# Alpaca Discovery

Draft 0.5

## Introduction

Clients can discover Windows COM based drivers through the ASCOM's registry-based Chooser capability, however, Alpaca devices can run on any operating system and may be on separate devices to clients. In consequence, Alpaca devices need a discover mechanism to enable clients to be aware of their existence and this document describes the Alpaca Discovery Protocol and provides advice on implementation.

## Definitions

For the following document Device will refer to something (a driver or device) that exposes the Alpaca interface and Client will refer to client applications that want to locate and use the Device's API(s).

- **DISCOVERY PORT**: this is the port to which the Client Broadcasts the discovery message and on which the Device listens. The Alpaca default discovery port is 32227.
- **DISCOVERY MESSAGE**: is the message sent by the client via broadcast on the DISCOVERY PORT.
- **RESPONSE MESSAGE**: is the message that the Device sends back via unicast to the client.
- **ALPACA PORT**: is the port on which the Alpaca management and device APIs are available.
- **ASCOM DEVICE**: An implementation of an ASCOM device interface such as ITelescope or IFocuser that can be accessed through the Alpaca Device API protocol.
- **UNIQUE ID**: A string identifier for an ASCOM DEVICE that is globally unique.
- **ALPACA DEVICE:** Hardware or software that supports the Alpaca Management and Alpaca Device API protocols to provide access to one or more ASCOM DEVICES.

## Alpaca Discovery Protocol[1]

Clients find devices through a UDP based protocol that uses:

- the IPv4 network broadcast address
- a designated IP port number, whose default is 32227
- a structured DISCOVERY MESSAGE
- a structured RESPONSE MESSAGE

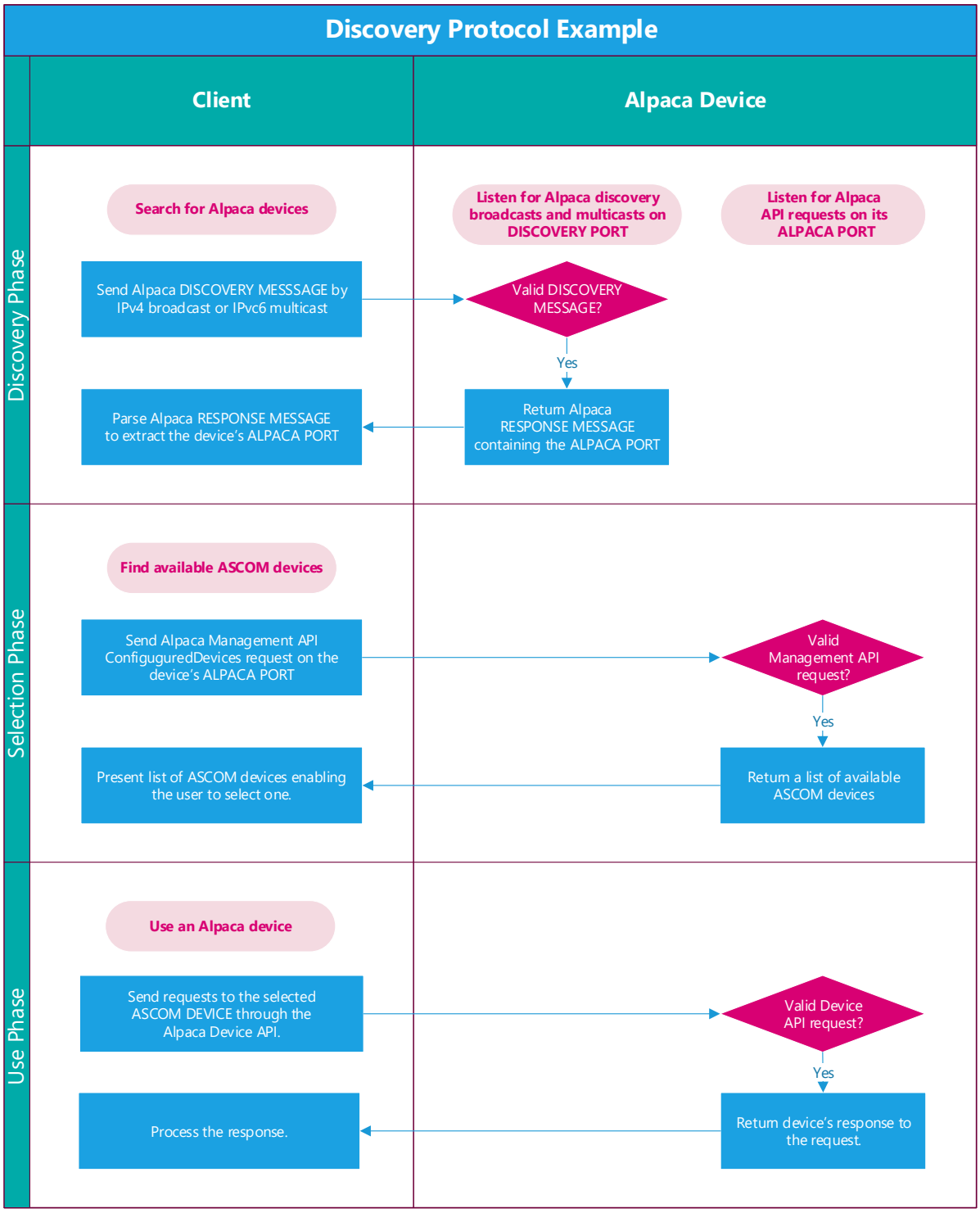
To search for and use ALPACA DEVICES, a client should:

