

```

[Identification]
    OptionType = NetAdapter
[PlatformsSupported]
    ISA
    EISA
    PCMCIA
    "Jazz-Internal Bus"
[Options]
    IBMTOK
[GeneralConstants]
from      = ""
to        = ""
retaddr   = ""
KeyNull   = ""
StringNull = ""
MAXIMUM_ALLOWED = 33554432
RegistryErrorIndex = NO_ERROR
KeyProduct = ""
KeyParameters = ""
TRUE      = 1
FALSE     = 0
NoTitle   = 0
ExitState = "Active"
OldVersionExisted = $(FALSE)
DriverPath = $(!STF_NTPATH)\drivers
[FileConstants]
UtilityInf = "UTILITY.INF"
ParamInf   = "NCPARAM.INF"
subroutineinf = "SUBROUTN.INF"
SoftwareType = "driver"
NetEventDLL = "%SystemRoot%\System32\netevent.dll"
IoLogMsgDLL = "%SystemRoot%\System32\IoLogMsg.dll"
SpeedList = {16,4}
Manufacturer = "Microsoft"
ProductMajorVersion = "4"
ProductMinorVersion = "0"
ProductVersion = $(ProductMajorVersion)".$(ProductMinorVersion)"
ProductSoftwareName = "IbmTok"
ProductSoftwareImagePath = "\SystemRoot\System32\drivers\ibmtok.sys"
NetRuleSoftwareType = "ibmtokSys ndisDriver ibmtokDriver"
NetRuleSoftwareUse = $(SoftwareType)
NetRuleSoftwareBindForm = ""IBMTokSys"" yes no container"
NetRuleSoftwareClass = {"ibmtokDriver basic"}
ProductHardwareName = "IbmTok"
NetRuleHardwareType = "ibmtok ibmtokAdapter"
NetRuleHardwareBindForm = " yes yes container"
NetRuleHardwareClass = {"ibmtokAdapter basic"}
ProductOpSupport = 134
ProductKeyName = $(!NTN_SoftwareBase)"\"$(Manufacturer)"\"$(
(ProductSoftwareName)"\CurrentVersion"
ParamKeyName = $(!NTN_ServiceBase)"\"$(ProductHardwareName)"\Parameters"
[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 returnoptions
    endif
    set PlatformList = ^(PlatformsSupported, 1)
    Ifcontains(i) $($1) in $(PlatformList)
        goto returnoptions
    else
        set Status = STATUS_NOTSUPPORTED
        goto finish_ReturnOptions
    endif
else
    set Status = STATUS_NOLANGUAGE
    goto finish_ReturnOptions
endif
returnoptions = +
    set OptionList      = ^(Options, 1)
    set OptionTextList = ^(OptionsText$($0), 1)
    set Status          = STATUS_SUCCESSFUL
finish_ReturnOptions = +
    Return $(Status) $(OptionList) $(OptionTextList)
[InstallOption]
    set Option      = $($1)
    set SrcDir      = $($2)
    set AddCopy     = $($3)
    set DoCopy      = $($4)
    set DoConfig    = $($5)
    set LanguageList = ^(LanguagesSupported, 1)
    Ifcontains(i) $($0) NOT-IN $(LanguageList)
        Return STATUS_NOLANGUAGE
    endif
    Debug-Output "OEMNADTK.INF: STF_CWDIR is: "$(!STF_CWDIR)
    set-subst LF = "\n"
    read-syms GeneralConstants
    read-syms FileConstants
    Shell $(UtilityInf), GetBindingInfo, "IBM"
    ifint $($ShellCode) != $($!SHELL_CODE_OK)
        Debug-Output "ShellCode error: cannot get an error string."
        goto ShellCodeError
    endif
    set NetRuleSoftwareBindable = $($R1)
    read-syms DialogConstants$(!STF_LANGUAGE)
    ifstr(i) $($!NTN_Origination) == "NCPA"
        set Continue = $(OK)
    endif
    read-syms FileConstants$(!STF_LANGUAGE)
    detect date
    set-title $(FunctionTitle)
    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) == bind
        set StartLabel = bindingadapter
    else-Ifstr(i) $($!NTN_InstallMode) == Update
        set StartLabel = UpgradeSoftware
    else-Ifstr(i) $($!NTN_InstallMode) == configure
        set CommonStatus = STATUS_REBOOT
        set ActivateDetection = TRUE
        set StartLabel = configureadapter

```

```

Ifstr(i) $(ProductKeyName) == $(!NTN_RegBase)
    Debug-Output "Cannot configure the IBM software."
