

# COINET

## KNIHOVNA PRO USNADNĚNÍ PRÁCE S TCP/IP ZÁSOBNÍKEM

Příručka uživatele a programátora



**SofCon<sup>®</sup> spol. s r.o.**  
Střešovická 49  
162 00 Praha 6  
tel/fax: +420 220 180 454  
E-mail: [sofcon@sofcon.cz](mailto:sofcon@sofcon.cz)  
www: <http://www.sofcon.cz>

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.

SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsáno. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.

SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003

Datum posledního uložení dokumentu: 16.05.2003

(Datum vydání a posledního uložení dokumentu musí být stejné)

Upozornění:

V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

**Obsah :**

---

1.O dokumentu	5
1.1. Revize dokumentu	5
1.2. Účel dokumentu	5
1.3. Rozsah platnosti	5
1.4. Související dokumenty	5
2.Termíny a definice	5
3.Úvod	6
3.1.1. Účel knihovny CoINET	6
4.Funkce a procedury	6
4.1.1. Funkce NetOpenStack	6
4.1.2. Procedura NetCloseStack	7
4.1.3. Funkce NetOpenSocket	7
4.1.4. Procedura NetCloseSocket	8
5.Příklad	9



## 1. O dokumentu

---

### 1.1. Revize dokumentu

---

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Čr		První vydání
1.10	1.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000

### 1.2. Účel dokumentu

---

Tento dokument slouží jako popis jednotky pro usnadnění práce s TCP/IP zásobníkem.

### 1.3. Rozsah platnosti

---

Určen pro programátory a uživatele programového vybavení SofCon.

### 1.4. Související dokumenty

---

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem CoBase.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

## 2. Termíny a definice

---

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

## 3. Úvod

---

### 3.1.1. Účel knihovny CoINET

Knihovna CoINET slouží ke snadnějšímu vytváření zařízení TCP/IP zásobníku (stacku). Protože TCP/IP zásobník se skládá ze čtyř vrstev, je jeho vytvoření poněkud obtížné, byla vytvořena tato knihovna, která vytváření výrazně zjednodušuje. Stručný popis TCP/IP zásobníku je uveden v dokumentaci ke knihovně CoBase.

## 4. Funkce a procedury

---

Knihovna definuje čtyři funkce. Dvě pro vytvoření a zrušení TCP/IP zásobníku (NetOpenStack a NetCloseStack) a dvě pro vytváření a rušení zásuvek transportních protokolů (NetOpenSocket, NetCloseSocket).

### 4.1.1. Funkce NetOpenStack

Funkce **NetOpenStack** vytváří TCP/IP stack.

```
function NetOpenStack( const NicClass, NicParams, IpParams,
                      Suffix: String ): TCoStatus;
```

#### Parametry:

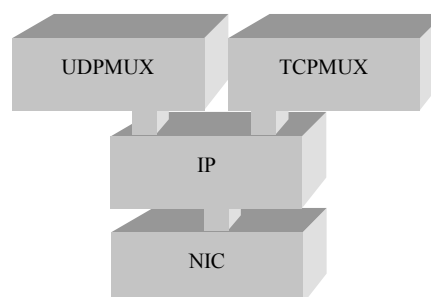
NicClass	Textový identifikátor třídy zařízení síťové karty (např. <b>ETH01</b> nebo <b>PKTDRV</b> )
NicParams	Konfigurační textový řetězec zařízení síťové karty. (viz. dokumentace ke konkrétnímu zařízení)
IpParams	Konfigurační textový řetězec zařízení protokolu IP (viz. dokumentace k zařízení protokolu IP, knihovna CoIPv4)
Suffix	Přípona názvu instance vytvořených zařízení. Použije se v případě, že máme v systému více síťových karet, v opačném případě zde uveďte prázdný řetězec.

#### Návratové hodnoty:

V případě úspěchu vrací funkce návratový kód **CST\_SUCCESS**. V opačném případě vrací chybový kód. (viz. funkce **CoCreateDevice**)

#### Poznámky:

Funkce **NetOpenStack** vytvoří a propojí čtyři instance zařízení:



**NICsuffix**                      Zařízení síťové karty  
specifikované parametry **NicClass** a **NicParams**.

<b>IPsuffix</b>	Zařízení IPv4 (IP protokol verze 4, TCoIPv4 z knihovny CoIPv4) specifikované parametrem <b>IpParams</b> .
<b>UDPMUXsuffix</b>	Multiplexer UDPMUX pro připojení UDP portů do TCP/IP stacku.
<b>TCPMUXsuffix</b>	Multiplexer TCPMUX pro připojení TCP portů do TCP/IP stacku.

Instance UDP multiplexeru resp. TCP multiplexeru je vytvořena jen v případě, že jsou tyto třídy zaregistrovány v globálním seznamu tříd (tzv. někde v programu jsou za klíčovým slovem **uses** uvedeny knihovny **CoUDP** resp. **CoTCP**)

#### 4.1.2. Procedura NetCloseStack

Procedura **NetCloseStack** uvolní TCP/IP stack vytvořený funkcí **NetOpenStack**.

```
procedure NetCloseStack( const Suffix: String );
```

##### Parametry:

Suffix Přípona názvu instance vytvořených zařízení. Musí se shodovat se stejnojmennou položkou funkce **NetOpenStack**.

