```
[Identification]
    OptionType = NetTransport
[PlatformsSupported]
    ISA
    EISA
    "Jazz-Internal Bus"
[Options]
    RASPPTP
[FileConstants]
    InfName                 = "OEMNXPPP.INF"
    UtilityInf              = "UTILITY.INF"
    ParamInf                = "NCPARAM.INF"
    subroutineinf           = "SUBROUTN.INF"
    SoftwareType            = "driver"
    Exit_Code               = 0
    NetEventDLL             = "%SystemRoot%\System32\netevent.dll"
    IoLogMsgDLL             = "%SystemRoot%\System32\IoLogMsg.dll"
    RASPPTPMsgDLL           = "%SystemRoot%\System32\raspptpl.dll;%SystemRoot%\
System32\netevent.dll"
    Manufacturer            = "Microsoft"
    ProductMajorVersion     = "4"
    ProductMinorVersion     = "0"
    ProductVersion          = $(ProductMajorVersion)"."$(ProductMinorVersion)
    ProductSoftwareName     = "RASPPTPM"
    ProductSoftwareImagePath = "\SystemRoot\System32\drivers\RASPPTPM.sys"
    NetworkCardKeyName      = $(!NTN_SoftwareBase)"\Microsoft\Windows NT\
CurrentVersion\NetworkCards"
    ProductKeyBase          = $(!NTN_SoftwareBase)"\"$(Manufacturer)
    NetRuleSoftwareType     = "RASPPTPSys RASPPTPDriver"
    NetRuleSoftwareUse      = $(SoftwareType)
    NetRuleSoftwareBindForm = """RASPPTPSys"" yes no container"
    NetRuleSoftwareClass    = {"RASPPTPDriver basic"}
    NetRuleSoftwareBindable = {"RASPPTPDriver RASPPTPAdapter non exclusive 100"}
    ProductHardwareName     = "RASPPTPM"
    NetRuleHardwareBindForm = " yes yes container"
    NetRuleHardwareRASPPTPType = "RASPPTP RASPPTPAdapter"
    NetRuleHardwareRASPPTPClass = {"RASPPTPAdapter basic"}
    BindableRASPPTPTxt      = {"RASPPTPDriver RASPPTPAdapter non exclusive 100"}
    ProductOpSupport     = 134
    ProductRASPPTPName      = "RASPPTP"
    ProductRASPPTPSvcType   = "transport"
    NetRuleRASPPTPClass     = "classRaspptpTransport ""basic"""
    NetRuleRASPPTPType      = "RaspptpTransport classRaspptpTransport"
    NetRuleRASPPTPUse       = "system"
    TapiDevices             = "TAPI DEVICES"
    TapiDeviceKey           = $(ProductHardwareName)
    TapiMediaType           = "VPN"
    ProductKeyName          = $(!NTN_SoftwareBase)"\"$(Manufacturer)"\"+
                              $(ProductSoftwareName)"\CurrentVersion"
    ParamKeyName            = $(!NTN_ServiceBase)"\"$(ProductHardwareName)+
                              "\Parameters"
    ProductRASPPTPKeyName  = $(!NTN_SoftwareBase)"\"$(Manufacturer)"\"+
                              "RASPPTP\CurrentVersion"
    LineType                = 1
    DebugFlags              = 0
[SMANFileConstants]
    PPTPProductMajorVersion         = "4"
    PPTPProductMinorVersion         = "0"
    PPTPProductVersion              = $(PPTPProductMajorVersion)"."$
```

```
(PPTPProductMinorVersion)
    PPTPProductSoftwareName         = "RASPPTPE"
    PPTPProductSoftwareImagePath    = "\SystemRoot\System32\drivers\RASPPTPE.sys"
    PPTPNetRuleSoftwareType         = "RASPPTPSys RASPPTPDriver"
    PPTPNetRuleSoftwareBindForm     = """RASPPTPSys"" yes no container"
    PPTPNetRuleSoftwareClass        = {"RASPPTPDriver basic"}
    PPTPNetRuleSoftwareBindable     = {"RASPPTPDriver RASPPTPAdapter non exclusive
100"}
    PPTPProductHardwareName         = "RASPPTPE"
    PPTPNetRuleHardwareRASPPTPType  = "RASPPTP RASPPTPAdapter"
    PPTPNetRuleHardwareRASPPTPClass = {"RASPPTPAdapter basic"}
    PPTPBindableRASPPTPTxt          = {"RASPPTPDriver RASPPTPAdapter non exclusive
100"}
    PPTPProductKeyName              = $(!NTN_SoftwareBase)"\"$(Manufacturer)"\"+
                                        $(PPTPProductSoftwareName)"\
CurrentVersion"
    PPTPParamKeyName                = $(!NTN_ServiceBase)"\"$
(PPTPProductHardwareName)+
                                        "\Parameters"
    !PPTPFProductName               = "RASPPTPF"
    !PPTPFProductImagePath          = "\SystemRoot\system32\drivers\raspptpf.sys"
    !RaspptpfKeyName                = $(!NTN_ServiceBase)"\"$(!PPTPFProductName)
[GeneralConstants]
    from                    = ""
    to                      = ""
    ExitCodeOk              = 0
    ExitCodeCancel          = 1
    ExitCodeFatal           = 2
    KeyNull                 = ""
    MAXIMUM_ALLOWED         = 33554432
    SERVICE_NO_CHANGE       = 4294967295
    RegistryErrorIndex      = NO_ERROR
    KeyProduct              = ""
    KeyParameters           = ""
    TRUE                    = 1
    FALSE                   = 0
    NoTitle                 = 0
    ExitState               = "Active"
    OldVersionExisted       = $(FALSE)
    DriverPath              = $(!STF_NTPATH)\drivers
[Date]
    Now                     = {} ? $(!LIBHANDLE) GetSystemDate
[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)
        IfStr(i) $($1) == ""
            GoTo SetOptions
        EndIf
        Set PlatformList = ^(PlatformsSupported, 1)
        IfContains(i) $($1) in $(PlatformList)
```

```
            GoTo SetOptions
        Else
            Set Status = STATUS_NOTSUPPORTED
            GoTo ExitReturnOptions
        EndIf
    Else
        Set Status = STATUS_NOLANGUAGE
        GoTo ExitReturnOptions
    EndIf
SetOptions = +
    Set OptionList     = ^(Options, 1)
    Set OptionTextList = ^(OptionsText$($0), 1)
    Set Status         = STATUS_SUCCESSFUL
ExitReturnOptions = +
    Return $(Status) $(OptionList) $(OptionTextList)
[InstallOption]
    Set Status   = STATUS_FAILED
    Set Option   = $($1)
    Set SrcDir   = $($2)
    Set RasDir   = $($2)
    Set AddCopy  = $($3)
    Set DoCopy   = $($4)
    Set DoConfig = $($5)
    Set LanguageList = ^(LanguagesSupported, 1)
    IfContains(i) $($0) NOT-IN $(LanguageList)
        Return STATUS_NOLANGUAGE
    EndIf
    Set-Subst LF = "\n"
    Read-Syms GeneralConstants
    Read-Syms FileConstants
    Read-Syms DialogConstants$(!STF_LANGUAGE)
    Read-Syms SMANFileConstants
    IfStr(i) $(!NTN_Origination) == "NCPA"
        Set Continue = $(OK)
    EndIf
    Read-Syms FileConstants$(!STF_LANGUAGE)
    Detect Date
    Set-Title  $(FunctionTitle)$(Option)
    Read-Syms SMANFileConstants$(!STF_LANGUAGE)
    Set to   = Begin
    Set from = Begin
    Set CommonStatus = STATUS_SUCCESSFUL
    EndWait
Begin = +
    Set ActivateDetection = FALSE
    IfStr(i) $(!NTN_InstallMode) == deinstall
        Set StartLabel = RemoveAdapter
    Else-IfStr(i) $(!NTN_InstallMode) == Update
        Set StartLabel = UpgradeSoftware
    Else-IfStr(i) $(!NTN_InstallMode) == bind
        Set StartLabel = BindingAdapter
    Else-IfStr(i) $(!NTN_InstallMode) == configure
        Set StartLabel = ConfigureAdapter
        Set ActivateDetection = TRUE
        Set CommonStatus = STATUS_REBOOT
        IfStr(i) $(ProductKeyName) == $(!NTN_RegBase)
            Debug-Output $(InfName)": Cannot configure the EtherWORKS 3 driver
software."
            Shell $(UtilityInf),RegistryErrorString,CANNOT_CONFIGURE_SOFTWARE
```

```
            IfInt $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output $(InfName)": ShellCode error: cannot get an error
string."
