layout manager - we will place items by hand
    resize(400,300);  // size the window
    setFont(new Font("Dialog",Font.PLAIN,12));
    add(title);
    add(title2);
    add(title3);
    add(title4);
    add(title5);
    add(checkHO);
    add(checkCat);
```

```
    add(buttonOK);
    add(buttonCancel);

    title.reshape       (10,20, 380,30);
    title2.reshape      (10,50, 380,30);
    title3.reshape      (10,80, 380,20);
    title4.reshape      (10,100, 380,30);
    title5.reshape      (80,120, 200,30);
    checkHO.reshape     (130,150 ,200,30);
    checkCat.reshape    (130,180,200,30);
    buttonOK.reshape    (110,250,60,30);
    buttonCancel.reshape(230,250,60,30);

    title.setFont(new Font("TimesRoman",Font.BOLD|Font.ITALIC,20));
    title2.setFont(new Font("Helvetica",Font.BOLD,18));

    repaint();
// End of window object part
  } //constructor

  void buttonOK_action(Event evt, Object arg){ //ButtonOK was hit
    if (checkHO.getState()==true)
    {
       Connect nn= new Connect();  // create connection window
       nn.show();
    }
    else
    {
       Browse bb= new Browse();   // or create browse window
       bb.show();
    }
       this.dispose();  // kill ourselves
  } //buttonOK_action

  void buttonCancel_action(Event evt, Object arg){ //ButtonCancel was hit
        System.exit(0);  // just exit
  } //buttonCancel_action

  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     buttonOK_action(evt,arg);
    else if (evt.target == buttonCancel) buttonCancel_action(evt,arg);
    else return false;
    return true;
  }//action

  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id==Event.WINDOW_DESTROY) {
        System.exit(0);  //exit on Alt-F4
    }
    return super.handleEvent(evt);
  }//handleEvent

  public static void main(String args[]){ //main routine
    Hotel nn= new Hotel();
    nn.show();
  }//main

}//class
```

18.6.2 Connect.java

This class creates the window shown in Figure 262 on page 329. It follows the same standard structure as used in Hotel.java.

```
//
// Import the classes that we will be using
//

import java.awt.*;
import java.applet.*;
import java.io.*;

public class Connect extends Frame {

/***********************************/
/* Define the objects for the panel */
/***********************************/
  CheckboxGroup actionChoice = new CheckboxGroup();
  Checkbox      checkPPP     = new Checkbox
   ("PPP", actionChoice, true);
  Checkbox      checkDials   = new Checkbox
   ("DIALs", actionChoice, false);
  Checkbox      checkLANDist = new Checkbox
   ("IBM LAN Distance", actionChoice, false);
  Checkbox      checkAsync   = new Checkbox
   ("Native Asynchronous", actionChoice, false);
  Checkbox      checkLAN     = new Checkbox
   ("I'm on the LAN Already!!!", actionChoice, false);
  Button        buttonOK     = new Button("OK");
  Button        buttonCancel = new Button("Cancel");
  Label         title        = new Label
  ("How do you want to connect:",Label.CENTER);
  Label         title2       = new Label
  ("Tel:");
  TextField     telephone    = new TextField(9);

  public Connect() {  //Constructor
    super("Hotel Application - Connection Options");

// Define window objects and size them
    setLayout(null);  // no layout manager - we will place items by hand
    resize(400,300);  // size the window
    setFont(new Font("Dialog",Font.PLAIN,12));
    add(checkPPP);
    add(checkDials);
    add(checkLANDist);
    add(checkAsync);
    add(checkLAN);
    add(buttonOK);
    add(buttonCancel);
    add(title);
    add(title2);
    add(telephone);

    checkPPP.reshape    (70,90,150,20);
    checkDials.reshape  (70,110,150,20);
    checkLANDist.reshape(70,130,150,20);
    checkAsync.reshape  (70,150,150,20);
    checkLAN.reshape    (70,170,170,20);
```

```
    buttonOK.reshape    (110,250,60,30);
    buttonCancel.reshape(230,250,60,30);
    title.reshape       (20,50,180,30);
    title2.reshape      (230,150,45,20);
    telephone.reshape   (275,150,80,20);

// set up telephone number field if asynch allowed
    if (Global.telephone.indexOf("-t")!=-1)
        Global.telephone=Global.telephone.substring(3);
    telephone.setText(Global.telephone);

// disable telephone number if not asynch
    if (Global.telephone.equals("")) {
        telephone.disable();
        checkAsync.disable();
        title2.disable();
        }

    repaint();
  }//constructor

  void buttonOK_action(Event evt, Object arg){ //ButtonOK was hit
    if      (checkPPP.getState()==true) {
         Global.connType = 1;
         DoOSCmd doppp= new DoOSCmd(Global.cmdPPP,Global.paramPPP);
         }
    else if (checkDials.getState()==true) {
         Global.connType = 2;
         DoOSCmd doDIALs = new DoOSCmd(Global.cmdDIALs,Global.paramDIALs);
         }
    else if (checkLANDist.getState()==true) {
         Global.connType = 3;
         DoOSCmd doLAND = new DoOSCmd(Global.cmdLANDist,Global.paramLANDist);
         }
    else if (checkAsync.getState()==true) {
         Global.connType = 4;
         Global.telephone="-t "+telephone.getText();
         }
    else if (checkLAN.getState()==true) {
         Global.connType = 5;
         }
// Make sure telephone number is empty if we don't need it
    if (checkAsync.getState()!=true) {
         Global.telephone="";
         }
    NVDM nvdm = new NVDM();
    nvdm.show();  //go to next window
    this.dispose();
  }//buttonOK_action

  void buttonCancel_action(Event evt, Object arg){ //ButtonCancel was hit
        Hotel newHotel = new Hotel();
        newHotel.show();
        this.dispose();
  }//buttonCancel_action

  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     buttonOK_action(evt,arg);
    else if (evt.target == buttonCancel) buttonCancel_action(evt,arg);
```

```
    else return false;
    return true;
  }//action


  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id==Event.WINDOW_DESTROY) {
        System.exit(0);  //exit on Alt-F4
    }
    return super.handleEvent(evt);
  }//handleEvent

}//class
```

18.6.3 NVDM.java

This class creates the window shown in Figure 263 on page 330.

```
import java.awt.*;
import java.applet.*;

public class NVDM extends Frame {
/***********************************/
/* Define the objects for the panel */
/***********************************/
  Label           label1       = new Label("Connect for:");
  Label           label2       = new Label("minutes");
  TextField       time         = new TextField(3);
  Button          buttonOK     = new Button("Connect");
  Button          buttonCancel = new Button("Cancel");

  public NVDM() { //constructor
    super("Hotel Application - Connect to Server");

// Define window objects and size them

    setLayout(null);  // no layout manager - we will place items by hand
    resize(260,150);  // size the window
    add(label1);
    add(label2);
    add(time);
    add(buttonOK);
    add(buttonCancel);

    label1.reshape      (50,40,80,20);
    time.reshape        (150,40,30,20);
    label2.reshape      (185,40,100,20);
    buttonOK.reshape    (50,100,60,30);
    buttonCancel.reshape(130,100,60,30);

    time.setText("10");

    repaint();
  }//constructor

  void buttonOK_action(Event evt, Object arg){ //ButtonOK was hit
    DoOSCmd dosvr = new DoOSCmd(Global.nvdm,"svr "+Global.server);
    dosvr.dispose();
    DoOSCmd donvdm = new DoOSCmd
      (Global.nvdm,"connect -d"+time.getText()+" "+Global.telephone);
```

```
    donvdm.dispose();
    Disconnect newdisc = new Disconnect();
    newdisc.show();
    this.dispose();
} //buttonOK_action

  void buttonCancel_action(Event evt, Object arg){ //ButtonCancel was hit
        Connect newConnect = new Connect();
        newConnect.show(); // go back to parent window
        this.dispose();
  }//buttonCancel_action

  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     buttonOK_action(evt,arg);
    else if (evt.target == buttonCancel) buttonCancel_action(evt,arg);
    else return false;
    return true;
  }//action


  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id==Event.WINDOW_DESTROY) {
        System.exit(0);  //exit on Alt-F4
    }
    return super.handleEvent(evt);
  }//handleEvent

}//class
```

18.6.4 Disconnect.java

This class creates the window shown in Figure 264 on page 330.

