```
[Identification]
    OptionType = NetService
[Options]
    RAS
[GeneralConstants]
    Chosen       = "ON"
    NotChosen    = "OFF"
    TRUE         = 1
    FALSE        = 0
    NoTitle      = 0
    FLibraryErrCtl = 1
    OldVersionExisted = $(FALSE)
[FileConstants]
    HandleNull   = ""
    PORTSDLGHANDLE = $(HandleNull)
    RasServerOption = "Server"
    RasClientOption = "Client"
    RasAdminOption  = "Admin"
    RasClientAndServerOption = "ClientAndServer"
    !Manufacturer        = "Microsoft"
    !ProductMajorVersion = "4"
    !ProductMinorVersion = "0"
    ProductVersion       = $(!ProductMajorVersion)"."$(!ProductMinorVersion)
    ProductOpSupport     = 134
    NetEventDLL = "%SystemRoot%\System32\netevent.dll"
    IoLogMsgDll = "%SystemRoot%\System32\drivers\IoLogMsg.dll"
    !RasMsgDll   = "%SystemRoot%\System32\rasmsg.dll"
    !RasEventTypeSupported = 31
    !HideComponent = 1
    HideBindings  = 0
    fReviewBindings = 1
    !RasInfName     = "OEMNSVRA.INF"
    ProductFullInfName = $(!STF_CWDDIR)$(!RasInfName)
    RasDir      = $(!STF_CWDDIR)RAS
    ProductPath = $(RasDir)"\"
    ProductRASName        = "RAS"
    ProductRASImagePath   = "%SystemRoot%\system32\ras"
    ProductRASSvcType     = "service"
    NetRuleRASClass       = "classRasService ""basic"""
    NetRuleRASType        = "RasService classRasService"
    NetRuleRASUse         = "system"
    !ProductPCIMACName     = "PCIMAC"
    ProductRASSVRName       = "RemoteAccess"
    ProductRASSVRImagePath  = "%SystemRoot%\system32\rassrv.exe"
    ProductRASSVRSvcType    = "service"
    NetRuleRASSVRClass      = "classRasServer ""basic"""
    NetRuleRASSVRType       = "RasServer classRasServer"
    NetRuleRASSVRUse        = $(ProductRASSVRSvcType)
    NetRuleRASSVRBindForm   = """RasServer"" yes yes container"
    NetRuleRASSVRBindable    = {"classRasServer netBiosTransport non non 100",+
                               "classRasServer ipxTransport non non 100"}
    ProductRASMANName       = "RasMan"
    ProductRASMANImagePath  = "%SystemRoot%\system32\rasman.exe"
    ProductRASMANSvcType    = "service"
    NetRuleRASMANClass      = "classRasManager ""basic"""
    NetRuleRASMANType       = "RasManager classRasManager"
    NetRuleRASMANUse        = $(ProductRASMANSvcType)
    NetRuleRASMANBindForm   = """RasManager"" yes no container"
    NetRuleRASMANBindable   = ""
    !ProductRASARPName       = "RasArp"
    !ProductRASARPImagePath  = "\SystemRoot\system32\drivers\rasarp.sys"
    !ProductRASISNRIPName       = "NwlnkRip"
    !ProductRASISNSAPName       = "NwSapAgent"
    !ProductRASISNRIPImagePath  = "\SystemRoot\system32\drivers\nwlnkrip.sys"
```

```
!ProductRASISNSAPImagePath  = "%SystemRoot%\system32\services.exe"
!ProductRASAUTODIALName       = "RasAuto"
!ProductRASAUTODIALImagePath  = "%SystemRoot%\system32\rasman.exe"
!ProductRASAUTODIALSvcType    = "service"
!ProductRASACDName       = "RasAcd"
!ProductRASACDImagePath  = "\SystemRoot\system32\drivers\rasacd.sys"
!ProductNDISWANName     = "NdisWan"
!ProductNDISWANImagePath = "\SystemRoot\system32\drivers\ndiswan.sys"
ProductNDISWANSvcType   = "kernel"
ProductNDISWANType      = "transport"
!NetRuleNDISWANType     = "ndisWanDrv ndisWanTransport"
!NetRuleNDISWANClass     = {"ndisWanTransport basic"}
NetRuleNDISWANUse       = $(ProductNDISWANType)" yes yes"
!NetRuleNDISWANBindForm  = """NdisWan"" yes no container"
!NetRuleNDISWANBindable  = +
        {"rasCapableTransport ndisWanAdapterDialIn non non 100",+
         "rasCapableTransport ndisWanAdapterDialOut non non 100",+
         "tcpipTransport ndisWanAdapterDialInIP non non 100",+
         "tcpipTransport ndisWanAdapterDialOutIP non non 100",+
         "netbtTransport ndisWanAdapterDialInIP non non 100",+
         "netbtTransport ndisWanAdapterDialOutIP non non 100",+
         "ipxTransport ndisWanAdapterDialInOutIPX non non 100",+
       "bhService    ndisWanAdapterBH non non 100",+
         "ndisWanTransport ndisWanAdapterBH non non 100",+
         "ndisWanTransport ndisWanAdapterDialIn non non 100",+
         "ndisWanTransport ndisWanAdapterDialOut non non 100",+
         "ndisWanTransport ndisWanAdapterDialInIP non non 100",+
         "ndisWanTransport ndisWanAdapterDialOutIP non non 100",+
         "ndisWanTransport ndisWanAdapterDialInOutIPX non non 100"}
ProductRASASYMACName      = "AsyncMac"
ProductRASASYMACImagePath = "\SystemRoot\system32\drivers\asyncmac.sys"
ProductRASASYMACType      = "driver"
ProductRASASYMACSvcType   = "kernel"
NetRuleRASASYMACType      = "rasAsyMacDrv rasAsyMacDriver"
NetRuleRASASYMACClass      = {"rasAsyMacDriver basic"}
NetRuleRASASYMACUse      = $(ProductRASASYMACType)
NetRuleRASASYMACBindForm  = """RasAsyMac"" yes no container"
!NetRuleRASASYMACBindable  = +
                   {"rasAsyMacDriver rasAsyMacAdapter non non 100"}
!ProductRASHUBDIALINName      = "RasHubDialin"
!ProductRASHUBDIALOUTName     = "RasHubDialout"
!ProductRASHUBDIALINIPName      = "RasHubDialinIp"
!ProductRASHUBDIALOUTIPName     = "RasHubDialoutIp"
!ProductRASHUBDIALINOUTIPXName  = "RasHubDialinoutIpx"
!ProductNDISWANDIALINName      = "NdisWanDialin"
!ProductNDISWANDIALOUTName           = "NdisWanDialout"
!ProductNDISWANDIALINIPName       = "NdisWanDialinIp"
!ProductNDISWANDIALOUTIPName      = "NdisWanDialoutIp"
!ProductNDISWANDIALINOUTIPXName   = "NdisWanDialinoutIpx"
!NetRuleHardwareBHType      = "ndisWanBH ndisWanAdapterBH"
!NetRuleHardwareBHClass     = {"ndisWanAdapterBH basic"}
!NetRuleHardwareBHBindForm  = " yes yes container"
!NetRuleHardwareNDISWANBindForm = " yes yes container"
!NetRuleHardwareDIALINType       = "ndiswandialin ndisWanAdapterDialIn"
!NetRuleHardwareDIALINClass      = {"ndisWanAdapterDialIn basic"}
!NetRuleHardwareDIALINBlock    = {"lanmanServer ndisWanAdapterDialIn",+
                         "tcpipTransport ndisWanAdapterDialIn",+
                         "netbtTransport ndisWanAdapterDialIn",+
                         "ipxTransport ndisWanAdapterDialin",+
                         "lanmanWorkstation ndisWanAdapterDialIn"}
!NetRuleHardwareDIALINIPType     = "ndiswandialinIP ndisWanAdapterDialInIP"
!NetRuleHardwareDIALINIPClass    = {"ndisWanAdapterDialInIP basic"}
!NetRuleHardwareDIALINIPBlock    = {"lanmanServer ndisWanAdapterDialInIP",+
                          "nbfTransport ndisWanAdapterDialinIP",+
```

```
                                "ipxTransport ndisWanAdapterDialinIP",+
                                "lanmanWorkstation ndisWanAdapterDialInIP"}
    !NetRuleHardwareDIALOUTType       = "ndiswandialout ndisWanAdapterDialOut"
    !NetRuleHardwareDIALOUTClass    = {"ndisWanAdapterDialOut basic"}
    !NetRuleHardwareDIALOUTBlock    = {"ipxTransport ndisWanAdapterDialOut"}
    !NetRuleHardwareDIALOUTIPType     = "ndiswandialoutIP
ndisWanAdapterDialOutIP"
    !NetRuleHardwareDIALOUTIPClass    = {"ndisWanAdapterDialOutIP basic"}
    !NetRuleHardwareDIALOUTIPBlock    = {"ipxTransport ndisWanAdapterDialOutIP",
+
                                "nbfTransport ndisWanAdapterDialOutIP"}
    !NetRuleHardwareDIALINOUTIPXType = "ndiswandialinoutIPX
ndisWanAdapterDialInOutIPX"
    !NetRuleHardwareDIALINOUTIPXClass = {"ndisWanAdapterDialInOutIPX basic"}
    !NetRuleHardwareDIALINOUTIPXBlock = {"nbfTransport
ndisWanAdapterDialInOutIPX",+
                                "tcpipTransport ndisWanAdapterDialInOutIPX",+
                                "netbtTransport ndisWanAdapterDialInOutIPX"}
    NetRuleHardwareRASASYMACType     = "rasAsyMac rasAsyMacAdapter"
    NetRuleHardwareRASASYMACBindForm = " yes yes container"
    NetRuleHardwareRASASYMACClass     = {"rasAsyMacAdapter basic"}
    !ProductNDISTAPIName        = "NdisTapi"
    !ProductNDISTAPIImagePath   = "\SystemRoot\system32\drivers\ndistapi.sys"
    !RasPerfKeyName             = "Performance"
    !RasPerfLibraryName         = "rasctrs.dll"
    !RasPerfOpenFunction        = "OpenRasPerformanceData"
    !RasPerfCloseFunction       = "CloseRasPerformanceData"
    !RasPerfCollectFunction     = "CollectRasPerformanceData"
    !RasCounterFileName         = "rasctrs.ini"
    ProductKeyBase  = $(!NTN_SoftwareBase)"\"$(!Manufacturer)
    ProductKeyName           = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASName)"\CurrentVersion"
    !NetworkCardKeyName      = $(!NTN_SoftwareBase)"\Microsoft\Windows NT\
CurrentVersion\NetworkCards"
    RasSvrKeyName            = $(!NTN_ServiceBase)"\"$(ProductRASSVRName)
    RasSvrParamKeyName       = $(!NTN_ServiceBase)"\"$(ProductRASSVRName)"\
Parameters"
    !RasManSvcKeyName        = $(!NTN_ServiceBase)"\"$(ProductRASMANName)
    !RasAcdKeyName           = $(!NTN_ServiceBase)"\"$(!ProductRASACDName)
    !RasArpKeyName           = $(!NTN_ServiceBase)"\"$(!ProductRASARPName)
    !RasIsnRipKeyName        = $(!NTN_ServiceBase)"\"$(!ProductRASISNRIPName)
    !RasIsnSapKeyName        = $(!NTN_ServiceBase)"\"$(!ProductRASISNSAPName)
    !NdisTapiKeyName         = $(!NTN_ServiceBase)"\"$(!ProductNDISTAPIName)
    !RasManKeyName           = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASMANName)"\CurrentVersion"
    !RasAutodialKeyName       = $(!NTN_ServiceBase)"\"$(!ProductRASAUTODIALName)
    RasManParamKeyName       = $(!NTN_ServiceBase)"\"$(ProductRASMANName)"\
Parameters"
    RasManLinkageKeyName     = $(!NTN_ServiceBase)"\"$(ProductRASMANName)"\
Linkage"
    NdisWanParamKeyName      = $(!NTN_ServiceBase)"\"$(!ProductNDISWANName)"\
Parameters"
    NdisWanLinkageKeyName    = $(!NTN_ServiceBase)"\"$(!ProductNDISWANName)"\
Linkage"
    RasAsyMacKeyName         = $(!NTN_ServiceBase)"\"$(ProductRASASYMACName)
    RasAsyMacParamKeyName    = $(!NTN_ServiceBase)"\"$(ProductRASASYMACName)"\
Parameters"
    RasMacLinkageKeyName     = $(!NTN_ServiceBase)"\"$(ProductRASASYMACName)"\
Linkage"
    RasTapiDevicesKeyName    = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASName)"\TAPI DEVICES"
    LinkageKeyName           = $(!NTN_ServiceBase)"\"$(Product$(Option)Name)"\
Linkage"
    !ProductNDISWANKeyName   = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$(!
```

```
ProductNDISWANName)"\CurrentVersion"
    ProductRASASYMACKeyName  = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASASYMACName)"\CurrentVersion"
    RasProtocolsKeyName      = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASName)"\PROTOCOLS"
    !UtilityInf     = "UTILITY.INF"
    subroutninf    = "SUBROUTN.INF"
    RascfgDll      = "RASCFG.DLL"
    Exit_Code      = 0
    ShellCode      = 0
    from      = ""
    to        = ""
    ExitCodeOk     = 0
    ExitCodeCancel = 1
    ExitCodeFatal  = 2
    KeyNull         = ""
    !MAXIMUM_ALLOWED   = 33554432
    !SERVICE_NO_CHANGE = 4294967295
    RegistryErrorIndex = NO_ERROR
    ServerSize  = 820116
    ClientSize  = 634260
    AdminSize   = 513962
    FInstallServer = $(Chosen)
    FInstallClient = $(Chosen)
    FInstallAdmin  = $(Chosen)
[UiVars]
    VolumeList      = {} ? $(!LIBHANDLE) GetHardDriveLetters
    VolumeFreeList  = {} ? $(!LIBHANDLE) GetHardDriveFreeSpace
    VolumeFSList    = {} ? $(!LIBHANDLE) GetHardDriveFileSystems
    DestVolume      = ""
    MinHelpID   = 25000
    MaxHelpID   = 25999
[SystemVars]
    !STF_PROCESSOR      = "" ? $(!LIBHANDLE) GetProcessor
    !STF_PLATFORM       = "" ? $(!LIBHANDLE) GetPlatform
    !STF_WINDOWSPATH    = "" ? $(!LIBHANDLE) GetWindowsNtDir
    !STF_NTPATH         = $(!STF_WINDOWSPATH)
    !STF_WINDOWSSYSPATH = "" ? $(!LIBHANDLE) GetWindowsNtSysDir
    !STF_COMPUTERNAME   = "" ? $(!LIBHANDLE) GetMyComputerName
    !STF_USERNAME       = "" ? $(!LIBHANDLE) GetMyUserName
    !STF_FLOPPYLIST     = {} ? $(!LIBHANDLE) GetFloppyDriveLetters
    !STF_UNUSEDDRIVES   = {} ? $(!LIBHANDLE) GetUnusedDrives
    !STF_LANGUAGE       = "" ? $(!LIBHANDLE) GetLanguage
    !STF_BUSTYPE        = "" ? $(!LIBHANDLE) GetMyBusType
    !STF_BUSTYPELIST    = "" ? $(!LIBHANDLE) GetMyBusTypeList
    !NTN_SoftwareBase   = "Software"
    !NTN_ServiceBase    = "System\CurrentControlSet\Services"
    !NTN_ScUseRegistry  = "NO"
[AvailableSystemMemory]
    SystemMemory    = "" ? $(!LIBHANDLE) GetMemorySize
    MinSystemMemory = 2048
[date]
    !CurrentDate = {} ? $(!LIBHANDLE) GetSystemDate
[HelpContextIDs]
    HC_RASOPTIONS     = 25020
    HC_CLIENTACCESS   = 25021
    HC_INSTALLOVER1   = 25022
    HC_INSTALLOVER2   = 25023
[Identify]
    read-syms Identification
    set Status     = STATUS_SUCCESSFUL
    set Identifier = $(OptionType)
    set Media      = #("Source Media Descriptions", 1, 1)
    Return $(Status) $(Identifier) $(Media)
```

```
[ReturnOptions]
    set Status        = STATUS_FAILED
    set OptionList     = {}
    set OptionTextList = {}
    set LanguageList = ^(LanguagesSupported, 1)
    Ifcontains(i) $($0) in $(LanguageList)
        goto returnoptions
    Else
        set Status = STATUS_NOLANGUAGE
        goto finish_ReturnOptions
    Endif
returnoptions = +
    set OptionList     = ^(Options, 1)
    set OptionTextList = ^(OptionsText$($0), 1)
    set Status         = STATUS_SUCCESSFUL
finish_ReturnOptions = +
    Return $(Status) $(OptionList) $(OptionTextList)
[Shell Commands]
    set Exit_Code = $(!STF_ERROR_GENERAL)
    LoadLibrary "x" $(!STF_SRCDIR)\setupdll.dll !LIBHANDLE
    LoadLibrary "x" $(!STF_SRCDIR)\ncpa.cpl !NCPA_HANDLE
    ifstr(i) $(!LIBHANDLE) == ""
       Debug-Output "OEMNSVRA.INF: unable to load setupdll.dll"
       Shell subroutn.inf, SetupMessage $(!STF_LANGUAGE) "FATAL" $
(UnableToLoadSetupdll)
       set Exit_Code = STATUS_USERCANCEL
       exit
    endif
    ifstr(i) $(!NCPA_HANDLE) == ""
       Debug-Output "OEMNSVRA.INF: unable to load ncpa.cpl"
       Shell subroutn.inf, SetupMessage $(!STF_LANGUAGE) "FATAL" $
(UnableToLoadNcpaCpl)
       set Exit_Code = STATUS_USERCANCEL
       exit
    endif
    ifstr(i) $(RAS_INSTALL_MODE) == ""
       set RAS_INSTALL_MODE = install
    endif
    read-syms SystemVars
    detect    SystemVars
    read-syms RasErrors$(!STF_LANGUAGE)
    OpenRegKey $(!REG_H_LOCAL) "" "System\CurrentControlSet\Control\
ProductOptions" 33554432 KeyProductOption
    ifstr(i) $(KeyProductOption) != ""
        GetRegValue $(KeyProductOption) "ProductType" ProductTypeList
        set TempProductType = *($(ProductTypeList),4)
        ifstr(i) $(TempProductType) == "winnt"
            set !STF_PRODUCT = WINNT
        else-ifstr(i) $(TempProductType) == "lanmannt"
            set !STF_PRODUCT = LANMANNT
        else-ifstr(i) $(TempProductType) == "servernt"
            set !STF_PRODUCT = SERVERNT
        else
            Debug-Output "OEMNSVRA.INF: invalid Product Type "$(TempProductType)
            Shell subroutn.inf, SetupMessage $(!STF_LANGUAGE) "FATAL" $
(UnknownProductType)
            set Exit_Code = STATUS_USERCANCEL
            exit
        endif
        CloseRegKey $(KeyProductOptions)
    else
        Debug-Output "OEMNSVRA.INF: unable to determine Product Type "
        Shell subroutn.inf, SetupMessage $(!STF_LANGUAGE) "FATAL" $
(NoProductType)
```

```
        set Exit_Code = STATUS_USERCANCEL
        exit
    endif
    Debug-Output "OEMNSVRA.INF: Product Type "$(!STF_PRODUCT)
    ifstr(i) $(RAS_INSTALL_MODE) != "install"
       ifstr(i) $(RAS_INSTALL_MODE) != "configure"
          ifstr(i) $(RAS_INSTALL_MODE) != "deinstall"
             Debug-Output "OEMNSVRA.INF: Invalid RAS_INSTALL_MODE "$
(RAS_INSTALL_MODE)
             Shell subroutn.inf, SetupMessage $(!STF_LANGUAGE) "WARNING" $
(InvalidMode)
             set Exit_Code = STATUS_USERCANCEL
             exit
          endif
       endif
    endif
    set !NTN_InstallMode = $(RAS_INSTALL_MODE)
    Shell "" InstallOption $(!STF_LANGUAGE) "RAS" $(!STF_SRCDIR) "YES" "YES"
"YES"
    ifint $($ShellCode) != $( !SHELL_CODE_OK)
       Debug-Output "OEMNSVRA.INF: Failed to shell out to InstallOption"
       exit
    endif
    set Exit_Code = $($R0)
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
       set Exit_Code = 1
    endif
    Debug-Output "Oemnsvra.inf: install returned "$($R0)
    FreeLibrary $(!LIBHANDLE)
    FreeLibrary $(!NCPA_HANDLE)
    exit
[InstallOption]
    Debug-Output "In InstallOption section"
    set Status   = STATUS_FAILED
    set Option      = $($1)
    set SourceDir   = $($2)
    set AddCopy     = $($3)
    set DoCopy      = $($4)
    set DoConfig    = $($5)
    set LanguageList = ^(LanguagesSupported, 1)
    Ifcontains(i) $($0) NOT-IN $(LanguageList)
        Return STATUS_NOLANGUAGE
    endif
    StartWait
    set-subst LF = "\n"
    read-syms UiVars
    detect    UiVars
    set !STF_NTPATH = $(!STF_WINDOWSSYSPATH)
    GetDriveInPath !STF_NTDRIVE $(!STF_WINDOWSSYSPATH)
    read-syms GeneralConstants
    read-syms FileConstants
    read-syms DialogConstants$(!STF_LANGUAGE)
    read-syms FileConstants$(!STF_LANGUAGE)
    read-syms RasErrors$(!STF_LANGUAGE)
    detect date
    read-syms HelpContextIDs
    SetHelpFile $(!STF_CWDDIR)rassetup.hlp $(MinHelpID) $(MaxHelpID)
    set-title  $(FunctionTitle)
    set to   = Begin
    set from = Begin
    set CommonStatus = STATUS_SUCCESSFUL
    Debug-Output "OEMNSVRA.INF: STF_CWDDIR is: "$(!STF_CWDDIR)
    Debug-Output "OEMNSVRA.INF: STF_LANGUAGE is: "$(!STF_LANGUAGE)
    Debug-Output "OEMNSVRA.INF: Option is: "$(Option)
```

```
    Debug-Output "OEMNSVRA.INF: NTN_SoftwareBase is: "$(!NTN_SoftwareBase)
    Debug-Output "OEMNSVRA.INF: NTN_ServiceBase is: "$(!NTN_ServiceBase)
Begin = +
    set !NETCARD_LIST    = {}
    set ServerInstalled = FALSE
    set ClientInstalled = FALSE
    set AdminInstalled  = FALSE
    set DoServer     = TRUE
    set DoClient     = TRUE
    set DoAdmin      = TRUE
    set DoServerOnly = FALSE
    set DoClientOnly = FALSE
    set DoAdminOnly  = FALSE
    set DoRas        = FALSE
    set DoRasSvr     = FALSE
    set DoRasMan     = FALSE
    set DoNdisWan     = FALSE
    set DoRasAsyMac  = FALSE
    IfStr(i) $(!NTN_InstallMode)    ==    deinstall
        Set StartLabel = RemoveRas
    else-Ifstr(i) $(!NTN_InstallMode) == Update
        set StartLabel = UpgradeRas
    else-Ifstr(i) $(!NTN_InstallMode) == configure
        set StartLabel = ConfigureRas
    else-Ifstr(i) $(!NTN_InstallMode) == bind
        set StartLabel = BindingsRas
    else
        set StartLabel = InstallRas
    endif
    set from = Begin
    set to = end
    goto $(StartLabel)
InstallRas = +
    StartWait
    Debug-Output "In InstallRas Section"
    Debug-Output "Origin is "$(!NTN_Origination)
    Debug-Output "Install Mode "$(!NTN_InstallMode)
    Debug-Output "Install phase "$(!NTN_InstallPhase)
    Debug-Output "Copy Flag is "$(DoCopy)
    Debug-Output "Option is "$(Option)
    Debug-Output "SrcDir is "$(SourceDir)
    Debug-Output "STF_WINDOWSPATH is "$(!STF_WINDOWSPATH)
    Debug-Output "Context Inf name "$(STF_CONTEXTINFNAME)
    Debug-Output "!STF_UNATTENDED is "$(!STF_UNATTENDED)
    Debug-Output "!STF_GUI_UNATTENDED is "$(!STF_GUI_UNATTENDED)
    Debug-Output "!STF_UNATTENDED_SECTION is "$(!STF_UNATTENDED_SECTION)
    set RasComponentsList = {}
    set NumInstalled = 0
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(!MAXIMUM_ALLOWED)
KeyProduct
    Ifstr $(KeyProduct) != $(KeyNull)
        Ifstr(i) $(!NTN_InstallPhase) == primary
           set CommonStatus = STATUS_USERCANCEL
           goto end
        EndIf
        GetRegValue $(KeyProduct) "MajorVersion" VersionInfo
        set VersionMajor = *($(VersionInfo), 4)
        GetRegValue $(KeyProduct) "MinorVersion" VersionInfo
        set VersionMinor = *($(VersionInfo), 4)
        set InstalledVersion = $(VersionMajor)"."$(VersionMinor)
        ifstr(i) $(ProductVersion) != $(InstalledVersion)
            Shell "" QueryRasUpgrade $(ProductRASTitle) $(InstalledVersion) +
                                     $(ProductVersion)
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
```

```
               goto ShellCodeError
            endif
            Ifstr(i) $($R1) == "OK"
                set from = InstallRas
                set to   = InstallRas1
                goto RemoveRas
            else
                goto end
            endif
        endif
        shell "" QueryComponentsInstalled $(ProductKeyName)
        Ifstr(i) $($R0) == STATUS_SUCCESSFUL
            Set RasComponentsList = $($R1)
            Set InstalledFlags = $($R2)
            Set ServerInstalled = *($(InstalledFlags),1)
            Set ClientInstalled = *($(InstalledFlags),2)
            Set AdminInstalled  = *($(InstalledFlags),3)
        Endif
        Debug-Output "Installed List is "$(RasComponentsList)
        Debug-Output "Installed Flags is "$(InstalledFlags)
        QueryListSize NumInstalled $(RasComponentsList)
        ifint $(NumInstalled) == 3
            read-syms VerExists$(!STF_LANGUAGE)
            set Text = $(Product$(Option)Title)$(Ver)$(ProductVersion)+
                       $(Text1)
            Shell $(subroutninf), SetupMessage $(!STF_LANGUAGE) "NONFATAL" $
(Text)
            Ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Goto fatal
            Else-Ifstr(i) $($R0) == STATUS_FAILED
                Goto fatal
            Else-Ifstr(i) $($R0) == STATUS_USERCANCEL
                Goto end
            Endif
            goto end
        endif
        CloseRegKey $(KeyProduct)
    else
InstallRas1 =+
       set fRemoveRas   = TRUE
    endif
    goto memorycheck
setupabort = +
    read-syms SetupAbortDlg$(!STF_LANGUAGE)
    ui start "SetupAbort"
    ifstr(i) $(DLGEVENT) == "CONTINUE"
        ui pop 1
        Exit
    else
        ui pop 1
        Exit
    endif
memorycheck = +
    read-syms AvailableSystemMemory
    detect AvailableSystemMemory
    Debug-Output "Available Memory is "$(SystemMemory)
    IfInt $(SystemMemory) < $(MinSystemMemory)
        read-syms FatalErrorMem$(!STF_LANGUAGE)
        shell "subroutn.inf" SetupMessage $(!STF_LANGUAGE) "FATAL" $(Fatal)
        goto setupabort
    Else
        goto InstallNetwork
    EndIf
InstallNetwork =+
```

```
    ifstr(i) $(DoServer) == FALSE
       ifstr(i) $(DoClient) == FALSE
           goto CopyResources
       endif
    endif
    goto CopyResources
CopyResources = +
    ifstr(i) $(!NTN_InstallMode) == "install"
        Ifstr(i) $(DoCopy) == "YES"
           Shell $(!UtilityInf), DoAskSource, $(!STF_CWDDIR), $(SourceDir) YES
           Ifint $($ShellCode) != $(!SHELL_CODE_OK)
               Goto ShellCodeError
           Else-Ifstr(i) $($R0) == STATUS_FAILED
               Shell $(!UtilityInf) RegistryErrorString "ASK_SOURCE_FAIL"
               ifint $($ShellCode) != $(!SHELL_CODE_OK)
                   goto ShellCodeError
               endif
               set Error = $($R0)
               goto fatal
           Else-Ifstr(i) $($R0) == STATUS_USERCANCEL
               Goto end
           Endif
           Set !STF_SRCDIR = $($R1)
           Set SourceDir  = $($R1)
           Debug-Output "SrcDir "$(SourceDir)
        endif
        Debug-Output "OEMNSVRA.INF: Copying files"
        Install InstallResources
        ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
            goto filecopycancel
        endif
        install InstallRasFiles
        ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
            goto filecopycancel
        endif
    endif
    StartWait
    ifstr(i) $(DoAdminOnly) == TRUE
        goto CommonCode
    endif
    goto PortsConfigure
ConfigureRas = +
    set OldVersionExisted = $(TRUE)
    read-syms StatusDeterminingConfig$(!STF_LANGUAGE)
    shell $(subroutninf) PushBillBoard NETSTATUSDLG $(ReadingConfig)
    Set BillboardVisible = 1
    StartWait
    shell "" QueryComponentsInstalled
    Ifstr(i) $($R0) == STATUS_SUCCESSFUL
        Set InstalledComps = $($R1)
        Set InstalledFlags = $($R2)
        Set DoServer      = *($(InstalledFlags),1)
        Set DoClient      = *($(InstalledFlags),2)
        Set DoAdmin       = *($(InstalledFlags),3)
        Set DoServerOnly  = *($(InstalledFlags),4)
        Set DoClientOnly  = *($(InstalledFlags),5)
        Set DoAdminOnly   = *($(InstalledFlags),6)
    Endif
    Set ServerInstalled = $(DoServer)
    Set ClientInstalled = $(DoClient)
    Set AdminInstalled  = $(DoAdmin)
    Debug-Output "DoServer "$(DoServer)
    Debug-Output "DoClient "$(DoClient)
    Debug-Output "DoAdmin "$(DoAdmin)
```

```
    Debug-Output "DoServerOnly "$(DoServerOnly)
    Debug-Output "DoClientOnly "$(DoClientOnly)
    Debug-Output "DoAdminOnly "$(DoAdminOnly)
    Ifint $(BillboardVisible) != 0
        Shell "subroutn.inf" PopBillboard
        Set BillboardVisible = 0
    Endif
    ifstr(i) $(DoAdminOnly) == TRUE
        read-syms NoConfigAdmin$(!STF_LANGUAGE)
        shell "subroutn.inf" SetupMessage $(!STF_LANGUAGE) "STATUS" $
(NoConfigAdmin)
        goto end
    endif
    goto PortsConfigure
PortsConfigure = +
    StartWait
    read-syms StatusDeterminingConfig$(!STF_LANGUAGE)
    shell $(subroutninf) PushBillBoard NETSTATUSDLG $(ReadingConfig)
    Set BillboardVisible = 1
    LoadLibrary "x" $(!STF_CWDDIR)rascfg.dll PORTSDLGHANDLE
    set Result = {}
    set fNetcardInstalled = FALSE
    shell "" IsNetcardInstalled
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
        set fNetcardInstalled = $($R1)
    endif
    Debug-Output "fNetcardInstalled Option "$(fNetcardInstalled)
CheckProtocolsInstalled =+
    shell "" QueryInstalledProtocols
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
        set fNetbeuiInstalled = $($R1)
        set fTcpIpInstalled   = $($R2)
        set fIpxInstalled     = $($R3)
    else
        read-syms QueryInstalledProtocolsError$(!STF_LANGUAGE)
        shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "FATAL" $(Text)
        goto end
    endif
    Ifint $(BillboardVisible) != 0
        Shell "subroutn.inf" PopBillboard
        Set BillboardVisible = 0
    Endif
    Debug-Output "!STF_UNATTENDED is "$(!STF_UNATTENDED)
    Debug-Output "!STF_GUI_UNATTENDED is "$(!STF_GUI_UNATTENDED)
    Debug-Output "!STF_UNATTENDED_SECTION is "$(!STF_UNATTENDED_SECTION)
    LibraryProcedure Result, $(PORTSDLGHANDLE), RasPortsConfig $(!STF_HWND) +
                     $(!NTN_InstallMode) $(fNetcardInstalled) $(ProductPath) +
                     $(fNetbeuiInstalled) $(fTcpIpInstalled) $(fIpxInstalled) +
                     $(!STF_GUI_UNATTENDED) $(!STF_UNATTENDED) $(!