                GoTo ShellCodeError
            EndIf
            Set Error = $($R0)
            Set from = end
            Set to = end
            GoTo NonFatalInfo
        EndIf
    Else
        Set StartLabel = InstallAdapter
        Set ActivateDetection = TRUE
        Set OEM_ABANDON_OPTIONS = {}
        Set OEM_ABANDON_SOFTWARE = FALSE
        Set OEM_ABANDON_ON = TRUE
        Set OEM_ABANDON_PPTP_SOTWARE = FALSE
    EndIf
    Debug-Output $(InfName)": ================================================="
    Debug-Output $(InfName)": STF_CWDIR is: "$(!STF_CWDIR)
    Debug-Output $(InfName)": STF_LANGUAGE is: "$(!STF_LANGUAGE)
    Debug-Output $(InfName)": Option is: "$(Option)
    Debug-Output $(InfName)": !STF_NCDETECT  is: "$(!STF_NCDETECT)
    Debug-Output $(InfName)": !STF_NCOPTION  is: "$(!STF_NCOPTION)
    Debug-Output $(InfName)": !STF_NCDETCARD is: "$(!STF_NCDETCARD)
    Debug-Output $(InfName)": !STF_NCDETINFO is: "$(!STF_NCDETINFO)
    Debug-Output $(InfName)": ================================================="
    Set DetectedCard = FALSE
    IfStr(i) $(ActivateDetection) != TRUE
        GoTo $(StartLabel)
    EndIf
    Debug-Output $(InfName)": Calling Param_SetDefaults"
    Shell $(ParamInf) Param_SetDefaults {}
    IfStr(i) $(!STF_NCDETECT) == YES
        IfStr(i) $(!STF_NCOPTION) == $(Option)
           Set DetectedCard = TRUE
           Debug-Output $(InfName)": Setting DetectedCard to TRUE"
        EndIf
    EndIf
    Shell "" DebugConfiguration "After parameter querying"
    Set from = FatalError
    Set to = FatalError
    GoTo $(StartLabel)
InstallAdapter = +
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED) KeyProduct
    IfStr $(KeyProduct) != $(KeyNull)
        CloseRegKey $(KeyProduct)
        IfStr(i) !(NTN_RegBase) == $(ProductKeyName)
           Shell $(UtilityInf), VerExistedDlg, $(ProductSoftwareTitle),+
               $(ProductVersion)
           IfInt $($ShellCode) != $(!SHELL_CODE_OK)
               Debug-Output $(InfName)": ShellCode error: cannot get an error
string."
               GoTo ShellCodeError
           EndIf
           GoTo end
        Else
           Shell $(UtilityInf), CardExistedDlg
           IfInt $($ShellCode) != $(!SHELL_CODE_OK)
```

```
                Debug-Output $(InfName)": ShellCode error: cannot get an error
string."
                GoTo ShellCodeError
            EndIf
            IfStr(i) $($R1) != "OK"
                Set CommonStatus = STATUS_USERCANCEL
                GoTo end
            EndIf
            Set OldVersionExisted = $(TRUE)
        EndIf
    EndIf
    Set CurrParamSettings = {}
    IfStr(i) $(DetectedCard) != TRUE
        GoTo AdapterSetup
    EndIf
    StartWait
        Shell $(ParamInf) Param_QueryCard $(!STF_NCDETCARD)
    EndWait
    IfStr(i) $($R0) != STATUS_SUCCESSFUL
        GoTo AdapterSetup
    EndIf
    Set DetectedParams = $($R1)
    Debug-Output $(InfName)": Calling Param_SetDefaults to merge detected params"
    Shell $(ParamInf) Param_SetDefaults $(DetectedParams)
    GoTo AdapterSetup
ConfigureAdapter = +
    Read-Syms InvokeConfigureDlg$(!STF_LANGUAGE)
    LoadLibrary "x" $(!STF_CWDDIR)raspptpc.dll RASPPTPDLGHANDLE
    Debug-Output $(InfName)": Getting info from GetChassisConfiguration"
    Debug-Output $(InfName)": NTN_InstallMode is "$(!NTN_InstallMode)
    LibraryProcedure Result, $(RASPPTPDLGHANDLE), GetChassisConfiguration $(!
NTN_InstallMode)
    Set NumberOfLineDevices = *($(Result), 1)
    Set AdapterNum = *($(Result), 2)
    FreeLibrary $(RASPPTPDLGHANDLE)
    ifstr(i) $(NumberOfLineDevices) != "EXITSETUP"
      ifstr(i) $(NumberOfLineDevices) != "RASNOCHANGE"
          Shell "" UpdateAddressList $(NumberOfLineDevices) $(AdapterNum) $
(ProductSoftwareName) +
                                                    $(TapiMediaType)
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "Cannot update address list for RAS"
          goto ShellCodeError
          endif
      endif
        Set CommonStatus = STATUS_SUCCESSFUL
    Else
        Set CommonStatus = STATUS_USERCANCEL
    EndIf
    goto end
AdapterSetup = +
    Shell "" DebugConfiguration "before Param_ParameterConfidence"
    Shell $(ParamInf) Param_ParameterConfidence
    IfStr(i) $($R0) != STATUS_SUCCESSFUL
        Debug-Output $(InfName)": parameter confidence too low to bypass
configuration"
      Shell "" DebugConfiguration "before AdapterOptions"
        GoTo AdapterOptions
    EndIf
```

```
    IfStr(i) $(DetectedCard) == TRUE
        IfStr(i) $(!STF_INSTALL_MODE) != CUSTOM
          Shell "" DebugConfiguration "before AdapterVerify"
            GoTo AdapterVerify
        EndIf
    EndIf
AdapterOptions = +
    Shell "" DebugConfiguration "inside AdapterOptions"
    Read-Syms FileDependentDlg$(!STF_LANGUAGE)
AdapterVerify = +
    Shell "" DebugConfiguration "inside AdapterVerify"
    Shell "" DebugConfiguration "after running dialog"
    IfStr(i) $(DetectedCard) != TRUE
        Shell $(ParamInf) Param_SaveValues
        Set NewParamSettings = $($R0)
        IfStr(i) $(CurrParamSettings) == {}
            Set DiffParamSettings = $(NewParamSettings)
        Else
            Shell $(ParamInf) Param_DiffValues $(CurrParamSettings)
            Set DiffParamSettings = $($R0)
        EndIf
        Debug-Output $(InfName)": Calling Param_VerifyResources"
        Shell $(ParamInf) Param_VerifyResources $(DiffParamSettings)
        IfStr(i) $($R0) == STATUS_SUCCESSFUL
            Debug-Output $(InfName)": Param_VerifyResources succeeded"
            GoTo SkipOptions
        EndIf
    Else
        Set CardVerifyIndex = $(!STF_NCDETCARD)
        Debug-Output $(InfName)": Calling Param_VerifyCard"
        Shell $(ParamInf) Param_VerifyCard $(!STF_NCDETCARD)
        IfStr(i) $($R0) == STATUS_SUCCESSFUL
            Debug-Output $(InfName)": Param_VerifyCard succeeded"
            GoTo SkipOptions
        EndIf
    EndIf
    Set from = AdapterOptions
    Set to = SkipOptions
    Shell $(UtilityInf),RegistryErrorString,VERIFY_WARNING
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error: cannot get an error string."
        GoTo ShellCodeError
    EndIf
    Set Error = $($R0)
    GoTo WarningMsg
SkipOptions =+
    IfInt $(OldVersionExisted) == $(TRUE)
        IfStr(i) $(!NTN_InstallMode) == configure
            GoTo WriteParameters
        EndIf
    EndIf
    StartWait
    IfInt $(OldVersionExisted) == $(FALSE)
        IfStr(i) $(!NTN_InstallMode) == "install"
           IfStr(i) $(DoCopy) == "YES"
              Shell $(UtilityInf), DoAskSource, $(!STF_CWDDIR), $(SrcDir) 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 FatalError
         Else-IfStr(i) $($R0) == STATUS_USERCANCEL
            GoTo SuccessfulOption
         EndIf
         Set SrcDir = $($R1)
      EndIf
      Install "Install-Option"
      IfStr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
         Shell $(UtilityInf) RegistryErrorString "UNABLE_COPY_FILE"
         IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
         EndIf
         Set Error = $($R0)
         GoTo FatalError
      EndIf
   EndIf
   Shell $(UtilityInf), InstallSoftwareProduct, $(Manufacturer),+
         $(ProductRASPPTPName), $(InfName)
   ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "InstallSoftware bombed out."
         goto ShellCodeError
   endif
   set RegistryErrorIndex = $($R0)
   Ifstr(i) $(RegistryErrorIndex) != "NO_ERROR"
       Debug-Output "REGISTRY PROBLEM"
       CloseRegKey $($R1)
       CloseRegKey $($R2)
       goto FatalRegistry
   endif
   Set SoftProductKey    = $($R1)
   Set SoftNetRuleKey    = $($R2)
   set NewValueList = +
       {{Infname ,$(NoTitle),$(!REG_VT_SZ),$(InfName)},+
        {SoftwareType,$(NoTitle),$(!REG_VT_SZ),$(ProductRASPPTPSvcType)},+
        {Title,$(NoTitle),$(!REG_VT_SZ), $(ProductRASPPTPTitle)},+
        {Description,$(NoTitle),$(!REG_VT_SZ),$(ProductRASPPTPDescription)},+
        {OperationsSupport,$(NoTitle),$(!REG_VT_DWORD),$(ProductOpSupport)},+
        {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$(ProductMajorVersion)},+
        {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$(ProductMinorVersion)},+
        {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),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), $(NetRuleRASPPTPClass)},+
```

```
            {type,$(NoTitle),$(!REG_VT_SZ),$(NetRuleRASPPTPType)},+
            {use,$(NoTitle),$(!REG_VT_SZ),$(NetRuleRASPPTPUse)}, +
            {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}, +
            {Infname ,$(NoTitle),$(!REG_VT_SZ),$(InfName)}}
        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
        Shell $(UtilityInf), AddSoftwareComponent, $(Manufacturer), +
            $(ProductSoftwareName), +
            $(ProductSoftwareName), +
            $(ProductSoftwareTitle), $(STF_CONTEXTINFNAME), +
            $(ProductSoftwareImagePath), "kernelautostart", "NDIS", {}, "",+
            $(RASPPTPMsgDLL)
        Set OEM_ABANDON_SOFTWARE = TRUE
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        Set KeyProduct      = $($R1)
        Set SoftNetRulesKey = $($R2)
        CloseRegKey $($R3)
        CloseRegKey $($R4)
        CloseRegKey $($R5)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            EndWait
            Debug-Output $(InfName)": Registry error: add software components"
            CloseRegKey $(KeyProduct)
            CloseRegKey $(SoftNetRulesKey)
            GoTo FatalRegistry
        EndIf
        Set NewValueList = {{SoftwareType,$(NoTitle),$(!REG_VT_SZ),$
(SoftwareType)},+
                            {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMajorVersion)},+
                            {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMinorVersion)},+
                            {Title,$(NoTitle),$(!REG_VT_SZ),$
(ProductSoftwareTitle)},+
                            {Description,$(NoTitle),$(!REG_VT_SZ),$
(ProductSoftwareDescription)},+
                            {ServiceName,$(NoTitle),$(!REG_VT_SZ),$
(ProductSoftwareName)},+
                            {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),1)}}
        Shell  $(UtilityInf), AddValueList, $(KeyProduct), $(NewValueList)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error."
