Déclaration de cUPNP

```
//Classe basique de requetage uPNP v1.2 - Tanguy PRUVOT 01/2007
//Fonctions : IP Externe, Ouverture et Fermeture de Port, Liste des Ports

//Note : Ferme le dernier port ouvert automatiquement à la destruction ou lors de l'ouverture d'un
nouveau port


//Ex. d'utilisation :

//oUPNP est un cUPNP("192.168.0.1",5678)
//oUPNP:OuvrePort("192.168.0.3",1234,"TCP")

//Informations techniques : http://www.upnp.org/

//v1.2 : Ajout de la fonction de Requete perso :Requete()
//       Les fonctions utilisent toutes la méthode commune :Requete
//       Récupération des résultats DANS le tableau dynamique :tResponse

//Pour vérifier que votre variable d'environnement uPNP est correcte, "WANIPConnection" par défaut dans
cette classe

//Vous pouvez faire la requete suivante, mais cela prend un certain temps.

//oUPNP:GetEnvironnement()

cUPNP est une classe

    sServerIP est chaîne
    nServerPort est entier
    sEnvironnement est chaîne

    sSeparator est chaîne
    sSeparatorHead est chaîne

    PUBLIC CONSTANT
        sLastDestination est chaîne
        nLastPort        est entier
        sLastProtocol    est chaîne

        tResponse   est un tableau dynamique de 0 par 2 chaînes
        sLastErrCode est chaîne
        sLastErrDesc est chaîne
FIN
```

Constructeur

```
PROCEDURE Constructeur(_sServerIP="",_ServerPort=5678,_sEnvironnement="WANIPConnection")

//Séparateur de ligne en XML
:sSeparator=""

//Séparateur de ligne dans la requete (HEAD)
:sSeparatorHead=RC

SI _sServerIP<>"" ALORS
    :sServerIP=_sServerIP
SINON
    sRouterIP est chaîne = NetAdresseIP()
    :sServerIP=Gauche(sRouterIP,Taille(sRouterIP)-Taille(ExtraitChaîne(sRouterIP,1,".",DepuisFin)))+"1"
FIN

:nServerPort=_ServerPort

:sEnvironnement=_sEnvironnement
```

Destructeur

```
PROCEDURE Destructeur()
```

Méthode NATOuvrePort

```
// Résumé : Ouvre un nouveau port par uPNP
// Syntaxe :
//[ <Résultat> = ] NATOuvrePort (<sDestination>, <nPort> [, <sProtocol> [, <nInternalPort> [,
<nLeaseDuration>]]])

//
// Paramètres :
//  sDestination : Adresse IP Interne
//  nPort : Port externe (accessible par internet)
//  sProtocol (valeur par défaut=TCP) : "TCP" ou "UDP"
//  nInternalPort (valeur par défaut=-1) : Port du PC en interne
//  nLeaseDuration (valeur par défaut=0) : Nombre de secondes de validité du port, 0=Infini
// Valeur de retour :
//  Type indéterminé :      Résultat, doit contenir le texte "Response" en cas de réussite
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE NATOuvrePort(sDestination,nPort,sProtocol="TCP",nInternalPort=-1,nLeaseDuration=0)

SI nInternalPort=-1 ALORS
    nInternalPort=nPort
FIN

:FermeLastPort()

P est entier
tParams est un tableau de 7 par 2 chaînes
P++;tParams[P,1]="NewRemoteHost";tParams[P,2]=sDestination
P++;tParams[P,1]="NewExternalPort";tParams[P,2]=nPort
P++;tParams[P,1]="NewProtocol";tParams[P,2]=sProtocol
P++;tParams[P,1]="NewInternalPort";tParams[P,2]=nInternalPort
P++;tParams[P,1]="NewEnabled";tParams[P,2]=1
P++;tParams[P,1]="NewPortMappingDescription";tParams[P,2]=""
P++;tParams[P,1]="NewLeaseDuration";tParams[P,2]=nLeaseDuration

RENVOYER :Requete("AddPortMapping",tParams)
```

Méthode NATFermePort

```
// Résumé : Supprime un port ouvert par uPNP
// Syntaxe :
//[ <Résultat> = ] NATFermePort (<sDestination>, <nPort> [, <sProtocol>])
//
// Paramètres :
//  sDestination : <indiquez ici le rôle de sDestination>
//  nPort : <indiquez ici le rôle de nPort>
//  sProtocol (valeur par défaut=TCP) : < indiquez ici le rôle de sProtocol >
// Valeur de retour :
//  Type indéterminé :   //     Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE NATFermePort(sDestination,nPort,sProtocol="TCP")

P est entier
tParams est un tableau de 3 par 2 chaînes
P++;tParams[P,1]="NewRemoteHost";tParams[P,2]=sDestination
P++;tParams[P,1]="NewExternalPort";tParams[P,2]=nPort
P++;tParams[P,1]="NewProtocol";tParams[P,2]=sProtocol

RENVOYER :Requete("DeletePortMapping",tParams)
```