STF_UNATTENDED_SECTION)
    StartWait
    set NewNumPorts   = 0
    set NewNumTapiPorts = 0
    set NewNumDialoutNBF = 0
    set NewNumDialinNBF  = 0
    set NewNumDialinIP = 0
    set NewNumDialoutIP = 0
    set NewNumDialinoutIPX = 0
    set fSerialInstalled = FALSE
    set fUnimodemInstalled = FALSE
    set fOtherInstalled = FALSE
    set fNetbeuiChosen = FALSE
    set fTcpIpChosen = FALSE
    set fIpxChosen = FALSE
```

```
    set fNetbeuiSelected = FALSE
    set fTcpIpSelected = FALSE
    set fIpxSelected = FALSE
    set fNetbeuiAllowed = FALSE
    set fTcpIpAllowed = FALSE
    set fIpxAllowed = FALSE
    set NewNumPorts       = *($(Result),1)
    set NewNumTapiPorts   = *($(Result),2)
    set NewNumPortsList = {}
    set NullString   = ""
    Debug-Output "PortsConfigure returned: "$(Result)
    Debug-Output "PortsConfigure NewNumPorts: "$(NewNumPorts)
    Debug-Output "PortsConfigure NewNumTapiPorts: "$(NewNumTapiPorts)
    ifstr(i) $(NewNumPorts) == "EXITSETUP"
        Debug-Output "PortsConfigre User selected ExitSetup."
        goto RemoveDiskFiles
    else-ifstr(i) $(NewNumPorts) == "NOPORTS"
        Debug-Output "PortsConfigre: No serial ports detected."
        shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "NONFATAL" $
(ErrorNoPorts)
        goto RemoveDiskFiles
    else-ifstr(i) $(NewNumPorts) == "BADARGS"
        Debug-Output "PortsConfigre: bad arguments to dll."
        shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "NONFATAL" $
(ErrorBadArgs)
        goto RemoveDiskFiles
    else-ifstr(i) $(NewNumPorts) == "FAILURE"
        Debug-Output "PortsConfigre: Unknown failure."
        shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "NONFATAL" $
(ErrorUnknown)
        goto RemoveDiskFiles
    endif
    goto PortsConfigure1
RemoveDiskFiles = +
    set CommonStatus = STATUS_USERCANCEL
    ifstr(i) $(!NTN_InstallMode) == "install"
        goto filecopycancel
    endif
    goto end
PortsConfigure1 = +
    set NewNumDialoutNBF = *($(Result),3)
    Debug-Output "PortsConfigure NewNumDialoutNBF: "$(NewNumDialoutNBF)
    set NewNumDialinNBF  = *($(Result),4)
    Debug-Output "PortsConfigure NewNumDialinNBF: "$(NewNumDialinNBF)
    set fSerialInstalled = *($(Result),5)
    Debug-Output "PortsConfigure fSerialInstalled: "$(fSerialInstalled)
    set fUnimodemInstalled = *($(Result),6)
    Debug-Output "PortsConfigure fUnimodemInstalled: "$(fUnimodemInstalled)
    set fOtherInstalled = *($(Result),7)
    Debug-Output "PortsConfigure fOtherInstalled: "$(fOtherInstalled)
    ifstr(i) $(fSerialInstalled) == TRUE
        set NewNumPortsList = >($(NewNumPortsList), $(NullString)$(NewNumPorts)$
(NullString))
    else-ifstr(i) $(fOtherInstalled) == TRUE
        set NewNumPortsList = >($(NewNumPortsList), $(NullString)$(NewNumPorts)$
(NullString))
    else-ifstr(i) $(fUnimodemInstalled) == TRUE
        set NewNumPortsList = >($(NewNumPortsList), $(NullString)$(NewNumPorts)$
(NullString))
    endif
    set fNetbeuiSelected = *($(Result),8)
    Debug-Output "PortsConfigure NetbeuiSelected: "$(fNetbeuiSelected)
    set fTcpIpSelected = *($(Result),9)
    Debug-Output "PortsConfigure TcpIpSelected: "$(fTcpIpSelected)
```

```
set fIpxSelected = *($(Result),10)
Debug-Output "PortsConfigure IpxSelected: "$(fIpxSelected)
set fNetbeuiAllowed = *($(Result),11)
Debug-Output "PortsConfigure NetbeuiAllowed: "$(fNetbeuiAllowed)
set fTcpIpAllowed = *($(Result),12)
Debug-Output "PortsConfigure TcpIpAllowed: "$(fTcpIpAllowed)
set fIpxAllowed = *($(Result),13)
Debug-Output "PortsConfigure IpxAllowed: "$(fIpxAllowed)
ifstr(i) $(fNetbeuiSelected) == TRUE
    set fNetbeuiChosen = TRUE
else
    set fNetbeuiChosen = $(fNetbeuiAllowed)
endif
ifstr(i) $(fTcpIpSelected) == TRUE
    set fTcpIpChosen = TRUE
else
    set fTcpIpChosen = $(fTcpIpAllowed)
endif
ifstr(i) $(fIpxSelected) == TRUE
    set fIpxChosen = TRUE
else
    set fIpxChosen = $(fIpxAllowed)
endif
ifstr(i) $(fTcpIpSelected) == TRUE
    set NewNumDialoutIP = $(NewNumDialoutNBF)
endif
ifint $(NewNumDialinNBF) != 0
     ifstr(i) $(fTcpIpAllowed) == TRUE
         set NewNumDialinIP = 1
     endif
endif
Debug-Output "NewNumDialinIP: "$(NewNumDialinIP)
Debug-Output "NewNumDialoutIP: "$(NewNumDialoutIP)
ifstr(i) $(fIpxChosen) == TRUE
    set NewNumDialinoutIPX = 1
    Debug-Output "IPX Selected - NewNumDialinoutIPX: "$(NewNumDialinoutIPX)
endif
ifstr(i) $(fIpxAllowed) == TRUE
    set EnableIpxRouter = 1
else
    set EnableIpxRouter = 0
endif
ifstr(i) $(fNetbeuiAllowed) == FALSE
     set NewNumDialinNBF = 0
endif
ifstr(i) $(fNetbeuiSelected) == FALSE
    set NewNumDialoutNBF = 0
endif
Debug-Output "NewNumDialinNbf: "$(NewNumDialinNBF)
Debug-Output "NewNumDialoutNbf: "$(NewNumDialoutNBF)
set NetBiosGtwyEnabled = 0
set EnableIpRouter     = 0
set EnableWanRouter    = 0
Shell "" GetNetworkAccess
ifint $($ShellCode) != $(!SHELL_CODE_OK)
    Debug-Output "error shelling GetNetworkAccess."
    goto ShellCodeError
endif
set NetBiosGtwyEnabled = $($R1)
set EnableIpRouter     = $($R2)
set EnableWanRouter    = $($R3)
ifint $(NetBiosGtwyEnabled) == 0
    ifstr(i) $(fNetbeuiAllowed) == TRUE
        set-add NewNumDialoutNBF  = $(NewNumDialoutNBF), $(NewNumDialinNBF)
```

```
            set     NewNumDialinNBF = 0
            Debug-Output "Mapped dialin ports to dialout ports "$
(NewNumDialoutNBF)
        endif
    endif
    set NewNumDialin = 0
    set NewNumDialout = 0
    set-add NewNumDialin = $(NewNumDialinNBF), $(NewNumDialinIP)
    ifstr(i) $(fIpxAllowed) == TRUE
        set-add NewNumDialin = $(NewNumDialin), $(NewNumDialinoutIPX)
    endif
    set-add NewNumDialout = $(NewNumDialoutNBF), $(NewNumDialoutIP)
    ifstr(i) $(fIpxSelected) == TRUE
        set-add NewNumDialout = $(NewNumDialout), $(NewNumDialinoutIPX)
    endif
    ifstr(i) $(fNetcardInstalled) == FALSE
        ifint $(NewNumDialin) != 0
            ifstr(i) $(!STF_INSTALL_MODE) != EXPRESS
               read-syms InstallLoopback$(!STF_LANGUAGE)
               Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "STATUS", $
(Message)
            else
               set AddCopy  = YES
               set DoCopy   = YES
               set DoConfig = YES
               set SaveNTN_InstallMode = $(!NTN_InstallMode)
               set !NTN_InstallMode = install
               Shell "oemnadlb.inf" InstallOption $(!STF_LANGUAGE) "LOOP" $(!
STF_SRCDIR) $(AddCopy) $(DoCopy) $(DoConfig)
               set !NTN_InstallMode = $(SaveNTN_InstallMode)
               Ifint $($ShellCode) != $(!SHELL_CODE_OK)
                   Debug-Output "OEMNSVRA.INF: INF oemnadlb.inf SHELL ERROR!"
                   read-syms InstallLoopback$(!STF_LANGUAGE)
                   Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE),
"STATUS", $(Error)
               Endif
            EndIf
        endif
    endif
    goto CommonCode
CommonCode = +
    ifstr(i) $(OldVersionExisted) == $(TRUE)
        ifstr(i) $(!NTN_InstallMode) == configure
           goto WriteParameters
        endif
    endif
    QueryListSize ListCount $(RasComponentsList)
    ifint $(ListCount) == 0
        goto FirstTimeInstall
    else
        ifstr(i) $(DoServer) == TRUE
            ifstr(i) $(ClientInstalled) == FALSE
                ifstr(i) $(AdminInstalled) == FALSE
                   set DoRas    = TRUE
                endif
                set DoRasSvr     = TRUE
                set DoRasMan     = TRUE
                set DoNdisWan    = TRUE
                set DoRasAsyMac  = TRUE
            else
                set DoRasSvr     = TRUE
            endif
            set RasComponentsList = >($(RasComponentsList), $(RasServerOption))
        endif
```

```
        ifstr(i) $(DoClient) == TRUE
            ifstr(i) $(ServerInstalled) == FALSE
                ifstr(i) $(AdminInstalled) == FALSE
                    set DoRas    = TRUE
                endif
                set DoRasMan     = TRUE
                set DoNdisWan    = TRUE
                set DoRasAsyMac  = TRUE
            endif
            set RasComponentsList = >($(RasComponentsList), $(RasClientOption))
        endif
        ifstr(i) $(DoAdmin) == TRUE
            ifstr(i) $(ServerInstalled) == FALSE
                ifstr(i) $(ClientInstalled) == FALSE
                    set DoRas = TRUE
                endif
            endif
            set RasComponentsList = >($(RasComponentsList), $(RasAdminOption))
        endif
        shell "" UpdateComponentsInstalled $(RasComponentsList) $
(ProductKeyName)
        goto FirstTimeInstall-1
    endif
FirstTimeInstall =+
    ifstr(i) $(DoServer) == TRUE
        set DoRas             = TRUE
        set DoRasSvr          = TRUE
        set DoRasMan          = TRUE
        set DoRasAutodial     = TRUE
        set DoNdisWan         = TRUE
        set DoRasAsyMac       = TRUE
        set RasComponentsList = >($(RasComponentsList), $(RasServerOption))
    endif
    ifstr(i) $(DoClient) == TRUE
        set DoRas             = TRUE
        set DoRasMan          = TRUE
        set DoRasAutodial     = TRUE
        set DoNdisWan         = TRUE
        set DoRasAsyMac       = TRUE
        set RasComponentsList = >($(RasComponentsList), $(RasClientOption))
    endif
    ifstr(i) $(DoAdmin) == TRUE
        set DoRas             = TRUE
        set RasComponentsList = >($(RasComponentsList), $(RasAdminOption))
    endif
FirstTimeInstall-1 = +
    ifstr(i) $(DoRas) == TRUE
        StartWait
        read-syms StatusUpdatingRegistry$(!STF_LANGUAGE)
        shell $(subroutninf) PushBillBoard NETSTATUSDLG $(CreatingRas)
        Set BillboardVisible = 1
        set ThisOption = RAS
        Shell $(!UtilityInf), InstallSoftwareProduct, $(!Manufacturer),+
              $(Product$(ThisOption)Name), $(!RasInfName)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
              Debug-Output "InstallSoftware bombed out."
              goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            Debug-Output "REGISTRY GOODAL"
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            goto fatalregistry
```

```
        endif
        Set SoftProductKey    = $($R1)
        Set SoftNetRuleKey    = $($R2)
        set NewValueList = +
            {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
             {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)SvcType)},+
             {Title,$(NoTitle),$(!REG_VT_SZ), $(Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Description)},+
             {PathName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)ImagePath)},+
             {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},
+
             {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},
+
             {RasComponents,$(NoTitle),$(!REG_VT_MULTI_SZ),$
(RasComponentsList)},+
             {OperationsSupport,$(NoTitle),$(!REG_VT_DWORD),$
(ProductOpSupport)}, +
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)}}
        Shell $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            CloseRegKey $(SoftProductKey)
            CloseRegKey $(SoftNetRuleKey)
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftProductKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
           CloseRegKey $(SoftNetRuleKey)
           goto fatalregistry
        endif
        set NewValueList = +
            {{class, $(NoTitle), $(!REG_VT_SZ), $(NetRule$(ThisOption)Class)},+
             {type,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Type)},+
             {use,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Use)}, +
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftNetRuleKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            goto fatalregistry
        endif
    endif
    ifstr(i) $(DoRasSvr) == TRUE
        set ThisOption = RASSVR
        set RasSpecificString = $(Product$(ThisOption)Name)
        Shell $(!UtilityInf), AddSoftwareComponent, $(!Manufacturer), +
              $(Product$(ThisOption)Name), $(Product$(ThisOption)Name), +
              $(Product$(ThisOption)DisplayName), +
              $(!RasInfName), $(Product$(ThisOption)ImagePath),+
              "service", "Network", {"LanmanServer","RasMan"}, "", +
              $(!RasMsgDll), $(!RasEventTypeSupported)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "DoRasSvr: AddSoftware bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
```

```
            Debug-Output "DoRasSvr: Registry Error "$(RegistryErrorIndex)
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            CloseRegKey $($R3)
            CloseRegKey $($R4)
            CloseRegKey $($R5)
            goto fatalregistry
        endif
        Set SoftProductKey    = $($R1)
        Set SoftNetRuleKey    = $($R2)
        set SoftServiceKey    = $($R3)
        Set SoftParamsKey     = $($R4)
        Set SoftLinkageKey    = $($R5)
        set NewValueList = +
            {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Name)},+
             {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)SvcType)},+
             {Title,$(NoTitle),$(!REG_VT_SZ),$(Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Description)},+
             {PathName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)ImagePath)},+
             {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},
+
             {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},
+
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
             {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "DoRasSvr:product: AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftProductKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            Debug-Output "DoRasSvr: Registry Error "$(RegistryErrorIndex)
            CloseRegKey $(SoftServiceKey)
            CloseRegKey $(SoftNetRuleKey)
            CloseRegKey $(SoftLinkageKey)
            CloseRegKey $(SoftParamsKey)
            goto fatalregistry
        endif
        set NewValueList = +
            {{class, $(NoTitle), $(!REG_VT_SZ), $(NetRule$(ThisOption)Class)},+
             {type,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Type)},+
             {use,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Use)}, +
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
             {bindform,$(NoTitle),$(!REG_VT_SZ),+
              $(NetRule$(ThisOption)BindForm)}, +
             {bindable,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(NetRule$(ThisOption)Bindable)}, +
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "DoRasSvr:netrules: AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftNetRuleKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftServiceKey)
```

```
            CloseRegKey $(SoftLinkageKey)
            CloseRegKey $(SoftParamsKey)
            Debug-Output "DoRasSvr: Registry Error "$(RegistryErrorIndex)
            goto fatalregistry
        endif
        set RasDependencies = {"LanmanServer","RasMan"}
        OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\NetBios" $(!
MAXIMUM_ALLOWED) KeyNetBios
        ifstr $(KeyNetBios) != $(KeyNull)
            set RasDependencies = >($(RasDependencies),"NetBios")
        endif
        Debug-Output "DoRasSvr: Adding RasDependencies "$(RasDependencies)
        set NewValueList = +
            {{OtherDependencies, $(NoTitle), $(!REG_VT_MULTI_SZ),+
              $(RasDependencies)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftLinkageKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "DoRasSvr:Linkage: AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftLinkageKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftServiceKey)
            Debug-Output "DoRasSvr: Registry Error "$(RegistryErrorIndex)
            goto fatalregistry
        endif
        set NewValueList = +
            {{AuthenticateRetries, $(NoTitle), $(!REG_VT_DWORD), 2},+
            {AuthenticateTime, $(NoTitle), $(!REG_VT_DWORD), 120},+
            {Autodisconnect, $(NoTitle), $(!REG_VT_DWORD), 20},+
            {EnableAudit, $(NoTitle), $(!REG_VT_DWORD), 1},+
            {CallbackTime, $(NoTitle), $(!REG_VT_DWORD), 2}}
        Shell  $(!UtilityInf), AddValueList, $(SoftParamsKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "AddValueList bombed out"
             goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftServiceKey)
            goto fatalregistry
        endif
        OpenRegKey $(SoftParamsKey) "" "NetBiosGateway" $(!MAXIMUM_ALLOWED)
KeyNetbiosGtwy
        ifstr $(KeyNtbiosGtwy) == $(KeyNull)
            CreateRegKey $(SoftParamsKey) {"NetBiosGateway",$
(NoTitle),GenericClass} "" $(!MAXIMUM_ALLOWED) "" KeyNetbiosGtwy
            OpenRegKey $(SoftParamsKey) "" "NetBiosGateway" $(!MAXIMUM_ALLOWED)
KeyNetbiosGtwy
        endif
        set NewValueList = +
            {{EnableBroadcast, $(NoTitle), $(!REG_VT_DWORD), 0},+
            {EnableNetbiosSessionsAuditing, $(NoTitle), $(!REG_VT_DWORD), 0},+
            {MaxDynMem, $(NoTitle), $(!REG_VT_DWORD), 655350},+
            {MaxNames, $(NoTitle), $(!REG_VT_DWORD), 255},+
            {MaxSessions, $(NoTitle), $(!REG_VT_DWORD), 255},+
            {MulticastForwardRate, $(NoTitle), $(!REG_VT_DWORD), 5},+
            {SizWorkbuf, $(NoTitle), $(!REG_VT_DWORD), 4500},+
            {RemoteListen, $(NoTitle), $(!REG_VT_DWORD), 1},+
            {NameUpdateTime, $(NoTitle), $(!REG_VT_DWORD), 120},+
            {MaxDgBufferedPerGroupName, $(NoTitle), $(!REG_VT_DWORD), 10},+
            {RcvDgSubmittedPerGroupName, $(NoTitle), $(!REG_VT_DWORD), 3},+
            {DisableMcastFwdWhenSessionTraffic, $(NoTitle), $(!REG_VT_DWORD),
```

```
1},+
            {MaxBcastDgBuffered, $(NoTitle), $(!REG_VT_DWORD), 32},+
            {NumRecvQueryIndications, $(NoTitle), $(!REG_VT_DWORD), 3}}
        Shell  $(!UtilityInf), AddValueList, $(KeyNetbiosGtwy), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "AddValueList bombed out"
             goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftServiceKey)
            goto fatalregistry
        endif
        CloseRegKey $(KeyNetbiosGtwy)
        CloseRegKey $(SoftParamsKey)
        Shell "" UpdatePerfmonInfoHelper $(SoftServiceKey)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "error shelling UpdatePerfmonInfoHelper."
            goto ShellCodeError
        endif
        Ifstr(i) $($R0) != STATUS_SUCCESSFUL
            Debug-Output "error returned by UpdatePerfmonInfoHelper."
        endif
        CloseRegKey $(SoftServiceKey)
    endif
InstallRasMan = +
    ifstr(i) $(DoRasMan) == TRUE
        set ThisOption = RASMAN
        set RasSpecificString = $(Product$(ThisOption)Name)
        Shell $(!UtilityInf), AddSoftwareComponent, $(!Manufacturer), +
              $(Product$(ThisOption)Name), $(Product$(ThisOption)Name), +
              $(Product$(ThisOption)DisplayName), +
              $(!RasInfName), $(Product$(ThisOption)ImagePath), "serviceshare",+
              "Network", {}, "", $(!RasMsgDll), $(!RasEventTypeSupported)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddSoftware bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            CloseRegKey $($R3)
            CloseRegKey $($R4)
            CloseRegKey $($R5)
            goto fatalregistry
        endif
        Set SoftProductKey    = $($R1)
        Set SoftNetRuleKey    = $($R2)
        CloseRegKey $($R3)
        Set SoftParamsKey     = $($R4)
        CloseRegKey $($R5)
        set NewValueList = +
            {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Name)},+
             {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)SvcType)},+
             {Title,$(NoTitle),$(!REG_VT_SZ),$(Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Description)},+
             {PathName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)ImagePath)},+
             {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},
```

```
+
             {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},
+
             {Review, $(NoTitle), $(!REG_VT_DWORD), $(fReviewBindings)}, +
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
             {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftProductKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftNetRuleKey)
            CloseRegKey $(SoftParamsKey)
            goto fatalregistry
        endif
        set NewValueList = +
            {{class, $(NoTitle), $(!REG_VT_SZ), $(NetRule$(ThisOption)Class)},+
             {type,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Type)},+
             {use,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Use)}, +
             {bindform,$(NoTitle),$(!REG_VT_SZ),+
              $(NetRule$(ThisOption)BindForm)}, +
             {bindable,$(NoTitle),$(!REG_VT_SZ),+
              $(NetRule$(ThisOption)Bindable)}, +
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftNetRuleKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            goto fatalregistry
        endif
        set NewValueList = {{Logging, $(NoTitle), $(!REG_VT_DWORD), 0}}
        Shell  $(!UtilityInf), AddValueList, $(SoftParamsKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "AddValueList bombed out"
             goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            goto fatalregistry
        endif
        CloseRegKey $(SoftParamsKey)
        Shell "" AddServiceDependency "RasMan" "tapisrv"
    endif
    ifstr(i) $(DoRasAutodial) == TRUE
        set ThisOption = RASAUTODIAL
        set RasSpecificString = $(!Product$(ThisOption)Name)
        Shell "" InstallSoftwareAndService $(ThisOption)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling InstallSoftwareAndService for "$
(ThisOption)
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error from InstallSoftwareAndService for"$
(ThisOption)
             goto end
```

```
        endif
        set DisableAutoDial = FALSE
        ifint $(NewNumDialout) == 0
           set DisableAutoDial = TRUE
        endif
        ifstr(i) $(DisableAutoDial) == TRUE
           OpenRegKey $(!REG_H_LOCAL) "" $(!RasAutodialKeyName) $(!
MAXIMUM_ALLOWED) KeyAutodial
           ifstr $(KeyAutodial) == $(KeyNull)
             Debug-Output "OEMNSVRA.INF: could not open RasAuto key"
           else
             SetRegValue $(KeyAutodial) {Start, $(NoTitle), $(!REG_VT_DWORD), 4}
             CloseRegKey $(KeyAutodial)
           endif
        endif
        Shell "" InstallRasAcdService
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling InstallRasAcdService"
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error from InstallRasAcdService"
             goto end
        endif
        Shell "" AddServiceDependency "RasAuto" "RasMan"
    endif
    ifstr(i) $(DoNdisWan) == TRUE
        set ThisOption = NDISWAN
        set RasSpecificString = $(Product$(ThisOption)Name)
        Shell $(!UtilityInf), AddSoftwareComponent, $(!Manufacturer), +
              $(!Product$(ThisOption)Name), $(!Product$(ThisOption)Name), +
              $(!Product$(ThisOption)DisplayName), +
              $(!RasInfName), $(!Product$(ThisOption)ImagePath),+
              "kernelautostart", "NDISWAN", {}, "", +
              $(!RasMsgDll), $(!RasEventTypeSupported)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddSoftware bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            CloseRegKey $($R3)
            CloseRegKey $($R4)
            CloseRegKey $($R5)
            goto fatalregistry
        endif
        Set SoftProductKey    = $($R1)
        Set SoftNetRuleKey    = $($R2)
        CloseRegKey $($R3)
        CloseRegKey $($R4)
        CloseRegKey $($R5)
        set NewValueList = +
            {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),+
              $(!Product$(ThisOption)Name)},+
             {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Type)},+
             {Title,$(NoTitle),$(!REG_VT_SZ),$(!Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(!Product$(ThisOption)Description)},+
             {PathName,$(NoTitle),$(!REG_VT_SZ),+
              $(!Product$(ThisOption)ImagePath)},+
```

```
             {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},
+
             {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},
+
             {BindControl,$(NoTitle),$(!REG_VT_DWORD),$(HideBindings)},+
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
             {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftProductKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftNetRuleKey)
            goto fatalregistry
        endif
        set NewValueList = +
        {{class, $(NoTitle), $(!REG_VT_MULTI_SZ), $(!NetRule$
(ThisOption)Class)},+
             {type,$(NoTitle),$(!REG_VT_SZ),$(!NetRule$(ThisOption)Type)},+
             {use,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Use)}, +
             {bindform,$(NoTitle),$(!REG_VT_SZ),+
              $(!NetRule$(ThisOption)BindForm)}, +
             {bindable,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(!NetRule$(ThisOption)Bindable)}, +
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftNetRuleKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            goto fatalregistry
        endif
        Shell "" InstallNdisWanBHAdapter
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
           Debug-Output "Cannot add NdisWan Blood hound adapter"
           goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             set RegistryErrorIndex = $($R0)
             Debug-Output "Error installing NdisWan Blood hound adapter"
             goto fatalregistry
        endif
        Shell "" AddNDISWANToServiceGroupOrder
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
           Debug-Output "Cannot add NDISWAN to ServiceGroupOrder"
           goto ShellCodeError
        endif
DoNdisWanEnd = +
    endif
    ifstr(i) $(DoRasAsyMac) == TRUE
        set ThisOption = RASASYMAC
        set RasSpecificString = $(Product$(ThisOption)Name)
        ifstr(i) $(fSerialInstalled) == TRUE
            set AsySoftwareType = "kernelautostart"
        else-ifstr(i) $(fOtherInstalled) == TRUE
            set AsySoftwareType = "kernelautostart"
        else-ifstr(i) $(fUnimodemInstalled) == TRUE
```

```
            set AsySoftwareType = "kernelautostart"
        else
            set AsySoftwareType = "kerneldisable"
        endif
        Shell $(!UtilityInf), AddSoftwareComponent, $(!Manufacturer), +
              $(Product$(ThisOption)Name), $(Product$(ThisOption)Name), +
              $(Product$(ThisOption)DisplayName), +
              $(!RasInfName), $(Product$(ThisOption)ImagePath), $
(AsySoftwareType),+
              "NDIS", {}, "", $(!RasMsgDll), $(!RasEventTypeSupported)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddSoftware bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            CloseRegKey $($R3)
            CloseRegKey $($R4)
            CloseRegKey $($R5)
            goto fatalregistry
        endif
        Set SoftProductKey    = $($R1)
        Set SoftNetRuleKey    = $($R2)
        CloseRegKey $($R3)
        CloseRegKey $($R4)
        CloseRegKey $($R5)
        set NewValueList = +
            {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Name)},+
             {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
          $(Product$(ThisOption)Type)},+
            {Title,$(NoTitle),$(!REG_VT_SZ),$(Product$(ThisOption)Title)},+
            {Description,$(NoTitle),$(!REG_VT_SZ),+
             $(Product$(ThisOption)Description)},+
            {PathName,$(NoTitle),$(!REG_VT_SZ),+
             $(Product$(ThisOption)ImagePath)},+
            {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},
+
            {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},
+
            {BindControl,$(NoTitle),$(!REG_VT_DWORD),$(HideBindings)},+
            {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
            {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftProductKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            CloseRegKey $(SoftNetRuleKey)
            goto fatalregistry
        endif
        set NewValueList = +
        {{class, $(NoTitle), $(!REG_VT_MULTI_SZ), $(NetRule$(ThisOption)Class)},
+
            {type,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Type)},+
            {use,$(NoTitle),$(!REG_VT_SZ),$(NetRule$(ThisOption)Use)}, +
            {bindform,$(NoTitle),$(!REG_VT_SZ),+
             $(NetRule$(ThisOption)BindForm)}, +
```

```
             {bindable,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(!NetRule$(ThisOption)Bindable)}, +
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "AddValueList bombed out"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $(SoftNetRuleKey)
        Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
            goto fatalregistry
        endif
        Shell $(!UtilityInf), AddHardwareComponent, +
              $(Product$(ThisOption)Name),$(!RasInfName),+
              $(Product$(ThisOption)KeyName)
        ifint $($R4) != -1
            Set !NETCARD_LIST = >($(!NETCARD_LIST), +
                                  {$(Product$(ThisOption)Name),+
                                  $(!NetworkCardKeyName)"\"$($R4)})
        endif
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
           Debug-Output "Cannot add hardware component"
           goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "Registry error: add hardware component"
            CloseRegKey $($R1)
            CloseRegKey $($R2)
            CloseRegKey $($R3)
            goto fatalregistry
        endif
        set KeyNetcard    = $($R1)
        set KeyParameters = $($R3)
        set KeyAdapterRules = $($R2)
        set AdapterNumber = $($R4)
        set NewValueList = +
            {{Manufacturer,$(NoTitle),$(!REG_VT_SZ),$(!Manufacturer)},+
             {Title,$(NoTitle),$(!REG_VT_SZ),+
              "["$($R4)"] "$(Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Description)},+
             {ProductName,$(NoTitle),$(!REG_VT_SZ),+
              $(Product$(ThisOption)Name)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),$($R5)},+
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
             {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell  $(!UtilityInf), AddValueList, $(KeyNetcard), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error"
            goto ShellCodeError
        endif
        CloseRegKey $(KeyNetcard)
        set TempProdName = """$(Product$(ThisOption)Name)$(AdapterNumber)"""
        set TempBindForm = $(TempProdName)$(NetRuleHardware$
(ThisOption)BindForm)
        set NewValueList = +
            {{type,$(NoTitle),$(!REG_VT_SZ),+
              $(NetRuleHardware$(ThisOption)Type)},+
             {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)}, +
             {class,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(NetRuleHardware$(ThisOption)Class)}, +
```

```
            {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
            {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(KeyAdapterRules), $(NewValueList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error."
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "Resgitry error: add value list."
            CloseRegKey $(KeyParameters)
            CloseRegKey $(KeyAdapterRules)
            goto fatalregistry
        endif
        CloseRegKey $(KeyAdapterRules)
        CloseRegKey $(KeyParameters)
    endif
    Shell "" InstallNdisTapiService
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling InstallNdisTapiService "
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         set RegistryErrorIndex = $($R0)
         Debug-Output "Error installing NdisTapi Service"
         goto fatalregistry
    endif
    goto WriteParameters
WriteParameters = +
    ifstr(i) $(DoAdminOnly) == TRUE
        goto WriteParametersOver
    endif
    StartWait
    read-syms StatusUpdatingRegistry$(!STF_LANGUAGE)
    shell $(subroutninf) PushBillBoard NETSTATUSDLG $(WritingRasParams)
    Set BillboardVisible = 1
    Debug-Output "Writing Registry Parameters."