```
import java.awt.*;
import java.applet.*;

public class Disconnect extends Frame {

/************************************/
/* Define the objects for the panel */
/************************************/
  Label           label1       = new Label
   ("You are connected to "+Global.server,Label.CENTER);
  Label           label2       = new Label
   ("If you wish to disconnect, click the button",Label.CENTER);
  Button          buttonOK     = new Button("Disconnect");

  public Disconnect() { //constructor
    super("Hotel Application - Connected");

// Define window objects and size them
    setLayout(null);  // no layout manager - we will place items by hand
    resize(300,200);  // size the window
    setFont(new Font("Dialog",Font.PLAIN,12));
    add(label1);
    add(label2);
    add(buttonOK);

    label1.reshape      (10,50,280,20);
    label2.reshape      (10,90,280,20);
```

```
    buttonOK.reshape    (110,150,80,30);

    repaint();
  }//constructor

  void buttonOK_action(Event evt, Object arg){
//issue NVDM disconnect
    DoOSCmd donvdm= new DoOSCmd(Global.nvdm,"disconnect");
    donvdm.dispose();
//check if we need to disconnect
    if      (Global.connType == 1) { //PPP
         DoOSCmd undoppp= new DoOSCmd(Global.dcmdPPP,Global.dparamPPP);
         }
    else if (Global.connType == 2) {  //DIALs
         DoOSCmd undoDIALs = new DoOSCmd(Global.dcmdDIALs,Global.dparamDIALs);
         }
    else if (Global.connType == 3) {  //LAN Distance
         DoOSCmd undoLAND = new DoOSCmd(Global.dcmdLANDist,Global.dparamLANDist);
         }
    else if (Global.connType == 4) {  //Async
         // Nothing to do
         }
    else if (Global.connType == 5) {  //LAN
         // Nothing to do
         }
        Hotel newHotel = new Hotel();
        newHotel.show();
        this.dispose();
  }//buttonOK_action

  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     buttonOK_action(evt,arg);
    else return false;
    return true;
  }//action

  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id==Event.WINDOW_DESTROY) {
        System.exit(0);  //exit on Alt-F4
    }
    return super.handleEvent(evt);
  }//handleEvent

}//class
```

18.6.5 Browse.java

This class creates the window shown in Figure 265 on page 331.

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;

public class Browse extends Frame {
/************************************/
/* Define the objects for the panel */
/************************************/
  CheckboxGroup   actionChoice = new CheckboxGroup();
  Checkbox        install      = new Checkbox
```

```
   ("Install Software", actionChoice, true);
  Checkbox        remove       = new Checkbox
   ("Remove Software", actionChoice, false);
  Choice          installable  = new Choice();
  Choice          removeable   = new Choice();
  Button          buttonOK     = new Button("OK");
  Button          buttonCancel = new Button("Cancel");
  Label           title        = new Label("Options:");
  Label           title2       = new Label
   ("Objects you can install:");
  Label           title3       = new Label
   ("Objects you can remove:");


// Variable for line feed character
  byte cReturnByte[] = {0x0A};
  String cReturn     = new String(cReturnByte,0,0,1);

  public Browse() {  //Constructor
    super("Hotel Application - Browser");

// Define window objects and size them
    setLayout(null);  // no layout manager - we will place items by hand
    resize(400,300);  // size the window
    setFont(new Font("Dialog",Font.PLAIN,12));
    add(install);
    add(remove);
    add(buttonOK);
    add(buttonCancel);
    add(title);
    add(title2);
    add(title3);

// Issue NVDM SVR MYSELF
    DoOSCmd svr = new DoOSCmd(Global.nvdm,"svr myself");
    svr.dispose();

//Issue NVDM LSCM * to get list of all objects
    DoOSCmd lscm = new DoOSCmd(Global.nvdm,"lscm *");
    String tmpstr = lscm.output; //string for output
    lscm.dispose();

    StringTokenizer st = new StringTokenizer(tmpstr, cReturn);
    String name="";

// Set defaults to be disabled
    installable.disable();
    removeable.disable();
    install.disable();
    remove.disable();
    buttonOK.disable();

// read output from lscm *. If status is available
// or Distributed then we can istall this so add to
// installable. If status is installed removable then
// we can remove it so add to removeable
    while (st.hasMoreTokens()) {
       String line = st.nextToken().trim();
       StringTokenizer subline = new StringTokenizer(line, ":");
```

```
        if (subline.hasMoreTokens())  {
            String key = subline.nextToken();
            if (key.equals("Global file name"))  {
               name = subline.nextToken();
            }
            else if (key.equals("Status"))  {
               String status = subline.nextToken().trim();
               if (status.equals("Available")) {     // Can be installed
                  installable.addItem(name);
                  installable.enable();
                  install.enable();
                  buttonOK.enable();
               }
               if (status.equals("Distributed")) {  // Can be installed
                  installable.addItem(name);
                  installable.enable();
                  install.enable();
                  buttonOK.enable();
               }
               if (status.indexOf("Installed, removable")!=-1) {
                  removeable.addItem(name);
                  removeable.enable();
                  remove.enable();
                  buttonOK.enable();
               }
            } //end-if status...
        } //end-if subline...
      }//while

    // Finish adding objects to screen
        add(installable);
        add(removeable);

        title.reshape       (20 ,25 ,100,20);
        install.reshape     (30 ,55 ,200,20);
        remove.reshape      (30 ,85 ,200,20);
        title2.reshape      (20 ,125,120,20);
        installable.reshape (145,125,200,30);
        title3.reshape      (20 ,155,120,20);
        removeable.reshape  (145,155,200,30);
        buttonOK.reshape    (100,230,80 ,30);
        buttonCancel.reshape(220,230,80 ,30);

        repaint();
      }//constructor

      void buttonOK_action(Event evt, Object arg){ //ButtonOK was hit
        if (install.getState()==true)
        { //install requested so issue command
          DoOSCmd inst = new DoOSCmd(Global.nvdm,"inst "+installable.getSelectedItem());
          inst.dispose();
        }
        else
        { //remove requested so issue command
          DoOSCmd rem = new DoOSCmd(Global.nvdm,"rem "+removeable.getSelectedItem());
          rem.dispose();
        }//end-if
    //
    // Now we would like to stop and start nvdm to
```

```
// force a read of the disconnected queue, but...
// something funny happens to FNDCMPS if we do this
// Instead we'll just issue a message
//
    DoOSCmd stop = new DoOSCmd("Help",
            "Your request will be actioned"+cReturn+
            "The next time you restart your computer"+cReturn+
            "or issue NVDM STOP and NVDM START from"+cReturn+
            "from the command line."+cReturn+cReturn+
            "(This is due to problems with FNDCMPS"+cReturn+
            "which we did not have time to investigate)");
//
//  exit - go back to top
//
    Hotel newHotel = new Hotel();
    newHotel.show();
    this.dispose();
  }//buttonOK_action

  void buttonCancel_action(Event evt, Object arg){ //ButtonCancel was hit
        Hotel newHotel = new Hotel();
        newHotel.show();
        this.dispose();
  }//buttonCancel_action


  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     buttonOK_action(evt,arg);
    else if (evt.target == buttonCancel) buttonCancel_action(evt,arg);
    else return false;
    return true;
  }//action

  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id==Event.WINDOW_DESTROY) {
        System.exit(0);  //exit on Alt-F4
    }
    return super.handleEvent(evt);
  }//handleEvent
}//class
```

