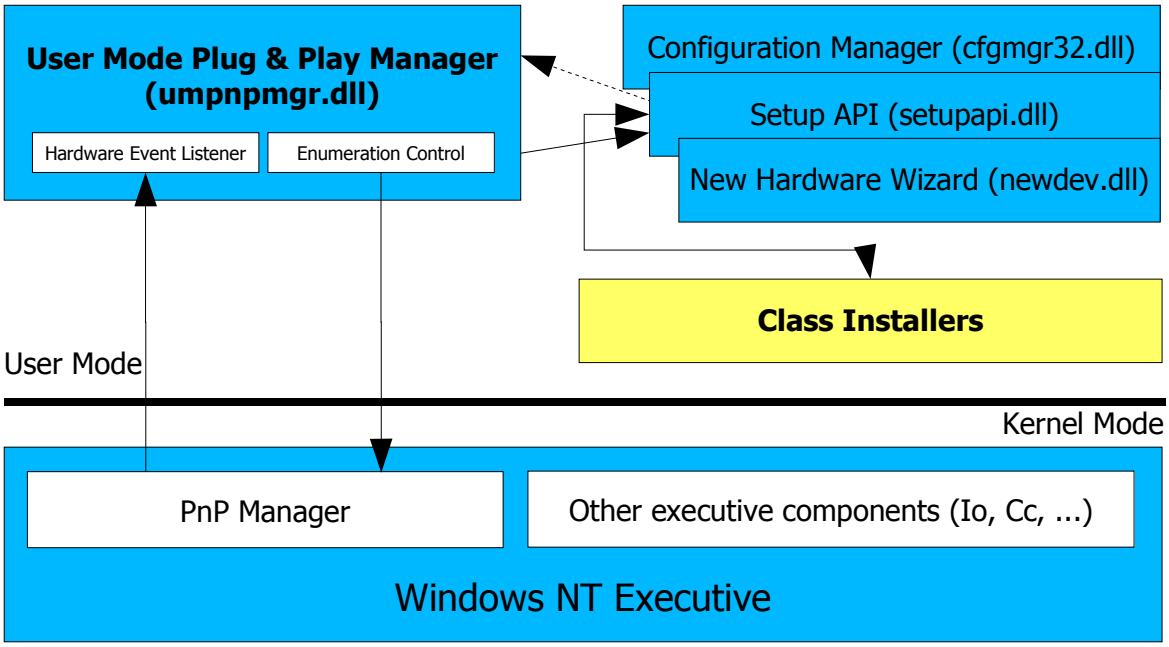
Anatomy of the Windows Plug & Play

Part I – User mode components



PnP Manager

The core of the Windows Plug & Play management is the PnP Manager that is part of the Windows NT Executive. It manages the device tree, resources (I/O ports, memory ranges, interrupts, DMA channels and bus numbers) and events (new device, device removal, ...). It's not really a user-mode component, but I had to mention it anyway.

User Mode Plug & Play Manager

is responsible for presenting the PnP events to the user. It's a service running in the context of the service manager. We can divide it into two separate components:

- **Hardware Event Listener**

  which handles the PnP event queue loop. It communicates with the kernel using a special API - NtGetPlugPlayEvent. This APIs is available for exclusive use by the UM PnP manager. The caller is required to have the SeTcbPrivilege privilege and even if this privilege is granted to some other application, the call to this API will not work (in the best case, actually it can even crash the system). It's a blocking call and the Kernel Mode PnP Manager stores most of the PnP event information in global variables. Hardware Event Listener contains a loop which repeatedly calls NtGetPlugPlayEvent and waits till it gets some event. Once an event is received, it's type is examined and appropriate action is taken (eg. showing the New Hardware Wizard for a “New Device” event).

Note:
The Win32 WM_DEVICECHANGE message is also sent by the Hardware Event Listener with the help of the BroadcastSystemMessage API.

- **Enumeration Control**

  Another special API used by the UM PnP manager is NtPlugPlayControl. It's an universal dispatch point for sending control commands to the kernel PnP Manager. The “Enumeration Control” part of the UM PnP manager consists of an RPC stub listening to the requests from the Setup DLL trio (SetupAPI, CfgMgr32 and NewDev) and translating them into the kernel format. An example of such request can be re-enumerating of a part of the device tree or getting device status.

Setup DLL trio

- **SetupAPI (setupapi.dll)**

  is the oldest member of the family. It's been present since the original NT 3.1 release and haven't changed much over the time. It contains the .INF file parser, helper functions for device installation (used by Class Installers) and log manager.

- **Configuration Manager (cfgmgr32.dll)**

  provides access to the device tree and generally to most features exposed by the UM PnP manager RPC control pipe. Nowadays these functions are present directly in SetupAPI and CfgMgr32.dll is just a forwarder.

- **New Hardware Wizard (newdev.dll)**

  contains User Interface APIs for device driver installation. The actual device installation is managed using SetupAPI calls.

Class Installers

are special DLLs that correspond to each device class (Net, Display, etc.) registered in the system. They handle the process of the installation for drivers of their type. That can include writing entries to registry, notifying kernel mode drivers of changes or other types of actions. The class installers for the most common device types (Storage, Display, Keyboard, Media, Monitor, Mouse, Net, NetClient, NetService, NetTrans, Ports, Printer, SCSIAdapter, Infrared, Image, TapeDrive, etc.) are already supplied by the system. In most cases one doesn't need to create his own class installer and can manage the job with a coinstaller. Coinstallers are yet another type of setup DLLs. They're distributed along with device drivers and can do device-specific installation or filter the class installer requests. When a new driver is installed, SetupAPI choses the corresponding class installer/coinstaller based on the .INF file and information in the registry (HKLM/System/CurrentControlSet/Control/Class).