    Shell $(UtilityInf),RegistryErrorString,CANNOT_CONFIGURE_SOFTWARE
    ifint $(ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "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 OEM_ABANDON_OPTIONS = {}
    set OEM_ABANDON_SOFTWARE = FALSE
    set OEM_ABANDON_ON = TRUE
    set ActivateDetection = TRUE
endif
set IOBaseAddress = 1
Debug-Output "OEMNADTK.INF:
=====
Debug-Output "OEMNADTK.INF: STF_CWDIR is: "$(!STF_CWDIR)
Debug-Output "OEMNADTK.INF: STF_LANGUAGE is: "$(!STF_LANGUAGE)
Debug-Output "OEMNADTK.INF: Option is: "$(Option)
Debug-Output "OEMNADTK.INF: !STF_NCDetect is: "$(!STF_NCDetect)
Debug-Output "OEMNADTK.INF: !STF_NCOPTION is: "$(!STF_NCOPTION)
Debug-Output "OEMNADTK.INF: !STF_NCDetectCard is: "$(!STF_NCDetectCard)
Debug-Output "OEMNADTK.INF: !STF_NCDetectInfo is: "$(!STF_NCDetectInfo)
Debug-Output "OEMNADTK.INF:
=====
Set DetectedCard = FALSE
set CardType = 1
set Pcmcia = FALSE
Ifstr(i) $(ActivateDetection) != TRUE
    Goto skipdetection
Endif
Set TypeList = {{IOADDR, IOADDR_Addr_List, IOBaseAddress},+
                {IRQ, IRQList, Token1IRQVal},+
                {MEMADDR, MMIODecList, Token1MMIOVal}}
Debug-Output "OEMNADTK.INF: Calling Param_BuildTypeLists"
Shell $(ParamInf) Param_BuildTypeLists $(Option) $(TypeList)
Set Status = $($R0)
ifstr(i) $(Status) != STATUS_SUCCESSFUL
    Goto fataldetect
Endif
Debug-Output "OEMNADTK.INF: Calling Param_SetDefaults"
Shell $(ParamInf) Param_SetDefaults {}
Shell $(ParamInf) HexListFromDecList $(MMIODecList)
Set MMIOList = $($R0)
Shell $(UtilityInf) SortList $(SpeedList) TRUE FALSE
Set Combo1List = $($R0)
Shell $(UtilityInf) SortList $(IRQList) TRUE FALSE
Set Combo2List = $($R0)
Shell $(UtilityInf) SortList $(MMIOList) TRUE FALSE
Set Combo3List = $($R0)
Ifstr(i) $(!STF_NCDetect) == YES
    Ifstr(i) $(!STF_NCOPTION) == $(Option)
        Set DetectedCard = TRUE
        Debug-Output "OEMNADTK.INF: Setting DetectedCard to TRUE"
    Endif
Endif
endif
skipdetection +=
set from = $(fatal)

```

```

    set to = $(fatal)
    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 "ShellCode error: cannot get an error string."