18.6.6 DoOSCmd.java

This class is responsible for all of the operating system calls.

```
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.io.*;
//
// This class issues operating system calls.
// it could be improved to read STDERR and
// to add a cancel button allowing the user
// to read the output. The problem with this
// is that control is returned to the calling
// class after the constructor finishes and the
// event created by the user clicking the button
// may happen much later.
// This is good enough for the simple example
// we are try to show here but should be extended
```

```
// if anyone takes this further.
//

public class DoOSCmd extends Frame{
/***********************************/
/* Define the objects for the panel */
/***********************************/
    TextArea  text  = new TextArea("", 80, 40);
    TextField com   = new TextField(40);
    Label     title = new Label("Command:");
    Label     title2= new Label("Output:");

    String   output="";  //to hold the STDOUT

    public DoOSCmd(String cmd, String params) {//constructor
      super("OS Command Progress");
      setFont(new Font("Dialog",Font.PLAIN,12));
// See if we are to process a command or issue help
      if (cmd.equals("Help")) {
          Help MyHelp = new Help(params);
          MyHelp.show();
      }
      else {
// define vars and add window items
          byte cReturnByte[] = new byte[1];
          cReturnByte[0]=(byte)0x0A;
          String cReturn = new String(cReturnByte,0,0,1);
          String tmpstr = "";
          StringBuffer sb = new StringBuffer(1);
          char car;
          setLayout(null);
          resize(400,300);
          add(text);
          add(com);
          add(title);
          add(title2);
          text.setEditable(false);
          com.setEditable(false);
          com.setText(cmd+" "+params);
          title.reshape(15,30,70,20);
          com.reshape(90,30,180,20);
          title2.reshape(15,55,70,20);
          text.reshape(15,80,370,200);
          show();
          repaint();

// Issue OS command and capture output
          Runtime r = Runtime.getRuntime();
          Process p = null;

// This next bit needs a little explanation:
// Intel platforms will take the parameters as a single
// string, but AIX (correctly) refuses. So we must
// split the string PARAMS into an array
// We use StringTokenizer to do this and see how
// many elements we will need by using countTokens
// Note the array must be exactly the right size - no null elements
          StringTokenizer st = new StringTokenizer(params.trim(), " ");
          String[] action=new String[st.countTokens()+1];
```

```
          action[0]=cmd;
          int i=0;
          while (st.hasMoreTokens()) {
            i++;
            action[i] = st.nextToken();
          }

// Now we can issue to command
          try {
               int c;
               p = r.exec(action);
               InputStream in = p.getInputStream();
               while ((c = in.read()) != -1) {
                   car=(char) c;
                   sb.setLength(0);
                   tmpstr=tmpstr + sb.append(car).toString();
                   if (c==0x0A) {
                       text.appendText(tmpstr);
                       output=output+tmpstr;
                       tmpstr="";
                   }//if
               }//while
               p.waitFor();
          }//try
          catch (Exception e) {
               System.exit(0);
          }//catch
        }//else
      this.dispose();
     } //constructor

} // end class
```

18.6.7 Global.java

```
import java.io.*;
import java.util.*;
class Global {
/*******************************************************/
/* Class for global variables.                         */
/*******************************************************/
// This is a static class
//
// Will read the first occurence of a variable in
// NVDM.CFG even if this line is commented out.
//
// No error checking for file not found.
//

    static String os;
    static String target;
    static String server;
    static String nvdm;
    static String softdist;
    static String telephone="";
    static String cmdPPP;
    static String cmdDIALs;
    static String cmdLANDist;
    static String paramPPP;
```

```
    static String paramDIALs;
    static String paramLANDist;
    static String dcmdPPP;
    static String dcmdDIALs;
    static String dcmdLANDist;
    static String dparamPPP;
    static String dparamDIALs;
    static String dparamLANDist;
    static int    connType;

    static {// Constructor
          readCFG();
          os = getValue("MACHINE TYPE:").trim();
          target = getValue("WORKSTATION NAME:");
          String fullProt = getValue("PROTOCOL:");
          if (fullProt.indexOf("ASY")!=-1) {
               telephone = fullProt.substring(fullProt.indexOf(" ")).trim();
          }
          String fullLine = getValue("SERVER:");
          if (fullLine.indexOf(" ")!=-1)
               server = fullLine.substring(0,fullLine.indexOf(" "));
          else
               server = fullLine;
    }

    public static String getValue(String lookFor) {
//
// Method to read NVDM.CFG - the wonders of Java string handling!
//
    int size;
    int loc;
    int len;

// First see if file exists
    File check = new File(softdist);
    if (check.exists() == false) {
          System.err.println("I can't find the file: "+softdist);
          System.err.println("Modify hotel.cfg to point to the right file");
          System.exit(0);
    }

// OK. Now read it
    try {
          InputStream f1 = new FileInputStream(softdist);
          size = f1.available();
          byte b[] = new byte[size];
          if (f1.read(b) != b.length) {
                System.err.println("Something bad happened");
          }
          String tmpstr = new String(b,0,0,b.length);
          byte cReturnByte[] = new byte[1];
          cReturnByte[0]=(byte)0x0A;
          String cReturn = new String(cReturnByte,0,0,1);
          loc = tmpstr.indexOf(lookFor);
          len = tmpstr.indexOf(cReturn,loc);
          f1.close();
          return(tmpstr.substring(loc+lookFor.length(),len).trim());
        }//try
    catch(IOException e)  {
```

```
            System.out.println("File not found error");
            }//catch
        return("Error");
        }//getValue


        public static String readCFG() {
//
// Method to read HOTEL.CFG
//
        int size;
        int loc;
        int len;
// First see if file exists
        File check = new File("hotel.cfg");
        if (check.exists() == false) {
            System.err.println("I can't find the file: hotel.cfg");
            System.exit(0);
        }

        try {
            InputStream f1 = new FileInputStream("hotel.cfg");
            size = f1.available();
            byte b[] = new byte[size];
            if (f1.read(b) != b.length) {
                System.err.println("Something bad happened");
            }
            String tmpstr = new String(b,0,0,b.length);

            StringTokenizer st = new StringTokenizer(tmpstr, "=+");

            nvdm         = st.nextToken().trim();
            softdist     = st.nextToken().trim();
// Connection Commands
            cmdPPP       = st.nextToken().trim();
            paramPPP     = st.nextToken().trim();
            cmdDIALs     = st.nextToken().trim();
            paramDIALs   = st.nextToken().trim();
            cmdLANDist   = st.nextToken().trim();
            paramLANDist = st.nextToken().trim();
// Disconnection Commands
            dcmdPPP       = st.nextToken().trim();
            dparamPPP     = st.nextToken().trim();
            dcmdDIALs     = st.nextToken().trim();
            dparamDIALs   = st.nextToken().trim();
            dcmdLANDist   = st.nextToken().trim();
            dparamLANDist = st.nextToken().trim();

            f1.close();
           }//try
        catch(IOException e)  {
            System.out.println("File not found error");
            }//catch
        return("Error");
        }//readCFG
}//class
```

18.6.8 Help.java

This is the help class. It is used if the application cannot perform an operating system command. Ideally, we could avoid calling this by developing the commands that are needed for all operating systems and specifying them in the hotel.cfg.

```
import java.awt.*;
import java.applet.*;
import java.util.*;
//
// General purpose help class to show some text
// and wait for the user to click OK
//

public class Help extends Frame {
   Button buttonOK = new Button("OK");
   Label  title    = new Label("Help:");

   public Help(String msg) {
      super("Help Window");
      setLayout(null);
      setFont(new Font("Dialog",Font.PLAIN,12));
      TextArea text = new TextArea(msg, 80, 40);
      resize(300,240);
      add(text);
      text.setEditable(false);
      add(buttonOK);
      add(title);
      title.reshape(20,20,100,20);
      text.reshape(20,50,260,120);
      buttonOK.reshape(120,190,60,30);

// Move the help window to the side
      Point p = this.location();
      this.move(p.x+300,p.y);

      show();
      repaint();
   }

  public boolean action(Event evt, Object arg ){  //action handler
    if      (evt.target == buttonOK)     this.dispose();
    return true;
  }


  public boolean handleEvent(Event evt) { //general event handler
    if (evt.id == Event.WINDOW_DESTROY) this.dispose();
    return super.handleEvent(evt);
  }//handleEvent

}
```