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            EndWait
            Debug-Output $(InfName)": Registry error: add value list."
```

```
            CloseRegKey $(KeyProduct)
            CloseRegKey $(SoftNetRulesKey)
            GoTo FatalRegistry
        EndIf
        Set NewValueList = {{type,$(NoTitle),$(!REG_VT_SZ),$(NetRuleSoftwareType)},
+
                            {use,$(NoTitle),$(!REG_VT_SZ),$(NetRuleSoftwareUse)}, +
                            {bindform,$(NoTitle),$(!REG_VT_SZ),$
(NetRuleSoftwareBindForm)}, +
                            {class,$(NoTitle),$(!REG_VT_MULTI_SZ),$
(NetRuleSoftwareClass)}, +
                            {bindable,$(NoTitle),$(!REG_VT_MULTI_SZ),$(Bindable$
(Option)Txt)}, +
                            {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}}
        Shell  $(UtilityInf), AddValueList, $(SoftNetRulesKey), $(NewValueList)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error."
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        CloseRegKey $(KeyProduct)
        CloseRegKey $(SoftNetRulesKey)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            EndWait
            Debug-Output $(InfName)": Resgitry error: add value list."
            GoTo FatalRegistry
        EndIf
        Shell "" AddServiceDependency $(ProductSoftwareName) $
(PPTPProductSoftwareName)
        Shell $(UtilityInf), AddSoftwareComponent, $(Manufacturer), +
            $(PPTPProductSoftwareName), +
            $(PPTPProductSoftwareName), +
            $(PPTPProductSoftwareTitle), $(STF_CONTEXTINFNAME), +
            $(PPTPProductSoftwareImagePath), "kernelauto", "NDIS", {}, "",+
            $(RASPPTPMsgDLL)
        Set OEM_ABANDON_PPTP_SOFTWARE = TRUE
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        Set PPTPKeyProduct      = $($R1)
        Set PPTPSoftNetRulesKey = $($R2)
        CloseRegKey $($R3)
        CloseRegKey $($R4)
        CloseRegKey $($R5)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            EndWait
            Debug-Output $(InfName)": Registry error: add sman software components"
            CloseRegKey $(PPTPKeyProduct)
            CloseRegKey $(PPTPSoftNetRulesKey)
            GoTo FatalRegistry
        EndIf
        Set NewValueList = {{SoftwareType,$(NoTitle),$(!REG_VT_SZ),$
(SoftwareType)},+
                            {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(PPTPProductMajorVersion)},+
                            {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(PPTPProductMinorVersion)},+
```

```
                            {Title,$(NoTitle),$(!REG_VT_SZ),$
(PPTPProductSoftwareTitle)},+
                            {Description,$(NoTitle),$(!REG_VT_SZ),$
(PPTPProductSoftwareDescription)},+
                            {ServiceName,$(NoTitle),$(!REG_VT_SZ),$
(PPTPProductSoftwareName)},+
                            {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),1)}}
        Shell  $(UtilityInf), AddValueList, $(PPTPKeyProduct), $(NewValueList)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error."
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            EndWait
            Debug-Output $(InfName)": Registry error: add value list."
            CloseRegKey $(PPTPKeyProduct)
            CloseRegKey $(PPTPSoftNetRulesKey)
            GoTo FatalRegistry
        EndIf
        Shell "" InstallRASPPTPFDriver
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output "Error shelling InstallRASPPTPFDriver"
             goto ShellCodeError
        endif
        ifstr(i) $($R0) != STATUS_SUCCESSFUL
             Debug-Output "Error from InstallRASPPTPFDriver"
             goto end
        endif
    EndIf
    Shell $(UtilityInf), AddHardwareComponent, $(ProductHardwareName),$
(STF_CONTEXTINFNAME),$(ProductKeyName)
    IfInt $($R4) != -1
        Set OEM_ABANDON_OPTIONS = >($(OEM_ABANDON_OPTIONS), $(!NTN_SoftwareBase)"\
Microsoft\Windows NT\CurrentVersion\NetworkCards\"$($R4))
    EndIf
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": Cannot add hardware component"
        GoTo ShellCodeError
    EndIf
    Set RegistryErrorIndex = $($R0)
    IfStr(i) $(RegistryErrorIndex) != NO_ERROR
        EndWait
        Debug-Output $(InfName)": Registry error: add hardware component"
        CloseRegKey $($R1)
        CloseRegKey $($R2)
        CloseRegKey $($R3)
        GoTo FatalRegistry
    EndIf
    Set KeyParameters = $($R3)
    Set KeyAdapterRules = $($R2)
    Set AdapterNumber = $($R4)
    Set NewValueList = {{Manufacturer,$(NoTitle),$(!REG_VT_SZ),$(Manufacturer)},+
                        {Title,$(NoTitle),$(!REG_VT_SZ),"["$($R4)"] "$
(ProductHardware$(Option)Title)},+
                        {Description,$(NoTitle),$(!REG_VT_SZ),$(ProductHardware$
(Option)Description)},+
                        {ProductName,$(NoTitle),$(!REG_VT_SZ),$
(ProductHardwareName)},+
```

```
                        {ServiceName,$(NoTitle),$(!REG_VT_SZ),$($R5)},+
                        {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),1)},+
                        {Hidden,$(NoTitle),$(!REG_VT_DWORD),1}}
    Shell  $(UtilityInf), AddValueList, $($R1), $(NewValueList)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error"
        GoTo ShellCodeError
    EndIf
    CloseRegKey $($R1)
    Set TempProdName = """"$(ProductHardwareName)$(AdapterNumber)""""
    Set TempBindForm = $(TempProdName)$(NetRuleHardwareBindForm)
    Set NewValueList = {{type,$(NoTitle),$(!REG_VT_SZ),$(NetRuleHardware$
(Option)Type)},+
                         {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)}, +
                         {class,$(NoTitle),$(!REG_VT_MULTI_SZ),$(NetRuleHardware$
(Option)Class)}, +
                         {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}}
    Shell  $(UtilityInf), AddValueList, $(KeyAdapterRules), $(NewValueList)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error."
        GoTo ShellCodeError
    EndIf
    Set RegistryErrorIndex = $($R0)
    IfStr(i) $(RegistryErrorIndex) != NO_ERROR
        EndWait
        Debug-Output $(InfName)": Registry error: add value list."
        CloseRegKey $(KeyParameters)
        CloseRegKey $(KeyAdapterRules)
        GoTo FatalRegistry
    EndIf
    CloseRegKey $(KeyAdapterRules)
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED) KeyProduct
    IfStr $(KeyProduct) != $(KeyNull)
        SetRegValue $(KeyProduct) {NetCard,$(NoTitle),$(!REG_VT_DWORD),$
(AdapterNumber)}
        CloseRegKey $(KeyProduct)
    EndIf
    GoTo WriteParameters
WriteParameters = +
    LoadLibrary "x" $(!STF_CWDDIR)raspptpc.dll RASPPTPDLGHANDLE
    Debug-Output $(InfName)": Getting info from GetChassisConfiguration"
    Debug-Output "!STF_UNATTENDED is "$(!STF_UNATTENDED)
    Debug-Output "!STF_GUI_UNATTENDED is "$(!STF_GUI_UNATTENDED)
    Debug-Output "!STF_UNATTENDED_SECTION is "$(!STF_UNATTENDED_SECTION)
    LibraryProcedure Result, $(RASPPTPDLGHANDLE), GetChassisConfiguration $(!
NTN_InstallMode) +
                              $(!STF_GUI_UNATTENDED) $(!STF_UNATTENDED) $(!