                goto ShellCodeError
            endif
            goto end
        else
            Shell $(UtilityInf), CardExistedDlg
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                Debug-Output "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
    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 "OEMNADTK.INF: Calling Param_SetDefaults to merge detected
params"
    Shell $(ParamInf) Param_SetDefaults $(DetectedParams)
    goto adapteroptions
configureadapter = +
    Ifstr $(KeyProduct) == $(KeyNull)
        OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_RegBase) $(MAXIMUM_ALLOWED)
KeyProduct
        Ifstr $(KeyProduct) == $(KeyNull)
            set RegistryErrorIndex = CANNOT_FIND_COMPONENT_SERVICE
            Debug-Output "Cannot find component product key"
            goto fatalregistry
        Endif
    Endif
    Debug-Output "OEMNADTK.INF: Shelling to FindService"
    Shell $(UtilityInf) FindService, $(KeyProduct)
    Ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "OEMNADTK.INF: FindService shell failure"
        Goto ShellCodeError
    Endif
    Ifstr(i) $($R0) != NO_ERROR
        Debug-Output "OEMNADTK.INF: FindService Shell error: "$($R0)
        Goto fatalregistry
    endif
    set KeyParameters = $($R2)
    CloseRegKey $($R1)

```

```

Ifstr $(KeyParameters) == $(KeyNull)
    set RegistryErrorIndex = "CANNOT_FIND_COMPONENT_SERVICE"
    Debug-Output "Cannot find component service"
    goto fatalregistry
endif
set OldVersionExisted = $(TRUE )
set ValueName = ""
set ValueData = ""
set ValueStr = ""
set ValueList = {}
EnumRegValue $(KeyParameters) ValueList
ForListDo $(ValueList)
    set ValueItem = $($ )
    set ValueName = *$(ValueItem),1)
    set ValueData = *$(ValueItem),4)
    Ifstr(i) $(ValueName) == "IOBaseAddress"
        set IOBaseAddress = $(ValueData)
    else-Ifstr(i) $(ValueName) == "NetworkAddress"
        set NetworkAddress = $(ValueData)
    else-ifstr(i) $(ValueName) == "BusType"
        set BusInterfaceType = $(ValueData)
    else-ifstr(i) $(ValueName) == "BusNumber"
        set BusNumber = $(ValueData)
    else-ifstr(i) $(ValueName) == "InterruptNumber"
        set Token1IRQVal = $(ValueData)
    else-ifstr(i) $(ValueName) == "RingSpeed"
        set Token1Speed = $(ValueData)
    else-ifstr(i) $(ValueName) == "MemoryMappedBaseAddress"
        set Token1MMIOVal = $(ValueData)
    else-ifstr(i) $(ValueName) == "MemoryMappedBaseAddress_1"
        set Token1RAM = $(ValueData)
    else-ifstr(i) $(ValueName) == "MemoryMappedSize"
        set Token1MMIOSize = $(ValueData)
    else-ifstr(i) $(ValueName) == "MemoryMappedSize_1"
        set Token1RAMSize = $(ValueData)
    else-ifstr(i) $(ValueName) == "Pcmcia"
        set Pcmcia = $(ValueData)
    endif
EndForListDo
ifstr(i) $(IOBaseAddress) == 2592
    set IOBaseAddress = 1
else-ifstr(i) $(IOBaseAddress) == 2596
    set IOBaseAddress = 2
else-ifstr(i) $(IOBaseAddress) == ""
    set IOBaseAddress = 1
endif
adaptersetup +=
Shell $(ParamInf) Param_ParameterConfidence
Ifstr(i) $($R0) != STATUS_SUCCESSFUL
    Debug-Output "OEMNADTK.INF: parameter confidence too low to bypass
configuration"
    Goto adapteroptions
Endif
Ifstr(i) $(DetectedCard) == TRUE
    Ifstr(i) $(!STF_INSTALL_MODE) != CUSTOM
        set BusInterfaceType = *$(!STF_NCDETINFO),5)
        set BusNumber = *$(!STF_NCDETINFO),6)
        Goto adapterverify
    Endif
Endif
goto adapteroptions
adapteroptions = +
set from = adapteroptions
ifstr(i) $(IOBaseAddress) == 2592

```

```

        set IOBaseAddress = 1
    else-ifstr(i) $(IOBaseAddress) == 2596
        set IOBaseAddress = 2
    else-ifstr(i) $(IOBaseAddress) == ""
        set IOBaseAddress = 1
    endif
    ifstr(i) $(!STF_GUI_UNATTENDED) == "YES"
        ifstr(i) $(!AutoNetInterfaceType) != ""
            set BusInterfaceType = $(!AutoNetInterfaceType)
        else
            set BusInterfaceType = 1
        endif
        ifstr(i) $(!AutoNetBusNumber) != ""
            set BusNumber = $(!AutoNetBusNumber)
        else
            set BusNumber = 0
        endif
        goto adapterverify
    endif
    set RadioIn = {$(IOBaseAddress)}
    read-syms FileDependentDlg$(!STF_LANGUAGE)
    ui start "InputDlg"
    ifstr(i) $(DLGEVENT) == "CONTINUE"
        set IOBaseAddress = *$(RadioOut),1)
        set NetworkAddress = *(EditTextOut),1)
        ui pop 1
    else-ifstr(i) $(DLGEVENT) == "BACK"
        set CommonStatus = STATUS_USERCANCEL
        Debug-Output "Action: exit. Bye."
        ui pop 1
        goto end
    else
        ui pop 1
        Debug-Output "Action: unknown. Bye."