    ifstr(i) $(DoServer) != TRUE
        goto WriteRasMan
    endif
    Debug-Output "OEMNSVRA.INF: Product type "$(!STF_PRODUCT)
    Debug-Output "OEMNSVRA.INF: Total dialin ports "$(NewNumDialin)
    set RasStartValue = 3
    ifstr(i) $(!STF_PRODUCT) != "WINNT"
        ifint $(NewNumDialin) != 0
            set RasStartValue = 2
        endif
    endif
    set KeySvr = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasSvrKeyName) $(!MAXIMUM_ALLOWED) KeySvr
    ifstr $(KeySvr) == $(KeyNull)
        Debug-Output "OEMNSVRA.INF: could not open RemoteAccess key"
    else
        ifstr(i) $(!NTN_InstallMode) != install
            GetRegValue $(KeySvr) "Start" StartList
            ifint $(RegLastError) == 0
                set RasStartValue = *($(StartList), 4)
            endif
        endif
        ifint $(NewNumDialin) == 0
            ifint $(RasStartValue) != 4
                set RasStartValue = 3
            endif
        endif
```

```
        Debug-Output "OEMNSVRA.INF: Setting RemoteAccess start type to "$
(RasStartValue)
        SetRegValue $(KeySvr) {Start,$(NoTitle),$(!REG_VT_DWORD),$
(RasStartValue)}
        CloseRegKey $(KeySvr)
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(RasSvrParamKeyName) $(!MAXIMUM_ALLOWED)
KeySrvParams
    ifstr $(KeySrvParams) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF: could not open RasSvr Params key"
       set RegistryErrorIndex = UNABLE_OPEN_SERVICE_PARAMETERS
       goto fatalregistry
    endif
    set NewValueList = +
        {{NetBiosGatewayEnabled, $(NoTitle), $(!REG_VT_DWORD), $
(NetBiosGtwyEnabled)}}
    Shell  $(!UtilityInf), AddValueList, $(KeySrvParams), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "AddValueList bombed out"
         goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto fatalregistry
    endif
    CloseRegKey $(KeySrvParams)
WriteRasMan = +
    OpenRegKey $(!REG_H_LOCAL) "" $(RasManParamKeyName) $(!MAXIMUM_ALLOWED)
KeyRasManParams
    ifstr $(KeyRasManParams) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF: could not open RasMan Params key"
       set RegistryErrorIndex = UNABLE_OPEN_SERVICE_PARAMETERS
       goto fatalregistry
    endif
    set MediaValue = {"rastapi"}
    ifstr(i) $(fSerialInstalled) == TRUE
        set MediaValue = >($(MediaValue), "rasser")
    endif
    set OtherConfigKey = $(!NTN_SoftwareBase)"\"$(!Manufacturer)"\"$
(ProductRASName)"\OTHER DEVICES\CONFIGURED"
    set KeyOtherMedia = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(OtherConfigKey) $(!MAXIMUM_ALLOWED)  +
               KeyOtherMedia
    ifstr $(KeyOtherMedia) != $(KeyNull)
        EnumRegKey $(KeyOtherMedia) OtherList
        Debug-Output "oemnsvra.inf: Other Media list "$(OtherList)
        ForListDo $(OtherList)
            set MediaName = *($($),1)
            set KeyMedia = $(KeyNull)
            OpenRegKey $(KeyOtherMedia) "" $(MediaName) $(!MAXIMUM_ALLOWED)  +
                       KeyMedia
            ifstr $(KeyMedia) != $(KeyNull)
                GetRegValue $(KeyMedia), "MediaType" MediaType
                Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
                    set MediaTypeValue = *($(MediaType),4)
                    set MediaValue = >($(MediaValue), $(MediaTypeValue))
                endif
                CloseRegKey $(KeyMedia)
            endif
        EndForListDo
        CloseRegKey $(KeyOtherMedia)
    else
        Debug-Output "oemnsvra.inf: Other Media not configured"
    endif
```

```
    Debug-Output "Adding Medias as "$(MediaValue)
    set NewValueList = {{Medias,$(NoTitle),$(!REG_VT_MULTI_SZ),$(MediaValue)}}
    Shell  $(!UtilityInf), AddValueList, $(KeyRasManParams), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "AddValueList bombed out"
         goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto fatalregistry
    endif
    CloseRegKey $(KeyRasManParams)
WriteNdisWan = +
    set PrevNumPorts = 0
    set PrevNumDialin = 0
    set PrevNumDialout = 0
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacParamKeyName) $(!MAXIMUM_ALLOWED)
KeyMacParams
    ifstr $(KeyMacParams) == $(KeyNull)
       ifstr(i) $(!NTN_InstallMode) == configure
          Debug-Output "OEMNSVRA.INF: could not open AsyncMac Params key"
          set RegistryErrorIndex = UNABLE_OPEN_SERVICE_PARAMETERS
          goto fatalregistry
       endif
       goto WriteNdisWan1
    endif
    GetRegValue $(KeyMacParams), "Ports" PortsInfo
    GetRegValue $(KeyMacParams), "DialinNBF" DialinNBFInfo
    GetRegValue $(KeyMacParams), "DialoutNBF" DialoutNBFInfo
    GetRegValue $(KeyMacParams), "DialinIP" DialinIPInfo
    GetRegValue $(KeyMacParams), "DialoutIP" DialoutIPInfo
    GetRegValue $(KeyMacParams), "DialinoutIPX" DialinoutIPXInfo
    set PrevNumPorts         = *($(PortsInfo), 4)
    set PrevNumDialinNBF     = *($(DialinNBFInfo), 4)
    set PrevNumDialoutNBF    = *($(DialoutNBFInfo), 4)
    set PrevNumDialinIP      = *($(DialinIPInfo), 4)
    set PrevNumDialoutIP     = *($(DialoutIPInfo), 4)
    set PrevNumDialinoutIPX  = *($(DialinoutIPXInfo), 4)
    Debug-Output "Ports currently configured "$(PrevNumPorts)
    Debug-Output "Ports currently dialin NBF "$(PrevNumDialinNBF)
    Debug-Output "Ports currently dialout NBF "$(PrevNumDialoutNBF)
    Debug-Output "Ports currently dialin IP "$(PrevNumDialinIP)
    Debug-Output "Ports currently diaout IP "$(PrevNumDialoutIP)
    Debug-Output "Ports currently diainout IPX "$(PrevNumDialinoutIPX)
    CloseRegKey $(KeyMacParams)
WriteNdisWan1 =+
    set ThisOption = NDISWAN
    set RasSpecificString = $(!Product$(ThisOption)Name)
    OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanParamKeyName) $(!MAXIMUM_ALLOWED)
KeyHubParams
    ifstr $(KeyHubParams) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF: could not open NdisWan Params key"
       set RegistryErrorIndex = UNABLE_OPEN_SERVICE_PARAMETERS
       goto fatalregistry
    endif
    set NewValueList = +
        {{EndPoints, $(NoTitle), $(!REG_VT_MULTI_SZ), $(NewNumPortsList)}}
    Shell  $(!UtilityInf), AddValueList, $(KeyHubParams), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "AddValueList bombed out"
         goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(KeyHubParams)
```

```
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto fatalregistry
    endif
    ifint $(PrevNumDialinNBF) == $(NewNumDialinNBF)
      ifint $(PrevNumDialoutNBF) == $(NewNumDialoutNBF)
        ifint $(PrevNumDialinIP) == $(NewNumDialinIP)
          ifint $(PrevNumDialoutIP) == $(NewNumDialoutIP)
            ifint $(PrevNumDialinoutIPX) == $(NewNumDialinoutIPX)
              set CommonStatus = STATUS_USERCANCEL
              goto WriteRasAsyMac
            endif
          endif
        endif
      endif
    endif
    set NumAddDialinNBF      = 0
    set NumAddDialoutNBF     = 0
    set NumRemoveDialinNBF   = 0
    set NumRemoveDialoutNBF  = 0
    set NumAddDialinIP       = 0
    set NumAddDialoutIP      = 0
    set NumRemoveDialinIP    = 0
    set NumRemoveDialoutIP   = 0
    ifint $(PrevNumDialinNBF) > $(NewNumDialinNBF)
        set-sub NumRemoveDialinNBF = $(PrevNumDialinNBF), $(NewNumDialinNBF)
    else
        set-sub NumAddDialinNBF = $(NewNumDialinNBF), $(PrevNumDialinNBF)
    endif
    ifint $(PrevNumDialoutNBF) > $(NewNumDialoutNBF)
        set-sub NumRemoveDialoutNBF = $(PrevNumDialoutNBF), $(NewNumDialoutNBF)
    else
        set-sub NumAddDialoutNBF = $(NewNumDialoutNBF), $(PrevNumDialoutNBF)
    endif
    ifint $(PrevNumDialinIP) > $(NewNumDialinIP)
        set-sub NumRemoveDialinIP = $(PrevNumDialinIP), $(NewNumDialinIP)
    else
        set-sub NumAddDialinIP = $(NewNumDialinIP), $(PrevNumDialinIP)
    endif
    ifint $(PrevNumDialoutIP) > $(NewNumDialoutIP)
        set-sub NumRemoveDialoutIP = $(PrevNumDialoutIP), $(NewNumDialoutIP)
    else
        set-sub NumAddDialoutIP = $(NewNumDialoutIP), $(PrevNumDialoutIP)
    endif
    ifint $(PrevNumDialinoutIPX) > $(NewNumDialinoutIPX)
        set-sub NumRemoveDialinoutIPX = $(PrevNumDialinoutIPX), $
(NewNumDialinoutIPX)
    else
        set-sub NumAddDialinoutIPX = $(NewNumDialinoutIPX), $
(PrevNumDialinoutIPX)
    endif
    set NumAddTotal = 0
    set NumRemoveTotal = 0
    set-add NumAddTotal = $(NumAddDialinNBF), $(NumAddDialoutNBF)
    set-add NumAddTotal = $(NumAddTotal), $(NumAddDialinIP)
    set-add NumAddTotal = $(NumAddTotal), $(NumAddDialoutIP)
    set-add NumAddTotal = $(NumAddTotal), $(NumAddDialinoutIPX)
    set-add NumRemoveTotal = $(NumRemoveDialinNBF), $(NumRemoveDialoutNBF)
    set-add NumRemoveTotal = $(NumRemoveTotal), $(NumRemoveDialinIP)
    set-add NumRemoveTotal = $(NumRemoveTotal), $(NumRemoveDialoutIP)
    set-add NumRemoveTotal = $(NumRemoveTotal), $(NumRemoveDialinoutIPX)
    Debug-Output "NumAddDialinNBF = "$(NumAddDialinNBF)
    Debug-Output "NumAddDialoutNBF = "$(NumAddDialoutNBF)
    Debug-Output "NumAddDialinIP = "$(NumAddDialinIP)
    Debug-Output "NumAddDialoutIP = "$(NumAddDialoutIP)
```

```
    Debug-Output "NumAddDialinoutIPX = "$(NumAddDialinoutIPX)
    Debug-Output "NumAddTotal = "$(NumAddTotal)
    Debug-Output "NumRemoveDialinNBF = "$(NumRemoveDialinNBF)
    Debug-Output "NumRemoveDialinIP = "$(NumRemoveDialinIP)
    Debug-Output "NumRemoveDialinoutIPX = "$(NumRemoveDialinoutIPX)
    Debug-Output "NumRemoveTotal = "$(NumRemoveTotal)
    Shell "utility.inf" BaseServiceKey
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) == NO_ERROR
        Debug-Output "OEMNSVRA.INF: Opened the services base key"
        set ServicesBaseKey = $($R1)
    else
        set ServicesBaseKey = $(KeyNull)
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(MAXIMUM_ALLOWED)
KeyNetcards
    Ifstr(i) $(RegistryErrorIndex) == NO_ERROR
        Debug-Output "OEMNSVRA.INF: Opened the networkcardkey "
    endif
    set RasAdapterNumber = 1
AddNdisWanNetCard = +
     IfInt $(NumAddTotal) == 0
        goto RemoveNdisWanNetCard
     else
          Debug-Output "In the Add if loop. NumAddTotal = "$(NumAddTotal)
          shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                      $(WritingRasParamsAdd)" "$(NumAddTotal)
          set-sub NumAddTotal = $(NumAddTotal),1
          Shell $(!UtilityInf), AddHardwareComponent, +
               $(!Product$(ThisOption)Name),$(!RasInfName),+
               $(!Product$(ThisOption)KeyName), +
               $(ServicesBaseKey), $(KeyNetcards), $(RasAdapterNumber)
          ifint $($R4) != -1
             Set !NETCARD_LIST = >($(!NETCARD_LIST), +
                                  {$(!Product$(ThisOption)Name),+
                                  $(!NetworkCardKeyName)"\"$($R4)})
             set RasAdapterNumber = $($R4)
          endif
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Cannot add hardware component"
             goto ShellCodeError
          endif
          set RegistryErrorIndex = $($R0)
          Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
              Debug-Output "Registry error: add hardware component"
              CloseRegKey $($R1)
              CloseRegKey $($R2)
              CloseRegKey $($R3)
              goto fatalregistry
          endif
        set KeyNetcard    = $($R1)
        set KeyParameters = $($R3)
        set KeyAdapterRules = $($R2)
        set AdapterNumber = $($R4)
        set NetcardInfoList = +
            {{Manufacturer,$(NoTitle),$(!REG_VT_SZ),$(!Manufacturer)},+
             {Title,$(NoTitle),$(!REG_VT_SZ),+
              "["$($R4)"] "$(!Product$(ThisOption)Title)},+
             {Description,$(NoTitle),$(!REG_VT_SZ),+
              $(!Product$(ThisOption)Description)},+
             {ServiceName,$(NoTitle),$(!REG_VT_SZ),$($R5)},+
             {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
             {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
        Shell $(!UtilityInf), GetBusTypeNum
```

```
set BusTypeNum = $($R1)
ifint $(NumAddDialinNBF) > 0
    set-sub NumAddDialinNBF = $(NumAddDialinNBF), 1
    set NetcardInfoList = >($(NetcardInfoList), +
                           {ProductName,$(NoTitle),$(!REG_VT_SZ),+
                            $(!Product$(ThisOption)DIALINName)})
    set ServiceParamList = +
                {{EnumExportPref,$(NoTitle),$(!REG_VT_DWORD),0},+
                {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    set AdapterType = $(!NetRuleHardwareDIALINType)
    set AdapterClass = $(!NetRuleHardwareDIALINClass)
    set AdapterBlock = $(!NetRuleHardwareDIALINBlock)
else-ifint $(NumAddDialoutNBF) > 0
    set-sub NumAddDialoutNBF = $(NumAddDialoutNBF), 1
    set NetcardInfoList = >($(NetcardInfoList), +
                          {ProductName,$(NoTitle),$(!REG_VT_SZ),+
                           $(!Product$(ThisOption)DIALOUTName)})
    set ServiceParamList = +
                 {{EnumExportPref,$(NoTitle),$(!REG_VT_DWORD),1},+
                 {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    set AdapterType = $(!NetRuleHardwareDIALOUTType)
    set AdapterClass = $(!NetRuleHardwareDIALOUTClass)
    set AdapterBlock = $(!NetRuleHardwareDIALOUTBlock)
else-ifint $(NumAddDialinIP) > 0
    set-sub NumAddDialinIP = $(NumAddDialinIP), 1
    set NetcardInfoList = >($(NetcardInfoList), +
                          {ProductName,$(NoTitle),$(!REG_VT_SZ),+
                           $(!Product$(ThisOption)DIALINIPName)})
    set ServiceParamList = +
                 {{EnumExportPref,$(NoTitle),$(!REG_VT_DWORD),0},+
                 {AutoIPAddress,$(NoTitle),$(!REG_VT_DWORD),1},+
                 {ServerAdapter,$(NoTitle),$(!REG_VT_DWORD),1},+
                 {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    set AdapterType = $(!NetRuleHardwareDIALINIPType)
    set AdapterClass = $(!NetRuleHardwareDIALINIPClass)
    set AdapterBlock = $(!NetRuleHardwareDIALINIPBlock)
else-ifint $(NumAddDialoutIP) > 0
    set-sub NumAddDialoutIP = $(NumAddDialoutIP), 1
    set NetcardInfoList = >($(NetcardInfoList), +
                          {ProductName,$(NoTitle),$(!REG_VT_SZ),+
                           $(!Product$(ThisOption)DIALOUTIPName)})
    set ServiceParamList = +
                {{EnumExportPref,$(NoTitle),$(!REG_VT_DWORD),1},+
                {AutoIPAddress,$(NoTitle),$(!REG_VT_DWORD),1},+
                {ServerAdapter,$(NoTitle),$(!REG_VT_DWORD),0},+
                {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    set AdapterType = $(!NetRuleHardwareDIALOUTIPType)
    set AdapterClass = $(!NetRuleHardwareDIALOUTIPClass)
    set AdapterBlock = $(!NetRuleHardwareDIALOUTIPBlock)
else-ifint $(NumAddDialinoutIPX) > 0
    set-sub NumAddDialinoutIPX = $(NumAddDialinoutIPX), 1
    set NetcardInfoList = >($(NetcardInfoList), +
                          {ProductName,$(NoTitle),$(!REG_VT_SZ),+
                           $(!Product$(ThisOption)DIALINOUTIPXName)})
    set ServiceParamList = +
                 {{EnumExportPref,$(NoTitle),$(!REG_VT_DWORD),0},+
                 {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    set AdapterType = $(!NetRuleHardwareDIALINOUTIPXType)
    set AdapterClass = $(!NetRuleHardwareDIALINOUTIPXClass)
    set AdapterBlock = $(!NetRuleHardwareDIALINOUTIPXBlock)
endif
Shell  $(!UtilityInf), AddValueList, $(KeyNetcard), $(NetcardInfoList)
ifint $($ShellCode) != $(!SHELL_CODE_OK)
    Debug-Output "ShellCode error"
```

```
            goto ShellCodeError
        endif
        CloseRegKey $(KeyNetcard)
        set TempProdName = """$(!Product$(ThisOption)Name)$(AdapterNumber)"""
        set TempBindForm = $(TempProdName)$(!NetRuleHardware$
(ThisOption)BindForm)
        set AdapterRulesList = +
            {{type,$(NoTitle),$(!REG_VT_SZ),+
              $(AdapterType)},+
             {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)},+
             {class,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(AdapterClass)},+
             {block,$(NoTitle),$(!REG_VT_MULTI_SZ),+
              $(AdapterBlock)},+
             {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)},+
             {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
        Shell  $(!UtilityInf), AddValueList, $(KeyAdapterRules), $
(AdapterRulesList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error."
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "Resgitry error: add value list."
            CloseRegKey $(KeyParameters)
            CloseRegKey $(KeyAdapterRules)
            goto fatalregistry
        endif
        CloseRegKey $(KeyAdapterRules)
        Shell  $(!UtilityInf), AddValueList, $(KeyParameters), $
(ServiceParamList)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error."
            goto ShellCodeError
        endif
        CloseRegKey $(KeyParameters)
      goto AddNdisWanNetCard
     endif
RemoveNdisWanNetCard = +
     ifstr(i) $(ServicesBaseKey) != $(KeyNull)
         CloseRegKey $(ServicesBaseKey)
     endif
     ifstr(i) $(KeyNetcards) != $(KeyNull)
         CloseRegKey $(KeyNetcards)
     endif
     Ifint $(NumRemoveTotal) == 0
       goto WriteRasAsyMac
     endif
     Debug-Output "Removing NdisWan netcard entry..."
        OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
KeyNetcards
        ifstr $(KeyNetcards) == $(KeyNull)
             Debug-Output "OEMNSVRA.INF: could not open Netcards key"
             set RegistryErrorIndex = UNABLE_OPEN_NETWORKCARD_SECTION
             goto fatalregistry
        endif
        EnumRegKey $(KeyNetcards) NetcardsList
        set RemoveDialinList  = {}
        set RemoveDialinIpList  = {}
        set RemoveDialoutList = {}
        set RemoveDialoutIpList = {}
        set RemoveDialinoutIpxList = {}
        ForListDo $(NetcardsList)
```

```
            set KeyName = *($($),1)
            OpenRegKey $(KeyNetcards) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
            ifstr $(Card) == $(KeyNull)
                Debug-Output "OEMNSVRA.INF: could not open netcard key"
                set RegistryErrorIndex = UNABLE_OPEN_NETWORKCARD_SECTION
                goto fatalregistry
            endif
            GetRegValue $(Card), "ProductName" ProductNameInfo
            set CardProductName = *($(ProductNameInfo), 4)
              ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINName)
               set RemoveDialinList = >($(RemoveDialinList), +
                              {$(!ProductNDISWANName),+
                                        $(!NetworkCardKeyName)"\"$(KeyName)})
              else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINIPName)
               set RemoveDialinIpList = >($(RemoveDialinIpList), +
                              {$(!ProductNDISWANName),+
                                         $(!NetworkCardKeyName)"\"$(KeyName)})
              else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTName)
               set RemoveDialoutList = >($(RemoveDialoutList), +
                               {$(!ProductNDISWANName),+
                                        $(!NetworkCardKeyName)"\"$(KeyName)})
              else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTIPName)
               set RemoveDialoutIpList = >($(RemoveDialoutIpList), +
                               {$(!ProductNDISWANName),+
                                         $(!NetworkCardKeyName)"\"$(KeyName)})
              else-ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALINOUTIPXName)
               set RemoveDialinoutIpxList = >($(RemoveDialinoutIpxList), +
                                 {$(!ProductNDISWANName),+
                                          $(!NetworkCardKeyName)"\"$(KeyName)})
            endif
            CloseRegKey $(Card)
        EndForListDo
        CloseRegKey $(KeyNetcards)
        QueryListSize IpNetCards $(RemoveDialinIpList)
        ifint $(IpNetCards) != 0
            ForListDo  $(RemoveDialinIpList)
                ifint $(NumRemoveDialinIP) == 0
                    goto RemoveDialinIPX
                endif
                set-sub NumRemoveDialinIP = $(NumRemoveDialinIP), 1
                shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                     $(WritingRasParamsRemove)" "$
(NumRemoveTotal)
                set-sub NumRemoveTotal = $(NumRemoveTotal), 1
                debug-output "Removing hardware component: "$($)
                Shell $(!UtilityInf), RemoveHardwareComponent, +
                                    $(!Manufacturer), *($($),1), *($($),2)
            EndForListDo
        endif
RemoveDialinIPX =+
        QueryListSize IpxNetCards $(RemoveDialinoutIpxList)
        ifint $(IpxNetCards) != 0
            ForListDo  $(RemoveDialinoutIpxList)
                ifint $(NumRemoveDialinoutIPX) == 0
                    goto RemoveDialinNBF
                endif
                set-sub NumRemoveDialinoutIPX = $(NumRemoveDialinoutIPX), 1
                debug-output "Removing hardware component: "$($)
                shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                     $(WritingRasParamsRemove)" "$
(NumRemoveTotal)
                set-sub NumRemoveTotal = $(NumRemoveTotal), 1
                Shell $(!UtilityInf), RemoveHardwareComponent, +
```

```
                                    $(!Manufacturer), *($($),1), *($($),2)
            EndForListDo
        endif
RemoveDialinNBF =+
        ForListDo  $(RemoveDialinList)
            ifint $(NumRemoveDialinNBF) == 0
                goto RemoveDialout
            endif
            set-sub NumRemoveDialinNBF = $(NumRemoveDialinNBF), 1
            shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                $(WritingRasParamsRemove)" "$(NumRemoveTotal)
            set-sub NumRemoveTotal = $(NumRemoveTotal), 1
            debug-output "Removing hardware component: "$($)
            Shell $(!UtilityInf), RemoveHardwareComponent, $(!Manufacturer),+
                  *($($),1), *($($),2)
        EndForListDo
RemoveDialout = +
        QueryListSize IpNetCards $(RemoveDialoutIpList)
        ifint $(IpNetCards) != 0
            ForListDo  $(RemoveDialoutIpList)
                ifint $(NumRemoveDialoutIP) == 0
                    goto RemoveDialoutNBF
                endif
                set-sub NumRemoveDialoutIP = $(NumRemoveDialoutIP), 1
                shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                    $(WritingRasParamsRemove)" "$
(NumRemoveTotal)
                set-sub NumRemoveTotal = $(NumRemoveTotal), 1
                debug-output "Removing hardware component: "$($)
                Shell $(!UtilityInf), RemoveHardwareComponent, +
                                    $(!Manufacturer), *($($),1), *($($),2)
            EndForListDo
        endif
RemoveDialoutNBF =+
        ForListDo  $(RemoveDialoutList)
            ifint $(NumRemoveDialoutNBF) == 0
            goto WriteRasAsyMac
            endif
            set-sub NumRemoveDialoutNBF = $(NumRemoveDialoutNBF), 1
            shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                                $(WritingRasParamsRemove)" "$(NumRemoveTotal)
            set-sub NumRemoveTotal = $(NumRemoveTotal), 1
            debug-output "Removing hardware component: "$($)
            Shell $(!UtilityInf), RemoveHardwareComponent, $(!Manufacturer),+
                  *($($),1), *($($),2)
        EndForListDo
WriteRasAsyMac = +
    ifstr(i) $(fSerialInstalled) == TRUE
        set AsyStartValue = 2
    else-ifstr(i) $(fOtherInstalled) == TRUE
        set AsyStartValue = 2
    else-ifstr(i) $(fUnimodemInstalled) == TRUE
        set AsyStartValue = 2
    else
        set AsyStartValue = 4
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacKeyName) $(!MAXIMUM_ALLOWED) KeyMac
    ifstr $(KeyMac) == $(KeyNull)
        Debug-Output "OEMNSVRA.INF: could not open AsyncMac key"
    else
        SetRegValue $(KeyMac) {Start,$(NoTitle),$(!REG_VT_DWORD),$
(AsyStartValue)}
        CloseRegKey $(KeyMac)
    endif
```

```
    ifint $(PrevNumPorts) == $(NewNumPorts)
      ifint $(PrevNumDialinNBF) == $(NewNumDialinNBF)
        ifint $(PrevNumDialoutNBF) == $(NewNumDialoutNBF)
          ifint $(PrevNumDialinIP) == $(NewNumDialinIP)
            ifint $(PrevNumDialoutIP) == $(NewNumDialoutIP)
              ifint $(PrevNumDialinoutIPX) == $(NewNumDialinoutIPX)
                  EndWait
                  Ifint $(BillboardVisible) != 0
                      Shell "subroutn.inf" PopBillboard
                      Set BillboardVisible = 0
                  Endif
                  goto WriteProtocolInfo
              endif
            endif
          endif
        endif
      endif
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacParamKeyName) $(!MAXIMUM_ALLOWED)
KeyMacParams
    ifstr $(KeyMacParams) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF: could not open AsyncMac Params key"
       set RegistryErrorIndex = UNABLE_OPEN_SERVICE_PARAMETERS
       goto fatalregistry
    endif
    set NewValueList = +
        {{Ports, $(NoTitle), $(!REG_VT_DWORD), $(NewNumPorts)},+
         {DialinNBF, $(NoTitle), $(!REG_VT_DWORD), $(NewNumDialinNBF)},+
         {DialoutNBF, $(NoTitle), $(!REG_VT_DWORD), $(NewNumDialoutNBF)},+
         {DialinIP, $(NoTitle), $(!REG_VT_DWORD), $(NewNumDialinIP)},+
         {DialoutIP, $(NoTitle), $(!REG_VT_DWORD), $(NewNumDialoutIP)},+
         {DialinoutIPX, $(NoTitle), $(!REG_VT_DWORD), $(NewNumDialinoutIPX)}}
    Shell  $(!UtilityInf), AddValueList, $(KeyMacParams), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "AddValueList bombed out"
         goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(KeyMacParams)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto fatalregistry
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
KeyNetcards
    ifstr $(KeyNetcards) == $(KeyNull)
         Debug-Output "OEMNSVRA.INF: could not open Netcards key"
         set RegistryErrorIndex = UNABLE_OPEN_NETWORKCARD_SECTION
         goto fatalregistry
    endif
    EnumRegKey $(KeyNetcards) NetcardsList
    ForListDo $(NetcardsList)
       set KeyName = *($($),1)
       OpenRegKey $(KeyNetcards) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
       ifstr $(Card) == $(KeyNull)
          Debug-Output "OEMNSVRA.INF: could not open netcard key"
          set RegistryErrorIndex = UNABLE_OPEN_NETWORKCARD_SECTION
          goto fatalregistry
       endif
       GetRegValue $(Card), "ProductName" ProductNameInfo
       set CardProductName = *($(ProductNameInfo), 4)
       ifstr(i) $(CardProductName) == $(ProductRASASYMACName)
           goto RasMacCardFound
       endif
       CloseRegKey $(Card)
```

```
    EndForListDo
    CloseRegKey $(KeyNetcards)
RasMacCardFound = +
    Debug-Output "Shelling to find AsyncMac Service."
    Shell $(!UtilityInf) FindService, $(Card)
    CloseRegKey $(Card)
    ifint $($ShellCode)  != $(!SHELL_CODE_OK)
        Debug-Output "OEMNSVRA.INF: shell Findservice failed"
        goto ShellCodeError
    endif
    ifstr(i) $($R0) != NO_ERROR
        Debug-Output "OEMNSVRA.INF: findservice failed."
        goto fatalregistry
    endif
    set KeyParameters = $($R2)
    CloseRegKey $($R1)
    Shell $(!UtilityInf), GetBusTypeNum
    set BusTypeNum = $($R1)
    set NewValueList = +
        {{Ports, $(NoTitle), $(!REG_VT_DWORD), $(NewNumPorts)}, +
         {BusType, $(NoTitle), $(!REG_VT_DWORD), $(BusTypeNum)}}
    Shell  $(!UtilityInf), AddValueList, $(KeyParameters), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "AddValueList bombed out"
         goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(KeyParameters)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto fatalregistry
    endif
WriteProtocolInfo =+
    Shell "" SaveSelectedProtocols $(fNetbeuiSelected) $(fTcpIpSelected) +
                                   $(fIpxSelected) $(fNetbeuiAllowed) +
                                   $(fTcpIpAllowed) $(fIpxAllowed)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling SaveSelectedProtocols"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error Saving selected protocols"
         goto end
    endif
    Shell "" WritePPPParameters
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling WritePPPParameters"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error from WritePPPParameters"
         goto end
    endif
    ifstr(i) $(fNetbeuiChosen) == TRUE
        ifstr(i) $(fNetbeuiInstalled) == FALSE
            Shell "" InstallProtocol "NETBEUI"
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 Debug-Output "Error shelling InstallProtocol NETBEUI"
                 goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                 Debug-Output "Error installing NETBEUI"
                 goto end
            endif
        endif
```

```
endif
ifstr(i) $(fTcpIpChosen) == TRUE
    ifstr(i) $(fTcpIpInstalled) == FALSE
        Shell "" InstallProtocol "TCPIP"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling InstallProtocol TCPIP"
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error installing TCPIP"
             goto end
        endif
    endif
    Shell "" InstallRasArpService
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling InstallRasArpService"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error from InstallRasArpService"
         goto end
    endif
    set CommonStatus = STATUS_SUCCESSFUL
    Shell "" UpdateIPRouterInfo $(EnableIpRouter)
    Shell "" SaveTcpipInfo $(fTcpIpAllowed)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling SaveTcpipInfo"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error saving Tcpip info"
         goto end
    endif
else
    Shell "" RemoveRasArpService
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling RemoveRasArpService"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error from RemoveRasArpService"
         goto end
    endif
    Shell "" SaveTcpipInfo $(fTcpIpAllowed)
    set CommonStatus = STATUS_SUCCESSFUL
endif
ifstr(i) $(fIpxChosen) == TRUE
    ifstr(i) $(fIpxInstalled) == FALSE
        Shell "" InstallProtocol "IPX"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling InstallProtocol IPX"
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error installing IPX"
             goto end
        endif
    endif
    OpenRegKey $(!REG_H_LOCAL) ""  +
               $(!NTN_ServiceBase)"\NWLNKIPX\Parameters"  +
               $(!MAXIMUM_ALLOWED) KeyIpxParameters
    ifstr $(KeyIpxParameters) != $(KeyNull)
       GetRegValue $(KeyIpxParameters), "SingleNetworkActive"  +
                   SingleNetworkActive
```

```
            ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                SetRegValue $(KeyIpxParameters) +
                            {SingleNetworkActive, 0, $(!REG_VT_DWORD), 1}
            endif
            GetRegValue $(KeyIpxParameters), "DisableDialoutSap"  +
                        DisableDialoutSap
            ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                SetRegValue $(KeyIpxParameters) +
                            {DisableDialoutSap, 0, $(!REG_VT_DWORD), 1}
            endif
            GetRegValue $(KeyIpxParameters), "DisableDialinNetbios"  +
                        DisableDialinNetbios
            ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                SetRegValue $(KeyIpxParameters) +
                            {DisableDialinNetbios, 0, $(!REG_VT_DWORD), 1}
            endif
            CloseRegKey $(KeyIpxParameters)
        Endif
        else
            Debug-Output "OEMNSVRA.INF: error opening NWLNKIPX\Parameters key"
        endif
        Shell "" SaveIpxInfo $(EnableIpxRouter) $(fIpxAllowed)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling SaveIpxInfo"
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error saving Ipx info"
             goto end
        endif
        ifint $(EnableIpxRouter) == 1
            Shell "" InstallNwlnkRipService
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 Debug-Output "Error shelling InstallNwlnkRipService"
                 goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                 Debug-Output "Error from InstallNwlnkRipService"
                 goto end
            endif
            set CommonStatus = STATUS_SUCCESSFUL
            Shell "" InstallIsnSapService
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 Debug-Output "Error shelling InstallIsnSapService"
                 goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                 Debug-Output "Error from InstallIsnSapService"
                 goto end
            endif
            set CommonStatus = STATUS_SUCCESSFUL
        else
            Shell "" RemoveNwlnkRipService
            Shell "" RemoveIsnSapService
        endif
        Shell "" UpdateIPXRouterInfo $(EnableWanRouter)
        Ifstr(i) $(!NTN_InstallPhase) != primary
            shell $(subroutninf) PopBillBoard
            Set BillboardVisible = 0
        Endif
    else
        Shell "" SaveIpxInfo $(EnableIpxRouter) $(fIpxAllowed)
    endif
WriteParametersOver = +
```

```
    EndWait
    goto successful
RemoveRas = +
    StartWait
    set REMOVE_SOFTWARE = {}
    shell "" QueryComponentsInstalled
    Ifstr(i) $($R0) == STATUS_SUCCESSFUL
        Set InstalledComps = $($R1)
        Set InstalledFlags = $($R2)
        Set DoServer      = *($(InstalledFlags),1)
        Set DoClient      = *($(InstalledFlags),2)
        Set DoAdmin       = *($(InstalledFlags),3)
        Set DoServerOnly  = *($(InstalledFlags),4)
        Set DoClientOnly  = *($(InstalledFlags),5)
        Set DoAdminOnly   = *($(InstalledFlags),6)
    Endif
    Debug-Output "Installed List is "$(InstalledComps)
    Debug-Output "Installed Flags is "$(InstalledFlags)
    ifstr(i) $(DoAdminOnly) == TRUE
         set REMOVE_SOFTWARE = {$(ProductRASName)}
         goto RemoveSoftware
    endif
    set REMOVE_SOFTWARE = >($(REMOVE_SOFTWARE), $(ProductRASName))
    ifstr(i) $(DoServer) == TRUE
         set REMOVE_SOFTWARE = >($(REMOVE_SOFTWARE), $(ProductRASSVRName))
    endif
    set REMOVE_SOFTWARE = >($(REMOVE_SOFTWARE), $(ProductRASMANName))
    set REMOVE_SOFTWARE = >($(REMOVE_SOFTWARE), $(!ProductRASAUTODIALName))
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
KeyNetcards
    ifstr $(KeyNetcards) == $(KeyNull)
         Debug-Output "OEMNSVRA.INF: could not open Netcards key"
         goto RemoveSoftware
    endif
    EnumRegKey $(KeyNetcards) NetcardsList
    ForListDo $(NetcardsList)
       set KeyName = *($($),1)
       OpenRegKey $(KeyNetcards) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
       ifstr $(Card) == $(KeyNull)
           Debug-Output "OEMNSVRA.INF: could not open netcard key"
           goto RemoveSoftware
       endif
       GetRegValue $(Card), "ProductName" ProductNameInfo
       set CardProductName = *($(ProductNameInfo), 4)
       ifstr(i) $(CardProductName) == $(!ProductNDISWANName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                        {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
       else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                        {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
       else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                        {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
       else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINIPName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                        {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
       else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTIPName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                        {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
```

```
       else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINOUTIPXName)
          set !NETCARD_LIST = >($(!NETCARD_LIST), +
                       {$(!ProductNDISWANName),+
                                  $(!NetworkCardKeyName)"\"$(KeyName)})
       else-ifstr(i) $(CardProductName) == $(ProductRASASYMACName)
          set !NETCARD_LIST = >($(!NETCARD_LIST),+
                                 {$(ProductRASASYMACName),+
                                  $(!NetworkCardKeyName)"\"$(KeyName)})
       endif
       CloseRegKey $(Card)
    EndForListDo
    CloseRegKey $(KeyNetcards)
    read-syms StatusUpdatingRegistry$(!STF_LANGUAGE)
    QueryListSize NumRemove $(!NETCARD_LIST)
    ForListDo  $(!NETCARD_LIST)
        shell $(subroutninf) PushBillBoard NETSTATUSDLG +
                              $(RemovingAdapters)" "$(NumRemove)
        Set BillboardVisible = 1
        debug-output "Removing hardware component: "$($)
        Shell $(!UtilityInf), RemoveHardwareComponent, $(!Manufacturer),+
              *($($),1), *($($),2)
        set-sub NumRemove = $(NumRemove), 1
    EndForListDo
    ifint $(BillboardVisible) != 0
        Shell "subroutn.inf" PopBillboard
        Set BillboardVisible = 0
    Endif
RemoveSoftware = +
    Shell "" RemoveRasArpService
    Shell "" RemoveNdisTapiService
    Shell "" RemoveRasAcdService
    set RasTapiDevicesKey = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasTapiDevicesKeyName) $(!MAXIMUM_ALLOWED)
RasTapiDevicesKey
    ifstr $(RasTapiDevicesKey) != $(KeyNull)
       set TapiProviderList = {}
       EnumRegKey $(RasTapiDevicesKey) TapiProviderList
       ForListDo $(TapiProviderList)
           set ProviderName = *($($),1)
           Debug-Output "OEMNSVRA.INF: Removing "$(ProviderName)" dependency on
NdisTapi."
           Shell "" RemoveServiceDependency $(ProviderName) "NdisTapi"
       EndForListDo
    EndIf
    set fIpxAllowed = FALSE
    Shell "" QuerySelectedProtocols
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
        set fIpxAllowed = $($R6)
    else
        Debug-Output "RemoveRas: error QuerySelectedProtocols"
    endif
    ifstr(i) $(fIpxAllowed) == TRUE
        Shell "" RemoveNwlnkRipService
        Shell "" RemoveIsnSapService
    endif
    Debug-Output "Remove Software List "$(REMOVE_SOFTWARE)
    ForListDo  $(REMOVE_SOFTWARE)
        Debug-Output "Removing software component: "$($)
        ifstr(i) $($) == RAS
             OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyBase) +
                        $(!MAXIMUM_ALLOWED) ProductKey
             Ifstr $(ProductKey) == $(KeyNull)
                Debug-Output "OEMNSVRA.INF: could not open Software product key"
                 goto RemoveFiles
```

```
            endif
            DeleteRegTree $(ProductKey) $(Product$($)Name)
            CloseRegKey $(ProductKey)
        else
            Shell $(!UtilityInf), RemoveSoftwareComponent, $(!Manufacturer), $
($)
        endif
    EndForListDo
RemoveFiles = +
    Install RemoveRasFiles
    ifstr(i) $(DoServerOnly) == FALSE
        Install RemoveRasGroup
    endif
    Shell "" RemoveInfFromReviewPrograms
    shell "" RemoveNetGroupDependency
RemoveRasOver = +
    Debug-Output "OEMNSVRA.INF: Doing a forcible cleanup..."
    OpenRegKey $(!REG_H_LOCAL) ""  $(!NTN_SoftwareBase)"\Microsoft" +
               $(!MAXIMUM_ALLOWED) KeySoftware
    ifstr $(KeySoftware) != $(KeyNull)
        set SoftList = {"ASYNCMAC", "NDISWAN", "RAS", "RASMAN", +
                        "RASAUTO", "REMOTEACCESS"}
        ForListDo $(SoftList)
            DeleteRegTree $(KeySoftware) $($)
        EndForListDo
        set NetworkCardKey = $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) +
                   $(!MAXIMUM_ALLOWED) NetworkCardKey
        Ifstr(i) $(NetworkCardKey) != $(KeyNull)
            set NetcardsList = {}
            EnumRegKey $(NetworkCardKey) NetcardsList
            Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
              ForListDo $(NetcardsList)
               set KeyName = *($($),1)
               set Card = $(KeyNull)
               OpenRegKey $(NetworkCardKey) "" $(KeyName) $(!MAXIMUM_ALLOWED)
Card
               ifstr $(Card) == $(KeyNull)
                   Debug-Output "RemoveRas: could not open netcard key "$
(KeyName)
               else
                   GetRegValue $(Card), "ProductName" ProductNameInfo
                   Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                      Debug-Output "RemoveRas: ProductName not found."
                   else
                      set CardProductName = *($(ProductNameInfo), 4)
                      Debug-Output "RemoveRas: ProductName. "$(CardProductName)
                      ifstr(i) $(CardProductName) == $(!ProductNDISWANName)
                         DeleteRegTree $(NetworkCardKey) $(KeyName)
                      endif
                        ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALINName)
                           DeleteRegTree $(NetworkCardKey) $(KeyName)
                      endif
                          ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALOUTName)
                           DeleteRegTree $(NetworkCardKey) $(KeyName)
                      endif
                          ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALINIPName)
                           DeleteRegTree $(NetworkCardKey) $(KeyName)
                      endif
                          ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALOUTIPName)
```

```
                        DeleteRegTree $(NetworkCardKey) $(KeyName)
                    endif
                        ifstr(i) $(CardProductName) == $(!
ProductNDISWANDIALINOUTIPXName)
                        DeleteRegTree $(NetworkCardKey) $(KeyName)
                    endif
                    ifstr(i) $(CardProductName) == $(ProductRASASYMACName)
                        DeleteRegTree $(NetworkCardKey) $(KeyName)
                    endif
                endif
              EndForListDo
            else
              Debug-Output "RemoveRas: EnumRegKey failed."
            endif
        else
            Debug-Output "RemoveRas: failed to open "$(!NetworkCardKeyName)
        endif
        CloseRegKey $(KeySoftware)
    else
        Debug-Output "RemoveRas: failed to open software key"
    endif
    OpenRegKey $(!REG_H_LOCAL) ""  $(!NTN_ServiceBase) +
               $(!MAXIMUM_ALLOWED) KeyServices
    ifstr $(KeyServices) != $(KeyNull)
        set ServiceList = {"ASYNCMAC", "NDISWAN", "RASARP", "RASMAN", +
                           "REMOTEACCESS", "RASACD", "RASAUTO"}
        ForListDo $(ServiceList)
            DeleteRegTree $(KeyServices) $($)
        EndForListDo
        EnumRegKey $(KeyServices) ServiceList
        Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
           ForListDo $(ServiceList)
               set ServiceName = *($($),1)
               set Result = 1
            LibraryProcedure Result $(!LIBHANDLE) SetupStrncmp +
                                $(ServiceName) "NdisWan" 7
               ifint $(Result) == 0
                  Debug-Output "RemoveRas: Removing "$(ServiceName)
                    DeleteRegTree $(KeyServices) $(ServiceName)
               else
                     LibraryProcedure Result $(!LIBHANDLE) SetupStrncmp +
                                $(ServiceName) "AsyncMac" 8
                   ifint $(Result) == 0
                      Debug-Output "RemoveRas: Removing "$(ServiceName)
                        DeleteRegTree $(KeyServices) $(ServiceName)
                   endif
               endif
           EndForListDo
        endif
        CloseRegKey $(KeyServices)
    endif
    EndWait
    goto $(to)
BindingsRas = +
    Debug-Output "OEMNSVRA.INF:Review bindings is being called!!!."