STF_UNATTENDED_SECTION)
    Debug-Output $(InfName)": Result value is :"$(Result)
    Set NumberOfLineDevices = *($(Result), 1)
    FreeLibrary $(RASPPTPDLGHANDLE)
    ifstr(i) $(NumberOfLineDevices) == "EXITSETUP"
        Read-Syms InvokeCancelDlg$(!STF_LANGUAGE)
        Debug-Output $(InfName)": GetChassisConfiguration User selected ExitSetup"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", $
(InvokeCancelMsg)
        GoTo SetFailed
    endif
    Debug-Output $(InfName)": Number Of Line Devices are :"$(NumberOfLineDevices)
```

```
    Shell "" CreateFullTapiAddrList $(AdapterNumber)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "Cannot create the TAPI address list"
        goto ShellCodeError
    endif
    Set FullTapiAddrList = $($R0)
    Set TapiAddressList = {}
    ForListDo $(FullTapiAddrList)
        Set CurListItem = $(AdapterNumber)"-"$($)"-0"
        Set CurListNumber = $(#)
        IfInt $(CurListNumber) <= $(NumberOfLineDevices)
            Set TapiAddressList = >($(TapiAddressList), $(CurListItem))
        Endif
    EndForListDo
    Debug-Output $(InfName)": Modified Tapi Address List is :"$(TapiAddressList)
    Set NewValueList = {+
                        {AddressList,$(NoTitle),$(!REG_VT_MULTI_SZ),$
(TapiAddressList)},+
                        {DeviceName,$(NoTitle),$(!REG_VT_SZ),$
(ProductHardwareName)},+
                        {LineType,$(NoTitle),$(!REG_VT_DWORD),$(LineType)},+
                        {MediaType,$(NoTitle),$(!REG_VT_SZ),$(TapiMediaType)},+
                        }
    Shell  $(UtilityInf), AddValueList, $(KeyParameters), $(NewValueList)
    CloseRegKey $(KeyParameters)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error."
        GoTo ShellCodeError
    EndIf
    Set RegistryErrorIndex = $($R0)
    IfStr(i) $(RegistryErrorIndex) != NO_ERROR
        Debug-Output $(InfName)": Registry error: Add value list"
        GoTo FatalRegistry
    EndIf
    IfStr(i) $(!NTN_InstallMode) == configure
        GoTo SuccessfulOption
    EndIf
    OpenRegKey $(!REG_H_LOCAL) "" "SYSTEM\CurrentControlSet\Services\TcpIp\
Parameters" $(MAXIMUM_ALLOWED) TcpKey
    IfStr(i) $(TcpKey) != ""
        Debug-Output $(InfName)": setting tcpip\parameters\
PPTPTcpMaxDataRetransmissions to 9"
        SetRegValue $(TcpKey) {PPTPTcpMaxDataRetransmissions,$(NoTitle),$(!
REG_VT_DWORD), 9}
        CloseRegKey $(TcpKey)
    Else
        Debug-Output $(InfName)": Registry error: can't open services\tcpip key"
    EndIf
    OpenRegKey $(!REG_H_LOCAL) "" "SOFTWARE\Microsoft" $(MAXIMUM_ALLOWED) BaseKey
    shell "" HtCreateRegKey $(BaseKey) "TAPI DEVICES\RASPPTPM"
    IfStr(i) $($R0) != NO_ERROR
        Debug-Output $(InfName)": Error creating registry key!"
        GoTo FatalRegistry
    EndIf
    Set TapiDeviceKey = $($R1)
    Set NewValueList = {+
                        {Address,$(NoTitle),$(!REG_VT_MULTI_SZ),$(TapiAddressList)},
+
                        {"Media Type",$(NoTitle),$(!REG_VT_SZ),$(TapiMediaType)}}
```

```
    Shell  $(UtilityInf), AddValueList, $(TapiDeviceKey), $(NewValueList)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error."
        GoTo ShellCodeError
    EndIf
    Set RegistryErrorIndex = $($R0)
    IfStr(i) $(RegistryErrorIndex) != NO_ERROR
        Debug-Output $(InfName)": Registry error: Add value list"
        GoTo FatalRegistry
    EndIf
    CloseRegKey $(TapiDeviceKey)
    CloseRegKey $(BaseKey)
    ifstr(i) $(!STF_INSTALL_MODE) == EXPRESS
         GoTo InstallRas
    Else
         Shell "" InstallTCPIPIfNotInstalled $(SrcDir) $(AddCopy) $(DoCopy) $
(DoConfig)
         ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 Debug-Output "Cannot install TCP/IP if not already installed"
             goto ShellCodeError
         endif
         ifstr(i) $($R0) != STATUS_SUCCESSFUL
            Debug-Output "Error installing TCP/IP"
             set Error = "Error installing TCP/IP"
             goto FatalError
         endif
         Shell "" DebugGlobalFlags "After TCP/IP"
    EndIf
InstallRas = +
    ifstr(i) $(!STF_INSTALL_MODE) == EXPRESS
         GoTo SuccessfulOption
    Else
         Read-Syms InvokeRasDlg$(!STF_LANGUAGE)
         Shell "oemnsvra.inf" CheckRasInstalled
         IfInt $($ShellCode) != $(!SHELL_CODE_OK)
             Debug-Output $(InfName)": Error Shelling the RAS INF file
oemnsvra.inf"
             Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE),  +
                     "STATUS", $(InvokeRasError)
             IfInt $($ShellCode) != $(!SHELL_CODE_OK)
                 GoTo ShellCodeError
             EndIf
             GoTo RASSetup
         EndIf
         Set RasInstalled = $($R0)
         Debug-Output $(InfName)": Is RAS Installed? "$(RasInstalled)
         IfStr(i) $(RasInstalled) == FALSE
             Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", +
                     $(InvokeRasSetupMsg)
             IfInt $($ShellCode) != $(!SHELL_CODE_OK)
                 GoTo RASSetup
             EndIf
         Else
             Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", +
                     $(InvokeRasConfigMsg)
             IfInt $($ShellCode) != $(!SHELL_CODE_OK)
                 GoTo RASSetup
             EndIf
         EndIf
```

```
         EndWait
         GoTo RASSetup
    Endif
RASSetup =+
    Set AddCopy  = YES
    Set DoCopy   = YES
    Set DoConfig = YES
    IfStr(i) $(RasInstalled) == TRUE
       Set SaveNTN_InstallMode = $(!NTN_InstallMode)
       Set !NTN_InstallMode = configure
    EndIf
    Set PREV_SRCDIR = $(!STF_SRCDIR_OVERRIDE)
    Set !STF_SRCDIR_OVERRIDE = $(SrcDir)
    Shell "oemnsvra.inf" InstallOption $(!STF_LANGUAGE) "RAS" $(SrcDir) $(AddCopy)
$(DoCopy) $(DoConfig)
    set !STF_SRCDIR_OVERRIDE = $(PREV_SRCDIR)
    IfStr(i) $(RasInstalled) == TRUE
            Set !NTN_InstallMode = $(SaveNTN_InstallMode)
    EndIf
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": Error Shelling the RAS INF file oemnsvra.inf"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS",  +
                $(InvokeRasError)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
        GoTo SuccessfulOption
    EndIf
    EndWait
    GoTo SuccessfulOption
BindingAdapter =+
    Set Error = "Binding: Sorry, not yet implemented."