Méthode NATGetGenericPortMappingEntry

```
// Résumé : Renvoie une entrée des mappings de ports actuellement ouverts par uPNP (par tout utilisateur
réseau)

// Syntaxe :
//[ <Résultat> = ] NATGetGenericPortMappingEntry (<nId>)
//
// Paramètres :
//  nId : <indiquez ici le rôle de nId>
// Valeur de retour :
//  Type indéterminé :      //  Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
```

```
//
PROCEDURE NATGetGenericPortMappingEntry(nId)

P est entier
tParams est un tableau de 1 par 2 chaînes
P++;tParams[P,1]="NewPortMappingIndex";tParams[P,2]=nId

RENVOYER :Requete("GetGenericPortMappingEntry",tParams)
```

Méthode NAPAdresseIPExterne

```
// Résumé : Récupère l'adresse IP Externe du routeur (IP Internet)
// Syntaxe :
//[ <Résultat> = ] NAPAdresseIPExterne ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Type indéterminé :  //  Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE NAPAdresseIPExterne()

RENVOYER :Requete("GetExternalIPAddress")
```

Méthode xmlResVersChaine

```
// Résumé : Utilisée en interne, Nettoie le superflu (l'enveloppe) du résultat xml renvoyé par la
requete uPNP

// Syntaxe :
//[ <Résultat> = ] xmlResVersChaine (<sRes>)
//
// Paramètres :
//  sRes : <indiquez ici le rôle de sRes>
// Valeur de retour :
//  Type indéterminé : //   Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
FONCTION PRIVÉE xmlResVersChaine(sRes)

//<?xml version="1.0"?>
//<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body>

//<m:AddPortMappingResponse
xmlns:m="urn:schemas-upnp-org:service:WANIPConnection:1"></m:AddPortMappingResponse></SOAP-ENV:Body>
</SOAP-ENV:Envelope>


Dimension(:tResponse,0,2)

SI Position(sRes,"SOAP-ENV:Body") > 0 ALORS
    sRes=ExtraitChaîne(sRes,2,"SOAP-ENV:Body>")
    sRes=sRes[[A Taille(sRes)-2]]

    //<m:GetGenericPortMappingEntryResponse
    xmlns:m="urn:schemas-upnp-org:service:WANIPConnection:1"><NewRemoteHost
    xmlns:dt="urn:schemas-microsoft-com:datatypes"
    dt:dt="string">192.168.0.3</NewRemoteHost><NewExternalPort xmlns:dt="urn:schemas-microsoft-com:d

    atatypes" dt:dt="ui2">1234</NewExternalPort><NewProtocol
    xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">TCP</NewProtocol><NewInternalPort
    xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="ui2">1234</NewInternalPort><NewInternalClient
    xm

    lns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string"></NewInternalClient><NewEnabled
    xmlns:dt="urn:schemas-microsoft-com:datatypes"
    dt:dt="boolean">1</NewEnabled><NewPortMappingDescription
    xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="strin
g">
    //</NewPortMappingDescription><NewLeaseDuration xmlns:dt="urn:schemas-microsoft-com:datatypes"
    dt:dt="ui4">0</NewLeaseDuration></m:GetGenericPortMappingEntryResponse>
```

```
    //Extraction du résultat dans le tableau tResponse
    P,N est entier=3
    S est chaîne=ExtraitChaîne(sRes,N,"<")
    TANTQUE Position(S,"Response>")=0 ET S<>EOT
        SI Gauche(S,1)<>"/" ALORS
            P++
            Dimension(:tResponse,P,2)
            :tResponse[P][1]=ExtraitChaîne(S,1,"")
            :tResponse[P][2]=ExtraitChaîne(S,1,">",DepuisFin)
            SI :tResponse[P][2]~=RC OU :tResponse[P][2]~=Caract(10) ALORS
                :tResponse[P][2]=""
            FIN
        FIN
        N++
        S=ExtraitChaîne(sRes,N,"<")
    FIN
FIN

:sLastErrCode=""
:sLastErrDesc=""

//En cas d'erreur
SI Position(sRes,"SOAP-ENV:Fault")
    sRes=:xmlErrVersChaine(sRes)
    Erreur(sRes)
FIN

RENVOYER sRes
```

Méthode FermeLastPort

```
// Résumé : Utilisée en interne, ferme le dernier port ouvert grace à la méthode :NATOuvrePort
// Syntaxe :
//:FermeLastPort ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE PRIVÉE FermeLastPort()

SI :sLastDestination<>"" ALORS
    SI :NATFermePort(:sLastDestination,:nLastPort,:sLastProtocol)<>"" ALORS
        :sLastDestination=""
        :nLastPort=0
        :sLastProtocol=""
    FIN
FIN
```