    Shell "" UpdateSelectedProtocols
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "error shelling UpdateSelectedProtocols."
        goto ShellCodeError
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by UpdateSelectedProtocols."
    endif
    set fNetbeuiChosen = $($R1)
```

```
    set fTcpIpChosen   = $($R2)
    set fIpxChosen     = $($R3)
    ifstr(i) $(fNetbeuiChosen) == FALSE
        ifstr(i) $(fTcpIpChosen) == FALSE
            ifstr(i) $(fIpxChosen) == FALSE
                read-syms NoProtocolsDlg$(!STF_LANGUAGE)
                Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "STATUS", $
(NoProtocolsWarning)
                ifint $($ShellCode) != $(!SHELL_CODE_OK)
                    goto ShellCodeError
                endif
                goto end
            endif
        endif
    endif
    set WkstaConfigured = FALSE
    set SrvrConfigured  = FALSE
    shell "" IsNetworkConfigured
    Debug-Output "IsNetworkConfigured returned R0 "$($R0)
    Debug-Output "IsNetworkConfigured returned R1 "$($R1)
    Debug-Output "IsNetworkConfigured returned R2 "$($R2)
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
        set WkstaConfigured  = $($R1)
        set SrvrConfigured   = $($R2)
    else
        read-syms NetworkConfigError$(!STF_LANGUAGE)
        set Text = $(Text1)
        shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "FATAL" $(Text)
        goto end
    endif
    ifstr(i) $(WkstaConfigured) == TRUE
        ifstr(i) $(SrvrConfigured) == TRUE
            goto ResetProgramList
        else
            goto InstallNetworkError
        endif
    else
        goto InstallNetworkError
    endif
InstallNetworkError =+
    set CommonStatus = STATUS_USERCANCEL
    read-syms NetworkConfigError$(!STF_LANGUAGE)
    set Text = $(Text2)
    shell $(subroutninf) SetupMessage $(!STF_LANGUAGE) "STATUS" $(Text)
ResetProgramList =+
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasManKeyName) $(!MAXIMUM_ALLOWED)
KeyRasMan
    ifstr $(KeyRasMan) != $(KeyNull)
        GetRegValue $(KeyRasMan), "Review" ReviewInfo
        set ReviewValue = *($(ReviewInfo), 4)
        ifint $(ReviewValue) == 1
            Debug-Output "BindingsRas: resetting Review to 0"
            SetRegValue $(KeyRasMan) {Review,$(NoTitle),$(!REG_VT_DWORD), 0}
            Shell "" AddInfToReviewProgramsList
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "error shelling AddInfToReviewProgramsList."
                goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "error returned by AddInfToReviewProgramsList."
            endif
        endif
        CloseRegKey $(KeyRasMan)
    else
```

```
        Debug-Output "OEMNSVRA.INF: error opening SOFTWARE\RasMan\CurrentVersion
key."
    endif
    shell "" QueryComponentsInstalled
    Ifstr(i) $($R0) == STATUS_SUCCESSFUL
        Set InstalledComps = $($R1)
        Set InstalledFlags = $($R2)
        Set DoServer      = *($(InstalledFlags),1)
        Set DoClient      = *($(InstalledFlags),2)
        Set DoAdmin       = *($(InstalledFlags),3)
        Set DoServerOnly  = *($(InstalledFlags),4)
        Set DoClientOnly  = *($(InstalledFlags),5)
        Set DoAdminOnly   = *($(InstalledFlags),6)
    Endif
    Debug-Output "Installed List is "$(InstalledComps)
    Debug-Output "Installed Flags is "$(InstalledFlags)
    set AsyStartValue = 4
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacKeyName) $(!MAXIMUM_ALLOWED) KeyMac
    ifstr $(KeyMac) == $(KeyNull)
        Debug-Output "OEMNSVRA.INF: could not open AsyncMac key"
    else
        GetRegValue $(KeyMac), "Start" AsyStartInfo
        ifint $(RegLastError) == 0
            set AsyStartValue = *($(AsyStartInfo), 4)
        endif
        Debug-Output "OEMNSVRA.INF: AsyncMac start value "$(AsyStartValue)
        CloseRegKey $(KeyMac)
    endif
    set NdisWanParam = {}
    set NdisWanEndPointsLst = {}
    ifint $(AsyStartValue) != 4
        set RasMacLinkageKey = $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasMacLinkageKeyName) $(!
MAXIMUM_ALLOWED) RasMacLinkageKey
        ifstr $(RasMacLinkageKey) == $(KeyNull)
            Debug-Output "OEMNSVRA.INF: could not open AsyncMac linkage key"
            set RegistryErrorIndex = UNABLE_ACCESS_CONFIGURE_SERVICE
            goto fatalregistry
        endif
        GetRegValue $(RasMacLinkageKey), "Bind" BindInfo
      set NdisWanParam = *($(BindInfo), 4)
        CloseRegKey $(RasMacLinkageKey)
      set NdisWanParamKey = $(KeyNull)
      OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanParamKeyName) $(!MAXIMUM_ALLOWED)
NdisWanParamKey
      ifstr $(NdisWanParamKey) == $(KeyNull)
          Debug-Output "OEMNSVRA.INF: could not open NdisWanParamKey"
        else
          GetRegValue $(NdisWanParamKey), "EndPoints" EndPointsInfo
            set EndPointsLst = *($(EndPointsInfo), 4)
          set NdisWanEndPointsLst = >($(NdisWanEndPointsLst),*($(EndPointsLst),
1))
          CloseRegKey $(NdisWanParamKey)
        endif
    endif
    set ProviderList = {}
    set RasTapiDevicesKey = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasTapiDevicesKeyName) $(!MAXIMUM_ALLOWED)
RasTapiDevicesKey
    ifstr $(RasTapiDevicesKey) != $(KeyNull)
       set TapiProviderList = {}
       EnumRegKey $(RasTapiDevicesKey) TapiProviderList
       ForListDo $(TapiProviderList)
           set TapiAddress = {}
```

```
            set NumEndPoints = 0
            set ProviderName = *($($),1)
            Debug-Output "OEMNSVRA.INF: TAPI provider name = "$(ProviderName)
            OpenRegKey $(RasTapiDevicesKey) "" $(ProviderName) $(!
MAXIMUM_ALLOWED) Provider
            ifstr $(Provider) != $(KeyNull)
                GetRegValue $(Provider), "Address" TapiAddress
                set AdapterNum = ""
                set EndPointCount = 0
                set tmpNdisWanParam = {}
                set tmpNdisWanEndPointsLst = {}
                set DeviceName = ""
                ForListDo *($(TapiAddress),4)
                    Split-String $($) "- " Address
                    set NewAdapterNum = *($(Address), 1)
                    ifstr(i) $(AdapterNum) != $(NewAdapterNum)
                      ifstr(i) $(AdapterNum) != ""
                         set tmpNdisWanParam = >($(tmpNdisWanParam), $
(DeviceName))
                         set tmpNdisWanEndPointsLst = >($
(tmpNdisWanEndPointsLst), $(EndPointCount))
                         set EndPointCount = 0
                      endif
                      set AdapterNum = $(NewAdapterNum)
                      set DeviceName = "\Device\"$(ProviderName)$(AdapterNum)
                    endif
                    set-add EndPointCount = $(EndPointCount), 1
                EndForListDo
                ifstr $(DeviceName) != ""
                    set tmpNdisWanParam = >($(tmpNdisWanParam), $(DeviceName))
                    set tmpNdisWanEndPointsLst = >($(tmpNdisWanEndPointsLst), $
(EndPointCount))
                endif
                CloseRegKey $(Provider)
            else
               Debug-Output "OEMNSVRA.INF: could not open RAS\TAPI DEVICES\"$
(DriverName)" key."
            endif
            set ProviderKeyName = $(!NTN_ServiceBase)"\"$(ProviderName)
            set ProviderLinkageKeyName = $(!NTN_ServiceBase)"\"$(ProviderName)"\
Linkage"
            Debug-Output "TAPI provider key name "$(ProviderKeyName)
            set ProviderKey = $(KeyNull)
            set ProviderLinkageKey = $(KeyNull)
            OpenRegKey $(!REG_H_LOCAL) "" $(ProviderKeyName) $(!MAXIMUM_ALLOWED)
ProviderKey
            ifstr $(ProviderKey) == $(KeyNull)
              Debug-Output "OEMNSVRA.INF: could not open key "$(ProviderKeyName)
            else
              set DeleteFlag = 0
              set DeleteFlagInfo = {}
              GetRegValue $(ProviderKey),"DeleteFlag", DeleteFlagInfo
              set DeleteFlag = *($(DeleteFlagInfo), 4)
              ifint $(DeleteFlag) != 1
                set ProviderList = >($(ProviderList), $(ProviderName))
                Shell "" AddServiceDependency $(ProviderName) "NdisTapi"
                ForListDo $(tmpNdisWanParam)
                  set NdisWanParam = >($(NdisWanParam), $($))
                  set NdisWanEndPointsLst = >($(NdisWanEndPointsLst), *($
(tmpNdisWanEndPointsLst), $(#)))
                EndForListDo
              endif
              CloseRegKey $(ProviderKey)
            endif
```

```
       EndForListDo
       CloseRegKey $(RasTapiDevicesKey)
    else
        Debug-Output "OEMNSVRA.INF: could not open RAS\TAPI DEVICES key"
    endif
    set NumBindings = 0
    Debug-Output "NdisWan param Bind value = "$(NdisWanParam)
    QueryListSize NumBindings $(NdisWanParam)
    Debug-Output "Number of Ndiswan bindings = "$(NumBindings)
    ifint $(NumBindings) == 0
        Debug-Output "OEMNSVRA.INF: No ports are configured"
        read-syms NoPortsConfigured$(!STF_LANGUAGE)
        Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "STATUS", $
(NoPortsError)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            goto ShellCodeError
        endif
        goto end
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanParamKeyName) $(!MAXIMUM_ALLOWED)
NdisWanParamKey
    ifstr $(NdisWanParamKey) == $(KeyNull)
      Debug-Output "OEMNSVRA.INF: could not open NdisWanParamKey"
        set RegistryErrorIndex = UNABLE_ACCESS_CONFIGURE_SERVICE
        goto fatalregistry
    endif
    Debug-Output "Setting NdisWan param Bind to "$(NdisWanParam)
    set NewValueList = {{Bind, $(NoTitle), $(!REG_VT_MULTI_SZ), $
(NdisWanParam)}}
    Shell $(!UtilityInf), AddValueList, $(NdisWanParamKey), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      CloseRegKey $(NdisWanParamKey)
        goto ShellCodeError
    endif
    Debug-Output "Setting NdisWan param EndPoints to "$(NdisWanEndPointsLst)
    set NewValueList = {{EndPoints, $(NoTitle), $(!REG_VT_MULTI_SZ), $
(NdisWanEndPointsLst)}}
    Shell $(!UtilityInf), AddValueList, $(NdisWanParamKey), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      CloseRegKey $(NdisWanParamKey)
        goto ShellCodeError
    endif
    CloseRegKey $(NdisWanParamKey)
    LoadLibrary "x" $(!STF_CWDDIR)rascfg.dll PORTSDLGHANDLE
    LibraryProcedure Result, $(PORTSDLGHANDLE), InitRasmanSecurityDescriptor
    Debug-Output "Result of setting Rasman security descriptor "$(Result)
    LibraryProcedure Result, $(PORTSDLGHANDLE), InitRemoteSecurityDescriptor
    Debug-Output "Result of setting Remote security descriptor "$(Result)
    Shell "" UpdateCPList
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "error shelling UpdateCPList."
        goto ShellCodeError
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by UpdateCPList."
    endif
    ifstr(i) $(fTcpIpChosen) == TRUE
        shell "" UpdateLLInterface
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "error shelling UpdateLLInterface."
            goto ShellCodeError
        endif
        Ifstr(i) $($R0) != STATUS_SUCCESSFUL
            Debug-Output "error returned by UpdateLLInterface."
```

```
        endif
    endif
    Shell "" SetRasArpBindValueFromTcpIP
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "error shelling SetRasArpBindValueFromTcpIP."
        goto ShellCodeError
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by SetRasArpBindValueFromTcpIP."
    endif
    Debug-Output "Review bindings done."
    goto end
UpgradeRas = +
    Debug-Output "OEMNSVRA.INF: Upgrade with Option type "$(Option)
    ifstr(i) $(Option) != "RAS"
        Debug-Output "OEMNSVRA.INF: not upgrading due to nonras option"
        set CommonStatus = STATUS_NOEFFECT
        goto end
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(!MAXIMUM_ALLOWED)
KeyProduct
    Ifstr $(KeyProduct) != $(KeyNull)
        Shell $(!UtilityInf), GetInfFileNameFromRegistry, $(KeyProduct)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error"
            goto ShellCodeError
        endif
        set !UG_Filename = $($R0)
        ifstr(i) $(!UG_Filename) != ""
            Debug-Output "File Name is "$(!UG_Filename)
            StartWait
            read-syms UpgradeErrors$(!STF_LANGUAGE)
            read-syms StatusUpdatingRegistry$(!STF_LANGUAGE)
            Shell "subroutn.inf" PushBillboard NETSTATUSDLG $(UpdatingRas)
            Set BillboardVisible = 1
            shell "" QueryComponentsInstalled
            Ifstr(i) $($R0) == STATUS_SUCCESSFUL
                Set InstalledComps = $($R1)
                Set InstalledFlags = $($R2)
                Set DoServer      = *($(InstalledFlags),1)
                Set DoClient      = *($(InstalledFlags),2)
                Set DoAdmin       = *($(InstalledFlags),3)
                Set DoServerOnly  = *($(InstalledFlags),4)
                Set DoClientOnly  = *($(InstalledFlags),5)
                Set DoAdminOnly   = *($(InstalledFlags),6)
            Endif
            Debug-Output "Installed List is "$(InstalledComps)
            Debug-Output "Installed Flags is "$(InstalledFlags)
            Set ServerInstalled = $(DoServer)
            Set ClientInstalled = $(DoClient)
            Set AdminInstalled  = $(DoAdmin)
            Install InstallResources
            ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
                goto filecopycancel
            endif
            Install InstallRasFiles
            ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
                goto filecopycancel
            endif
            set fTcpIp = FALSE
            Shell "" QuerySelectedProtocols
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "error returned by QuerySelectedProtocols."
            endif
```

```
            set fNetbeuiSelected = $($R1)
            set fTcpIpSelected   = $($R2)
            set fIpxSelected     = $($R3)
            set fNetbeuiAllowed  = $($R4)
            set fTcpIpAllowed    = $($R5)
            set fIpxAllowed      = $($R6)
            ifstr(i) $(fTcpIpSelected) == TRUE
              set fTcpIp = "TRUE"
            else-ifstr(i) $(fTcpIpAllowed) == TRUE
              set fTcpIp = "TRUE"
            endif
            ifstr(i) $(fTcpIp) == TRUE
                Shell "" RemoveServiceDependency "TCPIP" "RASARP"
                Shell "" RemoveRasArpService
                Shell "" InstallRasArpService
            endif
            Shell "" SetRestoreConnectionTo1
            Shell "" RemoveServiceDependency "RemoteAccess" "NetLogon"
            Shell "" AddServiceDependency "RasMan" "tapisrv"
            Shell "" InstallSoftwareAndService RASAUTODIAL
            Shell "" AddServiceDependency "RasAuto" "RasMan"
            Shell "" RemoveRasAcdService
            Shell "" InstallRasAcdService
            OpenRegKey $(!REG_H_LOCAL) "" $(!RasManSvcKeyName) $(!
MAXIMUM_ALLOWED) KeyRasMan
            ifstr $(KeyRasMan) == $(KeyNull)
              Debug-Output "OEMNSVRA.INF: could not open RasMan key"
            else
              SetRegValue $(KeyRasMan) {Type, $(NoTitle), $(!REG_VT_DWORD), 32}
              CloseRegKey $(KeyRasMan)
            endif
            LoadLibrary "x" $(!STF_CWDDIR)rascfg.dll PORTSDLGHANDLE
            LibraryProcedure Result, $(PORTSDLGHANDLE), RenameRasHubToNdisWan
            Shell "" RenameRasHubToNdisWan
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling
RenameRasHubToNdisWan."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by
RenameRasHubToNdisWan."
                goto end
            endif
            Shell "" IsNdisWanBHAdapterInstalled
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
               Debug-Output "OEMNSVRA.INF:error shelling
IsNdisWanBHAdapterInstalled"
              goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Shell "" InstallNdisWanBHAdapter
                ifint $($ShellCode) != $(!SHELL_CODE_OK)
                   Debug-Output "OEMNSVRA.INF:error shelling
InstallNdisWanBHAdapter"
                   goto ShellCodeError
                endif
                ifstr(i) $($R0) != STATUS_SUCCESSFUL
                     set RegistryErrorIndex = $($R0)
                     Debug-Output "Error installing NdisWan Blood hound adapter"
                     goto fatalregistry
                endif
            endif
            Shell "" InstallNdisTapiService
```

```
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 Debug-Output "Error shelling InstallNdisTapiService "
                 goto ShellCodeError
            endif
            ifstr(i) $($R0) != STATUS_SUCCESSFUL
                 set RegistryErrorIndex = $($R0)
                 Debug-Output "Error installing NdisTapi Service"
                 goto fatalregistry
            endif
            Shell "" UpdateAsyncMacNetRules
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling
UpdateAsyncMacNetRules."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by
UpdateAsyncMacNetRules."
                goto end
            endif
            Shell "" UpdateNdisWanInfo
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling UpdateNdisWanInfo."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by UpdateNdisWanInfo."
                goto end
            endif
            Shell "" AddNDISWANToServiceGroupOrder
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
               Debug-Output "Cannot add NDISWAN to ServiceGroupOrder"
               goto ShellCodeError
            endif
            Shell "" UpdateAsyncMacParameters
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling
UpdateAsyncMacParameters."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by
UpdateAsyncMacParameters."
                goto end
            endif
            Shell "" UpdateAsyncMacStartType
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling
UpdateAsyncMacStartType."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by
UpdateAsyncMacStartType."
                goto end
            endif
            Shell "" UpgradeSelectedProtocols
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling
UpgradeSelectedProtocols."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by
```

```
UpgradeSelectedProtocols."
                goto end
            endif
            ifstr(i) $(!STF_STANDARDSERVERUPGRADE) == "YES"
                OpenRegKey $(!REG_H_LOCAL) "" $(RasProtocolsKeyName) +
                                               $(!MAXIMUM_ALLOWED) KeyProtocols
                ifstr $(KeyProtocols) != $(KeyNull)
                    set fDialin = FALSE
                    set TmpList = {}
                    GetRegValue $(KeyProtocols) "fNetbeuiAllowed" TmpList
                    ifint $(RegLastError) == 0
                        ifstr *($(TmpList), 4) == 1
                            set fDialin = TRUE
                        endif
                    endif
                    set TmpList = {}
                    ifstr(i) $(fDialin) == FALSE
                        GetRegValue $(KeyProtocols) "fTcpIpAllowed" TmpList
                        ifint $(RegLastError) == 0
                            ifstr *($(TmpList), 4) == 1
                                set fDialin = TRUE
                            endif
                        endif
                    endif
                    set TmpList = {}
                    ifstr(i) $(fDialin) == FALSE
                        GetRegValue $(KeyProtocols) "fIpxAllowed" TmpList
                        ifint $(RegLastError) == 0
                            ifstr *($(TmpList), 4) == 1
                                set fDialin = TRUE
                            endif
                        endif
                    endif
                    Debug-Output "OEMNSVRA.INF: Upgrade => Are dialin ports
configured? "$(fDialin)
                    ifstr(i) $(fDialin) == TRUE
                        OpenRegKey $(!REG_H_LOCAL) "" $(RasSvrKeyName) +
                                                      $(!MAXIMUM_ALLOWED) KeySvr
                        ifstr $(KeySvr) != $(KeyNull)
                            set  RasStartValue = 2
                            GetRegValue $(KeySvr) "Start" StartList
                            ifint $(RegLastError) == 0
                               set RasStartValue = *($(StartList), 4)
                            endif
                            ifint $(RasStartValue) != 4
                                Debug-Output "OEMNSVRA.INF: Changing
RemoteAccess Start value to 2"
                                SetRegValue $(KeySvr) {Start,$(NoTitle),$(!
REG_VT_DWORD), 2}
                            endif
                            CloseRegKey $(KeySvr)
                        else
                            Debug-Output "OEMNSVRA.INF: error opening
RemoteAccess service key"
                        endif
                    endif
                    CloseRegKey $(KeyProtocols)
                else
                    Debug-Output "OEMNSVRA.INF: could not open RAS\Protocols
key"
                endif
                OpenRegKey $(!REG_H_LOCAL) "" $(!NdisTapiKeyName)"\Parameters" $
(!MAXIMUM_ALLOWED) ParamKey
                Ifstr(i) $(ParamKey) != $(KeyNull)
```

```
                    SetRegValue $(ParamKey) {AsyncEventQueueSize, 0, $(!
REG_VT_DWORD), 3072}
                    CloseRegKey $(ParamKey)
                EndIf
            endif
            Shell "" UpgradeIpxInfo $(PORTSDLGHANDLE)
            Shell "" UpgradeIsdnInfo
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling UpgradeIsdnInfo."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by UpgradeIsdnInfo."
                set Error = $(UpgradeIsdnInfoError)
                goto fatal
            endif
            Shell "" UpdatePerfmonInfo
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling UpdatePerfmonInfo."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by UpdatePerfmonInfo."
                goto end
            endif
            Shell "" UpdateCPList
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "OEMNSVRA.INF:error shelling UpdateCPList."
                goto ShellCodeError
            endif
            Ifstr(i) $($R0) != STATUS_SUCCESSFUL
                Debug-Output "OEMNSVRA.INF:error returned by UpdateCPList."
                goto end
            endif
            ifstr(i) $(!STF_PRODUCT) != "WINNT"
                OpenRegKey $(!REG_H_LOCAL) "" $(!NdisTapiKeyName)"\Parameters" $
(!MAXIMUM_ALLOWED) ParamKey
                Ifstr(i) $(ParamKey) != $(KeyNull)
                    SetRegValue $(ParamKey) {AsyncEventQueueSize, 0, $(!
REG_VT_DWORD), 3072}
                    CloseRegKey $(ParamKey)
                EndIf
            EndIf
            Install RemoveRasGroup
            EndWait
            Ifint $(BillboardVisible) != 0
                Shell "subroutn.inf" PopBillboard
                Set BillboardVisible = 0
            Endif
        endif
        Shell "" UpdateSoftwareType
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF:error shelling UpdateSoftwareType."
            goto ShellCodeError
        endif
        Ifstr(i) $($R0) != STATUS_SUCCESSFUL
            Debug-Output "OEMNSVRA.INF:error returned by UpdateSoftwareType."
            goto end
        endif
        SetRegValue $(KeyProduct) {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!
ProductMajorVersion)}
        SetRegValue $(KeyProduct) {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!
ProductMinorVersion)}
        SetRegValue $(KeyProduct) {Description,$(NoTitle),$(!REG_VT_SZ),$
```

```
(ProductRASDescription)}
        SetRegValue $(KeyProduct) {OperationsSupport,$(NoTitle),$(!
REG_VT_DWORD),$(ProductOpSupport)}
        CloseRegKey $(KeyProduct)
    else
        Set RegistryErrorIndex = $($R0)
        goto fatalregistry
    endif
    goto end
successful = +
    Ifstr(i) $(!NTN_InstallPhase) == primary
        goto installstep1
    else-ifstr(i) $(!NTN_InstallMode) == configure
        goto installstep1
    endif
    ifint $(NewNumDialin) != 0
       ifstr(i) $(!STF_GUI_UNATTENDED) != YES
            read-syms SuccessfulInstall$(!STF_LANGUAGE)
            shell "subroutn.inf" SetupMessage $(!STF_LANGUAGE) "STATUS" $
(Success)
       endif
    endif
installstep1 = +
    goto end
warning = +
    Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "WARNING", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
    goto ShellCodeError
    endif
    ifstr(i) $($R1) == "OK"
       goto $(to)
    else-ifstr(i) $($R1) == "CANCEL"
       goto $(from)
    else
       Debug-Msg "Error Error Bad DLGEVENT"
       goto "end"
    endif
nonfatal = +
    Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "NONFATAL", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        goto ShellCodeError
    endif
    ifstr(i) $($R1) == "OK"
        goto $(from)
    else
        goto "end"
    endif
fatalregistry = +
    Shell $(!UtilityInf) RegistryErrorString $(RegistryErrorIndex)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
       Debug-Output "fatalregistry: shell to find RegistryErrorString failed."
       goto ShellCodeError
    endif
    ifstr(i) $(RasSpecificString) != ""
        set Error = $($R0)" - service "$(RasSpecificString)"."
    else
        set Error = $($R0)
    endif
    read-syms AbortMessage$(!STF_LANGUAGE)
    set Error = $(Error)$(!LF)$(!LF)$(AbortText)
    goto fatal
fatal =    +
    Shell $(subroutninf) SetupMessage, $(!STF_LANGUAGE), "FATAL", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
```

```
        Debug-Output "fatal: shell to SetupMessage failed."
    goto ShellCodeError
    endif
    goto setfailed
ShellCodeError = +
    read-syms ShellCodeErrorMsg$(!STF_LANGUAGE)
    ui start "Error Message"
    goto setfailed
setfailed = +
    set CommonStatus = STATUS_FAILED
    ifstr(i) $(fRemoveRas) == TRUE
        set fRemoveRas = FALSE
        set from = setfailed
        set to   = end
        goto RemoveRas
    endif
    goto end
filecopycancel =+
    set CommonStatus = STATUS_USERCANCEL
    ifstr(i) $(fRemoveRas) == TRUE
        set fRemoveRas = FALSE
        set from = setfailed
        set to   = end
        goto RemoveRas
    endif
end = +
    ifstr(i) $(PORTSDLGHANDLE) != $(HandleNull)
        Debug-Output "Unloading RASCFG.DLL"
        FreeLibrary $(PORTSDLGHANDLE)
    endif
    Debug-Output "ending at last!!"
    goto term
term = +
    Debug-Output "OEMNSVRA.INF:term: CommonStatus "$(CommonStatus)
    Return $(CommonStatus)
[BindingsReview]
    set Option   = $($1)
    set SrcDir   = $($2)
    set AddCopy  = $($3)
    set DoCopy   = $($4)
    set DoConfig = $($5)
    set Language = $(!STF_LANGUAGE)
    set SaveInstallMode = $(!NTN_InstallMode)
    set !NTN_InstallMode = bind
    Shell "" InstallOption $(Language) $(Option) $(SrcDir) $(AddCopy) $(DoCopy)
$(DoConfig)
    set !NTN_InstallMode = $(SaveInstallMode)
    set Status = $($R0)
    Return $(Status)
[SetRestoreConnectionTo1]
   Debug-Output "SetRestoreConnectionTo1 entry.."
   set KeyNull = ""
   set RestoreKeyName = "SYSTEM\CurrentControlSet\Control\networkprovider"
   OpenRegKey $(!REG_H_LOCAL) "" $(RestoreKeyName) $(!MAXIMUM_ALLOWED)
KeyRestore
   Ifstr(i) $(KeyRestore) != $(KeyNull)
       SetRegValue $(KeyRestore) {RestoreConnection, 0, $(!REG_VT_DWORD), 1}
       CloseRegKey $(KeyRestore)
   else
       Debug-Output "SetRestoreConnectionTo1 error opening key. "$
(RestoreKeyName)
   endif
   Debug-Output "SetRestoreConnectionTo1 exit."
   return
```

```
[InstallSoftwareAndService]
   set Status = STATUS_SUCCESSFUL
   set ThisOption = $($0)
   Debug-Output "InstallSoftwareAndService for "$(ThisOption)
   Shell $(!UtilityInf), AddSoftwareComponent, $(!Manufacturer), +
         $(!Product$(ThisOption)Name), $(!Product$(ThisOption)Name), +
         $(!Product$(ThisOption)DisplayName), +
         $(!RasInfName), $(!Product$(ThisOption)ImagePath), "autoserviceshare",+
         "", {}, "", $(!RasMsgDll), $(!RasEventTypeSupported)
   ifint $($ShellCode) != $(!SHELL_CODE_OK)
       Debug-Output "InstallSoftware: AddSoftware bombed out for "$(ThisOption)
       goto InstallSoftwareError
   endif
   set RegistryErrorIndex = $($R0)
   Ifstr(i) $(RegistryErrorIndex) == SERVICE_ALREADY_EXISTS
       return $(Status)
   EndIf
   Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
       CloseRegKey $($R1)
       CloseRegKey $($R2)
       CloseRegKey $($R3)
       CloseRegKey $($R4)
       CloseRegKey $($R5)
       goto InstallSoftwareError
   endif
   Set SoftProductKey    = $($R1)
   CloseRegKey $($R2)
   CloseRegKey $($R3)
   CloseRegKey $($R4)
   CloseRegKey $($R5)
   set NewValueList = +
       {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)},+
        {ServiceName,$(NoTitle),$(!REG_VT_SZ),+
         $(!Product$(ThisOption)Name)},+
        {SoftwareType,$(NoTitle),$(!REG_VT_SZ),+
         $(!Product$(ThisOption)SvcType)},+
        {Title,$(NoTitle),$(!REG_VT_SZ),$(!Product$(ThisOption)Title)},+
        {Description,$(NoTitle),$(!REG_VT_SZ),+
         $(!Product$(ThisOption)Description)},+
        {PathName,$(NoTitle),$(!REG_VT_SZ),+
         $(!Product$(ThisOption)ImagePath)},+
        {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMajorVersion)},+
        {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(!ProductMinorVersion)},+
        {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
        {Hidden,$(NoTitle),$(!REG_VT_DWORD),$(!HideComponent)}}
   Shell  $(!UtilityInf), AddValueList, $(SoftProductKey), $(NewValueList)
   ifint $($ShellCode) != $(!SHELL_CODE_OK)
       Debug-Output "InstallSoftware: AddValueList bombed out for "$(ThisOption)
       goto InstallSoftwareError
   endif
   set RegistryErrorIndex = $($R0)
   CloseRegKey $(SoftProductKey)
   Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
       goto InstallSoftwareError
   endif
   goto InstallSoftwareEnd
InstallSoftwareError =+
   read-syms ShellCodeErrorMsg$(!STF_LANGUAGE)
   ui start "Error Message"
   return STATUS_FAILED
InstallSoftwareEnd =+
   return $(Status)
[AddInfToReviewProgramsList]
    Debug-Output "AddInfToReviewPrograms: entry"
```

```
    set Status  = STATUS_FAILED
    set KeyNull = ""
    set NcpaKeyName = $(!NTN_SoftwareBase)"\Microsoft\Ncpa\CurrentVersion"
    set newList = {"OEMNSVRA.INF"}
    OpenRegKey $(!REG_H_LOCAL) "" $(NcpaKeyName) $(!MAXIMUM_ALLOWED) KeyNcpa
    Ifstr(i) $(KeyNcpa) != $(KeyNull)
        GetRegValue    $(KeyNcpa) "ReviewPrograms" TmpList
        ifint $(RegLastError) == 0
            Debug-Output "AddInf: old List "*($(TmpList), 4)
            ForListDo *($(TmpList),4)
                ifstr(i) $($) != "OEMNSVRA.INF"
                    set newList = >($(newList), $($))
                endif
            EndForListDo
            Debug-Output "OEMNSVRA.INF: AddInf: new List "$(newList)
            SetRegValue $(KeyNcpa) {ReviewPrograms, 0, +
                                    $(!REG_VT_MULTI_SZ),$(newList)}
            CloseRegKey $(KeyNcpa)
            Set Status = STATUS_SUCCESSFUL
        endif
    else
        Debug-Output "AddInfToReviewPrograms: error opening ncpa key."
    endif
    Debug-Output "AddInfToReviewPrograms: exit"
    return $(Status)
[RemoveInfFromReviewPrograms]
    Debug-Output "RemoveInfFromReviewPrograms: entry"
    set Status  = STATUS_FAILED
    set KeyNull = ""
    set NcpaKeyName = $(!NTN_SoftwareBase)"\Microsoft\Ncpa\CurrentVersion"
    set newList = {}
    OpenRegKey $(!REG_H_LOCAL) "" $(NcpaKeyName) $(!MAXIMUM_ALLOWED) KeyNcpa
    Ifstr(i) $(KeyNcpa) != $(KeyNull)
        GetRegValue    $(KeyNcpa) "ReviewPrograms" TmpList
        ifint $(RegLastError) == 0
            Debug-Output "RemoveInf: old List "*($(TmpList), 4)
            ForListDo *($(TmpList),4)
                ifstr(i) $($) != "OEMNSVRA.INF"
                    set newList = >($(newList), $($))
                endif
            EndForListDo
            Debug-Output "AddInf: new List "$(newList)
            SetRegValue $(KeyNcpa) {ReviewPrograms, 0, +
                                    $(!REG_VT_MULTI_SZ),$(newList)}
            CloseRegKey $(KeyNcpa)
            Set Status = STATUS_SUCCESSFUL
        endif
    else
        Debug-Output "RemoveInfFromReviewPrograms: error opening ncpa key."
    endif
    Debug-Output "RemoveInfFromReviewPrograms: exit"
    return $(Status)
[InstallResources]
    set STF_VITAL = NO
    Debug-Output  "In Copying Files"
    ifstr(i) $(AddCopy) == "YES"
        CreateDir $(RasDir)
        AddSectionFilesToCopyList Files-Resource $(SourceDir) $(!
STF_WINDOWSSYSPATH)
        AddSectionKeyFileToCopyList Files-Ras-Inf "rasico" $(SourceDir) $
(ProductPath)
        AddSectionKeyFileToCopyList Files-Ras-Inf "rasread" $(SourceDir) $
(ProductPath)
        ifstr(i) $(!NTN_InstallMode) == "Install"
```

```
            AddSectionKeyFileToCopyList Files-Ras-Inf "modem" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Inf "pad" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Inf "switch" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Scp "cis" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Scp "slip" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Scp "slipmenu" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Scp "pppmenu" $(SourceDir) $
(ProductPath)
            AddSectionKeyFileToCopyList Files-Ras-Scp "scriptdoc" $(SourceDir) $
(ProductPath)
        else-Ifstr(i) $(!NTN_InstallMode) == "Update"
            forlistdo {modem.new, pad.new, switch.new, cis.new, slip.new,
slipmenu.new, pppmenu.new, script.doc}
                  LibraryProcedure Status, $(!LIBHANDLE), DelFile +
                                           $(!STF_WINDOWSSYSPATH)"\RAS\"$($)
            set STF_RENAME = "MODEM.NEW"
            AddSectionKeyFileToCopyList Files-Ras-Inf "modem" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "PAD.NEW"
            AddSectionKeyFileToCopyList Files-Ras-Inf "pad" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "SWITCH.NEW"
            AddSectionKeyFileToCopyList Files-Ras-Inf "switch" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "CIS.SCP"
            AddSectionKeyFileToCopyList Files-Ras-Scp "cis" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "SLIP.SCP"
            AddSectionKeyFileToCopyList Files-Ras-Scp "slip" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "SLIPMENU.SCP"
            AddSectionKeyFileToCopyList Files-Ras-Scp "slipmenu" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "PPPMENU.SCP"
            AddSectionKeyFileToCopyList Files-Ras-Scp "pppmenu" $(SourceDir) $
(ProductPath)
            set STF_RENAME = "SCRIPT.DOC"
            AddSectionKeyFileToCopyList Files-Ras-Scp "scriptdoc" $(SourceDir) $
(ProductPath)
            set STF_RENAME = ""
        endif
    endif
    ifstr(i) $(!NTN_InstallMode) == "Update"
       set !STF_NCPA_FLUSH_COPYLIST = TRUE
       CopyFilesInCopyList
    else-ifstr(i) $(DoCopy) == "YES"
        set !STF_NCPA_FLUSH_COPYLIST = TRUE
        CopyFilesInCopyList
    endif
    Debug-Output  "Done Copying Files"
    exit
[RemoveResources]
    Debug-Output "In removing infs and dll"
    set RemoveList = {}
    set RenameList = {}
    set RemoveList = >($(RemoveList), #(Files-RemoveList, MODEMINF, 1))
    set RemoveList = >($(RemoveList), #(Files-RemoveList, PADINF, 1))
    set RemoveList = >($(RemoveList), #(Files-RemoveList, SWITCHINF, 1))
```

```
    set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSETUPHLP, 1))
    set RenameList = >($(RenameList), #(Files-RemoveList, RASRESDLL, 1))
    set RenameList = >($(RenameList), #(Files-RemoveList, RASCFGDLL, 1))
    set RenameList = >($(RenameList), #(Files-RemoveList, RASFIL32DLL, 1))
    ForListDo $(RemoveList)
        Debug-Output "Removing "$($)
        LibraryProcedure Status , $(!LIBHANDLE), DelFile $($)
        Debug-Output "Status is "$(Status)
    EndForListDo
    ForListDo $(RenameList)
        Split-String $($) "\" FilePath
        QueryListSize PathLen $(FilePath)
        Split-String *($(FilePath),$(PathLen)) "." FullFileName
        Set FileName = *($(FullFileName),1)
        Debug-Output "FileName is "$(FileName)
        LibraryProcedure STATUS, $(!LIBHANDLE),CheckFileExistance $(!