18.6.9 Sample hotel.cfg File for OS/2

```
nvdm=c:\softdist\nvdm.cfg+
ppp=com1 9600 connect "slattach ATM3 OK ATDT1234567
 CONNECT \r ogin: pppuser ssword: pppuser"+
connectc=@HOTEL+
Help=IBM LAN Distance.

We did not have time to test this with
LAN Distance.

To use it you need to edit the configuration
file hotel.cfg and remove this help. Replace
the keyword help with the command to start
LAN Distance and place the parameters after
the equals sign.+
Help=One way to disconnect PPP is to pull
the telephone connection out of the wall.
Another way is to reboot your PC.+
Help=We need a way to disconnect DIALs+
Help=We need a way to disconnect LANDist+
```

18.6.10 Sample hotel.cfg File for Windows 95/NT

```
nvdm=c:\softdist\nvdm.cfg+
Help=As mentioned in the text there is no
easy way to start PPP from the command
line in Windows 95 or NT.

Please start PPP manually and click
OK.+
connectc=@HOTEL+
Help=IBM LAN Distance.

We did not have time to test this with
LAN Distance.

To use it you need to edit the configuration
file hotel.cfg and remove this help. Replace
the keyword help with the command to start
LAN Distance and place the parameters after
the equals sign.+
Help=We need a way to disconnect PPP+
Help=We need a way to disconnect DIALs+
Help=We need a way to disconnect LANDist+
```

18.6.11 Sample hotel.cfg File for AIX

```
/usr/lpp/netviewdm/bin/nvdm=/usr/lpp/netviewdm/db/nvdm.cfg+
Help=PPP Not tested+
Help=Not available for AIX+
Help=Not available for AIX+
Help=PPP Not tested+
Help=Not available for AIX+
Help=Not available for AIX+
```

Part 6. Appendixes

© Copyright IBM Corp. 1997

Appendix A. Installing and Configuring the Mobile Client for AIX

Here we supply the information that is necessary to install the Mobile Client for the AIX platform and configure it to work with a Software Distribution for AIX server.

The AIX Mobile Client might not be used as frequently on desktop and laptop systems as the ones on the OS/2 and Windows platforms. However, there might be situations when it is appropriate to use a Mobile Client also on the AIX platform:

- On an RS/6000 notebook computer
- On RS/6000 systems connected to a server using a slow line

When using a slow line you can have Software Distribution for AIX control the modems directly using the asynchronous support feature. This will eliminate the need to establish the connection to the server through the modem manually and will have the entire process under control of Software Distribution for AIX. We will install the Mobile Client for AIX on rs600020 and connect it to rs600012 which has the Software Distribution for AIX server already installed.

A.1 Installing the Mobile Client for AIX

The Mobile Client feature of Software Distribution for AIX is installed such as any other components using SMIT.

Having the media with the install images available you should type `smit installp` from the AIX command line being the root user. This will pop up the following window:

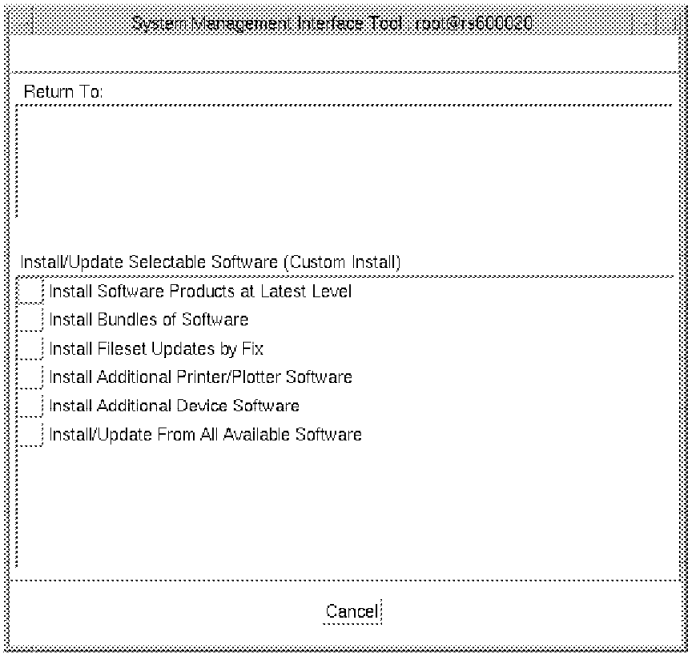


Figure 269. SMIT installp Window

© Copyright IBM Corp. 1997

Select **Install Software Products at Latest Level**. The following window will pop up:

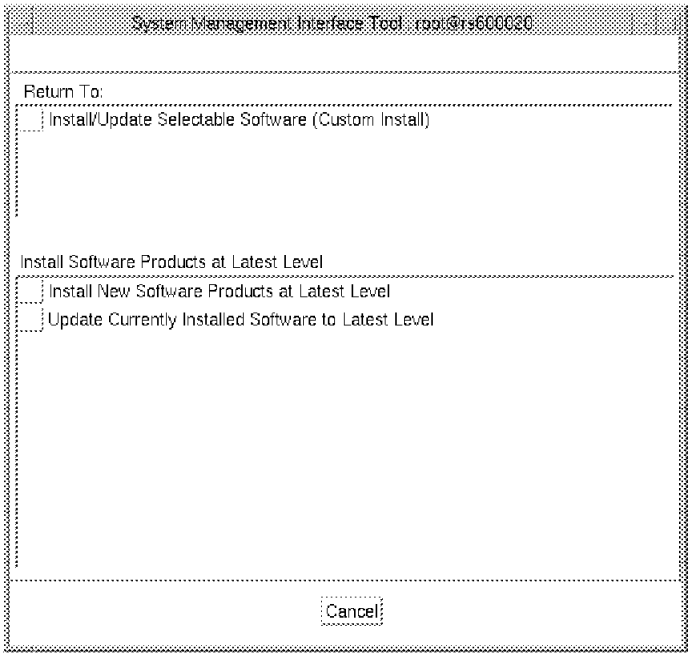


Figure 270. SMIT Install Software Products at Latest Level Window

Select **Install New Software Products at Latest Level**. The following window will pop up asking you for the source of the install images.

Figure 271. SMIT Input Device/Directory for Software Window

Enter the name of the directory where the install images are located and click on the **OK** button. In our example we enter `/usr/sys/inst.images`. The following window will pop up:

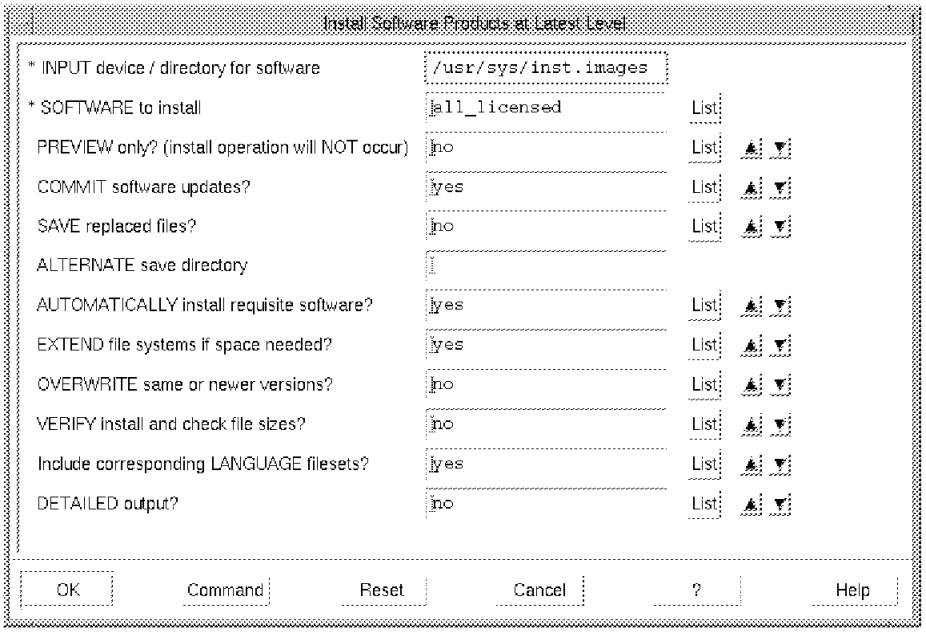


Figure 272. SMIT Install Software at Latest Level Window

Click on the **List** button right of the Software to Install input field to get a list of installable software. This should pop up a window similar to the following:

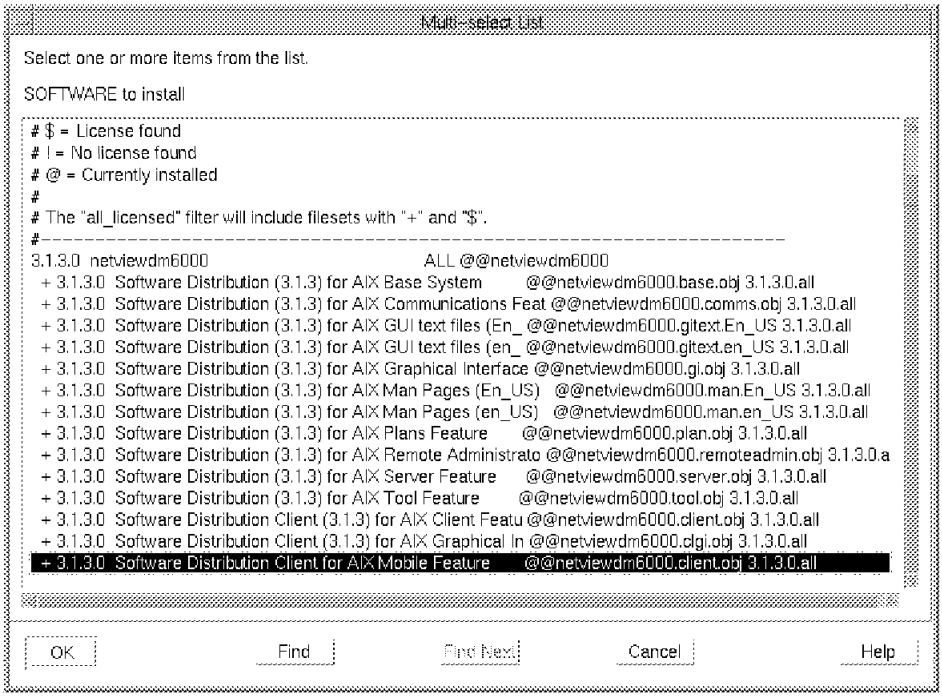


Figure 273. SMIT Software to Install Multi-Select List

Select **Software Distribution Client for AIX Mobile Feature** as shown in the above figure and then **OK**. You might also want to select Software Distribution Client (3.1.3) for AIX Graphical Interface before clicking the **OK** button. This will also install the graphical user interface for the Mobile Client.

Warning

We strongly recommend that you also install the GUI for the Mobile Client. The GUI component for the Mobile Client is the same that is also installed with the standard client.

However, you should be careful when installing the Mobile Feature with other features.

The install image for the Mobile Feature has the same name as the standard AIX Client Feature; for both the name is netviewdm6000.client.obj. If you want to install the GUI in combination with the Mobile Feature you should install the Mobile Feature first and then install the GUI in a separate step. This will ensure that you really install the Mobile Feature.

We experienced that if you select the GUI and Mobile Feature components at the same time, the standard AIX Client Feature is installed although the Mobile Feature was selected. This seems to be caused by the install images having the same name as mentioned above.

When installing the Mobile Feature you should also explicitly specify the name of the image file and not just the name of the directory where the install images reside. For example, if the directory /usr/sys/inst.images contains more images than that for the GUI and the Mobile Feature and the name of the install image for the mobile feature is netviewdm6000.mobclient.3.1.3.0 you should specify /usr/sys/inst.images/netviewdm6000.mobclient.3.1.3.0 as the source for your install images in Figure 271 on page 354. This will avoid that you accidentally install the standard client.

To check if you have really installed the Mobile Feature, you should issue the following commands after the installation:

```
cd /usr/lpp/netviewdm/db
ls
```

For a system with the Mobile Feature installed the output should look like the following:

```
nvdm.cfg   trgcfg
```

The file trgcfg will not be created on a standard client.

Select the **OK** button in the Install Software Products at Latest Level Window. This will start the installation of the Mobile Client for AIX.

You should see something similar to the following:

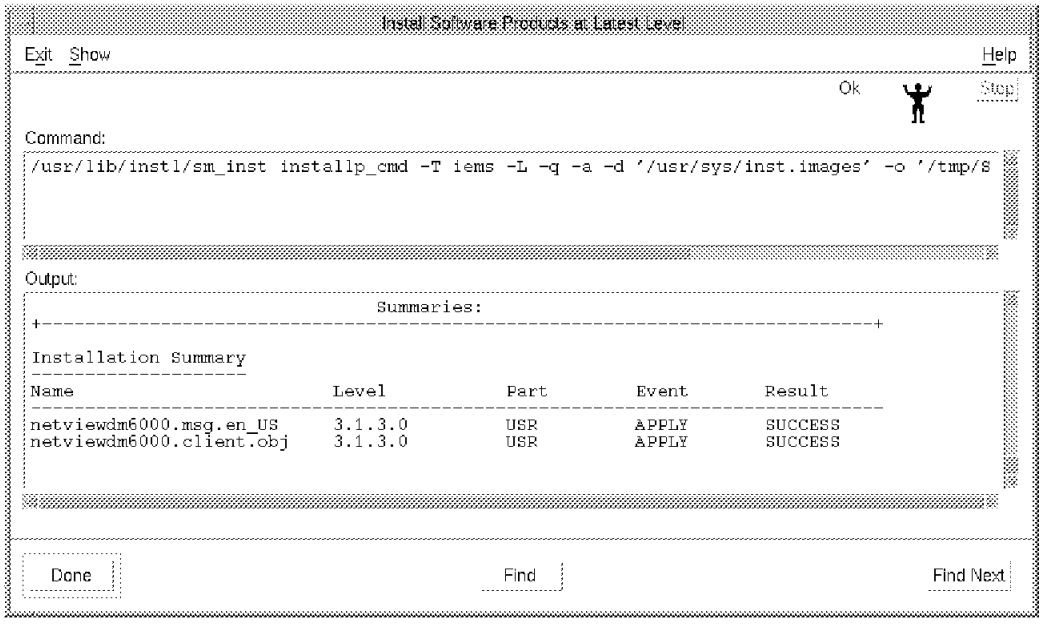


Figure 274. SMIT Install Software at Latest Level Window

Note

In the above example we chose only to install the Mobile Client feature without the GUI component. If you choose the GUI component too, you should have one additional line in the Installation Summary section.

The installation of the Mobile Client is now finished and you can exit SMIT.

A.2 Customizing the Mobile Client for AIX

Before you can start the Mobile Client you will need to edit the nvdm.cfg file on the client. We have modified the file in our example to look like the following:

```
WORKSTATION NAME:     rs600020
MESSAGE LOG LEVEL:    N
LAN AUTHORIZATION:    0
CONFIGURATION:        CLIENT
TARGET ADDRESS:       RS600020
MACHINE TYPE:         AIX
LOG FILE SIZE:        50000
TRACE FILE SIZE:      1000000
API TRACE FILE SIZE:  100
TCP/IP PORT:          729
SERVER:               rs600012
REPOSITORY:           /usr/lpp/netviewdm/repos
SERVICE AREA:         /usr/lpp/netviewdm/service
BACKUP AREA:          /usr/lpp/netviewdm/backup
WORK AREA:            /usr/lpp/netviewdm/work
```

Figure 275. nvdm.cfg File on rs600020

You need to specify an entry for SERVER and if you want to do a client auto-registration you will also need to supply an entry for TARGET ADDRESS.

Note

The server must also be configured to allow auto-registration of targets by specifying AUTOMATIC TARGET REGISTRATION: Y in its configuration file.