        goto end
    endif
    ifstr(i) $(!STF_NCDETINFO) == {}
        Shell $(UtilityInf),GetBusTypeDialog,$(ProductHardwareDescription) $
        (BusInterfaceType) $(BusNumber)
        ifint $(ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error."
            goto ShellCodeError
        endif
        set BusInterfaceType = $($R1)
        set BusNumber = $($R2)
    else
        set BusInterfaceType = *$(!STF_NCDETINFO),5)
        set BusNumber = *$(!STF_NCDETINFO),6)
    endif
    set Pcmcia = 0
    ifstr(i) $(BusInterfaceType) == 8
        set Pcmcia = 1
        set Token1MMIOSize = 8192
        set Token1RAMSize = 16384
        read-syms SecondDlg$(!STF_LANGUAGE)
        ifstr(i) $(Token1IRQVal) == ""
            set Token1IRQVal = *(IRQList), 1)
        endif
        ifstr(i) $(Token1MMIO) == ""
            set Token1MMIO = *(MMIOList), 1)
        endif
        ifstr(i) $(Token1Speed) == ""
            set Token1Speed = *(SpeedList), 1)
        endif
    endif

```

```

set Token1MMIO = *$(MMIOList), ~$(MMIODecList),+
$(Token1MMIOVal))
set Combo1Out = $(Token1Speed)
set Combo2Out = $(Token1IRQVal)
set Combo3Out = $(Token1MMIO)
ui start "Config"
ifstr(i) $(DLGEVENT) == "CONTINUE"
    set Token1Speed = $(Combo1Out)
    set Token1IRQVal = $(Combo2Out)
    set Token1MMIO = $(Combo3Out)
    ui pop 1
else-ifstr(i) $(DLGEVENT) == "BACK"
    set CommonStatus = STATUS_USERCANCEL
    Debug-Output "Action: exit. Bye."
    ui pop 1
    goto end
else
    ui pop 1
    Debug-Output "Action: unknown. Bye."
    goto end
endif
set Token1MMIOVal = *$(MMIODecList), ~$(MMIOList),+
$(Token1MMIO))
Set-add Token1RAM = $(Token1MMIOVal),8192
ifstr(i) $(IOBaseAddress) == 1
    set IOBaseAddress = 2592
else-ifstr(i) $(IOBaseAddress) == 2
    set IOBaseAddress = 2596
else-ifstr(i) $(IOBaseAddress) == ""
    set IOBaseAddress = 2592
endif
endif
adaperverify =+
Ifstr(i) $(DetectedCard) != TRUE
    Goto skipoptions
Endif
Debug-Output "OEMNADTK.INF: Calling Param_VerifyCard"
Shell $(ParamInf) Param_VerifyCard $(!STF_NCDETCARD)
Ifstr(i) $($R0) == STATUS_SUCCESSFUL
    Debug-Output "OEMNADTK.INF: Param_VerifyCard succeeded"
    Goto skipoptions
Endif
Set from = adapteroptions
Set to = skipoptions
Shell $(UtilityInf),RegistryErrorString,VERIFY_WARNING
ifint $($ShellCode) != $(!SHELL_CODE_OK)
    Debug-Output "ShellCode error: cannot get an error string."
    goto ShellCodeError
endif
set Error = $($R0)
Goto Warning
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
            endif
        endif
    endif
endif

```

```

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 successful
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 fatal
endif
endif
Shell $(UtilityInf), AddSoftwareComponent, $(Manufacturer), +
    $(ProductSoftwareName), +
    $(ProductSoftwareName), +
    $(ProductSoftwareTitle), $(STF_CONTEXTINFNAME), +
    $(ProductSoftwareImagePath), "kernel", "NDIS", {}, "", +
    $(NetEventDLL)
Set OEM_ABANDON_SOFTWARE = TRUE
ifint $($ShellCode) != $(!SHELL_CODE_OK)
    Debug-Output "ShellCode error"
    goto ShellCodeError
endif
set RegistryErrorIndex = $($R0)
Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
    EndWait
    Debug-Output "Registry error: add software components"
    CloseRegKey $($R1)
    CloseRegKey $($R2)
    CloseRegKey $($R3)
    CloseRegKey $($R4)
    CloseRegKey $($R5)
    goto fatalregistry
endif
Set SoftProductKey      = $($R1)
Set SoftNetRuleKey     = $($R2)
Set SoftServiceKey     = $($R3)
Set SoftParameterKey  = $($R4)
Set SoftLinkageKey     = $($R5)
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, $(SoftProductKey), $(NewValueList)

```



```

    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error."
        goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        EndWait
        Debug-Output "Resgitry error: add value list."