STF_WINDOWSSYSPATH)"\"$(FileName)".old"
        Debug-Output "CheckFile Status = "$(STATUS)
        ifstr(i) $(STATUS) == YES
            LibraryProcedure STATUS, $(!LIBHANDLE), DelFile $(!
STF_WINDOWSSYSPATH)"\"$(FileName)".old"
            Debug-Output "Delfile Status = "$(STATUS)
        endif
        Debug-Output "Renaming from "$($)
        Debug-Output "Renaming to "$(!STF_WINDOWSSYSPATH)"\"$(FileName)".old"
        LibraryProcedure Status1 , $(!LIBHANDLE), RenFile $($), $(!
STF_WINDOWSSYSPATH)"\"$(FileName)".old"
        Debug-Output "Status is "$(Status1)
        AddFileToDeleteList $(!STF_WINDOWSSYSPATH)"\"$(FileName)".old"
    EndForListDo
    exit
[InstallRasFiles]
    set STF_VITAL = NO
    set STF_OVERWRITE    = "VERIFYSOURCEOLDER"
    Debug-Output  "In InstallRasFiles Copying Files"
    Debug-Output  "ServerInstalled "$(ServerInstalled)
    Debug-Output  "ClientInstalled "$(ClientInstalled)
    ifstr(i) $(DoAdmin) == TRUE
        AddSectionFilesToCopyList Files-Ras-Admin $(SourceDir) +
                  $(!STF_WINDOWSSYSPATH)
        ifstr(i) $(DoAdminOnly) == TRUE
            goto InstallRasFiles1
        endif
    endif
    ifstr(i) $(DoClient) == TRUE
        AddSectionFilesToCopyList Files-Ras-Client $(SourceDir) +
                $(!STF_WINDOWSSYSPATH)
    endif
    ifstr(i) $(DoServer) == TRUE
        AddSectionFilesToCopyList Files-Ras-Server $(SourceDir) +
                $(!STF_WINDOWSSYSPATH)
    endif
    ifstr(i) $(!NTN_InstallMode) == "Update"
        AddSectionFilesToCopyList Files-Ras-Common $(SourceDir) +
                    $(!STF_WINDOWSSYSPATH)
        AddSectionFilesToCopyList Files-Ras-Drivers $(SourceDir) +
                    $(!STF_WINDOWSSYSPATH)\drivers
    else
        ifstr(i) $(ServerInstalled) == FALSE
            ifstr(i) $(ClientInstalled) == FALSE
                AddSectionFilesToCopyList Files-Ras-Common $(SourceDir) +
                            $(!STF_WINDOWSSYSPATH)
                AddSectionFilesToCopyList Files-Ras-Drivers $(SourceDir) +
                            $(!STF_WINDOWSSYSPATH)\drivers
```

```
            endif
         endif
    endif
InstallRasFiles1 = +
    ifstr(i) $(!NTN_InstallMode) == "Update"
       set !STF_NCPA_FLUSH_COPYLIST = TRUE
       CopyFilesInCopyList
    else
       ifstr(i) $(DoCopy) == "YES"
          set !STF_NCPA_FLUSH_COPYLIST = TRUE
          CopyFilesInCopyList
       endif
    endif
    Debug-Output  "Done Copying Files"
    exit
[RemoveRasFiles]
    ifstr(i) $(!NTN_InstallMode) == install
        ifstr(i) $(DoCopy) == "NO"
           Exit
        endif
    endif
    set RemoveList = {}
    set RenameList = {}
    set fCommonRemoved = FALSE
    ifstr(i) $(DoServer) == TRUE
       set fCommonRemoved = TRUE
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASGTWYDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASGPRXYDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSPRXYEXE, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSRVEXE, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCTRSDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCTRSINI, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCTRNMH, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSAUTHDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASADMINDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMANDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMANEXE, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMSGDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMXSDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSERDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPXCPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPPPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPPPENDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPAPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCHAPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSPAPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPCPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPHLPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASNBFCPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCCPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCBCPDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASNBIPCDLL, 1))
       set RenameList = >($(RenameList), #(Files-RemoveList, ASYNCMACSYS, 1))
       set RenameList = >($(RenameList), #(Files-RemoveList, NDISWANSYS, 1))
    endif
    ifstr(i) $(DoClient) == TRUE
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASDIALEXE, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPHONEHLP, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCAUTHDLL, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASAPI32DLL, 1))
       ifstr(i) $(fCommonRemoved) == FALSE
           set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMANDLL, 1))
           set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMANEXE, 1))
           set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMSGDLL, 1))
```

```
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASMXSDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSERDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPXCPDLL,
1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPPPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPPPENDLL,
1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASPAPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCHAPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASSPAPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPCPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASIPHLPDLL,
1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASNBFCPDLL,
1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCCPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASCBCPDLL, 1))
            set RemoveList = >($(RemoveList), #(Files-RemoveList, RASNBIPCDLL,
1))
            set RenameList = >($(RenameList), #(Files-RemoveList, ASYNCMACSYS,
1))
          set RenameList = >($(RenameList), #(Files-RemoveList, NDISWANSYS, 1))
        endif
    endif
    ifstr(i) $(DoAdmin) == TRUE
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASADMINHLP, 1))
       set RemoveList = >($(RemoveList), #(Files-RemoveList, RASGLOSSHLP, 1))
    endif
    ForListDo $(RemoveList)
        Debug-Output "Removing "$($)
        LibraryProcedure Status , $(!LIBHANDLE), DelFile $($)
        Debug-Output "Status is "$(Status)
    EndForListDo
    ForListDo $(RenameList)
        Split-String $($) "\" FilePath
        QueryListSize PathLen $(FilePath)
        Split-String *($(FilePath),$(PathLen)) "." FullFileName
        Set FileName = *($(FullFileName),1)
        Debug-Output "FileName is "$(FileName)
        LibraryProcedure STATUS, $(!LIBHANDLE),CheckFileExistance $(!
STF_WINDOWSSYSPATH)"\DRIVERS\"$(FileName)".old"
        Debug-Output "CheckFile Status = "$(STATUS)
        ifstr(i) $(STATUS) == YES
            LibraryProcedure STATUS, $(!LIBHANDLE), DelFile $(!
STF_WINDOWSSYSPATH)"\DRIVERS\"$(FileName)".old"
            Debug-Output "Delfile Status = "$(STATUS)
        endif
        Debug-Output "Renaming from "$($)
        Debug-Output "Renaming to "$(!STF_WINDOWSSYSPATH)"\DRIVERS\"$
(FileName)".old"
        LibraryProcedure Status1 , $(!LIBHANDLE), RenFile $($), $(!
STF_WINDOWSSYSPATH)"\DRIVERS\"$(FileName)".old"
        Debug-Output "Status is "$(Status1)
        AddFileToDeleteList $(!STF_WINDOWSSYSPATH)"\DRIVERS\"$(FileName)".old"
    EndForListDo
    exit
[RemoveRasGroup]
    Debug-Output "Removing icons from "$(RasGroup)
    CreateCommonProgManGroup $(RasGroup) ""
    RemoveCommonProgManGroup $(RasGroup)
    exit
[AddNDISWANToServiceGroupOrder]
    set GroupOrderName = "SYSTEM\CurrentControlSet\Control\ServiceGroupOrder"
    OpenRegKey $(!REG_H_LOCAL) "" $(GroupOrderName) $(!MAXIMUM_ALLOWED) KeyGroup
```

```
    set OldList = {}
    Ifstr(i) $(KeyGroup) != $(KeyNull)
        GetRegValue  $(KeyGroup) "List" TmpList
        ifint $(RegLastError) == 0
            ForListDo *($(TmpList), 4)
                set OldList = >($(OldList), $($))
            EndForListDo
        endif
        Debug-Output "AddNDISWANToServiceGroupOrder current list "$(OldList)
        Ifcontains(i) "NDISWAN" not-in $(OldList)
            set NewGroupList = {}
            ForListDo $(OldList)
                set NewGroupList = >($(NewGroupList), $($))
                ifstr(i) $($) == "NDIS"
                    set NewGroupList = >($(NewGroupList), "NDISWAN" )
                endif
            EndForListDo
            Debug-Output "AddNDISWANToServiceGroupOrder new list "$
(NewGroupList)
            SetRegValue $(KeyGroup) {List, 0,$(!REG_VT_MULTI_SZ),$
(NewGroupList)}
        endif
        CloseRegKey $(KeyGroup)
    else
        Debug-Output "AddNDISWANToServiceGroupOrder error opening
ServiceGroupOrder key."
    endif
    return
[CheckRasInstalled]
    set MAXIMUM_ALLOWED   = 33554432
    set ProductKeyName = $(!NTN_SoftwareBase)"\Microsoft\RAS\CurrentVersion"
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED)
KeyProduct
    Ifstr $(KeyProduct) != ""
        CloseRegKey $(KeyProduct)
        return "TRUE"
    else
        return "FALSE"
[QueryComponentsInstalled]
    Set Status            =    STATUS_FAILED
    Set ValueName         =    ""
    Set InstalledList     =    {}
    Set InstalledFlags    =    {}
    Set TmpList           =    {}
    Set KeyNull           =    ""
    Set NumberOfComps     =    0
    Set RasKeyName        =    $($0)
    Debug-Output "QueryComponentsInstalled: "$(RasKeyName)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) $(!MAXIMUM_ALLOWED) KeyProduct
    Ifstr(i) $(KeyProduct) != $(KeyNull)
        GetRegValue    $(KeyProduct) "RasComponents" TmpList
        Debug-Output "QueryComponentsInstalled: "$(TmpList)
        ForListDo *($(TmpList),4)
            set InstalledList = >($(InstalledList), $($))
        EndForListDo
        CloseRegKey $(KeyProduct)
        Ifcontains(i) "Server" in $(InstalledList)
            Set InstalledFlags = >($(InstalledFlags), TRUE)
        else
            Set InstalledFlags = >($(InstalledFlags), FALSE)
        endif
        Ifcontains(i) "Client" in $(InstalledList)
            Set InstalledFlags = >($(InstalledFlags), TRUE)
        else
```

```
            Set InstalledFlags = >($(InstalledFlags), FALSE)
        endif
        Ifcontains(i) "Admin" in $(InstalledList)
            Set InstalledFlags = >($(InstalledFlags), TRUE)
        else
            Set InstalledFlags = >($(InstalledFlags), FALSE)
        endif
        QueryListSize NumberOfComps $(InstalledList)
        ifint $(NumberOfComps) == 1
            Ifcontains(i) "Server" in $(InstalledList)
                Set InstalledFlags = >($(InstalledFlags), TRUE)
            else
                Set InstalledFlags = >($(InstalledFlags), FALSE)
            endif
            Ifcontains(i) "Client" in $(InstalledList)
                Set InstalledFlags = >($(InstalledFlags), TRUE)
            else
                Set InstalledFlags = >($(InstalledFlags), FALSE)
            endif
            Ifcontains(i) "Admin" in $(InstalledList)
                Set InstalledFlags = >($(InstalledFlags), TRUE)
            else
                Set InstalledFlags = >($(InstalledFlags), FALSE)
            endif
        else
            Set InstalledFlags = >($(InstalledFlags), FALSE)
            Set InstalledFlags = >($(InstalledFlags), FALSE)
            Set InstalledFlags = >($(InstalledFlags), FALSE)
        endif
        Set Status = STATUS_SUCCESSFUL
    Else
       Set Status = STATUS_NOT_FOUND
    EndIf
    Debug-Output "QueryComponentsInstalled: "$(InstalledList)
    Return $(Status) $(InstalledList) $(InstalledFlags)
[UpdateComponentsInstalled]
    Set Status          =    STATUS_FAILED
    Set InstalledList   =    $($0)
    Set RasKeyName      =    $($1)
    Set KeyNull         =    ""
    Debug-Output "UpdateComponentsInstalled: "$(RasKeyName)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) $(!MAXIMUM_ALLOWED) KeyProduct
    Ifstr(i) $(KeyProduct) != $(KeyNull)
        SetRegValue $(KeyProduct) {RasComponents, 0,$(!REG_VT_MULTI_SZ),$
(InstalledList)}
        CloseRegKey $(KeyProduct)
        Set Status = STATUS_SUCCESSFUL
    else
        Set Status = STATUS_FAILED
    endif
    return $(Status)
[GetNetworkAccess]
    set Status          = STATUS_SUCCESSFUL
    Set KeyNull         = ""
    set NbfNetAccess   = 0
    set TcpIpNetAccess = 0
    set IpxNetAccess   = 0
    set RasProtocolsKeyName = $(!NTN_SoftwareBase)"\Microsoft\RAS\PROTOCOLS"
    Debug-Output "GetNetworkAccess entry"
    set ProtocolKeyName = $(RasProtocolsKeyName)"\NBF"
    OpenRegKey $(!REG_H_LOCAL) "" $(ProtocolKeyName) $(!MAXIMUM_ALLOWED)
KeyProtocol
    Ifstr(i) $(KeyProtocol) != $(KeyNull)
       GetRegValue $(KeyProtocol), "NetBiosGatewayEnabled" NetworkAccess
```

```
       ifint $(RegLastError) == 0
           set NbfNetAccess = *($(NetworkAccess), 4)
       endif
       CloseRegKey $(KeyProtocol)
    else
       Debug-Output "GetNetworkAccess:error opening key "$(ProtocolKeyName)
    endif
    set ProtocolKeyName = $(RasProtocolsKeyName)"\IP"
    OpenRegKey $(!REG_H_LOCAL) "" $(ProtocolKeyName) $(!MAXIMUM_ALLOWED)
KeyProtocol
    Ifstr(i) $(KeyProtocol) != $(KeyNull)
       GetRegValue $(KeyProtocol), "AllowNetworkAccess" NetworkAccess
       ifint $(RegLastError) == 0
           set TcpIpNetAccess = *($(NetworkAccess), 4)
       endif
       CloseRegKey $(KeyProtocol)
    else
       Debug-Output "GetNetworkAccess:error opening key "$(ProtocolKeyName)
    endif
    set ProtocolKeyName = $(RasProtocolsKeyName)"\IPX"
    OpenRegKey $(!REG_H_LOCAL) "" $(ProtocolKeyName) $(!MAXIMUM_ALLOWED)
KeyProtocol
    Ifstr(i) $(KeyProtocol) != $(KeyNull)
       GetRegValue $(KeyProtocol), "AllowNetworkAccess" NetworkAccess
       ifint $(RegLastError) == 0
           set IpxNetAccess = *($(NetworkAccess), 4)
       endif
       CloseRegKey $(KeyProtocol)
    else
       Debug-Output "GetNetworkAccess:error opening key "$(ProtocolKeyName)
    endif
    Debug-Output "GetNetworkAccess exit"
    return $(Status) $(NbfNetAccess) $(TcpIpNetAccess) $(IpxNetAccess)
[IsNdisWanBHAdapterInstalled]
    set Status      = STATUS_FAILED
    Set KeyNull     = ""
    Debug-Output "IsNdisWanBHAdapterInstalled entry"
    set NetworkCardKey = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
NetworkCardKey
    Ifstr(i) $(NetworkCardKey) != $(KeyNull)
        set NetcardsList = {}
        EnumRegKey $(NetworkCardKey) NetcardsList
        Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
           Debug-Output "IsNdisWanBHAdapterInstalled: EnumRegKey failed."
           goto IsNdisWanBHAdapterInstalledEnd
        endif
        ForListDo $(NetcardsList)
           set KeyName = *($($),1)
           set Card = $(KeyNull)
           OpenRegKey $(NetworkCardKey) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
           ifstr $(Card) == $(KeyNull)
               Debug-Output "IsNdisWanBHAdapterInstalled: could not open netcard
key "$(KeyName)
           else
               GetRegValue $(Card), "ProductName" ProductNameInfo
               Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                  Debug-Output "IsNdisWanBHAdapterInstalled: ProductName not
found."
               else
                  set CardProductName = *($(ProductNameInfo), 4)
                  Debug-Output "IsNdisWanBHAdapterInstalled: ProductName. "$
(CardProductName)
                      ifstr(i) $(CardProductName) == $(!ProductNDISWANName)
```

```
                        set Status = STATUS_SUCCESSFUL
                        CloseRegKey $(Card)
                        goto IsNdisWanBHAdapterInstalledEnd
                    endif
                endif
                CloseRegKey $(Card)
            endif
        EndForListDo
    else
        Debug-Output "IsNdisWanBHAdapterInstalled: failed to open "$(!
NetworkCardKeyName)
    endif
IsNdisWanBHAdapterInstalledEnd =+
    ifstr(i) $(NetworkCardKey) != $(KeyNull)
        CloseRegKey $(NetworkCardKey)
    endif
    Debug-Output "IsNdisWanBHAdapterInstalled exit"
    Return $(Status)
[InstallNdisWanBHAdapter]
    set Status = STATUS_FAILED
    set ThisOption = NDISWAN
    Debug-Output "InstallNdisWanBHAdapter entry"
    Shell "utility.inf", AddHardwareComponent, +
          $(!Product$(ThisOption)Name),$(!RasInfName),+
          $(!Product$(ThisOption)KeyName)
    ifint $($R4) != -1
        Set !NETCARD_LIST = >($(!NETCARD_LIST), +
                              {$(!Product$(ThisOption)Name),+
                              $(!NetworkCardKeyName)"\"$($R4)})
    endif
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
       Debug-Output "InstallNdisWanBHAdapter:Shell error"
       goto InstallNdisWanBHAdapterEnd
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        Debug-Output "InstallNdisWanBHAdapter:Registry error: add hardware
component"
        CloseRegKey $($R1)
        CloseRegKey $($R2)
        CloseRegKey $($R3)
        goto InstallNdisWanBHAdapterEnd
    endif
    set KeyNetcard    = $($R1)
    set KeyParameters = $($R3)
    set KeyAdapterRules = $($R2)
    set AdapterNumber = $($R4)
    set NewValueList = +
        {{Manufacturer,$(NoTitle),$(!REG_VT_SZ),$(!Manufacturer)},+
         {Title,$(NoTitle),$(!REG_VT_SZ),+
          "["$($R4)"] "$(!Product$(ThisOption)Title)},+
         {Description,$(NoTitle),$(!REG_VT_SZ),+
          $(!Product$(ThisOption)Description)},+
         {ProductName,$(NoTitle),$(!REG_VT_SZ),+
          $(!Product$(ThisOption)Name)},+
         {ServiceName,$(NoTitle),$(!REG_VT_SZ),$($R5)},+
         {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(!CurrentDate),1)},+
         {Hidden,$(NoTitle),$(!REG_VT_DWORD),1}}
    Shell  "utility.inf", AddValueList, $(KeyNetcard), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "InstallNdisWanBHAdapter: ShellCode error"
        goto InstallNdisWanBHAdapterEnd
    endif
    CloseRegKey $(KeyNetcard)
```

```
    set TempProdName = """"$(!Product$(ThisOption)Name)$(AdapterNumber)""""
    set TempBindForm = $(TempProdName)$(!NetRuleHardwareBHBindForm)
    set NewValueList = +
        {{type,$(NoTitle),$(!REG_VT_SZ),+
          $(!NetRuleHardwareBHType)},+
         {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)}, +
         {class,$(NoTitle),$(!REG_VT_MULTI_SZ),+
          $(!NetRuleHardwareBHClass)}, +
         {InfOption,$(NoTitle),$(!REG_VT_SZ),$(ThisOption)}, +
         {Infname ,$(NoTitle),$(!REG_VT_SZ),$(!RasInfName)}}
    Shell  "utility.inf", AddValueList, $(KeyAdapterRules), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "InstallNdisWanBHAdapter: ShellCode error"
        goto InstallNdisWanBHAdapterEnd
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        Debug-Output "InstallNdisWanBHAdapter:Registry error: add hardware
component"
        CloseRegKey $(KeyParameters)
        CloseRegKey $(KeyAdapterRules)
        goto InstallNdisWanBHAdapterEnd
    endif
    CloseRegKey $(KeyAdapterRules)
    CloseRegKey $(KeyParameters)
    set Status = STATUS_SUCCESSFUL
InstallNdisWanBHAdapterEnd =+
    Debug-Output "InstallNdisWanBHAdapter exit"
    return $(Status)
[InstallRasArpService]
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "InstallRasArpService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasArpKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) == $(KeyNull)
        Shell "utility.inf", CreateService, $(!ProductRASARPName), +
            $(!ProductRASARPDisplayName), +
            $(!ProductRASARPImagePath), +
            "kernelautostart", "PNP_TDI", {"TCPIP"}, ""
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: InstallRasArpService : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $($R1)
        CloseRegKey $($R2)
        CloseRegKey $($R3)
        Ifstr(i) $(RegistryErrorIndex) == SERVICE_ALREADY_EXISTS
           return $(Status)
        EndIf
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: InstallRasArpService: Registry error
create service"
            return STATUS_FAILED
        endif
    endif
    Ifstr(i) $(KeyService) != $(KeyNull)
        CloseRegKey $(KeyService)
    endif
    Debug-Output "InstallRasArpService exit"
    return $(Status)
[RemoveRasArpService]
    set Status = STATUS_SUCCESSFUL
```

```
    set KeyNull = ""
    Debug-Output "RemoveRasArpService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasArpKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        Shell "utility.inf", RemoveService $(!ProductRASARPName) "YES"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: RemoveRasArpService : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: RemoveRasArpService: Registry error: "
            return STATUS_FAILED
        endif
        CloseRegKey $(KeyService)
    endif
    Debug-Output "RemoveRasArpService exit"
    return $(Status)
[InstallRasAcdService]
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "InstallRasAcdService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasAcdKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) == $(KeyNull)
        Shell "utility.inf", CreateService, $(!ProductRASACDName), +
            $(!ProductRASACDDisplayName), +
            $(!ProductRASACDImagePath), +
            "kernelautostart", "Streams Drivers", {}, ""
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: InstallRasArpService : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        CloseRegKey $($R1)
        CloseRegKey $($R2)
        CloseRegKey $($R3)
        Ifstr(i) $(RegistryErrorIndex) == SERVICE_ALREADY_EXISTS
           return $(Status)
        EndIf
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: InstallRasAcdService: Registry error
create service"
            return STATUS_FAILED
        endif
    endif
    Debug-Output "InstallRasArpService exit"
    return $(Status)
[RemoveRasAcdService]
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "RemoveRasAcdService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasAcdKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        Shell "utility.inf", RemoveService $(!ProductRASACDName) "YES"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: RemoveRasAcdService : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: RemoveRasAcdService: Registry error: "
```

```
            return STATUS_FAILED
        endif
        CloseRegKey $(KeyService)
    endif
    Debug-Output "RemoveRasAcdService exit"
    return $(Status)
[InstallNdisTapiService]
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "InstallNdisTapiService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!NdisTapiKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) == $(KeyNull)
        Shell "utility.inf", CreateService, $(!ProductNDISTAPIName), +
            $(!ProductNDISTAPIDisplayName), +
            $(!ProductNDISTAPIImagePath), +
            "kernelauto", "NDIS", {}, ""
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: InstallNdisTapiService : ShellCode
error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        set ParamKey = $($R2)
        CloseRegKey $($R1)
        CloseRegKey $($R3)
        Ifstr(i) $(RegistryErrorIndex) == SERVICE_ALREADY_EXISTS
           return $(Status)
        EndIf
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: InstallNdisTapiService: Registry error
create service"
            return STATUS_FAILED
        endif
        ifstr(i) $(!STF_PRODUCT) != "WINNT"
            SetRegValue $(ParamKey) {AsyncEventQueueSize, 0, $(!REG_VT_DWORD),
3072}
        else
            SetRegValue $(ParamKey) {AsyncEventQueueSize, 0, $(!REG_VT_DWORD),
768}
        endif
        CloseRegKey $(ParamKey)
    endif
    Ifstr(i) $(KeyService) != $(KeyNull)
        CloseRegKey $(KeyService)
    endif
    Debug-Output "InstallNdisTapiService exit"
    return $(Status)
[RemoveNdisTapiService]
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "RemoveNdisTapiService entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!NdisTapiKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        Shell "utility.inf", RemoveService $(!ProductNDISTAPIName) "YES"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNSVRA.INF: RemoveNdisTapiService : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNSVRA.INF: RemoveNdisTapiService: Registry error: "
            return STATUS_FAILED
```

```
        endif
        CloseRegKey $(KeyService)
    endif
    Debug-Output "RemoveNdisTapiService exit"
    return $(Status)
[InstallNwlnkRipService]
    Debug-Output "InstallNwlnkRipService entry"
    set Status  = STATUS_FAILED
    set KeyNull = ""
    Set SrcDir  = $(!STF_SRCDIR)
    ifstr(i) $(!NTN_InstallMode) == "install"
       set !STF_SRCDIR_OVERRIDE = $(SrcDir)
    endif
    set AddCopy  = YES
    set DoCopy   = YES
    set DoConfig = YES
    set SaveNTN_InstallMode = $(!NTN_InstallMode)
    set !NTN_InstallMode = install
    Shell "oemnsvrr.inf" InstallOption $(!STF_LANGUAGE) "NWLNKRIP"  +
          $(SrcDir) $(AddCopy) $(DoCopy) $(DoConfig) "RAS"
    set !NTN_InstallMode = $(SaveNTN_InstallMode)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "Error installing NWLNKRIP service"
        Goto InstallNwlnkRipServiceEnd
    Endif
    Set InstallStatus = $($R0)
    Ifstr(i) $(InstallStatus) != STATUS_SUCCESSFUL
        Ifstr(i) $(InstallStatus) != STATUS_USERCANCEL
            Debug-Output "InstallNwlnkRipService returned "$(InstallStatus)
            Goto InstallNwlnkRipServiceEnd
        Endif
    Endif
    OpenRegKey $(!REG_H_LOCAL) "" $(!RasIsnRipKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    ifstr $(KeyService) != ""
        OpenRegKey $(KeyService) "" "Parameters" $(!MAXIMUM_ALLOWED) KeyParams
        ifstr(i) $(KeyParams) != ""
            GetRegValue $(KeyParams),"NetbiosRouting", NetbiosRoutingInfo
            Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
               set NetbiosRouting = *($(NetbiosRoutingInfo), 4)
            else
               set NetbiosRouting = 2
            endif
            ifint $(NetbiosRouting) == 0
               set NetbiosRouting = 2
            else-ifint $(NetbiosRouting) == 1
               set NetbiosRouting = 3
            endif
            SetRegValue $(KeyParams) {NetbiosRouting, $(NoTitle), $(!