In our example we want to do an auto-registration on server rs600012 which has already been installed and configured.

Now that we have customized the nvdm.cfg file on the client we can start the Mobile Client.

A.3 Starting the Mobile Client for AIX

To start the Mobile Client on AIX type:

```
nvdm start
```

The first time the Mobile Client is started it will start the fndcmps process and also perform a connect to the Software Distribution for AIX server, in our case rs600012.

The next time you reboot your system the client should automatically be started since the installation of the Mobile Client component has added the following line to /etc/inittab which will start the client:

```
NetViewDM/6000:2:once: /etc/rc.ndm
```

Note

In our environment the installation did not automatically add the port used for TCP/IP communications to the server (729) to the /etc/services file. Before starting the client you should check if the port is there.

A.4 Using the Graphical Interface of the Mobile Client for AIX

The graphical user interface you can use on a Mobile Client for AIX is slightly different from that of a standard client.

To show the differences we start the Graphical Interface by typing:

```
nvdmgi &
```

When the GUI comes up as shown in Figure 276 on page 359 you will see that in addition to logging on to the Software Distribution for AIX server you will also be able to select MYSELF to log on to the local catalog of the Mobile Client.

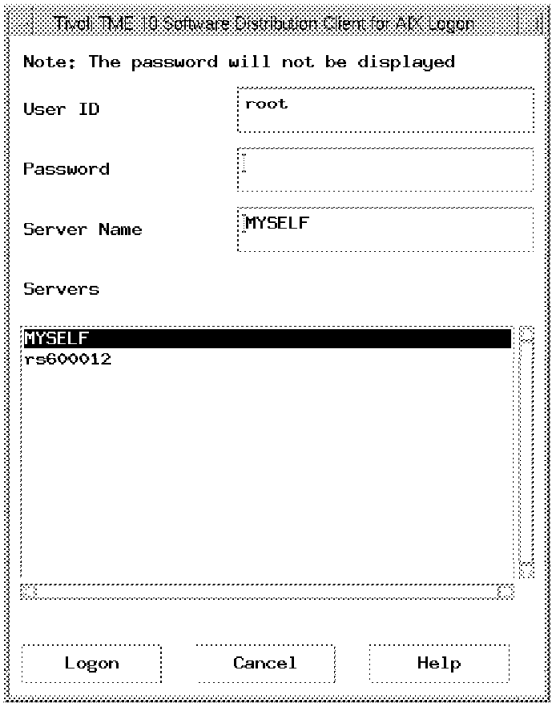


Figure 276. Mobile Client for AIX Graphical User Interface

To do so, select **MYSELF** and the **OK** button. The following window will appear:

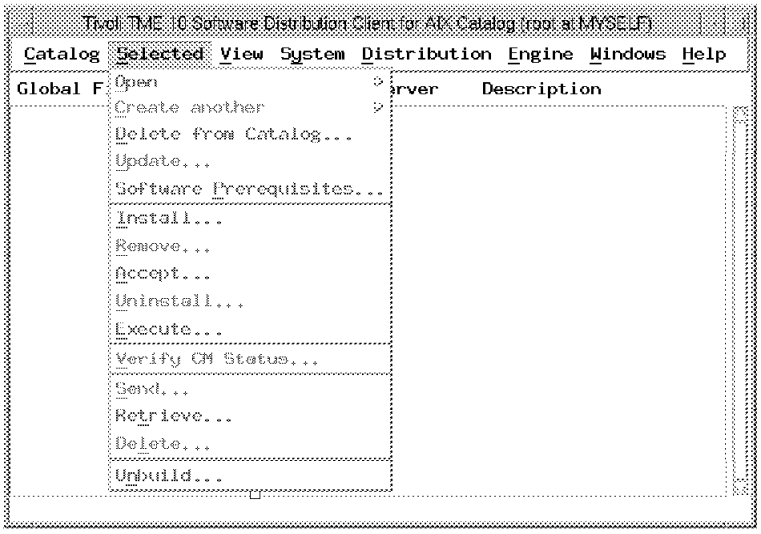


Figure 277. Mobile Client for AIX Catalog Window

This is the Catalog Window of the Mobile Client. You should notice that the window looks similar to the Catalog window of a Software Distribution for AIX server, but you will not be able to perform most of the tasks available on the server, indicated by the corresponding menu items not being selectable.

A.5 Observations on the Software Distribution for AIX Server

To see what happens at the Software Distribution for AIX Server rs600012 in our example we start the GUI of the server, which can be done on the server directly or by using the GUI of the client as shown in Figure 276 on page 359 selecting rs600012.

From the menu bar we select **Windows** and then **Requests** from the pull-down menu.

The following window will pop up.

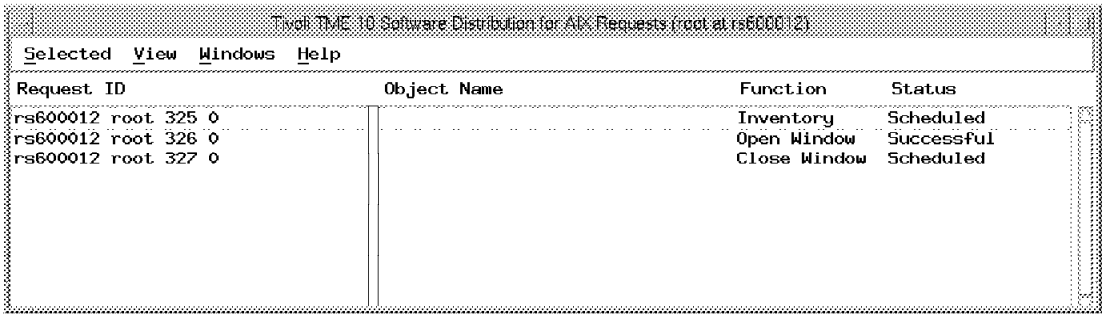


Figure 278. Software Distribution for AIX Requests Window

When the Mobile Client is started the first time it will automatically connect to the Software Distribution for AIX server and perform an auto-registration. Since the target is new, the server will automatically schedule an inventory request to retrieve the new targets inventory files.

This can be seen in Figure 278 (Request ID 325).

You can also see in Figure 278 that the request to open a connection window was succesful, so a connection between rs600012 and rs600020 can be opened.

Once the connection is established, the server will execute the scheduled inventory request an retrieve the inventory files from rs600012. After a short time this should have been performed, indicated by the Inventory request being Successful:

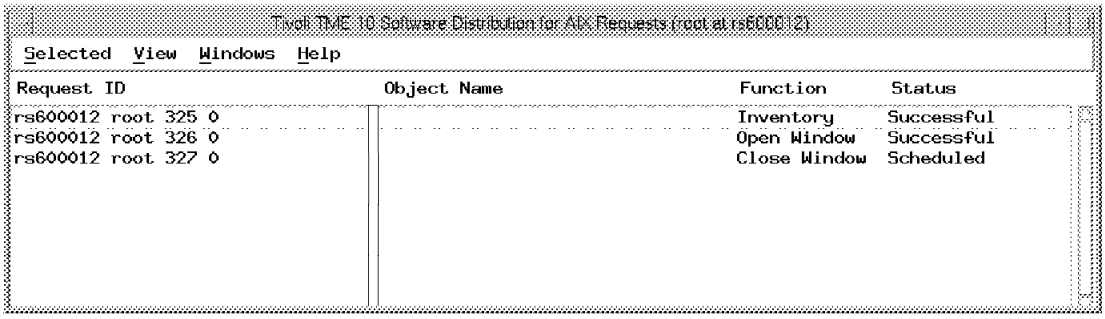


Figure 279. Software Distribution for AIX Requests Window

In order to see that the server really retrieves the inventory files from the client, we created a hardware inventory file on rs600012 before starting the Mobile Client.

There is an inventory discovery program delivered with the Software Distribution for AIX server and client that can be used to automatically create a hardware inventory of AIX systems, including information about CPU, disk space, graphic adapters, etc.

To create a hardware inventory, type the following being root at rs600012:

```
cd /usr/lpp/netviewdm
./bin/fndinvhw
```

This will create a hardware inventory file in /usr/lpp/netviewdm/fndhwinv.