Méthode xmlErrVersChaine

```
// Résumé : Utilisée en interne, nettoie la réponse d'erreur XML et renvoie la chaine d'erreur, ou son
code

// Syntaxe :
//[ <Résultat> = ] xmlErrVersChaine (<sRes>)
//
// Paramètres :
//  sRes : errorDescription
// Valeur de retour :
//  chaîne :  //    Type Chaine
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
//  sRes : <indiquez ici le rôle de sRes>
FONCTION PRIVÉE xmlErrVersChaine(sRes)

//<SOAP-ENV:Fault><faultcode>SOAP-ENV:Client</faultcode><faultstring>UPnPError</faultstring>
//<detail>
//<UPnPError xmlns="urn:schemas-upnp-org:control-1-0">
//<errorCode>718</errorCode><errorDescription>ConflictInMappingEntry</errorDescription>
```

```
//</UPnPError>
//</detail>
//</SOAP-ENV:Fault>

S est chaîne

SI Position(sRes,"errorDescription") > 0 ALORS
    S=ExtraitChaîne(sRes,2,"errorDescription>")
    sRes=S[[A Taille(S)-2]]
    :sLastErrDesc=sRes
FIN
 SI Position(sRes,"errorCode") > 0 ALORS
    S=ExtraitChaîne(sRes,2,"errorCode>")
    sRes=S[[A Taille(S)-2]]
    :sLastErrCode=sRes
FIN

RENVOYER :sLastErrDesc
```

Méthode Requete

```
// Résumé : Requete Perso
// Syntaxe :
//[ <Résultat> = ] Requete (<sCommand> [, <tParams> [, <sEnv> [, <nEnvIndex>]]])
//
// Paramètres :
//  sCommand : Commande uPNP
//  tParams : Tableau de paramètre : est un tableau de X par 2 chaines
//  sEnv (valeur par défaut=WANIPConnection) : <indiquez ici le rôle de sEnv>
//  nEnvIndex (valeur par défaut=1) : <indiquez ici le rôle de nEnvIndex>
// Valeur de retour :
//  Type indéterminé :     //   Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
FONCTION Requete(sCommand,tParams=Null,sEnv=:sEnvironnement,nEnvIndex=1)

SI tParams<>Null ALORS
    SI TableauInfo(tParams,tiNombreColonnes)<>2 ALORS
        Erreur("Le tableau doit avoir 2 colonnes, nom du paramètre + valeur")
    FIN
FIN

P est entier

sXML est chaîne = ...
"<?xml version=""1.0"" encoding=""utf-8""?>"+:sSeparator+...
"<s:Envelope s:encodingStyle=""http://schemas.xmlsoap.org/soap/encoding/"" xmlns:s=""http://schemas.xmlsoap.org/soap/envelope/"">"+:
sSeparator+...
"<s:Body>"+:sSeparator+...
"<u:"+sCommand+" xmlns:u=""urn:schemas-upnp-org:service:"+sEnv+":"+nEnvIndex+""""

SI tParams<>Null ALORS

    sXML+=">"+:sSeparator

    POUR P=1 A TableauInfo(tParams,tiNombreLignes)
        SI tParams[P,2]<>"" ALORS
            sXML += "<"+tParams[P,1]+">"+tParams[P,2]+"</"+tParams[P,1]+">"+:sSeparator
        SINON
            sXML += "<"+tParams[P,1]+" />"+:sSeparator
        FIN
    FIN

    sXML+="</u:"+sCommand+">"+:sSeparator

SINON

    sXML+=" />"+:sSeparator

FIN

sXML +=...
"</s:Body>"+:sSeparator+...
"</s:Envelope>"
```

```
SI HTTPRequête("http://"+:sServerIP+":"+:nServerPort+"/"+sEnv,"","charset: utf-8"+:sSeparatorHead+"SOAPACTION: "
"urn:schemas-upnp-org:service:"+sEnv+":"+nEnvIndex+"#"+sCommand+""""+:sSeparatorHead,ChaîneVersUTF8(sXML)+:
sSeparatorHead,"text/xml") ALORS
    RENVOYER :xmlResVersChaine(HTTPDonneRésultat(HTTPRésultat))
SINON
    Erreur(ErreurInfo())
    RENVOYER ""
FIN
```

Méthode tabReponseVersChaine

```
// Résumé : Renvoie le résultat sous la forme
<Var1Name>+TAB+<Var1Value>+RC+<Var2Name>+TAB+<Var2Value>....

// Syntaxe :
//[ <Résultat> = ] tabReponseVersChaine ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  chaîne :  //    Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.
//
FONCTION tabReponseVersChaine()

sRes est chaîne
R est entier
POUR R=1 A TableauInfo(:tResponse,tiNombreLignes)
    sRes+=:tResponse[R,1]+TAB+:tResponse[R,2]+RC
FIN

SI sRes<>"" ALORS
    //Retrait du dernier RC
    sRes=sRes[[A Taille(sRes)-2]]
FIN

RENVOYER sRes
```

Méthode GetEnvironnement

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//GetEnvironnement ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//// Exemple :
// Indiquez ici un exemple d'utilisation.

PROCEDURE GetEnvironnement()

SI :Requete("GetDefaultConnectionService",Null,"Layer3Forwarding") ALORS
    SI Dimension(:tResponse)>0 ALORS
        :sEnvironnement=:tResponse[1,2]
    FIN
FIN
```