REG_VT_DWORD), $(NetbiosRouting)}
            CloseRegKey $(KeyParams)
        endif
        CloseRegKey $(KeyService)
    endif
    Shell "" AddServiceDependency "RemoteAccess" $(!ProductRASISNRIPName)
    ifstr(i) $($R0) == STATUS_FAILED
        Debug-Output "InstallNwlnkRipService: error adding service dependency"
    endif
    set Status = STATUS_SUCCESSFUL
InstallNwlnkRipServiceEnd =+
    Debug-Output "InstallNwlnkRipService exit"
    return $(Status)
[RemoveNwlnkRipService]
    Debug-Output "RemoveNwlnkRipService entry"
```

```
    set Status = STATUS_FAILED
    set KeyNull = ""
    Set SrcDir  = $(!STF_SRCDIR)
    set AddCopy  = YES
    set DoCopy   = YES
    set DoConfig = YES
    set SaveNTN_InstallMode = $(!NTN_InstallMode)
    set !NTN_InstallMode = deinstall
    Shell "oemnsvrr.inf" InstallOption $(!STF_LANGUAGE) "NWLNKRIP"  +
          $(SrcDir) $(AddCopy) $(DoCopy) $(DoConfig) "RAS"
    set !NTN_InstallMode = $(SaveNTN_InstallMode)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "Error removing NWLNKRIP service"
        Goto RemoveNwlnkRipServiceEnd
    Endif
    Set RemoveStatus = $($R0)
    Ifstr(i) $(RemoveStatus) != STATUS_SUCCESSFUL
        Ifstr(i) $(RemoveStatus) != STATUS_USERCANCEL
            Debug-Output "RemoveNwlnkRipService returned "$(RemoveStatus)
            Goto RemoveNwlnkRipServiceEnd
        Endif
    Endif
    Shell "" RemoveServiceDependency "RemoteAccess" $(!ProductRASISNRIPName)
    ifstr(i) $($R0) == STATUS_FAILED
        Debug-Output "RemoveNwlnkRipService: error removing service dependency"
    endif
    set Status = STATUS_SUCCESSFUL
RemoveNwlnkRipServiceEnd =+
    Debug-Output "RemoveNwlnkRipService exit"
    return $(Status)
[InstallIsnSapService]
    Debug-Output "InstallIsnSapService entry"
    set Status  = STATUS_FAILED
    set KeyNull = ""
    Set SrcDir  = $(!STF_SRCDIR)
    ifstr(i) $(!NTN_InstallMode) == "install"
       set !STF_SRCDIR_OVERRIDE = $(SrcDir)
    endif
    set AddCopy  = YES
    set DoCopy   = YES
    set DoConfig = YES
    set SaveNTN_InstallMode = $(!NTN_InstallMode)
    set !NTN_InstallMode = install
    Shell "oemnsvsa.inf" InstallOption $(!STF_LANGUAGE) "SAP"  +
          $(SrcDir) $(AddCopy) $(DoCopy) $(DoConfig) "RAS"
    set !NTN_InstallMode = $(SaveNTN_InstallMode)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "Error installing SAP agent"
        Goto InstallIsnSapServiceEnd
    Endif
    Set InstallStatus = $($R0)
    Ifstr(i) $(InstallStatus) != STATUS_SUCCESSFUL
        Ifstr(i) $(InstallStatus) != STATUS_USERCANCEL
            Debug-Output "InstallSapService returned "$(InstallStatus)
            Goto InstallIsnSapServiceEnd
        Endif
    Endif
    Shell "" AddServiceDependency "RemoteAccess" $(!ProductRASISNSAPName)
    ifstr(i) $($R0) == STATUS_FAILED
        Debug-Output "InstallIsnSapService: error adding service dependency"
    endif
    set Status = STATUS_SUCCESSFUL
InstallIsnSapServiceEnd =+
    Debug-Output "InstallIsnSapService exit"
```

```
    return $(Status)
[RemoveIsnSapService]
    Debug-Output "RemoveIsnSapService entry"
    set Status = STATUS_FAILED
    set KeyNull = ""
    Set SrcDir  = $(!STF_SRCDIR)
    set AddCopy  = YES
    set DoCopy   = YES
    set DoConfig = YES
    set SaveNTN_InstallMode = $(!NTN_InstallMode)
    set !NTN_InstallMode = deinstall
    Shell "oemnsvsa.inf" InstallOption $(!STF_LANGUAGE) "SAP"  +
          $(SrcDir) $(AddCopy) $(DoCopy) $(DoConfig) "RAS"
    set !NTN_InstallMode = $(SaveNTN_InstallMode)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "Error removing SAP agent"
        Goto RemoveIsnSapServiceEnd
    Endif
    Set RemoveStatus = $($R0)
    Ifstr(i) $(RemoveStatus) != STATUS_SUCCESSFUL
        Ifstr(i) $(RemoveStatus) != STATUS_USERCANCEL
            Debug-Output "RemoveIsnSapService returned "$(RemoveStatus)
            Goto RemoveIsnSapServiceEnd
        Endif
    Endif
    Shell "" RemoveServiceDependency "RemoteAccess" $(!ProductRASISNSAPName)
    ifstr(i) $($R0) == STATUS_FAILED
        Debug-Output "RemoveIsnSapService: error removing service dependency"
    endif
    set Status = STATUS_SUCCESSFUL
RemoveIsnSapServiceEnd =+
    Debug-Output "RemoveIsnSapService exit"
    return $(Status)
[WritePPPParameters]
    Debug-Output "WritePPPParameters: entry"
    Set Status = STATUS_FAILED
    set KeyNull = ""
    set RasManKeyName =  $(!NTN_ServiceBase)"\RasMan"
    set RasManPPPKeyName =  $(!NTN_ServiceBase)"\RasMan\PPP"
    set RasProtocolsKeyName = $(!NTN_SoftwareBase)"\Microsoft\RAS\PROTOCOLS"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasManKeyName) $(!MAXIMUM_ALLOWED) KeyRasMan
    ifstr $(KeyRasMan) != $(KeyNull)
       OpenRegKey $(KeyRasMan) "" "PPP" $(!MAXIMUM_ALLOWED) KeyRasManPPP
       set NewValueList = {}
       ifstr $(KeyRasManPPP) != $(KeyNull)
           EnumRegValue $(KeyRasManPPP) NewValueList
       else
           CreateRegKey $(KeyRasMan) +
                   {"PPP",0,GenericClass} "" $(!MAXIMUM_ALLOWED) "" KeyRasManPPP
           CloseRegKey $(KeyRasMan)
           ifstr $(KeyRasManPPP) == $(KeyNull)
               Debug-Output "WritePPPParameters: error creating RasMan\ppp key"
               goto WritePPPParametersEnd
           endif
       endif
    else
       Debug-Output "WritePPPParameters: error opening RasMan key"
       goto UpdateCPListEnd
    endif
    ifstr(i) $(NewValueList) == {}
        set NewValueList = {{MaxConfigure, 0, $(!REG_VT_DWORD), 10}, +
                            {MaxTerminate, 0, $(!REG_VT_DWORD), 2}, +
                            {MaxFailure,   0, $(!REG_VT_DWORD), 10}. +
                            {MaxReject,    0, $(!REG_VT_DWORD), 5}. +
```

```
                            {NegotiateTime, 0, $(!REG_VT_DWORD), 150}, +
                            {Logging, 0, $(!REG_VT_DWORD), 0}, +
                            {RestartTimer, 0, $(!REG_VT_DWORD), 3}}
    endif
    forlistdo $(NewValueList)
        SetRegValue $(KeyRasManPPP) $($)
    endforlistdo
    set ForceEncryptedPassword = 2
    set ForceEncryptedData     = 0
    OpenRegKey $(!REG_H_LOCAL) "" $(RasProtocolsKeyName) $(!MAXIMUM_ALLOWED)
KeyRasProtocols
    ifstr $(KeyRasProtocols) != $(KeyNull)
        GetRegValue $(KeyRasProtocols) "ForceEncryptedPassword" ForceValue
        ifint $(RegLastError) == 0
            set ForceEncryptedPassword = *($(ForceValue), 4)
        endif
        GetRegValue $(KeyRasProtocols) "ForceEncryptedData" ForceValue
        ifint $(RegLastError) == 0
            set ForceEncryptedData = *($(ForceValue), 4)
        endif
        SetRegValue $(KeyRasManPPP) {ForceEncryptedPassword, 0, +
                                     $(!REG_VT_DWORD),$(ForceEncryptedPassword)}
        SetRegValue $(KeyRasManPPP) {ForceEncryptedData, 0, +
                                     $(!REG_VT_DWORD),$(ForceEncryptedData)}
        CloseRegKey $(KeyRasProtocols)
    endif
    CloseRegKey $(KeyRasManPPP)
    set Status = STATUS_SUCCESSFUL
WritePPPParametersEnd =+
    Debug-Output "WritePPPParameters: exit"
    return $(Status)
[UpdateCPList]
    Debug-Output "UpdateCPList: entry"
    Set Status = STATUS_FAILED
    set KeyNull = ""
    set RasManPPPKeyName =  $(!NTN_ServiceBase)"\RasMan\PPP"
    Shell "" WritePPPParameters
    Shell "" QuerySelectedProtocols
    ifstr(i) $($R0) == STATUS_SUCCESSFUL
        set fNetbeuiSelected   = $($R1)
        set fTcpIpSelected     = $($R2)
        set fIpxSelected       = $($R3)
        set fNetbeuiAllowed    = $($R4)
        set fTcpIpAllowed      = $($R5)
        set fIpxAllowed        = $($R6)
    else
        Debug-Output "UpdateCPList: error QuerySelectedProtocols"
        goto UpdateCPListEnd
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(RasManPPPKeyName) $(!MAXIMUM_ALLOWED)
KeyRasManPPP
    ifstr $(KeyRasManPPP) != $(KeyNull)
        EnumRegKey $(KeyRasManPPP) OldCPList
        ifstr(i) $(fNetbeuiSelected) == TRUE
            set fNetbeuiChosen = TRUE
        else
            set fNetbeuiChosen = $(fNetbeuiAllowed)
        endif
        ifstr(i) $(fTcpIpSelected) == TRUE
            set fTcpIpChosen = TRUE
        else
            set fTcpIpChosen = $(fTcpIpAllowed)
        endif
        ifstr(i) $(fIpxSelected) == TRUE
```

```
            set fIpxChosen = TRUE
        else
            set fIpxChosen = $(fIpxAllowed)
        endif
        set CPList = {"PAP", "CHAP", "SPAP", "CBCP", "COMPCP"}
        ifstr(i) $(fNetbeuiChosen) == TRUE
            set CPList = >($(CPList), "NBFCP")
        endif
        ifstr(i) $(fTcpIpChosen) == TRUE
            set CPList = >($(CPList), "IPCP")
        endif
        ifstr(i) $(fIpxChosen) == TRUE
            set CPList = >($(CPList), "IPXCP")
        endif
        set CurrentCPList = {}
        ForListDo $(OldCPList)
           set CPName = *($($),1)
           ifcontains(i) $(CPName) not-in $(CPList)
               DeleteRegTree $(KeyRasManPPP) $(CPName)
           else
               set CurrentCPList = >($(CurrentCPList), $(CPName))
           endif
        EndForListDo
        forlistdo $(CPList)
            ifcontains(i) $($) not-in $(CurrentCPList)
                CreateRegKey $(KeyRasManPPP) +
                         {$($),0,GenericClass} "" $(!MAXIMUM_ALLOWED) "" KeyCp
                set path = #(CP-List, $($), 1)
                set NewValueList = {{Path, 0, $(!REG_VT_EXPAND_SZ), $(path)}}
                ifstr $(KeyCp) != $(KeyNull)
                    Shell  "utility.inf", AddValueList, $(KeyCp), +
                                                          $(NewValueList)
                    ifint $($ShellCode) != $(!SHELL_CODE_OK)
                        Debug-Output "UpdateCPList:AddValueList bombed out"
                        goto UpdateCPListEnd
                    endif
                    set RegistryErrorIndex = $($R0)
                    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
                        goto UpdateCPListEnd
                    endif
                    CloseRegKey $(KeyCp)
                else
                    Debug-Output "UpdateCPList:error creating key"
                endif
            endif
        endforlistdo
        CloseRegKey $(KeyRasManPPP)
    else
        Debug-Output "UpdateCPList:error opening RASMAN\PPP key"
        goto UpdateCPListEnd
    endif
    set Status = STATUS_SUCCESSFUL
UpdateCPListEnd =+
    Debug-Output "UpdateCPList: exit"
    return $(Status)
[UpdateSoftwareType]
    Debug-Output "UpdateSoftwareType: entry"
    set ProductKeyName = $(!NTN_SoftwareBase)"\Microsoft\RAS\CurrentVersion"
    set Status = STATUS_FAILED
    set KeyCurrentVersion = ""
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(!MAXIMUM_ALLOWED)
KeyCurrentVersion
    ifstr $(KeyCurrentVersion) == ""
       Debug-Output "OEMNSVRA.INF:UpdateSoftwareType: couldn't open "$
```

```
(ProductKeyName)
       goto UpdateSoftwareTypeEnd
    endif
    SetRegValue $(KeyCurrentVersion) {SoftwareType, 0, $(!REG_VT_SZ),
"service" }
    CloseRegKey $(KeyCurrentVersion)
    set Status = STATUS_SUCCESSFUL
UpdateSoftwareTypeEnd=+
    Debug-Output "UpdateSoftwareType: exit"
    return $(Status)
[UpdateLLInterface]
    Debug-Output "UpdateLLInterface: entry"
    set Status = STATUS_FAILED
    set KeyNull = ""
    set LLInterface = "\Device\RASARP"
    set IPLinkageKeyName = $(!NTN_ServiceBase)"\TCPIP\Linkage"
    OpenRegKey $(!REG_H_LOCAL) "" $(IPLinkageKeyName) $(!MAXIMUM_ALLOWED)
KeyIpLinkage
    ifstr $(KeyIpLinkage) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF:UpdateLLInterface: couldn't open IP Linkage
key"
       goto UpdateLLInterfaceEnd
    endif
    set NdisWanList = {}
    GetRegValue    $(KeyIpLinkage) "Bind" TmpList
    ForListDo *($(TmpList),4)
        Split-String $($), "\", BindList
        QueryListSize ListSize $(BindList)
        set ServiceName = *($(BindList), $(ListSize))
        LibraryProcedure Result $(!LIBHANDLE) SetupStrncmp $(ServiceName)
"NdisWan" 7
        ifint $(Result) == 0
            set NdisWanList = >($(NdisWanList), $(ServiceName))
        endif
    EndForListDo
    CloseRegKey $(KeyIpLinkage)
    QueryListSize ListSize $(NdisWanList)
    ifint $(ListSize) >= 1
      ForListDo $(NdisWanList)
        set ServiceKeyName = $(!NTN_ServiceBase)"\"$($)"\Parameters\TCPIP"
          OpenRegKey $(!REG_H_LOCAL) "" $(ServiceKeyName) $(!MAXIMUM_ALLOWED)
KeyNdisWan
          ifstr $(KeyNdisWan) != $(KeyNull)
                SetRegValue $(KeyNdisWan) {IPAddress, 0, +
                              $(!REG_VT_MULTI_SZ), {"0.0.0.0"}}
                SetRegValue $(KeyNdisWan) {SubnetMask, 0, +
                              $(!REG_VT_MULTI_SZ), {"0.0.0.0"}}
                SetRegValue $(KeyNdisWan) {DefaultGateway, 0, +
                              $(!REG_VT_MULTI_SZ), {""}}
                SetRegValue $(KeyNdisWan) {LLInterface, 0,+
                                           $(!REG_VT_SZ), $(LLInterface)}
                SetRegValue $(KeyNdisWan) {EnableDHCP, 0, +
                                            $(!REG_VT_DWORD), 0}
          else
              CreateRegKey $(!REG_H_LOCAL) +
                           {$(ServiceKeyName),0,GenericClass} "" +
                                $(!MAXIMUM_ALLOWED) "" KeyNdisWan
              ifstr $(KeyNdisWan) != $(KeyNull)
                  SetRegValue $(KeyNdisWan) {IPAddress, 0, +
                                       $(!REG_VT_MULTI_SZ), {"0.0.0.0"}}
                  SetRegValue $(KeyNdisWan) {SubnetMask, 0, +
                                $(!REG_VT_MULTI_SZ), {"0.0.0.0"}}
                  SetRegValue $(KeyNdisWan) {DefaultGateway, 0, +
                                $(!REG_VT_MULTI_SZ), {""}}
```

```
                    SetRegValue $(KeyNdisWan) {LLInterface, 0,+
                                        $(!REG_VT_SZ), $(LLInterface)}
                    SetRegValue $(KeyNdisWan) {EnableDHCP, 0, +
                                        $(!REG_VT_DWORD), 0}
                    SetRegValue $(KeyNdisWan) {UseZeroBroadcast, 0, +
                                        $(!REG_VT_DWORD), 0}
          else
                  Debug-Output "OEMNSVRA.INF:UpdateLLInterface: couldn't create
NdisWan params-tcpip key"
               goto UpdateLLInterfaceEnd
          endif
        endif
          CloseRegKey $(KeyNdisWan)
        EndForListDo
    endif
    set Status = STATUS_SUCCESSFUL
UpdateLLInterfaceEnd =+
    Debug-Output "UpdateLLInterface: exit"
    return $(Status)
[SetRasArpBindValueFromTcpIP]
    Debug-Output "SetRasArpBindValueFromTcpIP: entry"
    set Status = STATUS_FAILED
    set KeyNull = ""
    set IPLinkageKeyName = $(!NTN_ServiceBase)"\TCPIP\Linkage"
    OpenRegKey $(!REG_H_LOCAL) "" $(IPLinkageKeyName) $(!MAXIMUM_ALLOWED)
KeyIpLinkage
    ifstr $(KeyIpLinkage) == $(KeyNull)
       Debug-Output "OEMNSVRA.INF:SetRasArpBindValueFromTcpIP: couldn't open IP
Linkage key"
       goto SetRasArpBindValueFromTcpIPEnd
    endif
    set NdisWanList = {}
    GetRegValue $(KeyIpLinkage) "Bind" TmpList
    ForListDo *($(TmpList),4)
        Split-String $($), "\", BindList
        QueryListSize ListSize $(BindList)
        set ServiceName = *($(BindList), $(ListSize))
        LibraryProcedure Result $(!LIBHANDLE) SetupStrncmp $(ServiceName)
"NdisWan" 7
        ifint $(Result) == 0
            set NdisWanList = >($(NdisWanList), $($))
        endif
    EndForListDo
    set RasArpLinkageKeyName = $(!NTN_ServiceBase)"\RASARP\Linkage"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasArpLinkageKeyName) $(!MAXIMUM_ALLOWED)
KeyRasArpLinkage
    ifstr $(KeyRasArpLinkage) != $(KeyNull)
        SetRegValue $(KeyRasArpLinkage) {Bind, 0, $(!REG_VT_MULTI_SZ), $
(NdisWanList)}
        CloseRegKey $(KeyRasArpLinkage)
    endif
    CloseRegKey $(KeyIpLinkage)
    set Status = STATUS_SUCCESSFUL
SetRasArpBindValueFromTcpIPEnd=+
    Debug-Output "SetRasArpBindValueFromTcpIP: entry"
    return $(Status)
[UpdateNetGroupDependency]
    Set  Status = STATUS_SUCCESSFUL
    set  KeyNull = ""
    set  fNetbeuiInstalled  = $($0)
    set  fTcpIpInstalled    = $($1)
    set  fIpxInstalled      = $($2)
    Debug-Output "UpdateNetGroupDependency entry"
    set ServiceList =  {}
```

```
    ifstr(i) $(fNetbeuiInstalled) == TRUE
        set ServiceList = >($(ServiceList), "nbf")
    endif
    ifstr(i) $(fTcpIpInstalled) == TRUE
        set ServiceList = >($(ServiceList), "tcpip")
    endif
    ifstr(i) $(fIpxInstalled) == TRUE
        set ServiceList = >($(ServiceList), "nwlnkipx")
    endif
    ForListDo $(ServiceList)
        set Service = $($)
        Debug-Output "OEMNSVRA.INF: Changing group dependency of "$(Service)
        OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\"$(Service) +
                   $(!MAXIMUM_ALLOWED) KeyService
        ifstr $(KeyService) != $(KeyNull)
            set newGroupList = {"+NDIS", "+NDISWAN"}
            GetRegValue    $(KeyService) "DependOnGroup" GrpList
            ifint $(RegLastError) == 0
                Debug-Output "UpdateNetGroupDependency: old group List "*($
(GrpList), 4)
                ForListDo *($(GrpList),4)
                    ifstr(i) $($) != "NDIS"
                        ifstr(i) $($) != "NDISWAN"
                            set newGroup = "+"$($)
                            set newGroupList = >($(newGroupList), $(newGroup))
                        endif
                    endif
                EndForListDo
            endif
            GetRegValue $(KeyService) "DependOnService" ServiceList
            ifint $(RegLastError) == 0
                Debug-Output "UpdateNetGroupDependency: old service List "*($
(ServiceList), 4)
                ForListDo *($(ServiceList),4)
                    set newGroupList = >($(newGroupList), $($))
                EndForListDo
            endif
            Debug-Output "OEMNSVRA.INF: UpdateNetGroupDependency: new depend
List "$(newGroupList)
            LibraryProcedure Result, $(!LIBHANDLE), SetupChangeServiceConfig, $
(Service) $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE),
"", "", $(newGroupList), "", "", ""
            CloseRegKey $(KeyService)
        else
            Debug-Output "UpdateNetGroupDependency: failed to open service
linkage key"$(Service)
        endif
    EndForListDo
    Debug-Output "UpdateNetGroupDependency exit"
    return $(Status)
[RemoveNetGroupDependency]
    Set  Status = STATUS_SUCCESSFUL
    set  KeyNull = ""
    set  fNetbeuiInstalled  = $($0)
    set  fTcpIpInstalled    = $($1)
    set  fIpxInstalled      = $($2)
    Debug-Output "UpdateNetGroupDependency entry"
    set ServiceList =  {"nbf", "tcpip", "nwlnkipx"}
    ForListDo $(ServiceList)
        set Service = $($)
        Debug-Output "OEMNSVRA.INF: Changing group dependency of "$(Service)
        OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\"$(Service) +
                   $(!MAXIMUM_ALLOWED) KeyService
        ifstr $(KeyService) != $(KeyNull)
```

```
          set DeleteFlag = ""
          GetRegValue $(KeyService),"DeleteFlag", DeleteFlagInfo
          set DeleteFlag = *($(DeleteFlagInfo), 4)
          ifint $(DeleteFlag) != 1
            set newGroupList = {}
            GetRegValue    $(KeyService) "DependOnGroup" GrpList
            ifint $(RegLastError) == 0
                Debug-Output "UpdateNetGroupDependency: old List "*($(GrpList),
4)
                ForListDo *($(GrpList),4)
                    ifstr(i) $($) != "NDISWAN"
                        set newGroup = "+"$($)
                        set newGroupList = >($(newGroupList), $(newGroup))
                    endif
                EndForListDo
            endif
            GetRegValue $(KeyService) "DependOnService" ServiceList
            ifint $(RegLastError) == 0
                Debug-Output "UpdateNetGroupDependency: old service List "*($
(ServiceList), 4)
                ForListDo *($(ServiceList),4)
                    set newGroupList = >($(newGroupList), $($))
                EndForListDo
            endif
            Debug-Output "OEMNSVRA.INF: UpdateNetGroupDependency: new depend
List "$(newGroupList)
            LibraryProcedure Result, $(!LIBHANDLE), SetupChangeServiceConfig, $
(Service) $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE),
"", "", $(newGroupList), "", "", ""
            CloseRegKey $(KeyService)
          endif
        else
            Debug-Output "UpdateNetGroupDependency: failed to open service
linkage key"$(Service)
        endif
    EndForListDo
    Debug-Output "UpdateNetGroupDependency exit"
    return $(Status)
[UpdateIPRouterInfo]
    set EnableRouter = $($0)
    set KeyNull      = ""
    Debug-Output "UpdateIPRouterInfo entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\TCPIP\parameters" $(!
MAXIMUM_ALLOWED) KeyIpParams
    ifstr $(KeyIpParams) != $(KeyNull)
        SetRegValue $(KeyIpParams) {IPEnableRouter, 0,+
                                    $(!REG_VT_DWORD), $(EnableRouter)}
        CloseRegKey $(KeyIpParams)
    endif
    Debug-Output "UpdateIPRouterInfo exit"
    return STATUS_SUCCESSFUL
[UpdateIPXRouterInfo]
    set Status = STATUS_FAILED
    set EnableRouter = $($0)
    set KeyNull      = ""
    Debug-Output "UpdateIPXRouterInfo entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\NWLNKIPX\NetConfig" $(!
MAXIMUM_ALLOWED) KeyIpxNetConfig
    ifstr $(KeyIpxNetConfig) != $(KeyNull)
        EnumRegKey $(KeyIpxNetConfig) DriverList
        ForListDo $(DriverList)
           set DriverName = *($($),1)
           OpenRegKey $(KeyIpxNetConfig) "" $(DriverName) $(!MAXIMUM_ALLOWED)
Driver
```

```
            ifstr $(Driver) == $(KeyNull)
                Debug-Output "UpdateIPXRouterInfo: could not open key ipx\
netconfig\"$(DriverName)
                CloseRegKey $(KeyIpxNetConfig)
                return $(Status)
            endif
            SetRegValue $(Driver) {EnableWanRouter, 0,+
                                    $(!REG_VT_DWORD), $(EnableRouter)}
            CloseRegKey $(Driver)
        EndForListDo
        CloseRegKey $(KeyIpxNetConfig)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "UpdateIPXRouterInfo: error opening ipx\netconfig key"
    endif
    Debug-Output "UpdateIPXRouterInfo exit"
    return $(Status)
[IsNetcardInstalled]
    Set Status                = STATUS_FAILED
    Set fNetcardInstalled     = FALSE
    Set KeyNull               = ""
    set ProductRASASYMACName     = "AsyncMac"
    set NetworkCardKey = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
NetworkCardKey
    Ifstr(i) $(NetworkCardKey) != $(KeyNull)
        set NetcardsList = {}
        EnumRegKey $(NetworkCardKey) NetcardsList
        Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
           Debug-Output "IsNetcardInstalled: EnumRegKey failed."
           goto IsNetcardInstalledEnd
        endif
        ForListDo $(NetcardsList)
           set KeyName = *($($),1)
           set Card = $(KeyNull)
           OpenRegKey $(NetworkCardKey) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
           ifstr $(Card) == $(KeyNull)
               Debug-Output "IsNetcardInstalled: could not open netcard key "$
(KeyName)
           else
               GetRegValue $(Card), "ProductName" ProductNameInfo
               Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                  Debug-Output "IsNetcardInstalled: ProductName not found."
               else
                  set CardProductName = *($(ProductNameInfo), 4)
                  Debug-Output "IsNetcardInstalled: ProductName. "$
(CardProductName)
                      ifstr(i) $(CardProductName) != $(!ProductNDISWANName)
                       ifstr(i) $(CardProductName) != $(!
ProductNDISWANDIALINName)
                        ifstr(i) $(CardProductName) != $(!
ProductNDISWANDIALOUTName)
                         ifstr(i) $(CardProductName) != $(!
ProductNDISWANDIALINIPName)
                          ifstr(i) $(CardProductName) != $(!
ProductNDISWANDIALOUTIPName)
                            ifstr(i) $(CardProductName) != $(!
ProductNDISWANDIALINOUTIPXName)
                        ifstr(i) $(CardProductName) != $(ProductRASASYMACName)
                         ifstr(i) $(CardProductName) != $(!ProductPCIMACName)
                           GetRegValue $(Card), "Manufacturer" Manufacturer
                           Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
                             ifstr(i) *($(Manufacturer), 4) != "Digiboard"
                               Debug-Output "found installed netcard"
```

```
                                set fNetcardInstalled = TRUE
                                set Status = STATUS_SUCCESSFUL
                                CloseRegKey $(Card)
                                goto IsNetcardInstalledEnd
                              endif
                            endif
                          endif
                        endif
                      endif
                    endif
                  endif
                endif
                CloseRegKey $(Card)
            endif
        EndForListDo
    else
        Debug-Output "IsNetcardInstalled: failed to open "$(!NetworkCardKeyName)
    endif
IsNetcardInstalledEnd =+
    ifstr(i) $(NetworkCardKey) != $(KeyNull)
        CloseRegKey $(NetworkCardKey)
    endif
    Return $(Status) $(fNetcardInstalled)
[IsNetworkConfigured]
    set WkstaKeyName   = $(!NTN_ServiceBase)"\LanmanWorkstation"
    set SrvrKeyName    = $(!NTN_ServiceBase)"\LanmanServer"
    Set KeyNull        =    ""
    Set status = STATUS_FAILED
    Set WkstaConfigured  = FALSE
    Set SrvrConfigured   = FALSE
    Debug-Output "IsNetworkConfigured WkstaKeyName "$(WkstaKeyName)
    set KeyProduct = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(WkstaKeyName) $(!MAXIMUM_ALLOWED) KeyProduct
    Ifstr(i) $(KeyProduct) != $(KeyNull)
        GetRegValue $(KeyProduct),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set WkstaConfigured = TRUE
        endif
        CloseRegKey $(KeyProduct)
    endif
    Debug-Output "IsNetworkConfigured SrvrKeyName "$(SrvrKeyName)
    set KeyProduct = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(SrvrKeyName) $(!MAXIMUM_ALLOWED) KeyProduct
    Ifstr(i) $(KeyProduct) != $(KeyNull)
        GetRegValue $(KeyProduct),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set SrvrConfigured = TRUE
        endif
        CloseRegKey $(KeyProduct)
    endif
    set status = STATUS_SUCCESSFUL
    Debug-Output "IsNetworkConfigured exit"
    Return $(status) $(WkstaConfigured) $(SrvrConfigured)
[QueryInstalledProtocols]
    Debug-Output "QueryInstalledProtocols entry "
    set NbfKeyName        = $(!NTN_ServiceBase)"\Nbf"
    set TcpIpKeyName      = $(!NTN_ServiceBase)"\TcpIp"
    set IpxKeyName        = $(!NTN_ServiceBase)"\NWLNKIPX"
    set AppleTalkKeyName  = $(!NTN_ServiceBase)"\AppleTalk"
```

```
    Set KeyNull        =    ""
    Set status = STATUS_FAILED
    Set NbfInstalled     = FALSE
    Set TcpIpInstalled   = FALSE
    Set IpxInstalled     = FALSE
    Set AppleTalkInstalled = FALSE
    set KeyService = $(KeyNull)
    set DeleteFlag = 0
    set DeleteFlagInfo = {}
    OpenRegKey $(!REG_H_LOCAL) "" $(NbfKeyName) $(!MAXIMUM_ALLOWED) KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        GetRegValue $(KeyService),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set NbfInstalled = TRUE
        endif
        CloseRegKey $(KeyService)
    endif
    set KeyService = $(KeyNull)
    set DeleteFlag = 0
    set DeleteFlagInfo = {}
    OpenRegKey $(!REG_H_LOCAL) "" $(TcpIpKeyName) $(!MAXIMUM_ALLOWED) KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        GetRegValue $(KeyService),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set TcpIpInstalled = TRUE
        endif
        CloseRegKey $(KeyService)
    endif
    set KeyService = $(KeyNull)
    set DeleteFlag = 0
    set DeleteFlagInfo = {}
    OpenRegKey $(!REG_H_LOCAL) "" $(IpxKeyName) $(!MAXIMUM_ALLOWED) KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        GetRegValue $(KeyService),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set IpxInstalled = TRUE
        endif
        CloseRegKey $(KeyService)
    endif
    set KeyService = $(KeyNull)
    set DeleteFlag = 0
    set DeleteFlagInfo = {}
    OpenRegKey $(!REG_H_LOCAL) "" $(AppleTalkKeyName) $(!MAXIMUM_ALLOWED)
KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        GetRegValue $(KeyService),"DeleteFlag", DeleteFlagInfo
        set DeleteFlag = *($(DeleteFlagInfo), 4)
        ifint $(DeleteFlag) != 1
            set AppleTalkInstalled = TRUE
        endif
        CloseRegKey $(KeyService)
    endif
    set status = STATUS_SUCCESSFUL
    Debug-Output "QueryInstalledProtocols exit"
    Return $(status) $(NbfInstalled) $(TcpIpInstalled) $(IpxInstalled) +
                     $(AppleTalkInstalled)
[ProtocolInfo]
    NETBEUI              = "OEMNXPNB.INF", "NBF", 1
    TCPIP                = "OEMNXPTC.INF", "TC", 2
    IPX                  = "OEMNXPIP.INF", "NWLNKIPX", 3
[InstallProtocol]
```

```
    Set Status                = STATUS_FAILED
    Set KeyNull               = ""
    set Protocol              = $($0)
    Debug-Output "OEMNSVRA.INF: Label: InstallProtocol"
    Set ThisInfName = #(ProtocolInfo, $(Protocol), 1)
    Set InfOption = #(ProtocolInfo, $(Protocol), 2)
    Set BbIndex = #(ProtocolInfo, $(Protocol), 3)
    Debug-Output "OEMNSVRA.INF: Installing "$(ThisInfName)
    Ifint $(BbIndex) != 0
        read-syms Billboard$(BbIndex)$(!STF_LANGUAGE)
        Shell "subroutn.inf" PushBillboard NETSTATUSDLG $(Status)
        Set BillboardVisible = 1
    Else-ifint $(BillboardVisible) != 0
        Shell "subroutn.inf" PopBillboard
        Set BillboardVisible = 0
    Endif
    set AddCopy  = YES
    set DoCopy   = YES
    set DoConfig = YES
    set InvokedByRas = YES
    set SaveNTN_InstallMode = $(!NTN_InstallMode)
    set SaveSTF_INSTALL_MODE = $(!STF_INSTALL_MODE)
    set !NTN_InstallMode = install
    set !STF_INSTALL_MODE = EXPRESS
    Shell $(ThisInfName) InstallOption $(!STF_LANGUAGE) $(InfOption) $(!
STF_SRCDIR) $(AddCopy) $(DoCopy) $(DoConfig) $(InvokedByRas)
    set !NTN_InstallMode = $(SaveNTN_InstallMode)
    set !STF_INSTALL_MODE = $(SaveSTF_INSTALL_MODE)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "OEMNSVRA.INF: INF "$(ThisInfName)" SHELL ERROR!"
        Goto InstallProtocolEscape
    Endif
    Set InstallProtocolStatus = $($R0)
    Ifstr(i) $(InstallProtocolStatus) != STATUS_SUCCESSFUL
        Ifstr(i) $(InstallProtocolStatus) != STATUS_USERCANCEL
            Debug-Output "OEMNSVRA.INF: INF "$(ThisInfName)" returned "$
(InstallProtocolStatus)
            Goto InstallProtocolEscape
        Endif
    Endif
InstallProtocolEscape = +
    set Status = $(InstallProtocolStatus)
    return $(Status)
[UpgradeSelectedProtocols]
    Debug-Output "UpgradeSelectedProtocols entry."
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set RasKeyName =  $(!NTN_SoftwareBase)"\Microsoft\Ras"
    set RasProtocolName =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols"
    set RasServiceParmName =  $(!NTN_ServiceBase)"\Remoteaccess\Parameters"
    set AsyMacServiceParmName = $(!NTN_ServiceBase)"\AsyncMac\Parameters"
    set RasManPPPName = $(!NTN_ServiceBase)"\RasMan\PPP"
    set GtwyEnabled = 1
    OpenRegKey $(!REG_H_LOCAL) "" $(RasServiceParmName) $(!MAXIMUM_ALLOWED)
KeyParam
    ifstr $(KeyParam) != $(KeyNull)
        GetRegValue $(KeyParam), "NetBiosGatewayEnabled", GtwyEnabledList
        ifint $(RegLastError) == 0
            set GtwyEnabled = *($(GtwyEnabledList), 4)
        endif
        CloseRegKey $(KeyParam)
    else
        Debug-Output "UpgradeSelectedProtocols: error opening RemoteAccess\
Parameters key"
```

```
    endif
    set DialinNBF    = 0
    set DialinIP     = 0
    set DialoutNBF   = 0
    set DialoutIP    = 0
    set DialinoutIPX = 0
    OpenRegKey $(!REG_H_LOCAL) "" $(AsyMacServiceParmName) $(!MAXIMUM_ALLOWED)
KeyParam
    ifstr $(KeyParam) != $(KeyNull)
        GetRegValue $(KeyParam), "DialinNBF", DialinNBFList
        ifint $(RegLastError) == 0
            set DialinNBF = *($(DialinNBFList), 4)
        endif
        GetRegValue $(KeyParam), "DialinIP", DialinIPList
        ifint $(RegLastError) == 0
            set DialinIP = *($(DialinIPList), 4)
        endif
        GetRegValue $(KeyParam), "DialoutNBF", DialoutNBFList
        ifint $(RegLastError) == 0
            set DialoutNBF = *($(DialoutNBFList), 4)
        endif
        GetRegValue $(KeyParam), "DialoutIP", DialoutIPList
        ifint $(RegLastError) == 0
            set DialoutIP = *($(DialoutIPList), 4)
        endif
        GetRegValue $(KeyParam), "DialinoutIPX", DialinoutIPXList
        ifint $(RegLastError) == 0
            set DialinoutIPX = *($(DialinoutIPXList), 4)
        endif
        CloseRegKey $(KeyParam)
    else
        Debug-Output "UpgradeSelectedProtocols: error opening AsyncMac\
Parameters key"
    endif
    set fDownLevel = FALSE
    OpenRegKey $(!REG_H_LOCAL) "" $(RasProtocolName) $(!MAXIMUM_ALLOWED)
RasProtocolKey
    ifstr $(RasProtocolKey) == $(KeyNull)
        set fDownLevel = TRUE
        OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) $(!MAXIMUM_ALLOWED) RasKey
        CreateRegKey $(RasKey) {"Protocols",0,GenericClass} "" $(!
MAXIMUM_ALLOWED) "" RasProtocolKey
        OpenRegKey $(RasKey) "" "Protocols" $(!MAXIMUM_ALLOWED) RasProtocolKey
        CloseRegKey $(RasKey)
    endif
    ifstr $(RasProtocolKey) != $(KeyNull)
        set MultilinkList = {}
        GetRegValue $(RasProtocolKey), "Multilink", MultiLinkList
        ifint $(RegLastError) != 0
           Debug-Output "Upgradeselectedprotocols: defaulting Multilink to 0"
           SetRegValue $(RasProtocolKey) {Multilink, 0, $(!REG_VT_DWORD), 0}
        endif
        set NetbeuiSelected = 0
        set NetbeuiAllowed  = 0
        set TcpIpSelected = 0
        set TcpIpAllowed = 0
        set IpxSelected = 0
        set IpxAllowed = 0
        ifint $(DialoutNBF) != 0
            set NetbeuiSelected = 1
        endif
        ifint $(GtwyEnabled) == 1
          ifint $(DialinNBF) != 0
            set NetbeuiAllowed = 1
```

```
          endif
        else
          ifint $(DialoutNBF) != 0
            set NetbeuiAllowed = 1
          endif
        endif
        ifstr(i) $(fDownLevel) != TRUE
          ifint $(DialoutIP) != 0
            set TcpIpSelected = 1
          endif
          ifint $(DialinIP) != 0
            set TcpIpAllowed = 1
          endif
          ifint $(DialinoutIPX) != 0
            set IpxSelected = 1
            set IpxAllowed  = 1
          endif
        endif
        set ItemList = {}
        GetRegValue $(RasProtocolKey), "fNetbeuiSelected", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fNetbeuiSelected, 0,+
                                          $(!REG_VT_DWORD),$(NetbeuiSelected)}
        endif
        GetRegValue $(RasProtocolKey), "fTcpIpSelected", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fTcpIpSelected, 0,+
                                          $(!REG_VT_DWORD),$(TcpIpSelected)}
        endif
        GetRegValue $(RasProtocolKey), "fIpxSelected", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fIpxSelected, 0,+
                                          $(!REG_VT_DWORD),$(IpxSelected)}
        endif
        GetRegValue $(RasProtocolKey), "fNetbeuiAllowed", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fNetbeuiAllowed, 0,+
                                          $(!REG_VT_DWORD),$(NetbeuiAllowed)}
        endif
        GetRegValue $(RasProtocolKey), "fTcpIpAllowed", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fTcpIpAllowed, 0,+
                                          $(!REG_VT_DWORD),$(TcpIpAllowed)}
        endif
        GetRegValue $(RasProtocolKey), "fIpxAllowed", ItemList
        ifint $(RegLastError) != 0
           SetRegValue $(RasProtocolKey) {fIpxAllowed, 0,+
                                          $(!REG_VT_DWORD),$(IpxAllowed)}
        endif
        OpenRegKey $(RasProtocolKey) "" "NBF" $(!MAXIMUM_ALLOWED) NbfKey
        ifstr $(NbfKey) == $(KeyNull)
            CreateRegKey $(RasProtocolKey) {"NBF",0,GenericClass} "" $(!
MAXIMUM_ALLOWED) "" NbfKey
            SetRegValue $(NbfKey) {NetBiosGatewayEnabled, 0,+
                                   $(!REG_VT_DWORD),$(GtwyEnabled)}
            CloseRegKey $(NbfKey)
        endif
        ifstr(i) $(fDownLevel) != TRUE
            OpenRegKey $(RasProtocolKey) "" "IP" $(!MAXIMUM_ALLOWED) IpKey
            ifstr $(IpKey) != $(KeyNull)
                GetRegValue $(IpKey), "AllowNetworkAccess", ItemList
                ifint $(RegLastError) != 0
                   SetRegValue $(IpKey) {AllowNetworkAccess, 0,+
                                         $(!REG_VT_DWORD),$(GtwyEnabled)}
```

```
                endif
                CloseRegKey $(IpKey)
            endif
            OpenRegKey $(RasProtocolKey) "" "IPX" $(!MAXIMUM_ALLOWED) IpxKey
            ifstr $(IpxKey) != $(KeyNull)
                GetRegValue $(IpxKey), "AllowNetworkAccess", ItemList
                ifint $(RegLastError) != 0
                SetRegValue $(IpxKey) {AllowNetworkAccess, 0,+
                                        $(!REG_VT_DWORD),$(GtwyEnabled)}
                endif
                CloseRegKey $(IpxKey)
            endif
        endif
        set ForceEncryptedPassword = 2
        set ForceEncryptedData = 0
        ifstr(i) $(fDownLevel) != TRUE
            GetRegValue $(RasProtocolKey), "ForceEncryptedData", DataList
            ifint $(RegLastError) != 0
                GetRegValue $(RasProtocolKey), "ForceEncryptedPassword",
PasswdList
                ifint $(RegLastError) == 0
                   set ForceEncryptedPassword = *($(PasswdList), 4)
                   ifint $(ForceEncryptedPassword) == 1
                       set ForceEncryptedPassword = 2
                   endif
                endif
            else
                goto  UpgradeSelectedProtocolsEnd
            endif
        endif
        SetRegValue $(RasProtocolKey) {ForceEncryptedPassword, 0,+
                                $(!REG_VT_DWORD), $(ForceEncryptedPassword)}
        SetRegValue $(RasProtocolKey) {ForceEncryptedData, 0,+
                                $(!REG_VT_DWORD), $(ForceEncryptedData)}
UpgradeSelectedProtocolsEnd =+
        CloseRegKey $(RasProtocolKey)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "UpgradeSelectedProtocols: error opening Ras\protocols key"
    endif
    Debug-Output "UpgradeSelectedProtocols exit."
    return $(Status)
[SaveSelectedProtocols]
    Debug-Output "SaveSelectedProtocols entry."
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    ifstr(i) $($0) == TRUE
        set NetbeuiSelected = 1
    else
        set NetbeuiSelected = 0
    endif
    ifstr(i) $($1) == TRUE
        set TcpIpSelected = 1
    else
        set TcpIpSelected = 0
    endif
    ifstr(i) $($2) == TRUE
        set IpxSelected = 1
    else
        set IpxSelected = 0
    endif
    ifstr(i) $($3) == TRUE
        set NetbeuiAllowed = 1
    else
```

```
        set NetbeuiAllowed = 0
    endif
    ifstr(i) $($4) == TRUE
        set TcpIpAllowed = 1
    else
        set TcpIpAllowed = 0
    endif
    ifstr(i) $($5) == TRUE
        set IpxAllowed = 1
    else
        set IpxAllowed = 0
    endif
    set RasKeyName =  $(!NTN_SoftwareBase)"\Microsoft\Ras"
    set RasProtocolName =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasProtocolName) $(!MAXIMUM_ALLOWED)
RasProtocolKey
    ifstr $(RasProtocolKey) == $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) $(!MAXIMUM_ALLOWED) RasKey
        CreateRegKey $(RasKey) {"Protocols",0,GenericClass} "" $(!
