# ACPI Scan Handlers


**Author:** Rafael J. Wysocki <rafael.j.wysocki@intel.com>

During system initialization and ACPI-based device hot-add, the ACPI namespace is scanned in search of device objects that generally represent various pieces of hardware. This causes a struct acpi_device object to be created and registered with the driver core for every device object in the ACPI namespace and the hierarchy of those struct acpi_device objects reflects the namespace layout (i.e. parent device objects in the namespace are represented by parent struct acpi_device objects and analogously for their children). Those struct acpi_device objects are referred to as "device nodes" in what follows, but they should not be confused with struct device_node objects used by the Device Trees parsing code (although their role is analogous to the role of those objects).

During ACPI-based device hot-remove device nodes representing pieces of hardware being removed are unregistered and deleted.

The core ACPI namespace scanning code in drivers/acpi/scan.c carries out basic initialization of device nodes, such as retrieving common configuration information from the device objects represented by them and populating them with appropriate data, but some of them require additional handling after they have been registered. For example, if the given device node represents a PCI host bridge, its registration should cause the PCI bus under that bridge to be enumerated and PCI devices on that bus to be registered with the driver core. Similarly, if the device node represents a PCI interrupt link, it is necessary to configure that link so that the kernel can use it.

Those additional configuration tasks usually depend on the type of the hardware component represented by the given device node which can be determined on the basis of the device node's hardware ID (HID). They are performed by objects called ACPI scan handlers represented by the following structure:

```
struct acpi_scan_handler {
        const struct acpi_device_id *ids;
        struct list_head list_node;
        int (*attach)(struct acpi_device *dev, const struct acpi_device_id *id);
        void (*detach)(struct acpi_device *dev);
};
```

where ids is the list of IDs of device nodes the given handler is supposed to take care of, list_node is the hook to the global list of ACPI scan handlers maintained by the ACPI core and the .attach() and .detach() callbacks are executed, respectively, after registration of new device nodes and before unregistration of device nodes the handler attached to previously.

The namespace scanning function, acpi_bus_scan(), first registers all of the device nodes in the given namespace scope with the driver core. Then, it tries to match a scan handler against each of them using the ids arrays of the available scan handlers. If a matching scan handler is found, its .attach() callback is executed for the given device node. If that callback returns 1, that means that the handler has claimed the device node and is now responsible for carrying out any additional configuration tasks related to it. It also will be responsible for preparing the device node for unregistration in that case. The device node's handler field is then populated with the address of the scan handler that has claimed it.

If the .attach() callback returns 0, it means that the device node is not interesting to the given scan handler and may be matched against the next scan handler in the list. If it returns a (negative) error code, that means that the namespace scan should be terminated due to a serious error. The error code returned should then reflect the type of the error.

The namespace trimming function, acpi_bus_trim(), first executes .detach() callbacks from the scan handlers of all device nodes in the given namespace scope (if they have scan handlers). Next, it unregisters all of the device nodes in that scope.

ACPI scan handlers can be added to the list maintained by the ACPI core with the help of the acpi_scan_add_handler() function taking a pointer to the new scan handler as an argument. The order in which scan handlers are added to the list is the order in which they are matched against device nodes during namespace scans.

All scan handles must be added to the list before acpi_bus_scan() is run for the first time and they cannot be removed from it.