        CloseRegKey $(SoftProductKey)
        CloseRegKey $(SoftNetRuleKey)
        CloseRegKey $(SoftServiceKey)
        CloseRegKey $(SoftParameterKey)
        CloseRegKey $(SoftLinkageKey)
        goto fatalregistry
    endif
    set NewValueList = {{type,$(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),$
(NetRuleSoftwareBindable)}, +
                        {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}}
    Shell $(UtilityInf), AddValueList, $(SoftNetRuleKey), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error."
        goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(SoftProductKey)
    CloseRegKey $(SoftNetRuleKey)
    CloseRegKey $(SoftServiceKey)
    CloseRegKey $(SoftParameterKey)
    CloseRegKey $(SoftLinkageKey)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        EndWait
        Debug-Output "Resgitry error: add value list."
        goto fatalregistry
    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 "Cannot add hardware component"
    goto ShellCodeError
endif
set RegistryErrorIndex = $($R0)
Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
    EndWait
    Debug-Output "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),$(!REG_VT_SZ),"["$($R4)"] "$
(ProductHardwareTitle)},+
        {Description,$(NoTitle),$(!REG_VT_SZ),$
(ProductHardwareDescription)},+
        {ProductName,$(NoTitle),$(!REG_VT_SZ),$
(ProductHardwareName)},+
        {ServiceName,$(NoTitle),$(!REG_VT_SZ),$(R5)},+
        {OperationsSupport,$(NoTitle),$(!REG_VT_DWORD),$
(ProductOpSupport)},+
        {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($Now,1)}}
    Shell $(UtilityInf), AddValueList, $($R1), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error"
        goto ShellCodeError
    endif
    CloseRegKey $($R1)
    set TempProdName = """"$(ProductHardwareName)$($AdapterNumber)""""
    set TempBindForm = $(TempProdName)$($NetRuleHardwareBindForm)
    set NewValueList = {{type,$(NoTitle),$(!REG_VT_SZ),$(NetRuleHardwareType)},+
        {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)}, +
        {class,$(NoTitle),$(!REG_VT_MULTI_SZ),$(
+
        {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}}
    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
        EndWait
        Debug-Output "Resgistry error: add value list."
        CloseRegKey $(KeyParameters)
        CloseRegKey $(KeyAdapterRules)
        goto fatalregistry
    endif
    CloseRegKey $(KeyAdapterRules)
    goto writeparameters
writeparameters = +
    ifstr(i) $(BusInterfaceType) == 8
        set NewValueList = {{IoBaseAddress,$(NoTitle),$(!REG_VT_DWORD),$
+
        {IoLength,$(NoTitle),$(!REG_VT_DWORD),7},+
        {BusType,$(NoTitle),$(!REG_VT_DWORD),$
+
        {BusNumber,$(NoTitle),$(!REG_VT_DWORD),$(BusNumber)},+
        {CardType,$(NoTitle),$(!REG_VT_DWORD),1},+
        {MediaType,$(NoTitle),$(!REG_VT_DWORD),2},+
        {NetworkAddress,$(NoTitle),$(!REG_VT_SZ),$(
+
        {InterruptNumber,$(NoTitle),$(!REG_VT_DWORD),$
+
        {RingSpeed,$(NoTitle),$(!REG_VT_DWORD),$(Token1Speed)},+
        {MemoryMappedBaseAddress,$(NoTitle),$(!REG_VT_DWORD),$
+
        {MemoryMappedBaseAddress_1,$(NoTitle),$(!REG_VT_DWORD),$
+
        {PCCardMemoryWindowOffset,$(NoTitle),$(!REG_VT_DWORD),$
+
        {PCCardMemoryWindowOffset_1,$(NoTitle),$(!REG_VT_DWORD),$
+

```

```

        {MemoryMappedSize,$(NoTitle),$(!REG_VT_DWORD),$
(Token1MMIOSize)},+
        {MemoryMappedSize_1,$(NoTitle),$(!REG_VT_DWORD),$
(Token1RAMSize)},+
        {RingSpeed,$(NoTitle),$(!REG_VT_DWORD),$(Token1Speed)},+
        {Pcmcia,$(NoTitle),$(!REG_VT_DWORD),$(Pcmcia)}}
    else
        set NewValueList = {{IoBaseAddress,$(NoTitle),$(!REG_VT_DWORD),$
(IOBaseAddress)},+
        {BusType,$(NoTitle),$(!REG_VT_DWORD),$
(BusInterfaceType)},+
        {BusNumber,$(NoTitle),$(!REG_VT_DWORD),$(BusNumber)},+
        {MediaType,$(NoTitle),$(!REG_VT_DWORD),2},+
        {NetworkAddress,$(NoTitle),$(!REG_VT_SZ),$
(NetworkAddress)},+
        {Pcmcia,$(NoTitle),$(!REG_VT_DWORD),$(Pcmcia)}}
    endif
    Shell $(UtilityInf), AddValueList, $(KeyParameters), $(NewValueList)
    ifstr(i) $(!STF_GUI_UNATTENDED) == "YES"
        Shell $(UtilityInf),AddDefaultNetCardParameters,$(KeyParameters)
    endif
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error."
        goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    CloseRegKey $(KeyParameters)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        Debug-Output "Registry error: Add value list"
        goto fatalregistry
    endif
    EndWait
    goto successful
bindingadapter =+
    set Error = "Binding: Sorry, not yet implemented."
    goto fatal
removeadapter = +
    Ifstr(i) $(ProductKeyName) == $(!NTN_RegBase)
        Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
        $(ProductSoftwareName)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            goto fatalregistry
        endif
    else
        Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
        $(ProductSoftwareName), $(!NTN_RegBase)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
            Debug-Output "ShellCode error"
            goto ShellCodeError
        endif
        set RegistryErrorIndex = $($R0)
        Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
            goto fatalregistry
        endif
    endif
    goto end
UpgradeSoftware = +
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED)
KeyProduct

```

```

Ifstr $(KeyProduct) != $(KeyNull)
    install "Install-Update"
    ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
        goto fatal
    endif
    SetRegValue $(KeyProduct) {MajorVersion,$(NoTitle),$(!REG_VT_SZ),$
(ProductMajorVersion)}
    SetRegValue $(KeyProduct) {MinorVersion,$(NoTitle),$(!REG_VT_SZ),$
(ProductMinorVersion)}
    CloseRegKey $(KeyProduct)
else
    goto fatalregistry
endif
set iSearch = 1
nextnetcard = +
Shell $(UtilityInf), FindNextNetworkCard, $(ProductHardwareName), $(iSearch)
set KeyNetcard = $($R0)
set iSearch = $($R1)
Debug-Output "OemNadEp.Inf: FindNextNetworkCard "$(KeyNetcard)","$(iSearch)
Ifstr $(KeyNetcard) != $(KeyNull)
    Debug-Output "OemNadEp.Inf: Setting OperationsSupport value"
    SetRegValue $(KeyNetcard) {OperationsSupport,$(NoTitle),$(!
REG_VT_DWORD),$(ProductOpSupport)}
    GetRegValue $(KeyNetcard), "ServiceName", ServiceNameList
    set AdapterServiceName = *$(ServiceNameList),4)
    OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_ServiceBase)"\"$
(AdapterServiceName)"\Parameters" $(!MAXIMUM_ALLOWED) ParametersKey
    Ifstr $(ParametersKey) == $(KeyNull)
        Debug-Output "Error opening BaseKey "$(!NTN_ServiceBase)"\"$
(AdapterServiceName)"\Parameters"
        CloseRegKey $(ParametersKey)
        goto fatalregistry
    Endif
    SetRegValue $(ParametersKey) {MediaType,$(NoTitle),$(!REG_VT_DWORD),2}
    CloseRegKey $(ParametersKey)
    CloseRegKey $(KeyNetcard)
    goto nextnetcard
Endif
goto end
successful = +
goto end
abandon = +
ForListDo $(OEM_ABANDON_OPTIONS)
    Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
$(ProductSoftwareName), $(ShellCode)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "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 "ShellCode error"
        goto ShellCodeError
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
        goto fatalregistry

```

```

        endif
    endif
    goto end
warning = +
    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)
    else
        goto "end"
    endif
nonfatalinfo = +
    Set CommonStatus = STATUS_USERCANCEL
    Set Severity = STATUS
    goto nonfatalmsg
nonfatal = +
    Set Severity = NONFATAL
    goto nonfatalmsg
nonfatalmsg = +
    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)
    else
        goto "end"
    endif
fatalregistry = +
    Shell $(UtilityInf) RegistryErrorString $(RegistryErrorIndex)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        goto ShellCodeError
    endif
    set Error = $($R0)
    goto fatal
fataldetect = +
    Shell $(UtilityInf),RegistryErrorString,CANNOT_DETECT
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error: cannot get an error string."
