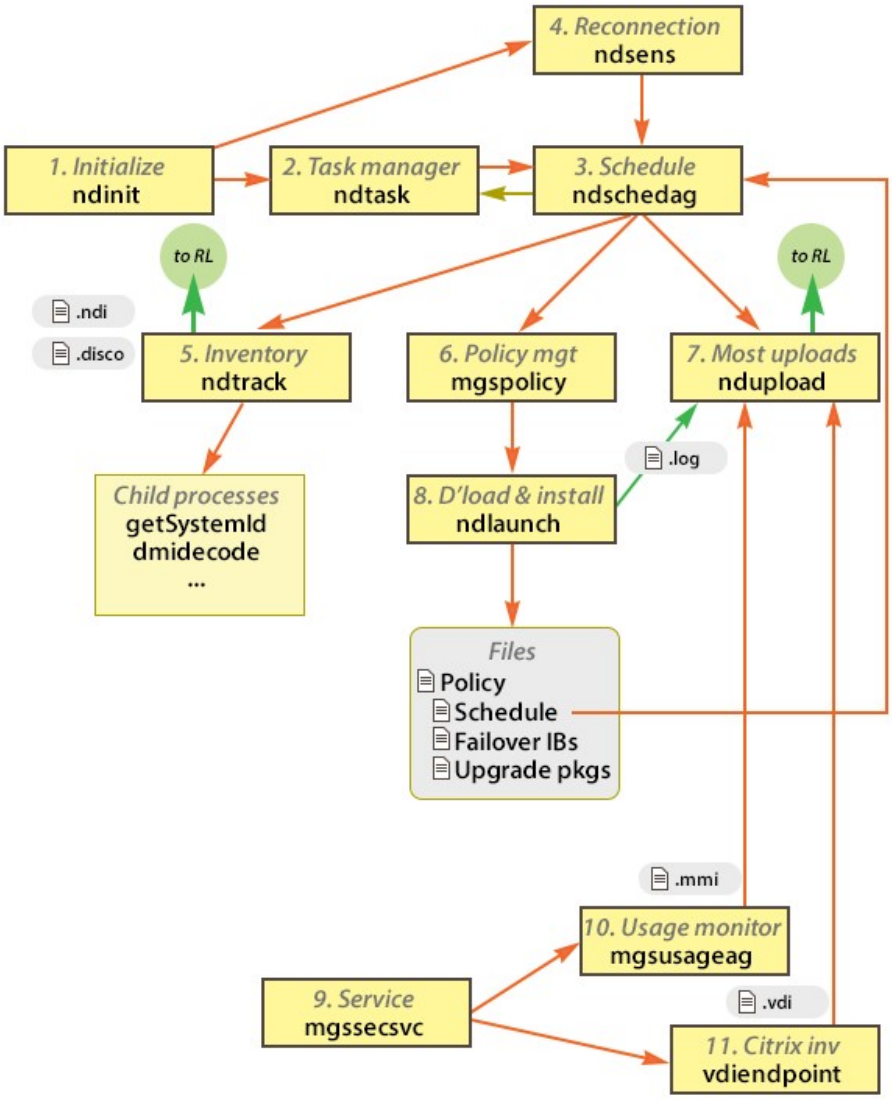

1. The `ndinit` component exists only on Windows. It is automatically initialized as a service on machine reboot, and is responsible for starting `ndtask` and, on reconnection to the network, `ndsens`.
2. The `ndtask` component is functionally identical with `mstask` (part of the Microsoft Task Scheduler), but is available across platforms. On UNIX-like platforms, it runs as a service. It functions rather like a to-do list, but does not start tasks directly. Instead, it invokes the `ndschedag` component to trigger tasks when required.
3. The schedule component (`ndschedag`) has two main functions:
   - When handed a new schedule and invoked by `ndlaunch`, `ndschedag` unpacks the contents of the schedule file (see Schedule Files (.nds)) and saves the details in separate task files (`.nts`) for use by `ndtask`. It then sends an IPC message to `ndtask` to process the updated task list.

- When a specified task is triggered, `ndschedag` is always invoked (most often by `ndtask`, and occasionally on Windows by `ndsens`), and then coordinates execution of the appropriate components (in particular, inventory gathering, policy updates, and data uploads).

4. The `ndsens` component (available only on Windows) is invoked only when the network connection is reestablished for the device on which the FlexNet inventory agent is running. When, after a disconnection, the device reconnects to the network, `ndsens` next invokes `ndschedag` to check for any scheduled events with an `OnConnect` trigger type (these are normally the `Update Machine Policy`, `Update Client Settings`, and `Upload Client Files` events). There is no equivalent functionality for the FlexNet inventory agent installed on UNIX-like machines.