If you had already started the Mobile Client before, you can type the following to refresh the inventory on the server:

```
nvdm inv
nvdm connect
```

If you did not start the client before, just type `nvdm start`.

After the inventory request has executed successfully, use the server GUI to look at the gathered hardware information:

Select **Windows** from the menu bar and then **Targets** from the pull-down menu. Then click on target rs600012 and select **Open** from the menu-bar and then **Details** from the pull-down menu.

Press the **Hardware** button. You should see hardware information similar as shown in Figure 280 on page 362.

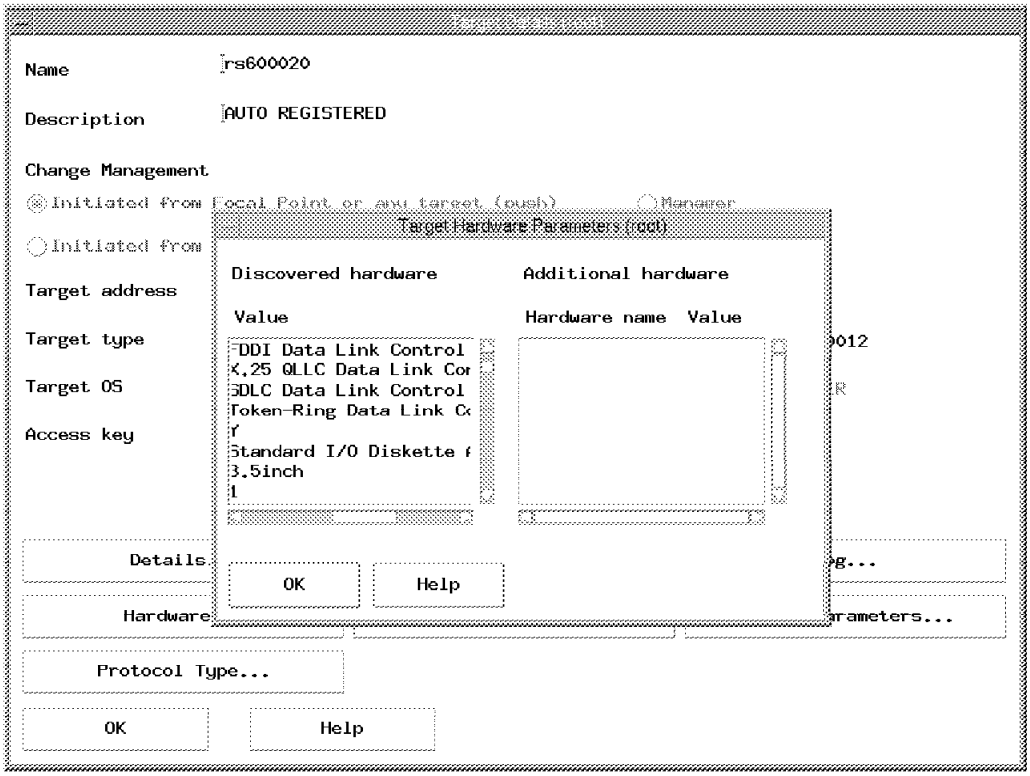


Figure 280. Target Hardware Parameters Window

Appendix B. Asynchronous Support for Windows 3.1 Clients

In this appendix we briefly describe the native asynchronous support for Windows 3.1 Mobile Clients. You should familiarize yourself with Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181 and 4.3, “Installation and Setup of the Windows 3.1 Mobile Client” on page 52 before reading this section.

B.1 Assumptions

In this appendix we also assume you already have a working Software Distribution for AIX server configured for asynchronous support as described in Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181. We also assume that you have a working Windows 3.1 Mobile Client installed in the manner described in 4.3, “Installation and Setup of the Windows 3.1 Mobile Client” on page 52. This PC must be connected to a modem or have a built-in or PCMCIA modem that is functioning normally.

B.2 Configuring the Windows 3.1 Client

Modify the NVDM.CFG file (found by default in the directory C:\SOFTDIST), to include support for the asynchronous protocol. An example is shown in Figure 281.

```
WORKSTATION NAME:                   win31asy
PROTOCOL:                           ASY 9876543
SERVER:                             rs600012 asy 1234567
REPOSITORY:                         C:\SOFTDIST\REPOS
WORK AREA:                          C:\SOFTDIST\WORK
BACKUP AREA:                        C:\SOFTDIST\BACKUP
SERVICE AREA:                       C:\SOFTDIST\SERVICE
CONFIGURATION:                      CLIENT
MESSAGE LOG LEVEL:                  D
LOG FILE SIZE:                      64000
API TRACE FILE SIZE:                64000
TRACE FILE SIZE:                    64000
MACHINE TYPE:                       WINDOWS
MAX USER INTERFACES:                20
MAX ATTEMPTS:                       5
TARGET MODE:                        PUSH
TARGET ADDRESS:                     win31asy
PORT:                               COM1
INIT:                               AT
DIAL:                               ATDT9,
INVENTORY PROGRAM:                  fndinv
```

Figure 281. NVDM.CFG for Asynchronous Support on Windows 3.1

This configuration file is very similar to the file used for asynchronous support in OS/2. Ensure that the port definition is correct for your modem and add any parameters that you require to the initialization string. The PROTOCOL: entry should contain the phone number that the Mobile Client is usually connected to. (This can be overridden in the connection request and a temporary value

© Copyright IBM Corp. 1997

supplied.) The SERVER: entry should have the phone number of the Software Distribution for AIX server. Note that in the DIAL: value we specify a 9 for an outside line.

Note

If you have already used your Mobile Client, then you should tidy up the installation by deleting the files in the directories DB, WORK, QUEUE and UICFG. Also run FNDDBINI from a DOS prompt.

B.3 Checking the Installation

There are two main things to check for:

1. Ensure that the WIN.INI has been updated as shown in Figure 282.
2. Check that you have the DLLs required for asynchronous support. These are shown in Figure 283.

```
[windows]
...
run=FNDCMPS.EXE
...
```

Figure 282. Extract from WIN.INI

```
 Volume in drive C is PCDOS_6
 Volume Serial Number is 2161-83EF
 Directory of C:\SOFTDIST\BIN

FNDRXA16 DLL      34,244 11-11-96   4:08p
FNDRXA32 DLL       6,144 11-07-96  10:29p
FNDRXASY DLL      52,736 11-07-96  10:29p
        3 file(s)       93,124 bytes
                   225,034,240 bytes free
```

Figure 283. Files Required for Asynchronous Support

Note

We initially installed our Mobile Client to use TCP/IP and then made the changes described above to switch to asynchronous support. We found that if we removed the underlying TCP/IP product, then we received an error when FNDCMPS.EXE was started. Apparently the code still issues a WINSOCK call even though it is configured to use asynchronous communications. The WINSOCK call fails because TCP/IP is not present.

By leaving the TCP/IP support installed everything worked fine, even if the LAN connection was not present and no TCP/IP functions could be used.

The above symptoms may be because of the early code level we used.

Please check with your IBM representative for current version information.

B.4 Using Asynchronous Support

Once Windows has been started and FNDCMPS.EXE is running, you can use the Mobile Client as you would expect. For troubleshooting the asynchronous connection, for example, checking modems and ports, you should refer to Chapter 11, “Using the Asynchronous Protocol Support for Mobile Clients” on page 181 where there is a detailed description of the setup.

Appendix C. Debugging Suggestions

This appendix contains some tips on debugging that may prove useful when working with the Mobile Client. These are commands and techniques that we used while making this redbook.

C.1 Enabling Traces

Several traces are available for TME 10 Software Distribution. The most detailed traces are of use only to the developers, but many of the other traces proved useful to us.

C.1.1 Application Traces

The application trace in Software Distribution 3.1.3 can be used to trace the data flow between a server and a client. The application trace, also called API trace can be initiated with the command:

```
nvdm tron
```

The command must be initiated at the client site and the server site.

