Subject: Re: [SysInfo - Improvement - Koldo] Better way to find distribution version & more
Posted by kasome on Tue, 20 May 2014 10:59:19 GMT
View Forum Message <> Reply to Message

Hello Koldo,

There is a bug in upp\bazaar\SysInfo\SysInfo.cpp

Here is the original code

```
Array<NetAdapter> GetAdapterInfo()
{
 Array<NetAdapter> res;
 Index<String> macs;

 int sck = socket(PF_INET, SOCK_DGRAM, 0);
 if(sck < 0)
  return res;

 int bufSize = 4096;
 struct ifconf ifc = { 0 };
 byte buf[bufSize];
 for(int iTry = 0; iTry < 4; iTry++) {
  ifc.ifc_len = bufSize;
  ifc.ifc_buf = (__caddr_t)buf;

  if (ioctl(sck, SIOCGIFCONF, &ifc) < 0) {
   close(sck);
   return res;
  }

  if(ifc.ifc_len != bufSize)
   break;

  bufSize *= 2;
  if(iTry >= 3) {
   close(sck);
   return res;
  }
 }

 int nIfaces = ifc.ifc_len / sizeof(struct ifreq);
 struct ifreq *iface = ifc.ifc_req;
 for(int iIface = 0; iIface < nIfaces; iIface++) {
  String MAC;

#ifdef SIOCGIFHWADDR
```

```
  if(ioctl(sck, SIOCGIFHWADDR, iface) < 0)
   continue;

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
      (byte)iface->ifr_hwaddr.sa_data[0],
      (byte)iface->ifr_hwaddr.sa_data[1],
      (byte)iface->ifr_hwaddr.sa_data[2],
      (byte)iface->ifr_hwaddr.sa_data[3],
      (byte)iface->ifr_hwaddr.sa_data[4],
      (byte)iface->ifr_hwaddr.sa_data[5]
  );

#elif SIOCGENADDR
  if (ioctl(sck, SIOCGENADDR, iface) < 0)
   continue;

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
      (byte)iface->ifr_enaddr[0],
      (byte)iface->ifr_enaddr[1],
      (byte)iface->ifr_enaddr[2],
      (byte)iface->ifr_enaddr[3],
      (byte)iface->ifr_enaddr[4],
      (byte)iface->ifr_enaddr[5]
  );

#elif __MACH__ || __NetBSD__ || __OpenBSD__ || __FreeBSD__
  int mib[6];
  mib[0] = CTL_NET;
  mib[1] = AF_ROUTE;
  mib[2] = 0;
  mib[3] = AF_LINK;
  mib[4] = NET_RT_IFLIST;
  mib[5] = if_nametoindex(iface->ifr_name);
  if (mib[5] == 0) // nameless interface ? skip
   continue;

  int len = 0;
  if (sysctl(mib, 6, NULL, (size_t*)&len, NULL, 0) != 0)
   continue; // sysctl error, just leave MAC empty

  char macbuf[len];
  if (!macbuf)  // can't allocat buffer, skip this MAC
   continue;

  if (sysctl(mib, 6, macbuf, (size_t*)&len, NULL, 0) != 0)
   continue;

  struct if_msghdr *ifm = (struct if_msghdr *)macbuf;
```

```
  struct sockaddr_dl *sdl = (struct sockaddr_dl *)(ifm + 1);
  byte ptr[] = (byte **)LLADDR(sdl);

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
     ptr[0], ptr[1], ptr[2], ptr[3], ptr[4], ptr[5]);

#else
#error OS Distribution Not Recognized
#endif

  if(MAC == "00:00:00:00:00:00")
   MAC.Clear();

  if(!MAC.IsEmpty() && macs.Find(MAC) >= 0)
   continue;
  macs.Add(MAC);

  NetAdapter &adapter = res.Add();

  adapter.mac = MAC;

  adapter.fullname = iface->ifr_name;

  // set interface type from name
  if(adapter.fullname.StartsWith("eth"))
   adapter.type = "ETHERNET";
  else if(adapter.fullname.StartsWith("lo"))
   adapter.type = "SOFTWARE_LOOPBACK";
  else if(adapter.fullname.StartsWith("ppp"))
   adapter.type = "MODEM";
  else if(adapter.fullname.StartsWith("hci"))
   adapter.type = "BLUETOOTH";
  else if(adapter.fullname.StartsWith("tr"))
   adapter.type = "TOKENRING";
  else if(adapter.fullname.StartsWith("vbox") || adapter.fullname.StartsWith("wifi") ||
    adapter.fullname.StartsWith("ath"))
   adapter.type = "VIRTUALBOX";
  else if(adapter.fullname.StartsWith("wlan"))
   adapter.type = "IEEE80211";
  else if(adapter.fullname.StartsWith("vmnet"))
   adapter.type = "VMWARE";
  else
   adapter.type = "OTHER";

  adapter.description = adapter.fullname;

  if (!ioctl(sck, SIOCGIFADDR, iface))
   continue;
```

```
        adapter.ip4 = inet_ntoa(((struct sockaddr_in *)&(iface->ifr_addr))->sin_addr);

  iface++;
 }
 close(sck);

 return res;
}
```