5. All inventory gathering and discovery work on the local device is the responsibility of the tracker, or `ndtrack`. Once triggered by `ndschedag`, this collects inventory data in `.ndi` files, and discovery data in `.disco` files. Depending on configuration, compressed archives of `.ndi` files are produced, and archives of `.disco` files may also be produced. By default (in the case of the full FlexNet inventory agent), the tracker also tries an immediate upload of gathered data files to the `ManageSoftRL` file share on its chosen inventory beacon. If configuration or some transient networking problem prevents this upload, the files are saved on the local file system, where they are subsequently picked up by the `ndupload` component. The tracker also invokes other components of the FlexNet inventory agent, such as `getSystemId` on Windows and a supplied version of `dmidecode` on certain legacy Linux platforms. For more information about operating system elements that are also invoked by the tracker, see the platform-specific listings under Common: Child Processes Invoked by the Tracker.

6. The policy component (`mgspolicy`) is responsible for managing the various rules that the control the overall FlexNet inventory agent. Historically it had a significant role (when "client-side policy" was an option); but now that all agent policy is prepared by the inventory beacon ("server-side policy"), its main role is to invoke the download and installation component (`ndlaunch`) to check for, and if necessary download, changed policy.

7. The upload component (`ndupload`) is responsible for most uploads from the local device to its chosen inventory beacon, using the same `ManageSoftRL` file share mentioned above. (When, instead, the tracker uploads discovery and inventory results, it is using a shared library of common code from `ndupload`, so that the upload functionality is identical. This integration supports uploads in other configuration of the tracker, such as in the Core deployment case and the Zero-footprint case, when the separate `ndupload` component is not available.) The uploader may be invoked directly by other components to upload files handed off through the command line; or it may be invoked by the scheduler to look in defined folders on the local device and upload all files present there. In this way, `ndupload` provides a scheduled catch-up service to upload files which the tracker failed to upload, and saved on disk instead. (And it also follows that the *absence* of the separate `ndupload` component in the Core deployment and Zero-footprint cases means that there can be no catch-up uploads after a transient failure of uploads by the tracker.)

8. The download and installation component, `ndlaunch`, is the only component that downloads files from an inventory beacon to the FlexNet inventory agent on the local device. In each case, it checks for changed content, and only downloads files that have been updated. The first file checked is the device policy for the FlexNet inventory agent. As well as summarizing the applicable rules for agent behavior, the policy file (see Policy Files (.npl)) points to a number of other resources that the launcher downloads on demand (when changes have occurred). These include:

   - The agent configuration file
   - The current version of `InventorySettings.xml`
   - The list of available inventory beacons to which the FlexNet inventory agent may upload data ('available'

inventory beacons are generally those with IIS configured for anonymous authentication)

- The upgrade packages that support self-upgrade of the installed FlexNet inventory agent
- The agent schedule, which is either of:
  - The default global schedule set for all installed agents in the web interface of FlexNet Manager Suite
  - The special schedule used for the FlexNet inventory agent on devices linked to IBM PVU licenses, when FlexNet Manager Suite is configured for 30-minute inventory updates and sub-capacity license calculations (as an alternative to ILMT).

  If `ndlaunch` downloads a revised schedule, it invokes `ndschedag` to respond appropriately.

Depending on the agent configuration and the available upgrade packages, the launcher may download a new version of the FlexNet inventory agent, and automatically install it. The launcher also saves log files for all its activities to the local disk, from where they are subsequently uploaded by `ndupload` to support status reporting on the central application server. This means that central reporting is dependent on the schedule for uploads, which is typically set to once per day, overnight.

9. Another service that runs exclusively on Microsoft Windows is `mgssecsvc` (there is no equivalent on UNIX-like platforms). Like `ndinit`, it is automatically initialized as a service on machine reboot. It exists solely as a wrapper for its child processes (which are implemented as DLLs on Windows, and so are running whenever the service is running).
10. The component that monitors application usage (when you have usage tracking configured) is `mgsusageag`, which is a library exercised by `mgssecsvc` on Windows. On UNIX-like platforms, `mgsusageag` is a service in its own right. Because of the ephemeral nature of usage data, `mgsusageag` invokes the uploader any time it has usage data (an `.mmi` file) to upload. This means that application usage data is uploaded asynchronously with relation to the upload schedule saved on the local device.
11. Similarly, for use when gathering virtual desktop inventory, the `vdiendpoint` component is implemented as a shared library on Microsoft Windows (there is no equivalent on UNIX-like platforms). Since the use of the local device as an end-point for virtual applications is (like usage data) also ephemeral, transient information, the resulting `.vdi` file is handed off immediately to `ndupload` for immediate transfer, without reference to the upload schedule.

Deployment Overview: Where to, and How

Each of the FlexNet inventory agent and the FlexNet inventory core components can be deployed in different ways, and to different places within your network hierarchy. These decisions affect both the immediate deployment effort, and the ongoing management effort. In some cases, the options also impact functionality and system requirements. (These distinct names for the two entities were defined in What Can Be Used for FlexNet Inventory Collection.)

FlexNet inventory agent

FlexNet inventory agent must always be installed on the target inventory device. There are two main deployment options:

- Automatic deployment through FlexNet Manager Suite, managed by the inventory beacons, in accordance with the targets you declare in the web interface. This process is called "adoption", since it 'adopts' the target device into a