Trace output for the application trace is written to the file fndapi. This file is located in the product directory C:SOFTDIST for OS/2 and all Windows platforms and /usr/lpp/netviewdm for the AIX platform.

To stop the API trace issue the command:

```
nvdm troff
```

You should also refer to the Software Distribution Command Line Reference for information about the API trace.

C.1.2 Tracing Internal Components

The tracing of internal components in the Software Distribution 3.1.3 product is normally not required unless requested by the development lab. We only provide the information on how to initiate tracing with no explanation of the contents of the trace files.

The trace files always have the names:

- FNDTRC1
- FNDTRC2

These files are located in the product directory C:SOFTDIST for all OS/2 and Windows platforms and /usr/lpp/netviewdm for the AIX platform.

There are different environment variables used to specify the type of trace and the trace detail level. When using the environment variables to initiate tracing you need to stop and restart TME 10 Software Distribution.

- FNDITRC=0

  Specifies that an internal trace of all the components in Software Distribution 3.1.3 is taken at the deepest level available.

© Copyright IBM Corp. 1997

> **Note**
>
> This trace has a strong influence on performance. The amount of trace messages written is very high, so you should allow enough space for the trace files or messages are overwritten.

- FNDITRC_CO=6

  This setting excludes the tracing of the common components.

- FNDITRC_DAS=6

  This setting excludes the tracing of the data access services component.

To set the environment variable for Software Distribution 3.1.3 on the AIX platform specify:

```
export FNDITRC=0
```

To set tracing on an OS/2 or any Windows platform specify:

```
set FNDITRC=0
```

Tracing can be stopped by setting the environment variables again, but this time without any value, for example:

```
set FNDITRC=
```

Again you have to stop and start TME 10 Software Distribution to activate the new setting. If you do not stop tracing by resetting the variables, the trace will continue to be taken even after product restart.

C.1.3 Dynamic Tracing

The internal trace can be switched on dynamically, meaning it is only set until next product start. To initiate dynamic tracing specify the command:

```
nvdm trace -ln comp_id -vn comp_id
```

Where the parameter -l specifies the trace level and comp_id the name of the component to be traced, n can have a value between 0 or 6. The parameter -v has the same value as -l and is only used to display the setting of the trace after entering the command. Valid component IDs are:

- co - for common routines
- cm - for change manager (DACA)
- rb - for the request block handler routine
- rx - for the request transfer (RB-API)
- tc - for the transmission controller
- cc - for the command line interface
- gi - for the graphical user interface

Refer to the Software Distribution Command Line Reference for a complete list of valid components.

C.2 FNDLOG

The FNDLOG is located in the product directory C:SOFTDIST for OS/2 and all Windows platforms and /usr/lpp/netviewdm for the AIX platform. It is an invaluable source of information about what TME 10 Software Distribution is up to. On AIX it is useful to keep a window open viewing the FNDLOG. This can be achieved by using the command:

```
tail -f /usr/lpp/netviewdm/fndlog
```

The option -f causes the view to be updated every time the file changes.

On OS/2 the same thing is possible using a version of the tail utility.

IBM Internal

IBM staff can obtain an OS/2 version of tail from OS2TOOLS in the package AIXLIKE.

This can be run in a window by entering:

```
mode co132,50
tail -f c:\softdist\fndlog
```

The mode command makes the window bigger to allow you to scroll back to see messages that have passed too quickly.

C.3 Renaming a Client

It is often useful when testing to reset a client to appear as though it were a totally new machine to TME 10 Software Distribution. This can be done for OS/2 and Windows without re-installing by issuing the following commands:

```
NVDM STOP
DEL \SOFTDIST\WORK
<answer yes when asked to confirm>
DEL \SOFTDIST\UICFG
<answer yes when asked to confirm>
DEL \SOFTDIST\QUEUE
<answer yes when asked to confirm>
DEL \SOFTDIST\DB
<answer yes when asked to confirm>
```

Now edit the NVDM.CFG and change the WORKSTATION NAME: and TARGET ADDRESS: to names that have not been used previously. Now from the command line enter:

```
FNDDBINI
NVDM START
```

C.4 Miscellaneous

To see the Mobile Client's queue enter:

```
NVDM SVR MYSELF
NVDM LSQ
```

If you delete a request and the server does not remove it quickly enough for you, try issuing an NVDM START to force it to re-read the queues. This is often

useful when you want to delete a target but have to first remove pending requests.

Appendix D. Special Notices

This publication is intended to help technical personnel understand and implement the Mobile Client function of TME 10 Software Distribution. The information in this publication is not intended as the specification of any programming interfaces that are provided by TME 10 Software Distribution. See the PUBLICATIONS section of the IBM Programming Announcement for TME 10 Software Distribution for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

© Copyright IBM Corp. 1997

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIXwindows
APPN	BookManager
BookMaster	DatagLANce
IBM	LAN Distance
Micro Channel	NetFinity
NetView	Presentation Manager
PS/ValuePoint	RISC System/6000
RS/6000	ThinkPad
ValuePoint	WebExplorer
Workplace Shell	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Apple (name and logo)	Apple Computer, Incorporated
C + +	American Telephone and Telegraph Company, Incorporated
Freelance Graphics	Lotus Development Corporation
Hayes	Hayes Microcomputer Products, Incorporated
HotJava	Sun Microsystems, Incorporated
Intel	Intel Corporation
IPX	Novell, Incorporated
Java	Sun Microsystems, Incorporated
Lotus	Lotus Development Corporation
Lotus Notes	Lotus Development Corporation
Lotus Organizer	Lotus Development Corporation
Lotus 1-2-3	Lotus Development Corporation
Lycos	Carnegie Mellon University
MS-DOS	Microsoft Corporation
NetWare	Novell, Incorporated
Newton (name and light bulb logo)	Apple Computer, Incorporated
NFS	Sun Microsystems Incorporated
Novell	Novell, Incorporated
Pentium	Intel Corporation
PostScript	Adobe Systems, Incorporated
Sun Microsystems	Sun Microsystems, Incorporated
Tivoli	Tivoli Systems Inc., an IBM Company
Tivoli Management Environment	Tivoli Systems Inc., an IBM Company

Tivoli Management Framework	Tivoli Systems Inc., an IBM Company
Tivoli Management Platform	Tivoli Systems Inc., an IBM Company
Tivoli/Courier	Tivoli Systems Inc., an IBM Company
TME	Tivoli Systems Inc., an IBM Company
TME 10	Tivoli Systems Inc., an IBM Company
Visual Basic	Microsoft Corporation
Windows NT	Microsoft Corporation
Win32	Microsoft Corporation
Win32s	Microsoft Corporation
X-Windows	Massachusetts Institute of Technology
X/Open	X/Open Company Limited
386	Intel Corporation
486	Intel Corporation

Other trademarks are trademarks of their respective companies.

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 377.

- *The NetView Distribution Manager/6000 Cookbook*, GG24-4246
- *NetView DM/6000: Agents and Advanced Scenarios*, GG24-4490
- *Software Distribution for AIX: A Solution for Installation and Configuration of Pristine AIX Environments*, SG24-4508
- *LAN Management Processes Using NetFinity*, SG24-4517
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *The CID Guide*, SG24-4295
- *IBM 8235 Dial-in Access to LANs Server Concepts and Implementation*, SG24-4816
- *TCP/IP Implementation in an OS/2 Warp Environment*, SG24-4730

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

E.3 Other Publications

These publications are also relevant as further information sources:

- *Tivoli TME 10 Software Distribution (3.1.3) for AIX*, SH19-4333
- *Tivoli TME 10 Software Distribution (3.1.3) for OS/2*, SH19-4334
- *Tivoli TME 10 Software Distribution (3.1.3) for Windows NT*, SH19-4335
- *Tivoli TME 10 Software Distribution Client Installation and Configuration*, SH19-4337

© Copyright IBM Corp. 1997

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

© Copyright IBM Corp. 1997

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number _____
- Credit card number _____

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Index

Special Characters

Numerics

A

B

C

© Copyright IBM Corp. 1997

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

IBM®

Printed in U.S.A.