MAXIMUM_ALLOWED) "" RasProtocolKey
        OpenRegKey $(RasKey) "" "Protocols" $(!MAXIMUM_ALLOWED) RasProtocolKey
        CloseRegKey $(RasKey)
    endif
    ifstr $(RasProtocolKey) != $(KeyNull)
        SetRegValue $(RasProtocolKey) {fNetbeuiSelected, 0,+
                                       $(!REG_VT_DWORD),$(NetbeuiSelected)}
        SetRegValue $(RasProtocolKey) {fTcpIpSelected, 0,+
                                       $(!REG_VT_DWORD),$(TcpIpSelected)}
        SetRegValue $(RasProtocolKey) {fIpxSelected, 0,+
                                       $(!REG_VT_DWORD),$(IpxSelected)}
        SetRegValue $(RasProtocolKey) {fNetbeuiAllowed, 0,+
                                       $(!REG_VT_DWORD),$(NetbeuiAllowed)}
        SetRegValue $(RasProtocolKey) {fTcpIpAllowed, 0,+
                                       $(!REG_VT_DWORD),$(TcpIpAllowed)}
        SetRegValue $(RasProtocolKey) {fIpxAllowed, 0,+
                                       $(!REG_VT_DWORD),$(IpxAllowed)}
        CloseRegKey $(RasProtocolKey)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "error opening Ras\protocols key"
    endif
    Debug-Output "SaveSelectedProtocols exit."
    return $(Status)
[QuerySelectedProtocols]
    Debug-Output "QuerySelectedProtocols: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set fNetbeuiSelected = FALSE
    set fTcpIpSelected   = FALSE
    set fIpxSelected     = FALSE
    set fNetbeuiAllowed  = FALSE
    set fTcpIpAllowed    = FALSE
    set fIpxAllowed      = FALSE
    set NetbeuiSelected  = {}
    set TcpIpSelected    = {}
    set IpxSelected      = {}
    set NetbeuiAllowed  = {}
    set TcpIpAllowed    = {}
    set IpxAllowed      = {}
    set RasKeyName =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) $(!MAXIMUM_ALLOWED) RasKey
    ifstr $(RasKey) != $(KeyNull)
        GetRegValue $(RasKey), "fNetbeuiSelected", NetbeuiSelected
        ifint *($(NetbeuiSelected), 4) == 1
```

```
            set fNetbeuiSelected = TRUE
        else
            set fNetbeuiSelected = FALSE
        endif
        GetRegValue $(RasKey), "fTcpIpSelected", TcpIpSelected
        ifint *($(TcpIpSelected), 4) == 1
            set fTcpIpSelected = TRUE
        else
            set fTcpIpSelected = FALSE
        endif
        GetRegValue $(RasKey), "fIpxSelected", IpxSelected
        ifint *($(IpxSelected), 4) == 1
            set fIpxSelected = TRUE
        else
            set fIpxSelected = FALSE
        endif
        GetRegValue $(RasKey), "fNetbeuiAllowed", NetbeuiAllowed
        ifint *($(NetbeuiAllowed), 4) == 1
            set fNetbeuiAllowed = TRUE
        else
            set fNetbeuiAllowed = FALSE
        endif
        GetRegValue $(RasKey), "fTcpIpAllowed", TcpIpAllowed
        ifint *($(TcpIpAllowed), 4) == 1
            set fTcpIpAllowed = TRUE
        else
            set fTcpIpAllowed = FALSE
        endif
        GetRegValue $(RasKey), "fIpxAllowed", IpxAllowed
        ifint *($(IpxAllowed), 4) == 1
            set fIpxAllowed = TRUE
        else
            set fIpxAllowed = FALSE
        endif
        CloseRegKey $(RasKey)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "error opening Ras\protocols key"
    endif
    Debug-Output "QuerySelectedProtocols: exit"
    return $(Status) $(fNetbeuiSelected) $(fTcpIpSelected) $(fIpxSelected) +
                     $(fNetbeuiAllowed) $(fTcpIpAllowed) $(fIpxAllowed)
[UpdateSelectedProtocols]
    Debug-Output "UpdateSelectedProtocols: entry"
    set Status   = STATUS_FAILED
    Shell "" QuerySelectedProtocols
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "error shelling QuerySelectedProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by QuerySelectedProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    set fNetbeuiSelected = $($R1)
    set fTcpIpSelected   = $($R2)
    set fIpxSelected     = $($R3)
    set fNetbeuiAllowed = $($R4)
    set fTcpIpAllowed   = $($R5)
    set fIpxAllowed     = $($R6)
    Debug-Output "UpdateSelectedProtocols: Selected protocols NBF IP IPX "$
(fNetbeuiSelected)" "$(fTcpIpSelected)" "$(fIpxSelected)
    shell "" QueryInstalledProtocols
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
```

```
        Debug-Output "error shelling QueryInstalledProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by QueryInstalledProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    set fNetbeuiInstalled = $($R1)
    set fTcpIpInstalled   = $($R2)
    set fIpxInstalled     = $($R3)
    Debug-Output "UpdateSelectedProtocols: Installed Protocols NBF IP IPX "$
(fNetbeuiInstalled)" "$(fTcpIpInstalled)" "$(fIpxInstalled)
    set fNetbeuiSelectedNew = $(fNetbeuiSelected)
    set fTcpIpSelectedNew   = $(fTcpIpSelected)
    set fIpxSelectedNew     = $(fIpxSelected)
    set fNetbeuiAllowedNew = $(fNetbeuiAllowed)
    set fTcpIpAllowedNew   = $(fTcpIpAllowed)
    set fIpxAllowedNew     = $(fIpxAllowed)
    ifstr(i) $(fNetbeuiSelectedNew) == TRUE
        set fNetbeuiChosen = TRUE
    else
        set fNetbeuiChosen = $(fNetbeuiAllowedNew)
    endif
    ifstr(i) $(fTcpIpSelectedNew) == TRUE
        set fTcpIpChosen = TRUE
    else
        set fTcpIpChosen = $(fTcpIpAllowedNew)
    endif
    ifstr(i) $(fIpxSelectedNew) == TRUE
        set fIpxChosen = TRUE
    else
        set fIpxChosen = $(fIpxAllowedNew)
    endif
    ifstr(i) $(fNetbeuiInstalled) == FALSE
        set fNetbeuiSelectedNew = FALSE
        set fNetbeuiAllowedNew = FALSE
    endif
    ifstr(i) $(fTcpIpInstalled) == FALSE
        set fTcpIpSelectedNew = FALSE
        set fTcpIpAllowedNew = FALSE
    endif
    ifstr(i) $(fIpxInstalled) == FALSE
        set fIpxSelectedNew = FALSE
        set fIpxAllowedNew = FALSE
    endif
    ifstr(i) $(fNetbeuiSelectedNew) == $(fNetbeuiSelected)
      ifstr(i) $(fTcpIpSelectedNew) == $(fTcpIpSelected)
        ifstr(i) $(fIpxSelectedNew) == $(fIpxSelected)
          ifstr(i) $(fNetbeuiAllowedNew) == $(fNetbeuiAllowed)
            ifstr(i) $(fTcpIpAllowedNew) == $(fTcpIpAllowed)
              ifstr(i) $(fIpxAllowedNew) == $(fIpxAllowed)
                set Status = STATUS_SUCCESSFUL
                Debug-Output "UpdateSelectedProtocols: Nothing changed"
                goto UpdateSelectedProtocolsEnd
              endif
            endif
          endif
        endif
      endif
    endif
    Shell "" SaveSelectedProtocols $(fNetbeuiSelectedNew) $(fTcpIpSelectedNew) +
                                   $(fIpxSelectedNew) $(fNetbeuiAllowedNew) +
                                   $(fTcpIpAllowedNew) $(fIpxAllowedNew)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
```

```
        Debug-Output "error shelling SaveSelectedProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by SaveSelectedProtocols."
        goto UpdateSelectedProtocolsEnd
    endif
    ifstr(i) $(fNetbeuiSelectedNew) == TRUE
        set fNetbeuiChosen = TRUE
    else
        set fNetbeuiChosen = $(fNetbeuiAllowedNew)
    endif
    ifstr(i) $(fTcpIpSelectedNew) == TRUE
        set fTcpIpChosen = TRUE
    else
        set fTcpIpChosen = $(fTcpIpAllowedNew)
    endif
    ifstr(i) $(fIpxSelectedNew) == TRUE
        set fIpxChosen = TRUE
    else
        set fIpxChosen = $(fIpxAllowedNew)
    endif
    Shell "" UpdateNetCardInfo $(fNetbeuiChosen) $(fTcpIpChosen) +
                               $(fIpxChosen)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "error shelling UpdateNetCardInfo."
        goto UpdateSelectedProtocolsEnd
    endif
    Ifstr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output "error returned by UpdateNetCardInfo."
        goto UpdateSelectedProtocolsEnd
    endif
    set Status = STATUS_SUCCESSFUL
    ifstr(i) $(fNetbeuiAllowedNew) == FALSE
        Debug-Output "UpdateSelectedProtocols: Removing RemoteAccess NBF
dependency"
        Shell "" RemoveServiceDependency "RemoteAccess" "NBF"
    endif
    ifstr(i) $(fTcpIpChosen) == FALSE
        Debug-Output "UpdateSelectedProtocols: Removing RasArp service"
        shell "" RemoveRasArpService
    endif
    ifstr(i) $(fIpxAllowedNew) == FALSE
        Debug-Output "UpdateSelectedProtocols: Removing IPX router and SAP"
        shell "" RemoveNwlnkRipService
        shell "" RemoveIsnSapService
        Shell "" RemoveServiceDependency "RemoteAccess" "NWLNKIPX"
        ifstr(i) $($R0) == STATUS_FAILED
            Debug-Output "UpdateSelectedProtocols: error removing service
dependency of RemoteAccess on NWLNKIPX"
        endif
    endif
UpdateSelectedProtocolsEnd = +
    Debug-Output "UpdateSelectedProtocols: exit"
    return $(Status) $(fNetbeuiChosen) $(fTcpIpChosen) $(fIpxChosen)
[UpgradeIsdnInfo]
    Debug-Output "UpgradeIsdnInfo entry."
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set RasKeyName      =  $(!NTN_SoftwareBase)"\Microsoft\Ras"
    set TapiName        =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Tapi Devices"
    set RasIsdnName     =  $(!NTN_SoftwareBase)"\Microsoft\Ras\MEDIA\ISDN"
    set RasManParamName =  $(!NTN_ServiceBase)"\RasMan\Parameters"
    set PcimacName      =  "HARDWARE\DEVICEMAP\TAPI DEVICES\PCIMAC"
```

```
    set KeyIsdn = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(RasIsdnName) $(!MAXIMUM_ALLOWED) KeyIsdn
    ifstr $(KeyIsdn) != $(KeyNull)
        EnumRegKey $(KeyIsdn) PortList
        ifint $(RegLastError) != 0
            Debug-Output "UpgradeIsdnInfo: error enumerating MEDIAS\ISDN key."
            CloseRegKey $(KeyIsdn)
            set Status = STATUS_SUCCESSFUL
            goto UpgradeIsdnInfoEnd
        endif
        QueryListSize IsdnPorts $(PortList)
        ifint $(IsdnPorts) == 0
            set Status = STATUS_SUCCESSFUL
            CloseRegKey $(KeyIsdn)
            goto UpgradeIsdnInfoEnd
        endif
        set MediaType = "ISDN"
        set Addresses = {}
        OpenRegKey $(!REG_H_LOCAL) "" $(PcimacName) +
                                              $(!MAXIMUM_ALLOWED) KeyPcimac
        ifstr $(KeyPcimac) != $(KeyNull)
            GetRegValue $(KeyPcimac), "Address", AddressList
            ifint $(RegLastError) == 0
                set Addresses = *($(AddressList), 4)
            endif
            GetRegValue $(KeyPcimac), "Media Type", MediaValue
            ifint $(RegLastError) == 0
                set MediaType = *($(MediaValue), 4)
            endif
            CloseRegKey $(KeyPcimac)
        else
            Debug-Output "UpgradeIsdnInfo: error opening DEVICEMAP\TAPI DEVICES\
PCIMAC key."
            CloseRegKey $(KeyIsdn)
            goto UpgradeIsdnInfoEnd
        endif
        set AddressList = {}
        set NameList    = {}
        set UsageList   = {}
        set index = 1
        forlistdo $(PortList)
           set Address =  *($(Addresses), $(index))
           set-add index = $(index) , 1
           set AddressList = >($(AddressList), $(Address))
           set PortName = *($($), 1)
           set NameList = >($(NameList), Pcimac$(PortName))
           OpenRegKey $(KeyIsdn) "" $(PortName) $(!MAXIMUM_ALLOWED) KeyPort
           ifstr $(KeyPort) != $(KeyNull)
             GetRegValue $(KeyPcimac), "Usage", UsageValue
             ifint $(RegLastError) == 0
                 set Usage = *($(UsageValue), 4)
                 set UsageList = >($(UsageList), $(Usage))
             else
                Debug-Output "UpgradeIsdnInfo: error reading usage "$(PortName)
                set UsageList = >($(UsageList), "Client")
             endif
             CloseRegKey $(KeyPort)
           else
             Debug-Output "UpgradeIsdnInfo: error opening key "$(PortName)
             set UsageList = >($(UsageList), "Client")
           endif
        endforlistdo
        set KeyTapi = $(KeyNull)
        CreateRegKey $(!REG_H_LOCAL) +
```

```
                {$(TapiName),$(NoTitle),GenericClass}  +
                 "" $(!MAXIMUM_ALLOWED) "" KeyTapi
        ifstr $(KeyTapi) != $(KeyNull)
            CreateRegKey $(KeyTapi) +
                 {"Pcimac" ,$(NoTitle),GenericClass} +
                 "" $(!MAXIMUM_ALLOWED) "" KeyPcimac
            ifstr $(KeyPcimac) != $(KeyNull)
                Debug-Output "UpgradeIsdnInfo: MediaType "$(MediaType)
                Debug-Output "UpgradeIsdnInfo: Addresses "$(AddressList)
                Debug-Output "UpgradeIsdnInfo: Names  "$(NameList)
                Debug-Output "UpgradeIsdnInfo: Usage  "$(UsageList)
                SetRegValue $(KeyPcimac) {"Media Type", 0, +
                                          $(!REG_VT_SZ),$(MediaType)}
                SetRegValue $(KeyPcimac) {"Address", 0, +
                                          $(!REG_VT_MULTI_SZ),$(AddressList)}
                SetRegValue $(KeyPcimac) {"Friendly Name", 0, +
                                          $(!REG_VT_MULTI_SZ),$(NameList)}
                SetRegValue $(KeyPcimac) {"Usage", 0, +
                                          $(!REG_VT_MULTI_SZ),$(UsageList)}
                CloseRegKey $(KeyPcimac)
            else
                Debug-Output "UpgradeIsdnInfo: error creating RAS\TAPI DEVICES\
Pcimac key"
            endif
        else
             Debug-Output "UpgradeIsdnInfo: error creating RAS\TAPI DEVICES key"
        endif
        CloseRegKey $(KeyIsdn)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasKeyName) +
                                      $(!MAXIMUM_ALLOWED) KeyRas
        ifstr $(KeyRas) != $(KeyNull)
            DeleteRegTree $(KeyRas) "Media"
            CloseRegKey $(KeyRas)
        endif
        set KeyRasman = $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasManParamName) +
                                      $(!MAXIMUM_ALLOWED) KeyRasman
        ifstr $(KeyRasman) != $(KeyNull)
           GetRegValue $(KeyRasman), "Medias", MediaList
           ifint $(RegLastError) == 0
               set Medias = *($(MediaList), 4)
           else
               Debug-Output "UpgradeIsdnInfo: error reading Medias value "
           endif
           set NewMedias = {}
           forlistdo $(Medias)
               ifstr(i) $($) != "ISDN"
                  set NewMedias = >($(NewMedias), $($))
               endif
           endforlistdo
           set NewMedias = >($(NewMedias), "rastapi")
           Debug-Output "UpgradeIsdnInfo: NewMedias "$(NewMedias)
           SetRegValue $(KeyRasman) {"Medias", 0, +
                                          $(!REG_VT_MULTI_SZ),$(NewMedias)}
           CloseRegKey $(KeyRasman)
        else
          Debug-Output "UpgradeIsdnInfo: error opening Services\Rasman\
Parameters key"
          goto UpgradeIsdnInfoEnd
        endif
        set Status = STATUS_SUCCESSFUL
    else
        set Status = STATUS_SUCCESSFUL
    endif
```

```
UpgradeIsdnInfoEnd = +
    Debug-Output "UpgradeIsdnInfo exit."
    return $(Status)
[UpdateNetCardInfo]
    Debug-Output "UpdateNetCardInfo: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set fNetbeuiChosen = $($0)
    set fTcpIpChosen   = $($1)
    set fIpxChosen     = $($2)
    set RasAsyMacParamKeyName = $(!NTN_ServiceBase)"\AsyncMac\Parameters"
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
KeyNetcards
    ifstr $(KeyNetcards) == $(KeyNull)
         Debug-Output "UpdateNetCardInfo: could not open Netcards key"
         goto UpdateNetCardInfoEnd
    endif
    EnumRegKey $(KeyNetcards) NetcardsList
    set RemoveList  = {}
    ForListDo $(NetcardsList)
       set KeyName = *($($),1)
       OpenRegKey $(KeyNetcards) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
       ifstr $(Card) == $(KeyNull)
           Debug-Output "OEMNSVRA.INF: could not open netcard key"
           CloseRegKey $(KeyNetcards)
           goto UpdateNetCardInfoEnd
       endif
       GetRegValue $(Card), "ProductName" ProductNameInfo
       set CardProductName = *($(ProductNameInfo), 4)
       ifstr(i) $(fNetbeuiChosen) == FALSE
         ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINName)
              set RemoveList = >($(RemoveList), +
                         {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
         else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTName)
              set RemoveList = >($(RemoveList), +
                         {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
           endif
       endif
       ifstr(i) $(fTcpIpChosen) == FALSE
         ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINIPName)
              set RemoveList = >($(RemoveList), +
                         {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
         else-ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALOUTIPName)
              set RemoveList = >($(RemoveList), +
                         {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
           endif
       endif
       ifstr(i) $(fIpxChosen) == FALSE
         ifstr(i) $(CardProductName) == $(!ProductNDISWANDIALINOUTIPXName)
              set RemoveList = >($(RemoveList), +
                         {$(!ProductNDISWANName),+
                                 $(!NetworkCardKeyName)"\"$(KeyName)})
           endif
       endif
       CloseRegKey $(Card)
    EndForListDo
    CloseRegKey $(KeyNetcards)
    QueryListSize NetCards $(RemoveList)
    ifint $(NetCards) != 0
        ForListDo  $(RemoveList)
```

```
            debug-output "Removing hardware component: "$($)
            Shell "utility.inf", RemoveHardwareComponent, +
                                 $(!Manufacturer), *($($),1), *($($),2)
        EndForListDo
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacParamKeyName) $(!MAXIMUM_ALLOWED)
KeyMacParams
    ifstr $(KeyMacParams) == $(KeyNull)
       Debug-Output "UpdateNetCardInfo: could not open AsyncMac Params key"
       goto UpdateNetCardInfoEnd
    endif
    set NewValueList = {}
    ifstr(i) $(fNetbeuiChosen) == FALSE
        set NewValueList =  >($(NewValueList), +
                             {DialinNBF, $(NoTitle), $(!REG_VT_DWORD), 0})
        set NewValueList = >($(NewValueList), +
                             {DialoutNBF, $(NoTitle), $(!REG_VT_DWORD), 0})
    endif
    ifstr(i) $(fTcpIpChosen) == FALSE
        set NewValueList = >($(NewValueList), +
                            {DialinIP, $(NoTitle), $(!REG_VT_DWORD), 0 })
        set NewValueList = >($(NewValueList), +
                             {DialoutIP, $(NoTitle), $(!REG_VT_DWORD), 0 })
    endif
    ifstr(i) $(fIpxChosen) == FALSE
        set NewValueList = >($(NewValueList), +
                             {DialinoutIPX, $(NoTitle), $(!REG_VT_DWORD), 0})
    endif
    Shell  "utility.inf", AddValueList, $(KeyMacParams), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "UpdateNetCardInfo:AddValueList bombed out"
         goto UpdateNetCardInfoEnd
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(KeyMacParams)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        goto UpdateNetCardInfoEnd
    endif
    set Status = STATUS_SUCCESSFUL
UpdateNetCardInfoEnd =+
    Debug-Output "UpdateNetcardInfo: Exit"
    return $(Status)
[UpdateNdisWanInfo]
    Debug-Output "UpdateNdisWanNetInfo: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set NdisWanName = $(!NTN_SoftwareBase)"\Microsoft\NdisWan\CurrentVersion"
    set NdisWanRulesName = $(!NTN_SoftwareBase)"\Microsoft\NdisWan\
CurrentVersion\NetRules"
    set NdisWanSvcName = $(!NTN_ServiceBase)"\NdisWan"
    set RasManLinkageName = $(!NTN_ServiceBase)"\RasMan\Linkage"
    OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanName) $(!MAXIMUM_ALLOWED) KeyNdisWan
    ifstr $(KeyNdisWan) != $(KeyNull)
        SetRegValue $(KeyNdisWan) {Description, 0, +
                        $(!REG_VT_SZ),$(!ProductNDISWANDescription)}
        SetRegValue $(KeyNdisWan) {PathName, 0, +
                        $(!REG_VT_SZ),$(!ProductNDISWANImagePath)}
        SetRegValue $(KeyNdisWan) {ServiceName, 0, +
                        $(!REG_VT_SZ),$(!ProductNDISWANName)}
        SetRegValue $(KeyNdisWan) {Title, 0, +
                        $(!REG_VT_SZ),$(!ProductNDISWANTitle)}
        CloseRegKey $(KeyNdisWan)
    else
          Debug-Output "UpdateNdisWanInfo: could not open NdisWan key"
```

```
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanRulesName) $(!MAXIMUM_ALLOWED)
KeyRules
    ifstr $(KeyRules) != $(KeyNull)
        SetRegValue $(KeyRules) {bindable, 0, +
                        $(!REG_VT_MULTI_SZ),$(!NetRuleNDISWANBindable)}
        SetRegValue $(KeyRules) {bindform, 0, +
                        $(!REG_VT_SZ),$(!NetRuleNDISWANBindForm)}
        SetRegValue $(KeyRules) {class, 0, +
                        $(!REG_VT_MULTI_SZ),$(!NetRuleNDISWANClass)}
        SetRegValue $(KeyRules) {type, 0, +
                        $(!REG_VT_SZ),$(!NetRuleNDISWANType)}
        SetRegValue $(KeyRules) {InfOption, 0, +
                        $(!REG_VT_SZ), "NDISWAN"}
        CloseRegKey $(KeyRules)
    else
          Debug-Output "UpdateNdisWanInfo: could not open NdisWan NetRules key"
    endif
    OpenRegKey $(!REG_H_LOCAL) "" $(NdisWanSvcName) $(!MAXIMUM_ALLOWED)
KeyService
    ifstr $(KeyService) != $(KeyNull)
        SetRegValue $(KeyService) {DisplayName, 0, +
                        $(!REG_VT_SZ),$(!ProductNDISWANDisplayName)}
        SetRegValue $(KeyService) {ImagePath, 0, +
                        $(!REG_VT_EXPAND_SZ),$(!ProductNDISWANImagePath)}
        SetRegValue $(KeyService) {Start, 0, $(!REG_VT_DWORD),2}
        SetRegValue $(KeyService) {Group, 0, $(!REG_VT_SZ), "NDISWAN"}
        DeleteRegValue $(KeyService) "DependOnService"
        DeleteRegValue $(KeyService) "DependOnGroup"
        OpenRegKey $(KeyService) "" "Linkage" $(!MAXIMUM_ALLOWED) KeyLinkage
        ifstr $(KeyLinkage) != $(KeyNull)
            DeleteRegValue $(KeyLinkage) "OtherDependencies"
            CloseRegKey $(KeyLinkage)
        endif
        CloseRegKey $(KeyService)
    else
          Debug-Output "UpdateNdisWanInfo: could not open NdisWan services key"
    endif
    Shell "" RemoveServiceDependency "RasMan" "RasHub"
    set Status   = STATUS_SUCCESSFUL
UpdateNdisWanInfoEnd =+
    Debug-Output "UpdateNdisWanInfo: Exit"
    return $(Status)
[UpdateAsyncMacNetRules]
    Debug-Output "UpdateAsyncMacNetRules: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set AsyncMacRulesName = $(!NTN_SoftwareBase)"\Microsoft\AsyncMac\
CurrentVersion\NetRules"
    OpenRegKey $(!REG_H_LOCAL) "" $(AsyncMacRulesName) $(!MAXIMUM_ALLOWED)
KeyRules
    ifstr $(KeyRules) != $(KeyNull)
        set Status   = STATUS_SUCCESSFUL
        SetRegValue $(KeyRules) {bindable, 0, +
                        $(!REG_VT_MULTI_SZ),$(!NetRuleRASASYMACBindable)}
        CloseRegKey $(KeyRules)
    else
          Debug-Output "UpdateAsyncMacNetRules: could not open NdisWan NetRules
key"
    endif
    Debug-Output "UpdateAsyncMacNetRules: Exit"
    return $(Status)
[UpdateAsyncMacParameters]
    Debug-Output "UpdateAsyncMacParameters: entry"
```

```
    set Status   = STATUS_SUCCESSFUL
    Set KeyNull  = ""
    set RasAsyMacParamKeyName = $(!NTN_ServiceBase)"\AsyncMac\Parameters"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacParamKeyName) $(!MAXIMUM_ALLOWED)
KeyMacParams
    ifstr $(KeyMacParams) != $(KeyNull)
        GetRegValue $(KeyMacParams), "DialinNBF" DialinNBFInfo
        Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
            GetRegValue $(KeyMacParams), "Dialin" DialinNBFInfo
            set PrevNumDialinNBF  = *($(DialinNBFInfo), 4)
            SetRegValue $(KeyMacParams) {DialinNBF, 0, +
                                         $(!REG_VT_DWORD),$(PrevNumDialinNBF)}
            Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
                Debug-Output "UpdateAsyncMacParameters: deleting dialin key"
                DeleteRegValue $(KeyMacParams) "Dialin"
            endif
        endif
        GetRegValue $(KeyMacParams), "DialoutNBF" DialoutNBFInfo
        Ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
            GetRegValue $(KeyMacParams), "Dialout" DialoutNBFInfo
            set PrevNumDialoutNBF    = *($(DialoutNBFInfo), 4)
            SetRegValue $(KeyMacParams) {DialoutNBF, 0, +
                                         $(!REG_VT_DWORD),$(PrevNumDialoutNBF)}
            Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
                Debug-Output "UpdateAsyncMacParameters: deleting dialout key"
                DeleteRegValue $(KeyMacParams) "Dialout"
            endif
        endif
        CloseRegKey $(KeyMacParams)
    else
        Debug-Output "OEMNSVRA.INF: could not open AsyncMac Params key"
    endif
    Debug-Output "UpdateAsyncMacParameters: Exit"
    return $(Status)
[UpdateAsyncMacStartType]
    Debug-Output "UpdateAsyncMacStartType: entry"
    set Status   = STATUS_SUCCESSFUL
    Set KeyNull  = ""
    set RasAsyMacKeyName = $(!NTN_ServiceBase)"\AsyncMac"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasAsyMacKeyName) $(!MAXIMUM_ALLOWED) KeyMac
    ifstr $(KeyMac) != $(KeyNull)
        GetRegValue $(KeyMac), "Start" StartValue
        Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
            set Start = *($(StartValue), 4)
            ifint $(Start) == 3
                set Start = 2
                SetRegValue $(KeyMac) {Start, 0, $(!REG_VT_DWORD), $(Start)}
            endif
        endif
        CloseRegKey $(KeyMac)
    else
        Debug-Output "OEMNSVRA.INF: could not open AsyncMac key"
    endif
    Debug-Output "UpdateAsyncMacStartType: Exit"
    return $(Status)
[UpdatePerfmonInfo]
    Debug-Output "UpdatePerfmonInfo: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set RemoteAccessService = $(!NTN_ServiceBase)"\RemoteAccess"
    OpenRegKey $(!REG_H_LOCAL) "" $(RemoteAccessService) $(!MAXIMUM_ALLOWED)
KeyService
    ifstr $(KeyService) != $(KeyNull)
        Shell "" UpdatePerfmonInfoHelper $(KeyService)
```

```
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "error shelling UpdatePerfmonInfoHelper."
            goto UpdatePerfmonInfoEnd
        endif
        Ifstr(i) $($R0) != STATUS_SUCCESSFUL
            Debug-Output "error returned by UpdatePerfmonInfo."
            goto UpdatePerfmonInfoEnd
        endif
        CloseRegKey $(KeyService)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "UpdatePerfmonInfo: error opening RemoteAccess service key"
    endif
UpdatePerfmonInfoEnd =+
    Debug-Output "UpdatePerfmonInfo: Exit"
    return $(Status)
[UpdatePerfmonInfoHelper]
    Debug-Output "UpdatePerfmonInfoHelper: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set ServiceKey = $($0)
    set KeyPerformance = $(KeyNull)
    OpenRegKey $(ServiceKey) "" $(!RasPerfKeyName) $(!MAXIMUM_ALLOWED)
KeyPerformance
    ifstr $(KeyPerformance) == $(KeyNull)
        CreateRegKey $(ServiceKey) {$(!RasPerfKeyName),$(NoTitle),GenericClass}
"" $(!MAXIMUM_ALLOWED) "" KeyPerformance
        OpenRegKey $(ServiceKey) "" $(!RasPerfKeyName) $(!MAXIMUM_ALLOWED)
KeyPerformance
        ifstr $(KeyPerformance) == $(KeyNull)
            Debug-Output "Error creating Performance key"
            goto UpdatePerfmonInfoHelperEnd
        endif
    endif
    set NewValueList = +
         {{Library ,$(NoTitle),$(!REG_VT_SZ),$(!RasPerfLibraryName)},+
         {Open, $(NoTitle),$(!REG_VT_SZ),$(!RasPerfOpenFunction)},+
         {Close, $(NoTitle),$(!REG_VT_SZ),$(!RasPerfCloseFunction)},+
         {Collect, $(NoTitle),$(!REG_VT_SZ),$(!RasPerfCollectFunction)}}
    Shell  "Utility.Inf", AddValueList, $(KeyPerformance), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "UpdatePerfmonInfoHelper: error shelling AddValueList"
         goto UpdatePerfmonInfoHelperEnd
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
        Debug-Output "UpdatePerfmonInfoHelper: AddValueList bombed out"
        goto UpdatePerfmonInfoHelperEnd
    endif
    CloseRegKey $(KeyPerformance)
    set Status = STATUS_SUCCESSFUL
UpdatePerfmonInfoHelperEnd =+
    Debug-Output "UpdatePerfmonInfoHelper: Exit"
    return $(Status)
[RenameRasHubToNdisWan]
    Debug-Output "RenameRasHubToNdisWan: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    OpenRegKey $(!REG_H_LOCAL) "" $(!NetworkCardKeyName) $(!MAXIMUM_ALLOWED)
KeyNetcards
    ifstr $(KeyNetcards) == $(KeyNull)
         Debug-Output "RenameRasHubToNdisWan: could not open Netcards key"
         goto RenameRasHubToNdisWanEnd
    endif
```

```
    EnumRegKey $(KeyNetcards) NetcardsList
    ForListDo $(NetcardsList)
       set KeyName = *($($),1)
       OpenRegKey $(KeyNetcards) "" $(KeyName) $(!MAXIMUM_ALLOWED) Card
       ifstr $(Card) == $(KeyNull)
           Debug-Output "RenameRasHubToNdisWan: could not open netcard key"
           CloseRegKey $(KeyNetcards)
           goto RenameRasHubToNdisWanEnd
       endif
       GetRegValue $(Card), "ProductName" ProductNameInfo
       set CardProductName = *($(ProductNameInfo), 4)
         ifstr(i) $(CardProductName) == $(!ProductRASHUBDIALINName)
           set ThisOption = DIALIN
         else-ifstr(i) $(CardProductName) == $(!ProductRASHUBDIALOUTName)
           set ThisOption = DIALOUT
         else-ifstr(i) $(CardProductName) == $(!ProductRASHUBDIALINIPName)
           set ThisOption = DIALINIP
         else-ifstr(i) $(CardProductName) == $(!ProductRASHUBDIALOUTIPName)
           set ThisOption = DIALOUTIP
         else-ifstr(i) $(CardProductName) == $(!ProductRASHUBDIALINOUTIPXName)
           set ThisOption = DIALINOUTIPX
       else
           goto endloop
       endif
       SetRegValue $(Card) {ProductName, 0, +
                            $(!REG_VT_SZ),$(!ProductNDISWAN$(ThisOption)Name)}
       SetRegValue $(Card) {Description, 0, +
                            $(!REG_VT_SZ),$(!ProductNDISWANDescription)}
       SetRegValue $(Card) {ServiceName, 0, $(!REG_VT_SZ), +
                            $(!ProductNDISWANName)$(KeyName)}
       SetRegValue $(Card) {Title, 0, $(!REG_VT_SZ),+
                            "["$(KeyName)"] "$(!ProductNDISWANTitle)}
       OpenRegKey $(Card) "" "NetRules" $(!MAXIMUM_ALLOWED) NetRules
       ifstr $(NetRules) == $(KeyNull)
           CloseRegKey $(KeyNetCards)
           CloseRegKey $(Card)
           Debug-Output "RenameRasHubToNdisWan: could not open "$(KeyName)"\
NetRules key"
           goto RenameRasHubToNdisWanEnd
       Endif
       SetRegValue $(NetRules) {InfOption, 0, +
                            $(!REG_VT_SZ), "NDISWAN"}
       SetRegValue $(NetRules) {block, 0, +
                            $(!REG_VT_MULTI_SZ),$(!NetRuleHardware$
(ThisOption)Block)}
       SetRegValue $(NetRules) {class, 0, +
                            $(!REG_VT_MULTI_SZ),$(!NetRuleHardware$
(ThisOption)Class)}
       SetRegValue $(NetRules) {type, 0, +
                            $(!REG_VT_SZ),$(!NetRuleHardware$(ThisOption)Type)}
       Set TempProdName = """"$(!ProductNDISWANName)$(KeyName)""""
       Set TempBindForm = $(TempProdName)$(!NetRuleHardwareNDISWANBindForm)
       SetRegValue $(NetRules) {bindform, 0, $(!REG_VT_SZ),$(TempBindForm)}
       CloseRegKey $(NetRules)
endloop=+
       CloseRegKey $(Card)
    EndForListDo
    CloseRegKey $(KeyNetcards)
    set Status = STATUS_SUCCESSFUL
RenameRasHubToNdisWanEnd =+
    Debug-Output "RenameRasHubToNdisWan: Exit"
    return $(Status)
[UpgradeIpxInfo]
    Debug-Output "UpgradeIpxInfo: entry"
```

```
    set PORTSDLGHANDLE = $($0)
    set RasProtocolsName = $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols"
    set SapAgentName     = $(!NTN_SoftwareBase)"\Microsoft\NWSAPAGENT"
    set NwlnkRipName     = $(!NTN_SoftwareBase)"\Microsoft\NWLNKRIP"
    set IpxRouterName    = $(!NTN_ServiceBase)"\IpxRouter"
    OpenRegKey $(!REG_H_LOCAL) "" +
                    $(RasProtocolsName) $(!MAXIMUM_ALLOWED) KeyProtocols
    ifstr $(KeyProtocols) != ""
        set fIpxAllowed = 0
        GetRegValue $(KeyProtocols) "fIpxAllowed" IpxList
        ifint $(RegLastError) == 0
            set fIpxAllowed = *($(IpxList), 4)
        endif
        ifint $(fIpxAllowed) != 0
            Debug-Output "UpgradeIpxInfo: RAS is configured for IPX dialin"
            set KeyRouter = ""
            OpenRegKey $(!REG_H_LOCAL) "" +
                      $(IpxRouterName) $(!MAXIMUM_ALLOWED) KeyRouter
            ifstr $(KeyRouter) !=  ""
                CloseRegKey $(KeyRouter)
                Debug-Output "Renaming IpxRouter to NwlnkRip..."