1. Transmit a DISCOVERY MESSAGE to the DISCOVERY PORT by broadcast (IPv4).
    a. Discoverable ALPACA DEVICES will listen for IPv4 broadcast transmissions on the DISCOVERY PORT and assess each message to confirm whether it is a valid DISCOVERY MESSAGE.
    b. If the request is valid the device will return a RESPONSE MESSAGE indicating the device's ALPACA PORT.
2. Use the IP address from the RESPONSE MESSAGE together with the ALPACA PORT from the DISCOVERY RESPONSE to query the Alpaca Management API to determine which ASCOM DEVICES and device types are available.
3. When selected by the user, access specific devices through the ALPACA DEVICE API that also runs on the ALPACA PORT at the IP address of the initiator of the RESPONSE MESSAGE.

Implementation requirements are listed in *Appendix 1 - Implementation Requirements.*

[1] The IPv4 protocol described is finalised and an IPv6 protocol is being worked up

The following figure gives a conceptual overview of the discovery process.




More detailed implementations are shown in *Appendix 3 - Detailed Discovery Protocol Example.*

## Discovery Message Format

To provide for future extension, if required, the DISCOVERY MESSAGE has a structured format:

| Byte Number | Content |
|---|---|
| 0::14 | Fixed ASCII text: **alpacadiscovery** |
| 15 | ASCII Version number: 1 for the current version. The version number sequence is 1::9 then A::Z |
| 16::63 | ASCOM reserved for future expansion |

The current, version 1, discovery message therefore contains 16 bytes, comprising 15 bytes from the ASCII text alpacadiscovery together with a single ASCII version byte:

**alpacadiscovery1**

```
Hex: 0x61, 0x6C, 0x70, 0x61, 0x63, 0x61, 0x64, 0x69, 0x73, 0x63, 0x6F, 0x76, 0x65, 0x72, 0x79, 0x31
```

The discovery message “alpacadiscovery” has been registered to ASCOM in the IANA service registry:

https://www.iana.org/assignments/service-names-port-numbers

## Discovery Response Format

The ALPACA DEVICE response must be a JSON object containing the device’s ALPACA PORT e.g.:

```
{
  "AlpacaPort":12345
}
```

## Unique IDs (UID)

An ASCOM DEVICE’s UID is returned within the Alpaca Management API ConfiguredDevices response.

The UID is an ASCII string that MUST be unique to each ASCOM DEVICE. Its purpose is to help clients re-discover a previously used ASCOM DEVICE if its IP address changes.

1. The UID must be derived from a 48bit or larger space and converted to an ASCII string.
2. The UID must be exposed through the **UniqueID** field of the Alpaca Management API ConfiguredDevices response.
3. Manufacturers and developers must use appropriate algorithms to ensure that otherwise identical devices have different UIDs.
4. Alpaca Devices MUST send the same UIDs on every network interface to which the device is attached.
5. Once a UID has been assigned to an ASCOM DEVICE, it must never change. This means that:
    a. An ASCOM DEVICE served by an ALPACA DEVICE will always be uniquely identifiable through the assigned UID .
    b. UID must be retained when devices are powered down.