Here is the fixed code

```
Array<NetAdapter> GetAdapterInfo()
{
 Array<NetAdapter> res;
 Index<String> macs;

 int sck = socket(PF_INET, SOCK_DGRAM, 0);
 if(sck < 0)
  return res;

 int bufSize = 4096;
 struct ifconf ifc = { 0 };
 byte buf[bufSize];
 for(int iTry = 0; iTry < 4; iTry++) {
  ifc.ifc_len = bufSize;
  ifc.ifc_buf = (__caddr_t)buf;

  if (ioctl(sck, SIOCGIFCONF, &ifc) < 0) {
   close(sck);
   return res;
  }

  if(ifc.ifc_len != bufSize)
   break;

  bufSize *= 2;
  if(iTry >= 3) {
   close(sck);
   return res;
  }
 }

 int nIfaces = ifc.ifc_len / sizeof(struct ifreq);
 struct ifreq *iface = ifc.ifc_req;
 for(int iIface = 0; iIface < nIfaces; iIface++) {
  String MAC;
```

```
#ifdef SIOCGIFHWADDR
  if(ioctl(sck, SIOCGIFHWADDR, iface) < 0)
   continue;

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
      (byte)iface->ifr_hwaddr.sa_data[0],
      (byte)iface->ifr_hwaddr.sa_data[1],
      (byte)iface->ifr_hwaddr.sa_data[2],
      (byte)iface->ifr_hwaddr.sa_data[3],
      (byte)iface->ifr_hwaddr.sa_data[4],
      (byte)iface->ifr_hwaddr.sa_data[5]
  );

#elif SIOCGENADDR
  if (ioctl(sck, SIOCGENADDR, iface) < 0)
   continue;

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
      (byte)iface->ifr_enaddr[0],
      (byte)iface->ifr_enaddr[1],
      (byte)iface->ifr_enaddr[2],
      (byte)iface->ifr_enaddr[3],
      (byte)iface->ifr_enaddr[4],
      (byte)iface->ifr_enaddr[5]
  );

#elif __MACH__ || __NetBSD__ || __OpenBSD__ || __FreeBSD__
  int mib[6];
  mib[0] = CTL_NET;
  mib[1] = AF_ROUTE;
  mib[2] = 0;
  mib[3] = AF_LINK;
  mib[4] = NET_RT_IFLIST;
  mib[5] = if_nametoindex(iface->ifr_name);
  if (mib[5] == 0) // nameless interface ? skip
   continue;

  int len = 0;
  if (sysctl(mib, 6, NULL, (size_t*)&len, NULL, 0) != 0)
   continue; // sysctl error, just leave MAC empty

  char macbuf[len];
  if (!macbuf)  // can't allocat buffer, skip this MAC
   continue;

  if (sysctl(mib, 6, macbuf, (size_t*)&len, NULL, 0) != 0)
   continue;
```

```
  struct if_msghdr *ifm = (struct if_msghdr *)macbuf;
  struct sockaddr_dl *sdl = (struct sockaddr_dl *)(ifm + 1);
  byte ptr[] = (byte **)LLADDR(sdl);

  MAC = Format("%02x:%02x:%02x:%02x:%02x:%02x",
     ptr[0], ptr[1], ptr[2], ptr[3], ptr[4], ptr[5]);

#else
#error OS Distribution Not Recognized
#endif

  if(MAC == "00:00:00:00:00:00")
   MAC.Clear();

  if(!MAC.IsEmpty() && macs.Find(MAC) >= 0)
   continue;
  macs.Add(MAC);

  NetAdapter &adapter = res.Add();

  adapter.mac = MAC;

  adapter.fullname = iface->ifr_name;

  // set interface type from name
  if(adapter.fullname.StartsWith("eth"))
   adapter.type = "ETHERNET";
  else if(adapter.fullname.StartsWith("lo"))
   adapter.type = "SOFTWARE_LOOPBACK";
  else if(adapter.fullname.StartsWith("ppp"))
   adapter.type = "MODEM";
  else if(adapter.fullname.StartsWith("hci"))
   adapter.type = "BLUETOOTH";
  else if(adapter.fullname.StartsWith("tr"))
   adapter.type = "TOKENRING";
  else if(adapter.fullname.StartsWith("vbox") || adapter.fullname.StartsWith("wifi") ||
    adapter.fullname.StartsWith("ath"))
   adapter.type = "VIRTUALBOX";
  else if(adapter.fullname.StartsWith("wlan"))
   adapter.type = "IEEE80211";
  else if(adapter.fullname.StartsWith("vmnet"))
   adapter.type = "VMWARE";
  else
   adapter.type = "OTHER";

  adapter.description = adapter.fullname;
```

```
  if (ioctl(sck, SIOCGIFADDR, iface) < 0) {
   iface++;
   continue;
  }

        adapter.ip4 = inet_ntoa(((struct sockaddr_in *)&(iface->ifr_addr))->sin_addr);

  iface++;
 }
 close(sck);

 return res;
}
```

Hope that helps.