        goto ShellCodeError
    endif
    set Error = $($R0)
    Goto fatal
fatal = +
    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 = +
    goto term
term = +
    Return $(CommonStatus)
[Install-Option]
    set STF_VITAL = ""
    ifstr(i) $(AddCopy) == "YES"
        AddSectionFilesToCopyList Files-$(Option) $(SrcDir) $(!
STF_WINDOWSSYSPATH)\drivers
    endif
    ifstr(i) $(DoCopy) == "YES"
        set !STF_NCPA_FLUSH_COPYLIST = TRUE
        CopyFilesInCopyList
    endif
    ifstr(i) $(DoConfig) == "YES"
    endif
    Exit
[Install-Update]
    set STF_VITAL          = ""
    set STF_OVERWRITE     = "VERIFYSOURCEOLDER"
    AddSectionFilesToCopyList Files-$(Option) $(SrcDir) $(!STF_WINDOWSSYSPATH)\
drivers
    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-IBMTok]
1,IBMTOK.SYS , SIZE=34416
[LanguagesSupported]
    ENG
[OptionsTextENG]
    IBMTOK          = "IBM Token Ring (ISA/PCMCIA) Adapter"
[FileConstantsENG]
ProCaption    = "Windows NT Setup"
ProCancel     = "Cancel"

```

```

ProCancelMsg = "Windows NT Networking is not correctly installed. "+
               "Are you sure you want to cancel copying files?"
ProCancelCap = "Network Setup Message"
ProText1     = "Copying:"
ProText2     = "To:"
FunctionTitle = "IBM Token Ring Adapter Card Setup"
FunctionTitle1 = "IBM Token Ring PCMCIA Adapter Card Setup"
ProductSoftwareDescription = "IBM Token Ring (ISA/PCMCIA) Adapter Driver"
ProductHardwareDescription = "IBM Token Ring (ISA/PCMCIA) Adapter"
ProductSoftwareTitle = "IBM Token Ring Adapter (ISA/PCMCIA) Driver"
ProductHardwareTitle = "IBM Token Ring (ISA/PCMCIA) Adapter"
ShellCodeErrorTitle = "Error: "$(FunctionTitle)
ShellCodeErrorText = "Shell Code Error."
[DialogConstantsENG]
Help = "&Help"
Exit = "Cancel"
OK = "OK"
HelpContext = ""
Continue = "Continue"
Cancel = "Cancel"
[FileDependentDlgENG]
Group1 = "I/O Port Base Address"
Radio1 = "&Primary"
Radio2 = "&Secondary"
Edit1Label = "&Network Address:"
DlgType = "RadioCombination"
DlgTemplate = "IBM_TOKEN"
Caption = $(FunctionTitle)
ComboListItemsIn = {}
ComboListItemsOut = {}
EditTextLim = 17
EditTextIn = $(NetworkAddress)
OptionsGreyed = {}
HelpContext = $(!IDH_DB_OEMNADTK_INS)
RCctlFocusOn = 403
[SecondDlgENG]
Group1 = "Network Speed"
HelpContext = 1
Caption = $(FunctionTitle1)
Combo1Label = "Network &Speed:"
Combo2Label = "Card &IRQ Level:"
Combo3Label = "Memory &Base:"
Combo1List = $(SpeedList)
Combo1Out = $(Token1Speed)
Combo2List = $(IRQList)
Combo2Out = $(Token1IRQVal)
Combo3List = $(MMIOList)
Combo3Out = $(Token1MMIO)
DlgType = Combination
DlgTemplate = "WD"
ComboListItemsIn = {Combo1List, Combo2List, Combo3List}
ComboListItemsOut = {Combo1Out, Combo2Out, Combo3Out}
CBOptionsGreyed = {}
EditTextIn = ""
EditTextLim = ""
NotifyFields = {NO, NO, NO}
HelpContext = 1

```