    GoTo FatalError
RemoveAdapter = +
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED) KeyProduct
    IfStr $(KeyProduct) != $(KeyNull)
        GetRegValue $(KeyProduct),"NetCard", NetCardInfo
        Set NetCard = *($(NetCardInfo), 4)
        Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
            $(ProductSoftwareName), $(NetworkCardKeyName)"\"$(NetCard)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            GoTo FatalRegistry
        EndIf
    EndIf
    Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
      $(PPTPProductSoftwareName)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
      Debug-Output $(InfName)": ShellCode error"
        GoTo ShellCodeError
    EndIf
    Set RegistryErrorIndex = $($R0)
    IfStr(i) $(RegistryErrorIndex) != NO_ERROR
          GoTo FatalRegistry
    EndIf
```

```
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyBase) +
               $(MAXIMUM_ALLOWED) ProductKey
    Ifstr $(ProductKey) == $(KeyNull)
       Debug-Output "OEMNXPPP.INF: could not open Software product key"
    else
       DeleteRegTree $(ProductKey) $(ProductRASPPTPName)
       CloseRegKey $(ProductKey)
    endif
    Shell "" RemoveRASPPTPFDriver
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
         Debug-Output "Error shelling RemoveRASPPTPFDriver"
         goto ShellCodeError
    endif
    ifstr(i) $($R0) != STATUS_SUCCESSFUL
         Debug-Output "Error from RemoveRASPPTPFDriver"
         goto end
    endif
    OpenRegKey $(!REG_H_LOCAL) "" "SOFTWARE\Microsoft\TAPI DEVICES" $
(MAXIMUM_ALLOWED) BaseKey
    IfStr $(BaseKey) != $(KeyNull)
        DeleteRegTree $(BaseKey) $(ProductSoftwareName)
    EndIf
    OpenRegKey $(!REG_H_LOCAL) "" "SYSTEM\CurrentControlSet\Services\TcpIp\
Parameters" $(MAXIMUM_ALLOWED) TcpKey
    IfStr(i) $(TcpKey) != ""
        Debug-Output $(InfName)": removing tcpip\parameters\
PPTPTcpMaxDataRetransmissions value"
      DeleteRegValue $(TcpKey) "PPTPTcpMaxDataRetransmissions"
        CloseRegKey $(TcpKey)
    Else
        Debug-Output $(InfName)": Registry error: can't open services\tcpip key"
    EndIf
    Read-Syms InvokeRasDlg$(!STF_LANGUAGE)
    Shell "oemnsvra.inf" CheckRasInstalled
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": Error Shelling the RAS INF file oemnsvra.inf"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE),  +
                "STATUS", $(InvokeRasError)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
        GoTo RASSetup2
    EndIf
    Set RasInstalled = $($R0)
    Debug-Output $(InfName)": Is RAS Installed? "$(RasInstalled)
    IfStr(i) $(RasInstalled) == FALSE
        goto end
    Else
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", +
                $(InvokeRasConfigMsg2)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo RASSetup2
        EndIf
    EndIf
    EndWait
    GoTo RASSetup2
RASSetup2 =+
    Set AddCopy  = NO
    Set DoCopy   = NO
```

```
    Set DoConfig = NO
    Set SaveNTN_InstallMode = $(!NTN_InstallMode)
    Set !NTN_InstallMode = configure
    Shell "oemnsvra.inf" InstallOption $(!STF_LANGUAGE) "RAS" $(SrcDir) $(AddCopy)
$(DoCopy) $(DoConfig)
    Set !NTN_InstallMode = $(SaveNTN_InstallMode)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": Error Shelling the RAS INF file oemnsvra.inf"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS",  +
                $(InvokeRasError)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
        GoTo end
    EndIf
    GoTo end
UpgradeSoftware = +
        Debug-Output $(InfName)": comes into 1st part of upgrade"
        OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED)
KeyProduct
        IfStr $(KeyProduct) != $(KeyNull)
            GetRegValue $(KeyProduct),"MajorVersion", VersionInfo
            Set Version = *($(VersionInfo), 4)
            Shell $(UtilityInf), GetInfFileNameFromRegistry, $(KeyProduct)
            IfInt $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output $(InfName)": ShellCode error"
                GoTo ShellCodeError
            EndIf
            Set !UG_Filename = $($R0)
            IfStr(i) $(!UG_Filename) != ""
                Debug-Output $(InfName)": starts to install"
                install "Install-Update"
                IfStr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
                    GoTo FatalError
                EndIf
                LibraryProcedure Result, $(!LIBHANDLE), SetupChangeServiceConfig, $
(!PPTPFProductName) +
                                          $(SERVICE_NO_CHANGE), $
(SERVICE_NO_CHANGE), $(SERVICE_NO_CHANGE), +
                                          "", "TDI", "", "", "", ""
            EndIf
            SetRegValue $(KeyProduct) {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMajorVersion)}
            SetRegValue $(KeyProduct) {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMinorVersion)}
            IfInt $(Version) != $(ProductVersion)
            EndIf
            CloseRegKey $(KeyProduct)
        Else
            GoTo FatalRegistry
        EndIf
        Debug-Output $(InfName)": comes into PPTPE part of upgrade"
        OpenRegKey $(!REG_H_LOCAL) "" $(PPTPProductKeyName) $(MAXIMUM_ALLOWED)
PPTPKeyProduct
        IfStr $(PPTPKeyProduct) != $(KeyNull)
            SetRegValue $(PPTPKeyProduct) {MajorVersion,$(NoTitle),$(!
REG_VT_DWORD),$(PPTPProductMajorVersion)}
            SetRegValue $(PPTPKeyProduct) {MinorVersion,$(NoTitle),$(!
REG_VT_DWORD),$(PPTPProductMinorVersion)}
```

```
            CloseRegKey $(PPTPKeyProduct)
        Else
            GoTo FatalRegistry
        EndIf
        Debug-Output $(InfName)": comes into PPTP part of upgrade"
        OpenRegKey $(!REG_H_LOCAL) "" $(ProductRASPPTPKeyName) $(MAXIMUM_ALLOWED)
PPTPKeyProduct
        IfStr $(PPTPKeyProduct) != $(KeyNull)
            SetRegValue $(PPTPKeyProduct) {MajorVersion,$(NoTitle),$(!
REG_VT_DWORD),$(PPTPProductMajorVersion)}
            SetRegValue $(PPTPKeyProduct) {MinorVersion,$(NoTitle),$(!
REG_VT_DWORD),$(PPTPProductMinorVersion)}
            SetRegValue $(PPTPKeyProduct) {Description,$(NoTitle),$(!REG_VT_SZ),$
(ProductRASPPTPDescription)}
            SetRegValue $(PPTPKeyProduct) {OperationsSupport,$(NoTitle),$(!
REG_VT_DWORD),$(ProductOpSupport)}
            CloseRegKey $(PPTPKeyProduct)
        Else
            GoTo FatalRegistry
        EndIf
    GoTo end
SuccessfulOption = +
    GoTo end
Abandon = +
    ForListDo $(OEM_ABANDON_OPTIONS)
        Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
            $(ProductSoftwareName), $($)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            GoTo FatalRegistry
        EndIf
    EndForListDo
    IfStr(i) $(OEM_ABANDON_SOFTWARE) == TRUE
        Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
            $(ProductSoftwareName), FALSE
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            GoTo FatalRegistry
        EndIf
    EndIf
    IfStr(i) $(OEM_ABANDON_PPTP_SOFTWARE) == TRUE
        Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
            $(PPTPProductSoftwareName), FALSE
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)": ShellCode error"
            GoTo ShellCodeError
        EndIf
        Set RegistryErrorIndex = $($R0)
        IfStr(i) $(RegistryErrorIndex) != NO_ERROR
            GoTo FatalRegistry
        EndIf
```

```
    EndIf
    GoTo end
WarningMsg = +
    Shell $(subroutineinf) 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)
    EndIf
    GoTo end
NonFatalInfo = +
    Set Severity = STATUS
    Set CommonStatus = STATUS_USERCANCEL
    IfStr(i) $(Error) == ""
        Set Severity = NONFATAL
        Shell $(UtilityInf) RegistryErrorString "SETUP_FAIL"
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
        Set Error = $($R0)
    EndIf
    Shell $(subroutineinf) SetupMessage, $(!STF_LANGUAGE), $(Severity), $(Error)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        GoTo ShellCodeError
    EndIf
    IfStr(i) $($R1) == "OK"
        GoTo $(from)
    EndIf
    GoTo end
FatalRegistry = +
    Shell $(UtilityInf) RegistryErrorString $(RegistryErrorIndex)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        GoTo ShellCodeError
    EndIf
    Set Error = $($R0)
    GoTo FatalError
FatalDetect = +
    Shell $(UtilityInf),RegistryErrorString,CANNOT_DETECT
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": ShellCode error: cannot get an error string."