##### Poznámky:

Funkce odstraní instance vytvořené při volání funkce **NetOpenStack** z globálního seznamu. Pokud již tyto instance nejsou využívány, provede se zároveň jejich uvolnění z paměti

Jsou uvolňovány instance s názvy NICsuffix, IPsuffix, UDPMUXsuffix.

#### 4.1.3. Funkce NetOpenSocket

Funkce **NetOpenSocket** vytvoří zařízení transportní vrstvy a připojí ho k TCP/IP stacku.

```
function NetOpenSocket( const PortClass, PortParams, Suffix: String;  
                          var Sock: PCoDevice ): TCoStatus;
```

##### Parametry:

PortClass Textový identifikátor třídy zařízení transportní vrstvy. Např. UDP nebo TCP.

PortParams Konfigurační textové parametry vytvářeného zařízení transportní vrstvy (viz. dokumentace ke konkrétnímu transportnímu protokolu.)

Suffix Přípona názvů instancí zařízení TCP/IP stacku.

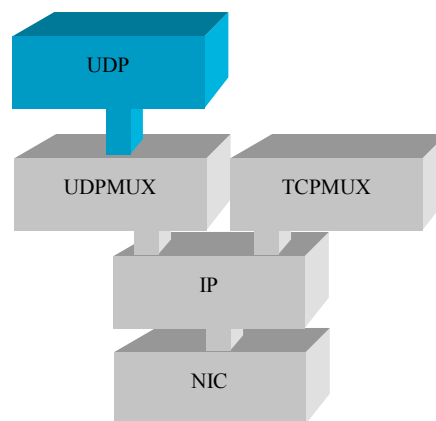
Sock Ukazatel na proměnnou, do které bude uložen ukazatel na instanci vytvořeného zařízení.

**Návratové hodnoty:**

V případě úspěchu vrací funkce návratový kód `CST_SUCCESS`. V opačném případě vrací chybový kód. (viz. funkce **CoCreateDevice**)

**Poznámky:**

Funkce se snaží vytvářené zařízení navázat na správné zařízení nižší vrstvy. Tiše se předpokládá, že instance zařízení nižší vrstvy je pojmenovaná a má název s předponou **PortClass**, tělem **MUX** a příponou **Suffix**. Takže například pokud vytváříme zařízení **UDP**, je potřeba, aby v systému bylo zaregistrované zařízení **UDPMUX**. To se jednoduše zajistí dřívějším voláním funkce **NetOpenStack**.

**Příklad:**

```
Status := NetOpenSocket( 'UDP', 'LPORT=5000', '', Sock );
```

**4.1.4. Procedura NetCloseSocket**

Procedura **NetCloseSocket** uzavře zařízení vytvořené funkcí **NetOpenSocket**.

```
procedure NetCloseSocket( var Sock: PCoDevice );
```

**Parametry:**

**Sock**            Ukazatel na instanci zařízení vytvořené funkcí **NetOpenSocket**.

**Poznámky:**

Funkce nastaví proměnnou **Sock** na hodnotu **nil**.



## 5. Příklad

---

### uses

```
CoBase,
CoEth01,
CoUDP,
CoINET;
```

*{ Ve výčtu použitých jednotek je potřeba uvést CoEth01 (ovladač konkrétní síťové karty) a CoUDP (protokol transportní vrstvy), přestože se dále v příkladu přímo nikde nevyužívají. Inicializační části těchto jednotek totiž zaregistrují zařízení v globálním seznamu tříd zařízení... }*

### const

```
cStackId = '';
```

### var

```
Status : TCoStatus; { Návratový kód funkcí }
Sock    : PCoDevice;
```

### begin

```
{ Vytvoření TCP/IP stacku, lokální IP adresa je 192.168.1.200 }
```

```
Status := NetOpenStack( 'ETH01', 'IOBASE=$300',
                        'IPADDR="192.168.1.200"', cStackId );
```

```
if Status <> CST_SUCCESS then
```

#### begin

```
  WriteLn( 'NetOpenStack() failed:', CoStatusToStr( Status ) );
  Exit;
```

```
end;
```

```
{ V tomto okamžiku máme vytvořené zařízení NIC, IP, UDPMUX a
  TCPMUX }
```

```
{ Vytvoříme instanci UDP portu, lokální port 5000 }
```

```
Status := NetOpenSocket( 'UDP', 'LPORT=5000', cStackId, Sock );
```

```
if Status <> CST_SUCCESS then
```

#### begin

```
  WriteLn( 'NetOpenSocket() failed: ', CoStatusToStr( Status ) );
  NetCloseStack( cSuffix );
  Exit;
```

```
end;
```

```
{ V Sock máme ukazatel na instanci UDP portu }
```

```
..
..
..
..
..
..
```

```
{ Po skončení práce ukazatel na instanci UDP portu uvolníme }
NetCloseSocket( Sock );
```

```
{ Zrušíme TCP/IP stack }
```

```
NetCloseStack( cStackId );
```

```
end;
```