                  LibraryProcedure Result $(PORTSDLGHANDLE)
RenameIpxRouterToNwlnkRip
                Shell "" RemoveServiceDependency "RemoteAccess" "IpxRouter"
                Shell "" AddServiceDependency "RemoteAccess" "NwlnkRip"
                Debug-Output "Renaming IpxRouter to NwlnkRip done..."
            endif
            OpenRegKey $(!REG_H_LOCAL) "" $(!RasIsnRipKeyName) +
                                        $(!MAXIMUM_ALLOWED) KeyService
            ifstr $(KeyService) != ""
                OpenRegKey $(KeyService) "" "Parameters" +
                                       $(!MAXIMUM_ALLOWED) KeyParams
                ifstr(i) $(KeyParams) != ""
                   GetRegValue $(KeyParams) "NetbiosRouting" NetbiosRoutingInfo
                   Ifint $(RegLastError) == $(!REG_ERROR_SUCCESS)
                      set NetbiosRouting = *($(NetbiosRoutingInfo), 4)
                   else
                      set NetbiosRouting = 2
                   endif
                   set NetbiosRouting = *($(NetbiosRoutingInfo), 4)
                   ifint $(NetbiosRouting) == 0
                      set NetbiosRouting = 2
                   else-ifint $(NetbiosRouting) == 1
                      set NetbiosRouting = 3
                   endif
                   SetRegValue $(KeyParams) {"NetbiosRouting" , 0, $(!
REG_VT_DWORD), $(NetbiosRouting)}
                endif
                OpenRegKey $(KeyService) "" "Parameters\UseRef" +
                                          $(!MAXIMUM_ALLOWED) KeyUseRef
                ifstr(i) $(KeyUseRef) == ""
                    ifstr(i) $(KeyParams) != ""
                       CreateRegKey $(KeyParams) {"UseRef", 0, GenericClass} +
                                       "" $(!MAXIMUM_ALLOWED) "" KeyUseRef
                        CloseRegKey $(KeyParams)
                    endif
                endif
                ifstr(i) $(KeyUseRef) != ""
                    SetRegValue $(KeyUseRef) {"RAS" , 0, $(!REG_VT_SZ), "1"}
                    CloseRegKey $(KeyUseRef)
                endif
                CloseRegKey $(KeyService)
            endif
            OpenRegKey $(!REG_H_LOCAL) "" $(SapAgentName) +
```

```
                                        $(!MAXIMUM_ALLOWED) KeyService
            ifstr $(KeyService) == ""
                OpenRegKey $(!REG_H_LOCAL) "" $(!RasIsnSapKeyName) +
                                            $(!MAXIMUM_ALLOWED) KeySap
                Ifstr(i) $(KeySap) != ""
                    Shell "utility.inf", RemoveService +
                                        $(!ProductRASISNSAPName) "YES"
                    Shell "" InstallIsnSapService
                endif
                CloseRegKey $(KeySap)
            else
                CloseRegKey $(KeyService)
            endif
            OpenRegKey $(!REG_H_LOCAL) "" $(NwlnkRipName) +
                                        $(!MAXIMUM_ALLOWED) KeyService
            ifstr $(KeyService) == ""
                OpenRegKey $(!REG_H_LOCAL) "" $(!RasIsnRipKeyName) +
                                            $(!MAXIMUM_ALLOWED) KeyRip
                Ifstr(i) $(KeyRip) != ""
                    Shell "" InstallNwlnkRipService
                CloseRegKey $(KeyRip)
            else
                CloseRegKey $(KeyService)
            endif
            OpenRegKey $(!REG_H_LOCAL) ""  +
                       $(!NTN_ServiceBase)"\NWLNKIPX\Parameters"  +
                       $(!MAXIMUM_ALLOWED) KeyIpxParameters
            ifstr $(KeyIpxParameters) != $(KeyNull)
                GetRegValue $(KeyIpxParameters), "SingleNetworkActive"  +
                            SingleNetworkActive
                ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                    SetRegValue $(KeyIpxParameters) +
                                {SingleNetworkActive, 0, $(!REG_VT_DWORD), 1}
                endif
                GetRegValue $(KeyIpxParameters), "DisableDialoutSap"  +
                            DisableDialoutSap
                ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                    SetRegValue $(KeyIpxParameters) +
                                {DisableDialoutSap, 0, $(!REG_VT_DWORD), 1}
                endif
                GetRegValue $(KeyIpxParameters), "DisableDialinNetbios"  +
                            DisableDialinNetbios
                ifint $(RegLastError) != $(!REG_ERROR_SUCCESS)
                    SetRegValue $(KeyIpxParameters) +
                                {DisableDialinNetbios, 0, $(!REG_VT_DWORD), 1}
                endif
                CloseRegKey $(KeyIpxParameters)
            Endif
            else
                Debug-Output "OEMNSVRA.INF: error opening NWLNKIPX\Parameters
key"
            endif
        endif
        CloseRegKey $(KeyProtocols)
    else
        Debug-Output "UpgradeIpxInfo: error opening RAS\Protocols key"
    endif
    Debug-Output "UpgradeIpxInfo: exit"
    return
[SaveTcpipInfo]
    Debug-Output "SaveTcpipInfo: entry - AddIpInfo = "$($0)
    set AddTcpInfo = $($0)
    set Status     = STATUS_FAILED
    Set KeyNull    = ""
```

```
    set RasIpKeyName =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols\Ip"
    set RasSvrParamKeyName = $(!NTN_ServiceBase)"\RemoteAccess\Parameters"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasIpKeyName) $(!MAXIMUM_ALLOWED) RasIpKey
    ifstr $(RasIpKey) != $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasSvrParamKeyName) $(!MAXIMUM_ALLOWED)
KeySrvParams
        ifstr $(KeySrvParams) == $(KeyNull)
           Debug-Output "SaveTcpipInfo: could not open RasSvr Params key"
           CloseRegKey $(RasIpKey)
           return $(Status)
        endif
        ifstr(i) $(AddTcpInfo) == TRUE
            CreateRegKey $(KeySrvParams) {"Ip",$(NoTitle),GenericClass} "" $(!
MAXIMUM_ALLOWED) "" RemoteIpKey
            OpenRegKey $(KeySrvParams) "" "Ip" $(!MAXIMUM_ALLOWED) RemoteIpKey
            ifstr $(RemoteIpKey) != $(KeyNull)
               EnumRegValue $(RasIpKey) NewValueList
               Shell  "utility.inf", AddValueList, $(RemoteIpKey), $
(NewValueList)
               CloseRegKey $(RemoteIpKey)
            endif
        else
            Debug-Output "SaveTpcipInfo: removing ip info"
            DeleteRegTree $(KeySrvParams) "IP"
        endif
        CloseRegKey $(RasIpKey)
        CloseRegKey $(KeySrvParams)
    endif
    Debug-Output "SaveTcpipInfo: exit"
    set Status = STATUS_SUCCESSFUL
    return $(Status)
[SaveIpxInfo]
    Debug-Output "SaveIpxInfo: entry RouterInstalled= "$($0)" AddIpxInfo = "$
($1)
    set RouterInstalled = $($0)
    set AddIpxInfo      = $($1)
    set Status          = STATUS_FAILED
    Set KeyNull         = ""
    set RasIpxKeyName =  $(!NTN_SoftwareBase)"\Microsoft\Ras\Protocols\Ipx"
    set RasSvrParamKeyName = $(!NTN_ServiceBase)"\RemoteAccess\Parameters"
    OpenRegKey $(!REG_H_LOCAL) "" $(RasIpxKeyName) $(!MAXIMUM_ALLOWED) RasIpxKey
    ifstr $(RasIpxKey) != $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(RasSvrParamKeyName) +
                                            $(!MAXIMUM_ALLOWED) KeySrvParams
        ifstr $(KeySrvParams) == $(KeyNull)
           Debug-Output "SaveTcpipInfo: could not open RasSvr Params key"
           CloseRegKey $(RasIpxKey)
           return $(Status) $(fRouterInstalled)
        endif
        ifstr(i) $(AddIpxInfo) == TRUE
            CreateRegKey $(KeySrvParams) {"Ipx",$(NoTitle),GenericClass} "" $(!
MAXIMUM_ALLOWED) "" RemoteIpxKey
            OpenRegKey $(KeySrvParams) "" "Ipx" $(!MAXIMUM_ALLOWED) RemoteIpxKey
            ifstr $(RemoteIpxKey) != $(KeyNull)
               EnumRegValue $(RasIpxKey) NewValueList
               Shell  "utility.inf", AddValueList, $(RemoteIpxKey), $
(NewValueList)
               SetRegValue $(RemoteIpxKey) {RouterInstalled, 0,$(!REG_VT_DWORD),
$(RouterInstalled)}
               CloseRegKey $(RemoteIpxKey)
            endif
        else
            Debug-Output "SaveIpxInfo: removing ipx info"
            DeleteRegTree $(KeySrvParams) "IPX"
```

```
        endif
        CloseRegKey $(KeySrvParams)
        CloseRegKey $(RasIpxKey)
    endif
    Debug-Output "SaveIpxInfo: exit"
    set Status = STATUS_SUCCESSFUL
    return $(Status)
[AddServiceDependency]
    Debug-Output "AddServiceDependency: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set DependentService = $(!NTN_ServiceBase)\$($0)
    set DependentName = $($0)
    set DependOn = $($1)
    set ServiceKey = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(DependentService)"\Linkage" $(!
MAXIMUM_ALLOWED) ServiceKey
    ifstr(i) $(ServiceKey) != $(KeyNull)
        GetRegValue $(ServiceKey) "OtherDependencies" ServicesList
        ifint $(RegLastError) != 0
           set ServiceValues = {}
        else
           set ServiceValues = *($(ServicesList),4)
        endif
        debug-output "AddServiceDependency: Old OtherDependencies: "$
(ServiceValues)
        ifstr(i) $(ServiceValues) == {}
            Set ServiceValues = {$(DependOn)}
        else-ifstr(i) $(ServiceValues) == ""
            Set ServiceValues = {$(DependOn)}
        else-ifcontains(i) $(DependOn) in $(ServiceValues)
            return STATUS_SUCCESSFUL
        else
            Set ServiceValues = >($(ServiceValues), $(DependOn))
        endif
        debug-output "AddServiceDependency: New OtherDependencies: "$
(ServiceValues)
        SetRegValue $(ServiceKey) {OtherDependencies, 0,+
                                   $(!REG_VT_MULTI_SZ), $(ServiceValues)}
        CloseRegKey $(ServiceKey)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "AddServiceDependency: error opening service "$
(DependentService)"\Linkage"
    endif
    set KeyService = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(DependentService) $(!MAXIMUM_ALLOWED)
KeyService
    ifstr $(KeyService) != $(KeyNull)
        set newDependList = {$(DependOn)}
        GetRegValue    $(KeyService) "DependOnService" ServiceList
        ifint $(RegLastError) == 0
            Debug-Output "AddServiceDependency: old DependOnService List "*($
(ServiceList), 4)
            ForListDo *($(ServiceList),4)
                ifstr(i) $($) != $(DependOn)
                   set newDependList = >($(newDependList), $($))
                endif
            EndForListDo
        endif
        GetRegValue    $(KeyService) "DependOnGroup" GrpList
        ifint $(RegLastError) == 0
            Debug-Output "AddServiceDependency: old DependOnGroup List "*($
(GrpList), 4)
```

```
            ForListDo *($(GrpList),4)
               set grp = "+"$($)
               set newDependList = >($(newDependList), $(grp))
            EndForListDo
        endif
        Debug-Output "OEMNSVRA.INF: AddServiceDependency: new Dependency List "$
(newDependList)
        LibraryProcedure Result, $(!LIBHANDLE), SetupChangeServiceConfig, $
(DependentName) $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE), $(!
SERVICE_NO_CHANGE), "", "", $(newDependList), "", "", ""
        CloseRegKey $(KeyService)
    else
        Debug-Output "AddServiceDependency: failed to open service linkage key"$
(DependentService)
    endif
    Debug-Output "AddServiceDependency: exit"
    return $(Status)
[RemoveServiceDependency]
    Debug-Output "RemoveServiceDependency: entry"
    set Status   = STATUS_FAILED
    Set KeyNull  = ""
    set DependentService = $(!NTN_ServiceBase)\$($0)
    set DependentName = $($0)
    set DependOn = $($1)
    OpenRegKey $(!REG_H_LOCAL) "" $(DependentService)"\Linkage" $(!
MAXIMUM_ALLOWED) ServiceKey
    ifstr(i) $(ServiceKey) != $(KeyNull)
        GetRegValue $(ServiceKey) "OtherDependencies" ServicesList
        ifint $(RegLastError) != 0
           set ServiceValues = {}
        else
           set ServiceValues = *($(ServicesList),4)
        endif
        debug-output "RemoveServiceDependency: old OtherDependencies list:"$
(ServicesList)
        set ServiceValues = *($(ServicesList),4)
        debug-output "ServiceValues: "$(ServiceValues)
        ifcontains(i) $(DependOn) in $(ServiceValues)
            set NewServiceValues = {}
            ForListDo $(ServiceValues)
                ifstr(i) $($) != $(DependOn)
                    set NewServiceValues = >($(NewServiceValues), $($))
                endif
            EndForListDo
            debug-output "RemoveServiceDependency: new OtherDependencies list:
"$(NewServiceValues)
            SetRegValue $(ServiceKey) {OtherDependencies, 0,+
                            $(!REG_VT_MULTI_SZ), $(NewServiceValues)}
        else
        endif
        CloseRegKey $(ServiceKey)
        set Status = STATUS_SUCCESSFUL
    else
        Debug-Output "RemoveServiceDependency: error opening service "$
(DependentService)"\Linkage"
    endif
    set KeyService = $(KeyNull)
    OpenRegKey $(!REG_H_LOCAL) "" $(DependentService) $(!MAXIMUM_ALLOWED)
KeyService
    ifstr $(KeyService) != $(KeyNull)
        set newDependList = {}
        GetRegValue    $(KeyService) "DeleteFlag" DeleteFlagValue
        ifint $(RegLastError) != 0
            GetRegValue    $(KeyService) "DependOnService" ServiceList
```

```
            ifint $(RegLastError) == 0
                Debug-Output "RemoveServiceDependency: old DependOnService List
"*($(ServiceList), 4)
                ifcontains(i) $(DependOn) in *($(ServiceList),4)
                    ForListDo *($(ServiceList),4)
                        ifstr(i) $($) != $(DependOn)
                           set newDependList = >($(newDependList), $($))
                        endif
                    EndForListDo
                    GetRegValue $(KeyService) "DependOnGroup" GrpList
                    ifint $(RegLastError) == 0
                        ForListDo *($(GrpList),4)
                           set grp = "+"$($)
                           set newDependList = >($(newDependList), $(grp))
                        EndForListDo
                    endif
                    Debug-Output "OEMNSVRA.INF:RemoveServiceDependency: new
DependOnService List "$(newDependList)
                    LibraryProcedure Result, $(!LIBHANDLE), +
                       SetupChangeServiceConfig, $(DependentName)  +
                       $(!SERVICE_NO_CHANGE), $(!SERVICE_NO_CHANGE), +
                       $(!SERVICE_NO_CHANGE), "", "",  +
                       $(newDependList), "", "", ""
                endif
            endif
        endif
        CloseRegKey $(KeyService)
    else
        Debug-Output "RemoveServiceDependency: failed to open service linkage
key"$(DependentService)
    endif
    Debug-Output "RemoveServiceDependency: exit"
    return $(Status)
[QueryUserQuit]
    set Status     = STATUS_FAILED
    set UserAction = CANCEL
    set STF_MB_TEXT  = "The changes will not be saved. "+
                       "Are you sure you want to exit Setup."
    read-syms ExitWarningDlg$(!STF_LANGUAGE)
    ui start "ExitWarning"
    ifstr(i) $(DLGEVENT) == "YES"
        set Status     = STATUS_SUCCESSFUL
        set UserAction = "OK"
    else-ifstr(i) $(DLGEVENT) == "NO"
        set Status     = STATUS_SUCCESSFUL
        set UserAction = "CANCEL"
    else
    endif
fin_QueryUserQuit = +
    Return $(Status) $(UserAction)
[QueryRasUpgrade]
    set Status     = STATUS_FAILED
    set UserAction = CANCEL
    read-syms RasUpgrade$(!STF_LANGUAGE)
    set Text = $($0)$(Ver)$($1)$(Text1)$(Ver)$($2)$(Text2)$(Text3)
    set STF_MB_TEXT  = $(Text)
    read-syms ExitWarningDlg$(!STF_LANGUAGE)
    ui start "RasUpgrade"
    ifstr(i) $(DLGEVENT) == "YES"
        set Status     = STATUS_SUCCESSFUL
        set UserAction = "OK"
    else-ifstr(i) $(DLGEVENT) == "NO"
        set Status     = STATUS_SUCCESSFUL
        set UserAction = "CANCEL"
```

```
    else
    endif
fin_QueryRasUpgrade = +
    Return $(Status) $(UserAction)
[CP-List]
NBFCP       = "%SystemRoot%\SYSTEM32\RASNBFCP.DLL"
IPCP        = "%SystemRoot%\SYSTEM32\RASIPCP.DLL"
IPXCP       = "%SystemRoot%\SYSTEM32\RASIPXCP.DLL"
PAP         = "%SystemRoot%\SYSTEM32\RASPAP.DLL"
CHAP        = "%SystemRoot%\SYSTEM32\RASCHAP.DLL"
SPAP        = "%SystemRoot%\SYSTEM32\RASSPAP.DLL"
CBCP        = "%SystemRoot%\SYSTEM32\RASCBCP.DLL"
COMPCP      = "%SystemRoot%\SYSTEM32\RASCCP.DLL"
[Files-RemoveList]
RASADMINDLL    = $(!STF_WINDOWSSYSPATH)\RASADMIN.DLL
RASGTWYDLL     = $(!STF_WINDOWSSYSPATH)\RASGTWY.DLL
RASGPRXYDLL    = $(!STF_WINDOWSSYSPATH)\RASGPRXY.DLL
RASSRVEXE      = $(!STF_WINDOWSSYSPATH)\RASSRV.EXE
RASCTRSDLL     = $(!STF_WINDOWSSYSPATH)\RASCTRS.DLL
RASCTRSINI     = $(!STF_WINDOWSSYSPATH)\RASCTRS.INI
RASCTRNMH      = $(!STF_WINDOWSSYSPATH)\RASCTRNM.H
RASSPRXYEXE    = $(!STF_WINDOWSSYSPATH)\RASSPRXY.EXE
RASSAUTHDLL    = $(!STF_WINDOWSSYSPATH)\RASSAUTH.DLL
RASDIALEXE     = $(!STF_WINDOWSSYSPATH)\RASDIAL.EXE
RASPHONEHLP    = $(!STF_WINDOWSSYSPATH)\RASPHONE.HLP
RASGLOSSHLP    = $(!STF_WINDOWSSYSPATH)\RASGLOSS.HLP
RASAPI32DLL    = $(!STF_WINDOWSSYSPATH)\RASAPI32.DLL
RASCAUTHDLL    = $(!STF_WINDOWSSYSPATH)\RASCAUTH.DLL
RASADMINHLP    = $(!STF_WINDOWSSYSPATH)\RASADMIN.HLP
RASMANDLL      = $(!STF_WINDOWSSYSPATH)\RASMAN.DLL
RASMANEXE      = $(!STF_WINDOWSSYSPATH)\RASMAN.EXE
RASMSGDLL      = $(!STF_WINDOWSSYSPATH)\RASMSG.DLL
RASMXSDLL      = $(!STF_WINDOWSSYSPATH)\RASMXS.DLL
RASSERDLL      = $(!STF_WINDOWSSYSPATH)\RASSER.DLL
RASIPXCPDLL    = $(!STF_WINDOWSSYSPATH)\RASIPXCP.DLL
RASPPPDLL      = $(!STF_WINDOWSSYSPATH)\RASPPP.DLL
RASPPPENDLL    = $(!STF_WINDOWSSYSPATH)\RASPPPEN.DLL
RASPAPDLL      = $(!STF_WINDOWSSYSPATH)\RASPAP.DLL
RASCHAPDLL     = $(!STF_WINDOWSSYSPATH)\RASCHAP.DLL
RASSPAPDLL     = $(!STF_WINDOWSSYSPATH)\RASSPAP.DLL
RASIPCPDLL     = $(!STF_WINDOWSSYSPATH)\RASIPCP.DLL
RASIPHLPDLL    = $(!STF_WINDOWSSYSPATH)\RASIPHLP.DLL
RASNBFCPDLL    = $(!STF_WINDOWSSYSPATH)\RASNBFCP.DLL
RASNBIPCDLL    = $(!STF_WINDOWSSYSPATH)\RASNBIPC.DLL
RASCCPDLL      = $(!STF_WINDOWSSYSPATH)\RASCCP.DLL
RASCBCPDLL     = $(!STF_WINDOWSSYSPATH)\RASCBCP.DLL
ASYNCMACSYS    = $(!STF_WINDOWSSYSPATH)\DRIVERS\ASYNCMAC.SYS
NDISWANSYS     = $(!STF_WINDOWSSYSPATH)\DRIVERS\NDISWAN.SYS
RASARPSYS      = $(!STF_WINDOWSSYSPATH)\DRIVERS\RASARP.SYS
[Source Media Descriptions]
    1 = "Windows NT Workstation CD-ROM" , TAGFILE = cdrom_w.40
[Signature]
    FileType = MICROSOFT_FILE
[GetSignature]
    read-syms Signature
    return $(FileType)
[ProductType]
STF_PRODUCT  = Winnt
STF_PLATFORM = I386
[Files-Ras-Admin]
1,RASADMIN.EXE , SIZE=126224 , NODELETESOURCE
1,RASSAPI.DLL , SIZE=22800 , NODELETESOURCE
1,RASADMIN.CNT , SIZE=1330
1,RASADMIN.HLP , SIZE=42084
```

```
[Files-Ras-Client]
1,RASPHONE.EXE , SIZE=52496 , NODELETESOURCE
1,RASSHELL.DLL , SIZE=18704 , NODELETESOURCE
1,RASAPI16.DLL , SIZE=4512
1,RASAPI32.DLL , SIZE=127248
1,RASCAUTH.DLL , SIZE=28432
1,RASCPL.CPL , SIZE=10512
1,RASDIAL.EXE , SIZE=17168
1,RASGLOSS.CNT , SIZE=2310
1,RASGLOSS.HLP , SIZE=23767
1,RASMON.EXE , SIZE=117520
1,RASPHONE.CNT , SIZE=7381
1,RASPHONE.HLP , SIZE=289828
1,RASSCRPT.DLL , SIZE=54544
[Files-Ras-Common]
1,RASADHLP.DLL , SIZE=10000
1,RASAUTO.DLL , SIZE=73488
1,RASAUTOU.EXE , SIZE=12048
1,RASCBCP.DLL , SIZE=9488
1,RASCCP.DLL , SIZE=11024
1,RASCHAP.DLL , SIZE=24848
1,RASDLG.DLL , SIZE=347408
1,RASIPCP.DLL , SIZE=16656
1,RASIPHLP.DLL , SIZE=27920
1,RASIPXCP.DLL , SIZE=22800
1,RASMAN.DLL , SIZE=61200
1,RASMAN.EXE , SIZE=8976
1,RASMSG.DLL , SIZE=28944
1,RASMXS.DLL , SIZE=26896
1,RASNBFCP.DLL , SIZE=15632
1,RASNBIPC.DLL , SIZE=8976
1,RASPAP.DLL , SIZE=13584
1,RASPPPEN.DLL , SIZE=49936
1,RASSER.DLL , SIZE=16656
1,RASSPAP.DLL , SIZE=15120
1,RASTAPI.DLL , SIZE=33040
[Files-Ras-Drivers]
1,ASYNCMAC.SYS , SIZE=39152
1,NDISTAPI.SYS , SIZE=8496
1,NDISWAN.SYS , SIZE=57104
1,RASACD.SYS , SIZE=8112
1,RASARP.SYS , SIZE=12912
[Files-Ras-Inf]
modem = 1,MODEM.INF , SIZE=999 , NODELETESOURCE
pad = 1,PAD.INF , SIZE=999 , NODELETESOURCE
rasico = 1,RAS.ICO , SIZE=999 , NODELETESOURCE
rasread = 1,RASREAD.TXT , SIZE=999 , NODELETESOURCE
switch = 1,SWITCH.INF , SIZE=999 , NODELETESOURCE
[Files-Ras-Scp]
cis = 1,CIS.SCP , SIZE=999 , NODELETESOURCE
pppmenu = 1,PPPMENU.SCP , SIZE=999 , NODELETESOURCE
scriptdoc = 1,SCRIPT.DOC , SIZE=999 , NODELETESOURCE
slip = 1,SLIP.SCP , SIZE=999 , NODELETESOURCE
slipmenu = 1,SLIPMENU.SCP , SIZE=999 , NODELETESOURCE
[Files-Ras-Server]
1,RASADMIN.DLL , SIZE=18192
1,RASCTRS.DLL , SIZE=15120
1,RASGPRXY.DLL , SIZE=15120
1,RASGTWY.DLL , SIZE=47888
1,RASSAUTH.DLL , SIZE=28432
1,RASSPRXY.EXE , SIZE=9488
1,RASSRV.EXE , SIZE=69392
[Files-Resource]
1,RASCFG.DLL , SIZE=154384 , NODELETESOURCE
```

```
1,RASFIL32.DLL , SIZE=12560 , NODELETESOURCE
1,RASSETUP.CNT , SIZE=99 , NODELETESOURCE
1,RASSETUP.HLP , SIZE=16757 , NODELETESOURCE
[LanguagesSupported]
    ENG
[OptionsTextENG]
    RAS = "Remote Access Service"
[FileConstantsENG]
    RasGroup    = "Remote Access Service"
    RasPhone    = "Remote Access"
    RasMon      = "Remote Access Monitor"
    RasAdmin    = "Remote Access Admin"
    ReadMe      = "Read Me"
    RasInternet = "Remote Access and the Internet"
    RasHelp     = "Remote Access Help"
    ProCaption   = "Remote Access Service Setup"
    ProCancel    = "Cance&l"
    ProCancelMsg = "Remote Access Service is not correctly installed. "+
                   "Are you sure you want to cancel copying files?"
    ProCancelCap = "Remote Access Service Setup Message"
    ProText1     = "Copying:"
    ProText2     = "To:"
    Error        = "Unable to determine proper source disk location; copy cannot
be performed."
    FunctionTitle   ="Remote Access Setup"
    ProductRASDescription               = "Enables users to work offsite as
though connected directly to a network."
    ProductRASTitle                     = "Remote Access Service"
    ProductRASDisplayName               = "Remote Access Service"
    ProductRASMANDescription            = "Windows NT Remote Access Connection
Manager"
    ProductRASMANTitle                  = "Remote Access Connection Manager"
    ProductRASMANDisplayName            = "Remote Access Connection Manager"
    !ProductRASAUTODIALDescription      = "Windows NT Remote Access Autodial
Manager"
    !ProductRASAUTODIALDisplayName      = "Remote Access Autodial Manager"
    !ProductRASAUTODIALTitle            = "Remote Access Autodial Manager"
    !ProductNDISWANDescription          = "Windows NT Remote Access WAN
Wrapper"
    !ProductNDISWANTitle                = "Remote Access WAN Wrapper"
    !ProductNDISWANDisplayName          = "Remote Access WAN Wrapper"
    ProductRASASYMACDescription         = "Windows NT Remote Access AsyMac
Adapter Driver"
    ProductRASASYMACTitle               = "Remote Access Mac"
    ProductRASASYMACDisplayName         = "Remote Access Mac"
    ProductRASSVRDescription            = "Windows NT Remote Access NetBios
Gateway"
    ProductRASSVRTitle                  = "Remote Access Server Service"
    ProductRASSVRDisplayName            = "Remote Access Server"
    !ProductRASISNRIPTitle              = "Remote Access NWLNKRIP Service"
    !ProductRASISNRIPDisplayName        = "Remote Access NWLNKRIP Service"
    !ProductRASISNSAPTitle              = "Remote Access NWLNKSAP Service"
    !ProductRASISNSAPDisplayName        = "Remote Access NWLNKSAP Service"
    !ProductRASARPTitle                 = "Remote Access ARP Service"
    !ProductRASARPDisplayName           = "Remote Access ARP Service"
    !ProductRASACDDisplayName           = "Remote Access Auto Connection
Driver"
    !ProductNDISTAPIDisplayName         = "Microsoft NDIS TAPI driver"
[DialogConstantsENG]
    Help        = "&Help"
    Exit        = "E&xit"
    OK          = "OK"
    HelpContext = ""
    Continue    = "C&ontinue"
```

```
    Cancel      = "C&ancel"
[ExitWarningDlgENG]
    DlgType      = "MessageBox"
    STF_MB_TITLE = "Remote Access Service Setup"
    STF_MB_TYPE  = 3
    STF_MB_ICON  = 5
    STF_MB_DEF   = 2
[SetupAbortDlgENG]
    DlgType     = "Info"
    DlgTemplate = "RESTART"
    Caption     = "Remote Access Service Setup"
    DlgText     = "Setup for Remote Access Service failed."$(!LF)$(!LF)+
                  "Please choose Exit to exit back to the Windows NT System."
    Windows     = "E&xit to &Windows NT"
[AbortMessageENG]
  AbortText  =  "Aborting Remote Access Setup. Please restart the system "+
                "to fix the problem."
[ShellCodeErrorMsgENG]
    DlgType      = "MessageBox"
    STF_MB_TITLE = "Error: "$(FunctionTitle)
    STF_MB_TEXT  = "An error occured running the shell command. "+
                   "Please restart the system to fix the problem."
    STF_MB_TYPE  = 1
    STF_MB_ICON  = 3
    STF_MB_DEF   = 1
[SuccessfulInstallENG]
  Success  = "Remote Access Service has been successfully installed. "$(!LF)$(!
LF)+
             "Use Remote Access Admin or User Manager in the Administrative "+
             "Tools Folder to assign Remote Access permissions to users."
[MsgDlgENG]
    DlgType      = "MessageBox"
    STF_MB_TYPE  = 1
    STF_MB_ICON  = 2
    STF_MB_DEF   = 2
[StatusUpdatingRegistryENG]
    CreatingRas             = "Please wait while Remote Access is installed..."
    WritingRasParams        = "Writing Remote Access parameters..."
    WritingRasParamsAdd     = "Writing Remote Access parameters...Adding RAS
binding "
    WritingRasParamsRemove  = "Writing Remote Access parameters...Removing RAS
binding "
    RemovingRas             = "Removing Remote Access Service..."
    RemovingAdapters        = "Remote Access Setup: Removing RAS binding "
    RemovingResources       = "Removing Remote Access resource files..."
    UpdatingRas             = "Please wait while Remote Access is updated..."
[StatusDeterminingConfigENG]
    ReadingConfig      = "Determining current configuration ..."
[StatusCopyingResENG]
    Status = "Copying DLLs and INF files to disk ..."
[NonFatalError1ENG]
    NonFatal  = "Setup cannot function properly without a user name."
[FatalErrorMemENG]
    Fatal  = "Not enough memory is available. Remote Access requires atleast 2
Meg."
[NoConfigAdminENG]
    NoConfigAdmin = "The only component installed currently is RasAdmin which "+
                    "cannot be configured."
[VerExistsENG]
    Ver   = " version "
    Text1 = " is already installed. "
[NetworkConfigErrorENG]
    Text1 = "Setup encountered an error while determining the "+
            "network configuration.  Please make sure that "+
```

```
            "the network is properly configured before attempting "+
            "to install Remote Access Service."
    Text2 = "The Microsoft Windows Workstation and Server Service must be "+
            "installed for proper operation of Remote Access Service."$(!LF)$(!
LF)+
            "To install Workstation or Server Service:"$(!LF)$(!LF)+
            "1. Invoke 'Network' control panel applet."$(!LF)+
            "2. Choose Add Software and select Workstation or Server from the
list."
[NoPortsConfiguredENG]
   NoPortsError =+
       "Remote Access is not configured with any ports. "+
       "You have an invalid RAS configuration. "+
       "Please configure Remote Access Service and add "+
       "ports using the Network option in Control Panel "+
       "after restarting the system."
[QueryInstalledProtocolsErrorENG]
    Text  = "Setup encountered an error while determining the "+
            "installed network protocols.  Try restarting the "+
            "system to correct the problem. Contact your system "+
            "administrator, if the problem persists."
[RasUpgradeENG]
    Ver = " ver "
    Text1 = " exists on the system. Are you sure you want to upgrade to"
    Text2 = "?. "
    Text3 = "Selecting YES will result in the new version replacing the old."
[InstallLoopbackENG]
Message = "RAS Setup has discovered that you have configured one or more "+
          "ports to receive calls and your computer does not have a "+
          "network adapter installed. Setup will install the MS Loopback "+
          "Adapter to enable remote clients to use resources on this computer."
Error   = "RAS setup encountered an error while invoking the OEMNADLB.INF file
"+
          "to install the MS Loopback Adapter. You can install the Loopback
Adapter "+
          "from the Network option in Control Panel after restarting the system.
"+
          "Remote Clients will not be able to dialin and use resources on this
computer "+
          "till you install the Loopback Adapter."
[NoNetbeuiDlgENG]
    NoNetbeuiWarning =+
        "The NetBEUI protocol is either not installed or not configured "+
        "for Remote Access.  NetBEUI provides the fastest throughput to "+
        "NetBIOS resources when accessed over a Remote Access connection. "+
        "If clients will access remote NetBIOS resources, configure RAS "+
        "to use NetBEUI. "$(!LF)$(!LF)+
        "You can add the NetBEUI protocol to your computer using the "+
        "Network option in Control Panel."
[NoProtocolsDlgENG]
    NoProtocolsWarning =+
       "Remote Access is not configured with any network protocols. "+
       "You have an invalid RAS configuration. "+
       "You can configure RAS to support "+
       "network protocols using the Network option in Control Panel "+
       "after restarting the system."
[UpgradeErrorsENG]
UpgradeIsdnInfoError = "Remote Access Setup encoutered an error while "+
                       "updating the ISDN information. Please remove and "+
                       "reinstall Remote Access to ensure proper operation."
[RasErrorsENG]
ErrorBadArgs = "OEMNSVRA.INF specified invalid number of arguments "+
               "to RASCFG.DLL. Remote Access Setup will terminate now."
ErrorNoPorts = "Dial-Up Networking setup failed to detect any installed "+
```

```
                "ports on this computer. Setup will terminate now."
ErrorUnknown = "Dial-Up Networking setup encountered an unknown failure "+
                "and will terminate now. Please contact your administrator "+
                "to deteremine the cause of the problem."
InvalidMode  = "Dial-Up Networking setup was invoked with an invalid "+
                "RasInstallMode environment variable. It should be set to "+
                "either install or configure. The value currently passed to "+
                "setup is "$(RasInstallMode)"."
UnableToLoadSetupdll = "Dial-Up Networking setup was unable to load setupdll.dll
"+
                       "in the system32 directory. Please make sure this file is
"+
                       "present on your computer. Setup will now exit. "
UnableToLoadNcpaCpl = "Dial-Up Networking setup was unable to load ncpa.cpl "+
                       "in the system32 directory. Please make sure this file is
"+
                       "present on your computer. Setup will now exit. "
NoProductType = "Dial-Up Networking setup was unable to determine your "+
                 "computer product type. Make sure that the following registry "+
                 "value exists on your computer. "+
                 "HEKY_LOCAL_MACHINE\System\CurrentControlSet\Control\
ProductOptions. "+
                 "Setup will now exit."
UnknownProductType = "Dial-Up Networking setup detected an unsupported product
type "$(TempProductType)". "+
                     "Make sure that the following registry value
HEKY_LOCAL_MACHINE\System\CurrentControlSet\Control\ProductOptions\ProductType
"+
                     "is one of the following valid values WinNT, LanManNT or
ServerNT. Setup will now exit."
[Billboard1ENG]
Status = "Remote Access Setup is installing the NetBEUI transport..."
[Billboard2ENG]
Status = "Remote Access Setup is installing the TCP/IP transport..."
[Billboard3ENG]
Status = "Remote Access Setup is installing NWLINK and related services..."
```