        GoTo ShellCodeError
    EndIf
    Set Error = $($R0)
    GoTo FatalError
FatalError = +
    IfStr(i) $(Error) == ""
        Shell $(UtilityInf) RegistryErrorString "SETUP_FAIL"
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
        Set Error = $($R0)
    EndIf
    Shell $(subroutineinf) SetupMessage, $(!STF_LANGUAGE), "FATAL", $(Error)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        GoTo ShellCodeError
    EndIf
```

```
    GoTo SetFailed
ShellCodeError = +
    Set DlgType      = "MessageBox"
    Set STF_MB_TITLE = $(ShellCodeErrorTitle)
    Set STF_MB_TEXT  = $(ShellCodeErrorText)
    Set STF_MB_TYPE  = 1
    Set STF_MB_ICON  = 3
    Set STF_MB_DEF   = 1
    UI Start "Error Message"
    GoTo SetFailed
SetFailed = +
    Set CommonStatus = STATUS_FAILED
    IfStr(i) $(OEM_ABANDON_ON) == TRUE
        Set OEM_ABANDON_ON = FALSE
        GoTo Abandon
    EndIf
    GoTo end
end = +
    Return $(CommonStatus)
[HtCreateRegKey]
    Debug-Output $(InfName)": Entering [HtCreateRegKey]"
    Set ECR_Result          = NO_ERROR
    Set ECR_BaseKeyHandle   = $($0)
    Set ECR_NewPath         = $($1)
    Set KeyNull             = ""
    Set MAXIMUM_ALLOWED     = 33554432
    Debug-Output $(InfName)": HtCreateRegKey - ECR_BaseKeyHandle = "$
(ECR_BaseKeyHandle)
    Debug-Output $(InfName)":                  ECR_NewPath       = "$(ECR_NewPath)
    Debug-Output $(InfName)":                  MAXIMUM_ALLOWED   = "$
(MAXIMUM_ALLOWED)
    Debug-Output $(InfName)":                  KeyNull           = "$(KeyNull)
    OpenRegKey $(ECR_BaseKeyHandle) "" $(ECR_NewPath) $(MAXIMUM_ALLOWED) +
               ECR_BaseKey
    Debug-Output $(InfName)": ECR_BaseKey = "$(ECR_BaseKey)
    Debug-Output $(InfName)":     OpenRegKey returned "$($R0)
    IfStr $(ECR_BaseKey) == $(KeyNull)
        Debug-Output $(InfName)": ECR_BaseKey == KeyNull"
    Else
        Debug-Output $(InfName)": ECR_BaseKey != KeyNull"
        Set ECR_KeyHandle = $(ECR_BaseKey)
        GoTo  ECR_Return
    EndIf
    Set ECR_TmpPath = ""
    Split-String $(ECR_NewPath) "\" ECR_PList
    Debug-Output $(InfName)": ECR_PList = "$(ECR_PList)
    ForListDo $(ECR_PList)
        IfStr(i) $($) != "\"
            IfInt $(#) == 1
                Set ECR_TmpPath = $($)
            Else
                Set ECR_TmpPath = $(ECR_TmpPath)"\"$($)
            EndIf
            Debug-Output $(InfName)": Determining if "$(ECR_TmpPath)" exists"
            OpenRegKey $(ECR_BaseKeyHandle) "" $(ECR_TmpPath) $(MAXIMUM_ALLOWED)
ECR_BaseKey
            IfStr $(ECR_BaseKey) == $(KeyNull)
                Debug-Output $(InfName)": Creating "$(ECR_TmpPath)
                CreateRegKey $(ECR_BaseKeyHandle)  {$(ECR_TmpPath),0,GenericClass}
```

```
"" $(MAXIMUM_ALLOWED) "" ECR_KeyHandle
                IfStr(i) $(ECR_KeyHandle) == $(KeyNull)
                    Set ECR_Result = $($R0)
                    GoTo ECR_Return
                EndIf
            EndIf
        EndIf
    EndForListDo
ECR_Return = +
    Return $(ECR_Result) $(ECR_KeyHandle)
[InstallTCPIPIfNotInstalled]
    Debug-Output $(InfName)"InstallTCPIPIfNotInstalled entry "
    Debug-Output $(InfName)"!NTN_ServiceBase is "$(!NTN_ServiceBase)
    set SrcDir   = $($0)
    set AddCopy  = $($1)
    set DoCopy   = $($2)
    set DoConfig = $($3)
    Read-Syms InvokeTCPIPDlg$(!STF_LANGUAGE)
    set TcpIpKeyName      = $(!NTN_ServiceBase)"\TcpIp"
    Set KeyNull        =    ""
    Set status = STATUS_FAILED
    Set TcpIpInstalled   = FALSE
      Set MAXIMUM_ALLOWED = 33554432
    set Protocol = "TCPIP"
    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
    Debug-Output $(InfName)"TCP/IP Installed? "$(TcpIpInstalled)
    Ifstr(i) $(TcpIpInstalled) == "FALSE"
        Debug-Output $(InfName)"Installing TCP/IP"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", +
            $(InvokeTCPSetupMsg)
        set InvokedByRas = FALSE
      Debug-Output $(InfName)": resetting stf_srcdir_override flag"
      Set PREV_SRCDIR = $(!STF_SRCDIR_OVERRIDE)
      Set !STF_SRCDIR_OVERRIDE = $(SrcDir)
        Shell "oemnxptc.inf" InstallOption $(!STF_LANGUAGE) "TC" $(SrcDir) $
(AddCopy) $(DoCopy) $(DoConfig) $(InvokedByRas)
      set !STF_SRCDIR_OVERRIDE = $(PREV_SRCDIR)
        Ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output $(InfName)"SHELL ERROR! during TCP/IP installation"
            Goto InstallTCPEscape
        Endif
        Set status = $($R0)
        Ifstr(i) $(status) != STATUS_SUCCESSFUL
            Ifstr(i) $(status) != STATUS_USERCANCEL
                Debug-Output $(InfName)"oemnxptc.inf  returned "$(status)
                Goto InstallTCPEscape
            Endif
        Endif
```

```
    Else
      set status = STATUS_SUCCESSFUL
    Endif
InstallTCPEscape = +
    set RetStatus = $(status)
    return $(RetStatus)
[AddUSRMediaGroupToServiceGroupOrder]
    Debug-Output $(InfName)": Entering [AddUSRMediaGroupToServiceGroupOrder]"
    Set MAXIMUM_ALLOWED     = 33554432
    Set KeyNull             = ""
    Read-Syms SMANFileConstants
    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 "AddUSRMediaGroupToServiceGroupOrder current list "$(OldList)
        Ifcontains(i) $(PPTPGroupName) not-in $(OldList)
            set NewGroupList = {}
            ForListDo $(OldList)
                ifstr(i) $($) == "NDIS"
                    set NewGroupList = >($(NewGroupList), $(PPTPGroupName))
                endif
                set NewGroupList = >($(NewGroupList), $($))
             EndForListDo
            Debug-Output "AddUSRMediaGroupToServiceGroupOrder new list "$
(NewGroupList)
            SetRegValue $(KeyGroup) {List, 0,$(!REG_VT_MULTI_SZ),$(NewGroupList)}
        endif
        CloseRegKey $(KeyGroup)
    else
        Debug-Output "AddUSRMediaGroupToServiceGroupOrder error opening
ServiceGroupOrder key."
        Shell $(subroutineinf) SetupMessage, $(!STF_LANGUAGE), "NONFATAL",
"ServiceGroupOrder Update Failed"
    endif
    return
[UpdateAddressList]
    Debug-Output $(InfName)": Entering [UpdateAddressList]"
      Set NumberLineDevices = $($0)
      Set AdapNum = $($1)
      Set DriverName = $($2)
      Set MediaType = $($3)
    Set MAXIMUM_ALLOWED     = 33554432
    Set KeyNull             = ""
      Debug-Output $(InfName)": number of line devices received value "$
(NumberLineDevices)
      Debug-Output $(InfName)": adapter number received value "$(AdapNum)
      Debug-Output $(InfName)": driver name received value "$(DriverName)
      Shell "" CreateFullTapiAddrList $(AdapNum)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
          Debug-Output "Cannot create the new TAPI address list"
        goto ShellCodeError
    endif
```

```
      Set NewTapiAddrList = $($R0)
      Set TapiAddressList = {}
      ForListDo $(NewTapiAddrList)
            Set CurListItem = $(AdapNum)"-"$($)"-0"
            Set CurListNumber = $(#)
            IfInt $(CurListNumber) <= $(NumberLineDevices)
                  Set TapiAddressList = >($(TapiAddressList), $(CurListItem))
            Endif
      EndForListDo
      Debug-Output $(InfName)": Modified new Tapi Address List is :"$
(TapiAddressList)
    set RASPPTPAddress = "SYSTEM\CurrentControlSet\Services\"$(DriverName)$
(AdapNum)"\Parameters"
      Debug-Output $(InfName)": RASPPTPAddress is "$(RASPPTPAddress)
    OpenRegKey $(!REG_H_LOCAL) "" $(RASPPTPAddress) $(MAXIMUM_ALLOWED) KeyParam
    Debug-Output $(InfName)": keyparam value is "$(KeyParam)
    Ifstr(i) $(KeyParam) != $(KeyNull)
      Debug-Output $(InfName)": entering to set usrwan2 value"
      DeleteRegValue $(KeyParam) "AddressList"
        SetRegValue $(KeyParam) {AddressList, 0,$(!REG_VT_MULTI_SZ),$
(TapiAddressList)}
      CloseRegKey $(KeyParam)
      EndIf
    OpenRegKey $(!REG_H_LOCAL) "" "SOFTWARE\Microsoft" $(MAXIMUM_ALLOWED) BaseKey
    Shell "" HtCreateRegKey $(BaseKey) "TAPI DEVICES\RASPPTPM"
    IfStr(i) $($R0) != NO_ERROR
        Debug-Output $(InfName)": Error creating registry key!"
        GoTo FatalRegistry
    EndIf
    Set TapiDeviceKey = $($R1)
      DeleteRegValue $(TapiDeviceKey) "Address"
      DeleteRegValue $(TapiDeviceKey) "Media Type"
    SetRegValue $(TapiDeviceKey) {Address, 0,$(!REG_VT_MULTI_SZ),$
(TapiAddressList)}
    SetRegValue $(TapiDeviceKey) {"Media Type", 0,$(!REG_VT_SZ),$(MediaType)}
    CloseRegKey $(TapiDeviceKey)
    CloseRegKey $(BaseKey)
    Read-Syms InvokeRasDlg$(!STF_LANGUAGE)
    Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS", +
          $(InvokeRasConfigMsg)
    Set AddCopy  = YES
    Set DoCopy   = YES
    Set DoConfig = YES
    Set SaveNTN_InstallMode = $(!NTN_InstallMode)
    Set !NTN_InstallMode = configure
    Shell "oemnsvra.inf" InstallOption $(!STF_LANGUAGE) "RAS" "" $(AddCopy) $
(DoCopy) $(DoConfig)
    Set !NTN_InstallMode = $(SaveNTN_InstallMode)
    IfInt $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output $(InfName)": Error Shelling the RAS INF file oemnsvra.inf"
        Shell "subroutn.inf" SetupMessage, $(!STF_LANGUAGE), "STATUS",  +
                $(InvokeRasError)
        IfInt $($ShellCode) != $(!SHELL_CODE_OK)
            GoTo ShellCodeError
        EndIf
    EndIf
    EndWait
      return
[CreateFullTapiAddrList]
```

```
    Debug-Output $(InfName)": Entering [TapiAddrList]"
      Set AdapterNum = $($0)
      Set FullTapiAddrList =  { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
+
                        "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
+
                        "21", "22", "23", "24", "25", "26", "27", "28", "29", "30",
+
                        "31", "32", "33", "34", "35", "36", "37", "38", "39", "40",
+
                        "41", "42", "43", "44", "45", "46", "47", "48", "49", "50",
+
                        "51", "52", "53", "54", "55", "56", "57", "58", "59", "60",
+
                        "61", "62", "63", "64", "65", "66", "67", "68", "69", "70",
+
                        "71", "72", "73", "74", "75", "76", "77", "78", "79", "80",
+
                        "81", "82", "83", "84", "85", "86", "87", "88", "89", "90",
+
                        "91", "92", "93", "94", "95", "96", "97", "98", "99",
"100", +
                        "101", "102", "103", "104", "105", "106", "107", "108",
"109", "110", +
                        "111", "112", "113", "114", "115", "116", "117", "118",
"119", "120", +
                        "121", "122", "123", "124", "125", "126", "127", "128",
"129", "130", +
                        "131", "132", "133", "134", "135", "136", "137", "138",
"139", "140", +
                        "141", "142", "143", "144", "145", "146", "147", "148",
"149", "150", +
                        "151", "152", "153", "154", "155", "156", "157", "158",
"159", "160", +
                        "161", "162", "163", "164", "165", "166", "167", "168",
"169", "170", +
                        "171", "172", "173", "174", "175", "176", "177", "178",
"179", "180", +
                        "181", "182", "183", "184", "185", "186", "187", "188",
"189", "190", +
                        "191", "192", "193", "194", "195", "196", "197", "198",
"199", "200", +
                        "201", "202", "203", "204", "205", "206", "207", "208",
"209", "210", +
                        "211", "212", "213", "214", "215", "216", "217", "218",
"219", "220", +
                        "221", "222", "223", "224", "225", "226", "227", "228",
"229", "230", +
                        "231", "232", "233", "234", "235", "236", "237", "238",
"239", "240", +
                        "241", "242", "243", "244", "245", "246", "247", "248",
"249", "250", +
                        "251", "252", "253", "254", "255", "256", "257", "258",
"259", "260", +
                        "261", "262", "263", "264", "265", "266", "267", "268",
"269", "270", +
                        "271", "272", "273", "274", "275", "276", "277", "278",
"279", "280", +
                        "281", "282", "283", "284", "285", "286", "287", "288",
```

```
"289", "290", +
                    "291", "292", "293", "294", "295", "296", "297", "298",
"299", "300", +
                    "301", "302", "303", "304", "305", "306", "307", "308",
"309", "310", +
                    "311", "312", "313", "314", "315", "316", "317", "318",
"319", "320", +
                    "321", "322", "323", "324", "325", "326", "327", "328",
"329", "330", +
                    "331", "332", "333", "334", "335", "336", "337", "338",
"339", "340", +
                    "341", "342", "343", "344", "345", "346", "347", "348",
"349", "350", +
                    "351", "352", "353", "354", "355", "356", "357", "358",
"359", "360", +
                    "361", "362", "363", "364", "365", "366", "367", "368",
"369", "370", +
                    "371", "372", "373", "374", "375", "376", "377", "378",
"379", "380", +
                    "381", "382", "383", "384", "385", "386", "387", "388",
"389", "390", +
                    "391", "392", "393", "394", "395", "396", "397", "398",
"399", "400", +
                    "401", "402", "403", "404", "405", "406", "407", "408",
"409", "410", +
                    "411", "412", "413", "414", "415", "416", "417", "418",
"419", "420", +
                    "421", "422", "423", "424", "425", "426", "427", "428",
"429", "430", +
                    "431", "432", "433", "434", "435", "436", "437", "438",
"439", "440", +
                    "441", "442", "443", "444", "445", "446", "447", "448",
"449", "450", +
                    "451", "452", "453", "454", "455", "456", "457", "458",
"459", "460", +
                    "461", "462", "463", "464", "465", "466", "467", "468",
"469", "470", +
                    "471", "472", "473", "474", "475", "476", "477", "478",
"479", "480", +
                    "481", "482", "483", "484", "485", "486", "487", "488",
"489", "490", +
                    "491", "492", "493", "494", "495", "496", "497", "498",
"499", "500", +
                          "501", "502", "503", "504", "505", "506", "507",
"508", "509", "510", +
                    "511", "512", "513", "514", "515", "516", "517", "518",
"519", "520", +
                    "521", "522", "523", "524", "525", "526", "527", "528",
"529", "530", +
                    "531", "532", "533", "534", "535", "536", "537", "538",
"539", "540", +
                    "541", "542", "543", "544", "545", "546", "547", "548",
"549", "550", +
                    "551", "552", "553", "554", "555", "556", "557", "558",
"559", "560", +
                    "561", "562", "563", "564", "565", "566", "567", "568",
"569", "570", +
                    "571", "572", "573", "574", "575", "576", "577", "578",
"579", "580", +
```

```
                    "581", "582", "583", "584", "585", "586", "587", "588",
"589", "590", +
                    "591", "592", "593", "594", "595", "596", "597", "598",
"599", "600", +
                    "601", "602", "603", "604", "605", "606", "607", "608",
"609", "610", +
                    "611", "612", "613", "614", "615", "616", "617", "618",
"619", "620", +
                    "621", "622", "623", "624", "625", "626", "627", "628",
"629", "630", +
                    "631", "632", "633", "634", "635", "636", "637", "638",
"639", "640", +
                    "641", "642", "643", "644", "645", "646", "647", "648",
"649", "650", +
                    "651", "652", "653", "654", "655", "656", "657", "658",
"659", "660", +
                    "661", "662", "663", "664", "665", "666", "667", "668",
"669", "670", +
                    "671", "672", "673", "674", "675", "676", "677", "678",
"679", "680", +
                    "681", "682", "683", "684", "685", "686", "687", "688",
"689", "690", +
                    "691", "692", "693", "694", "695", "696", "697", "698",
"699", "700", +
                    "701", "702", "703", "704", "705", "706", "707", "708",
"709", "710", +
                    "711", "712", "713", "714", "715", "716", "717", "718",
"719", "720", +
                    "721", "722", "723", "724", "725", "726", "727", "728",
"729", "730", +
                    "731", "732", "733", "734", "735", "736", "737", "738",
"739", "740", +
                    "741", "742", "743", "744", "745", "746", "747", "748",
"749", "750", +
                    "751", "752", "753", "754", "755", "756", "757", "758",
"759", "760", +
                    "761", "762", "763", "764", "765", "766", "767", "768",
"769", "770", +
                    "771", "772", "773", "774", "775", "776", "777", "778",
"779", "780", +
                    "781", "782", "783", "784", "785", "786", "787", "788",
"789", "790", +
                    "791", "792", "793", "794", "795", "796", "797", "798",
"799", "800", +
                    "801", "802", "803", "804", "805", "806", "807", "808",
"809", "810", +
                    "811", "812", "813", "814", "815", "816", "817", "818",
"819", "820", +
                    "821", "822", "823", "824", "825", "826", "827", "828",
"829", "830", +
                    "831", "832", "833", "834", "835", "836", "837", "838",
"839", "840", +
                    "841", "842", "843", "844", "845", "846", "847", "848",
"849", "850", +
                    "851", "852", "853", "854", "855", "856", "857", "858",
"859", "860", +
                    "861", "862", "863", "864", "865", "866", "867", "868",
"869", "870", +
                    "871", "872", "873", "874", "875", "876", "877", "878",
```

```
"879", "880", +
                        "881", "882", "883", "884", "885", "886", "887", "888",
"889", "890", +
                        "891", "892", "893", "894", "895", "896", "897", "898",
"899", "900", +
                        "901", "902", "903", "904", "905", "906", "907", "908",
"909", "910", +
                        "911", "912", "913", "914", "915", "916", "917", "918",
"919", "920", +
                        "921", "922", "923", "924", "925", "926", "927", "928",
"929", "930", +
                        "931", "932", "933", "934", "935", "936", "937", "938",
"939", "940", +
                        "941", "942", "943", "944", "945", "946", "947", "948",
"949", "950", +
                        "951", "952", "953", "954", "955", "956", "957", "958",
"959", "960", +
                        "961", "962", "963", "964", "965", "966", "967", "968",
"969", "970", +
                        "971", "972", "973", "974", "975", "976", "977", "978",
"979", "980", +
                        "981", "982", "983", "984", "985", "986", "987", "988",
"989", "990", +
                        "991", "992", "993", "994", "995", "996", "997", "998",
"999", "1000" }
      return $(FullTapiAddrList)
[InstallRASPPTPFDriver]
    Set MAXIMUM_ALLOWED = 33554432
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "InstallRASPPTPFDriver entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RaspptpfKeyName) $(MAXIMUM_ALLOWED) KeyService
    Ifstr(i) $(KeyService) == $(KeyNull)
        Shell "utility.inf", CreateService, $(!PPTPFProductName), +
            $(!PPTPFProductDisplayName), +
            $(!PPTPFProductImagePath), +
            "kernelautostart", "TDI", {"TCPIP"}, ""
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNXPPP.INF: InstallRASPPTPFDriver : 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 "OEMNXPPP.INF: InstallRASPPTPFDriver: Registry error
create service"
            return STATUS_FAILED
        endif
    endif
    Ifstr(i) $(KeyService) != $(KeyNull)
        CloseRegKey $(KeyService)
    endif
    Debug-Output "InstallRASPPTPFDriver exit"
    return $(Status)
```

```
[RemoveRASPPTPFDriver]
    Set MAXIMUM_ALLOWED = 33554432
    set Status = STATUS_SUCCESSFUL
    set KeyNull = ""
    Debug-Output "RemoveRASPPTPFDriver entry"
    OpenRegKey $(!REG_H_LOCAL) "" $(!RaspptpfKeyName) $(MAXIMUM_ALLOWED) KeyService
    Ifstr(i) $(KeyService) != $(KeyNull)
        Shell "utility.inf", RemoveService $(!PPTPFProductName) "YES"
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "OEMNXPPP.INF: RemoveRASPPTPFDriver : ShellCode error"
            return STATUS_FAILED
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            Debug-Output "OEMNXPPP.INF: RemoveRASPPTPFDriver: Registry error: "
            return STATUS_FAILED
        endif
        CloseRegKey $(KeyService)
    endif
    Debug-Output "RemoveRASPPTPFDriver exit"
    return $(Status)
[AddServiceDependency]
    Debug-Output "AddServiceDependency: entry"
    set MAXIMUM_ALLOWED = 33554432
    set SERVICE_NO_CHANGE = 4294967295
    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)
[DebugConfiguration]
    Debug-Output $(!p:InfName)": **CONFIGURATION STATE: "$($0)
    Debug-Output $(!p:InfName)": InterruptNumber is "$(!p:InterruptNumber)
    Debug-Output $(!p:InfName)": RamBaseAddress  is "$(!p:RamBaseAddress)
    Return
[DebugGlobalFlags]
    Debug-Output $(InfName)": **CONFIGURATION STATE: "$($0)
    Debug-Output $(InfName)": !STF_SRCDIR_USED value is "$(!STF_SRCDIR_USED)
    Debug-Output $(InfName)": !STF_SRCDIR_KEYED value is "$(!STF_SRCDIR_KEYED)
    Debug-Output $(InfName)": !STF_SRCDIR_OVERRIDE value is "$(!
STF_SRCDIR_OVERRIDE)
    Return
[Install-Option]
    Set STF_VITAL = ""
    IfStr(i) $(AddCopy) == "YES"
        AddSectionFilesToCopyList Files-RASPPTPDLL $(SrcDir) $(!STF_WINDOWSSYSPATH)
        AddSectionFilesToCopyList Files-RASPPTPDRV $(SrcDir) $(!
STF_WINDOWSSYSPATH)\drivers
    EndIf
    IfStr(i) $(DoCopy) == "YES"
       Set !STF_NCPA_FLUSH_COPYLIST = TRUE
       CopyFilesInCopyList
```

```
    Else
       LibraryProcedure STATUS, $(!NCPA_HANDLE), CopySingleFile $(!STF_HWND) $
(SrcDir)\RASPPTPC.DLL $(!STF_WINDOWSSYSPATH)\RASPPTPC.DLL
    EndIf
    Exit
[Install-Update]
   Set STF_VITAL        = ""
   Set STF_OVERWRITE    = "VERIFYSOURCEOLDER"
   AddSectionFilesToCopyList Files-RASPPTPDLL $(SrcDir) $(!STF_WINDOWSSYSPATH)
   AddSectionFilesToCopyList Files-RASPPTPDRV $(SrcDir) $(!STF_WINDOWSSYSPATH)\
drivers
   Set !STF_NCPA_FLUSH_COPYLIST = TRUE
   CopyFilesInCopyList
   Exit
[Source Media Descriptions]
    1 = "Windows NT Workstation CD-ROM" , TAGFILE = cdrom_w.40
[Signature]
    FileType = MICROSOFT_FILE
[GetSignature]
    read-syms Signature
    return $(FileType)
[ProductType]
STF_PRODUCT  = Winnt
STF_PLATFORM = I386
[Files-Inf]
2,    oemsetup.inf,      SIZE=1000, RENAME=$(!UG_Filename)
[Files-RASPPTPDLL]
1,RASPPTPC.DLL , SIZE=999
1,RASPPTPL.DLL , SIZE=999
[Files-RASPPTPDRV]
1,RASPPTPE.SYS , SIZE=999
1,RASPPTPF.SYS , SIZE=999
1,RASPPTPM.SYS , SIZE=999
1,RASPPTPU.SYS , SIZE=999
[LanguagesSupported]
    ENG
[OptionsTextENG]
    RASPPTP             = "Point to Point Tunneling ���Í�"
[FileConstantsENG]
    ProCaption          = "Windows NT ��ı���"
    ProCancel           = "��ۣ"
    ProCancelMsg        = "Windows NT ò[illegible]Ăİ[illegible]Ă��· ���B"+
                          "[illegible]?"
    ProCancelCap        = "ò[illegible] ��ı��  γ[illegible]"
    ProText1            = "[illegible]:"
    ProText2            = "[illegible]:"
    FunctionTitleRASPPTP                = "PPTP [illegible] ��ı���"
    ProductSoftwareDescription          = "PPTP ��"[illegible]"
    ProductHardwareRASPPTPDescription   = "PPTP [illegible]"
    ProductSoftwareTitle                = "PPTP [illegible] ��"[illegible]"
    ProductHardwareRASPPTPTitle         = "PPTP [illegible]"
    ProductRASPPTPDescription           = "[illegible]òĂŁ[illegible] ° ��[illegible] (PPP) [illegible]
ò[illegible]ZM[illegible]"+
                                          "[illegible]Íğ?[illegible]A[illegible] h[illegible] h[illegible]A[illegible]
[illegible]n[illegible]òĂ[illegible]J[illegible]òĂ[illegible]A[illegible]ò[illegible]ɱ[illegible]Ł[illegible]· ��B"
    ProductRASPPTPTitle                 = "Point To Point Tunneling ���Í�"
    ShellCodeErrorTitle                 = "[illegible]: "$(FunctionTitle)$(Option))
    ShellCodeErrorText                  = "��[illegible]Ł[illegible]B"
[SMANFileConstantsENG]
```

```
    PPTPFunctionTitleRASPPTP                = "PPTP ���� ��ı���"
    PPTPProductSoftwareDescription          = "PPTP ��"��"
    PPTPProductHardwareRASPPTPDescription   = "PPTP ������"
    PPTPProductSoftwareTitle                = "PPTP ���í� ��"��"
    PPTPProductHardwareRASPPTPTitle         = "PPTP ������"
    !PPTPFProductDisplayName                = "PPTP Filter Driver"
[DialogConstantsENG]
    Help                = "����(&H)"
    Exit                = "��♪■"
    OK                  = "OK"
    HelpContext         = ""
    Continue    = "���s"
    Cancel      = "��♪■"
[FileDependentDlgENG]
    DlgType             = "RadioCombination"
    DlgTemplate         = "NE2000"
    Caption             = $(FunctionTitle)$(Option)
    EditTextIn          = ""
    EditTextLim         = ""
    CBOptionsGreyed = {}
    NotifyFields        = {NO, NO}
[InvokeRasDlgENG]
    InvokeRasSetupMsg   = "ÿ■ ■■■■ ■■■■ (RAS) ■■♪İ· ■■ı■■B"+
                          "PPTP ■■ RAS ■■g■p■ł■■■■ ■■ "+
                          "RAS ■■ı■■ŕ■ PPTP ■■Ă■\■■■■■Ă■■■■■■■■B"
    InvokeRasConfigMsg  = "ÿ■ ■■■■ ■■■■ (RAS) ■■ı■■ŕ■N■■■■■· ■■B"+
                          "PPTP ■■ RAS ■■g■p■ł■■■■ ■■ "+
                          "RAS ■■ı■■ŕ■ PPTP ■■Ă■\■■■■■Ă■■■■■■■■B"
    InvokeRasConfigMsg2 = "ÿ■ ■■■■ ■■■■ (RAS) ■■ı■■ŕ■N■■■■■· ■■B"+
                          "RAS ■\■■■■ PPTP ■■Ă■■■■Ă■■■■■■■■B"
    InvokeRasError      = "RAS ■■ı■■■ INF ■■ (OEMNSVRA.INF) ■╪V■■ŁA"+
                          "PPTP ■■ı■■ıı■■o■■■· ■■■■B"+
                          "■♪■ system32 ■▂■↓ɑ■■■■■A"+
                          "■· ■■K■♪■■▂■ł■■■■m■F■■■Ă■■■■■■■■B"
[InvokeTCPIPDlgENG]
    InvokeTCPSetupMsg   = "TCP/IP ■Ͳ♪İ♪■■■Ă■■· ■■■B■♪İ♪■J■n■■■· ■■B"
    InvokeTCPError      = "TCP/IP ■■ı■■■ INF ■■ (OEMNXPTC.INF) ■╪V■■ŁA"+
                          "PPTP ■■ı■■ıı■■o■■■· ■■■■B"+
                          "■♪■ SYSTEM32 ■▂■↓ɑ■■■■■A"+
                          "■· ■■K■♪■■▂■ł■■■■m■F■■■Ă■■■■■■■■B"
[InvokeCancelDlgENG]
    InvokeCancelMsg = "PPTP ■■ı■■ŕ■■■■■■Ă■■· ■■■B"+
                      "PPTP ■■■■■■■♪İ♪■■■♁■A■■ı■■■ ■■^■■т■"+
                      "■Ď■s■■■Ă■■■■■■■B"
[InvokeConfigureDlgENG]
    InvokeConfigureMsg = "■úX■■L■■?■■■?Ź■ċN■■■■■Ă■■■■■■■B"
```