Approaches for generating UIDs are described in *Appendix 2 - Creating Unique IDs.*

# Appendix 1 - Implementation Requirements

## Alpaca Clients and Devices

1. Devices and Clients SHOULD support the Alpaca discovery protocol by default.
2. When discovery is supported Devices and Clients MUST provide a mechanism for the end user to change the DISCOVERY PORT from the default value if required.

## Alpaca Clients

1. Clients MAY broadcast the DISCOVERY MESSAGE multiple times because UDP is a non-guaranteed delivery protocol and packets may be lost.
2. Clients SHOULD combine the responses to remove duplicate responses.
3. Clients MUST be able to read the ALPACAPORT from a valid json message, even if there are additional variables in the message.
4. Clients MUST be able to handle responses from multiple Devices.
5. Client SHOULD offer a mechanism to report or log incorrect responses.
6. Clients SHOULD NOT try to connect to devices that respond incorrectly.
7. The ALPACA PORT must be configurable by the end user to support cases where intermediate proxies translate the operating port to a different value as seen by the client.
8. Clients MUST NOT bind their response socket to the DISCOVERY PORT. Clients SHOULD bind their response socket to a system assigned port.
9. Clients SHOULD broadcast the DISCOVERY MESSAGE to the unique broadcast address of each network interface rather than to address 255.255.255.255 in order to maximize compatibility with networking equipment and helping to ensure that the packets are routed correctly.

## Alpaca Devices

1. Devices must only respond to discovery broadcasts from interfaces where the Alpaca Management and Device APIs can be accessed.
2. Devices should allocate a minimum buffer size of 64 bytes to hold the received Alpaca discovery message to ensure that they can handle any potential future discovery protocol messages.
3. Devices MUST allow the ALPACA PORT to be changed by the end user to handle the Device being behind a proxy.
4. *The DISCOVERY RESPONSE MUST be made via unicast from a system assigned port and be directed to the port and IP address from which the client sent the DISCOVERY MESSAGE.*
5. Devices that run within a shared context such as Linux or Windows OS, must be implemented to share the DISCOVERY PORT with other Devices running within the shared context. This means that Devices MUST open the socket with the language equivalents of SO_REUSEADDR / SO_REUSEADDR / SO_REUSEPORT on Windows / Linux / OSX respectively. Devices in a shared context MUST NOT require or use root /admin access to ensure that other Devices can use the port.
6. Security requirements
    a. Devices SHOULD NOT be open to the Internet and SHOULD ONLY respond to trusted IP addresses on link-local and private networks.
    b. Devices SHOULD respond to every discovery request, although they SHOULD rate limit responses to prevent UDP amplification attacks.

Add Daniel's security section

# Appendix 2 - Creating Unique IDs

There are several potential ways of generating UIDs. Some examples are:

- A random number of 48 or more bits expressed as a hex string
- An EUI-48 or EUI-64 as described in IEEE EUI Tutorial expressed as a hex string
- The device's MAC address can be used by *Alpaca Devices* that present a single ASCOM DEVICE
- A UUID[2] as described in RFC 4122 expressed as a string: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX without surrounding quotes, square brackets or curly braces

The standard intentionally avoids specifying a source or format for the UID in order to provide flexibility for manufacturers and developers. For example, low power devices could use the EUI-48 or EUI-64 embedded in the device's networking chip while Linux or .NET developers could use a UUID[2] that is dynamically generated by the host.

## Options for Programmatic UUID[2]Creation

| Environment | Code Example |
|---|---|
| .NET | `string guidString = System.Guid.NewGuid().ToString();` |
| Windows C++ | `GUID guid;`<br>`CoCreateGuid(&guid);`<br><br>`OLECHAR* guidString;`<br>`StringFromCLSID(guid, &guidString);`<br><br>`// use guidString...`<br><br>`// ensure memory is freed`<br>`::CoTaskMemFree(guidString);` |
| Linux | `cat /proc/sys/kernel/random/uuid`<br>Returns: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX |
| MacOS | `uuidgen`<br>Returns: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX |

## Device Startup Checks

On startup the device should check whether it has already assigned UID to its ASCOM DEVICES and, if not, must create and persist them.

# Basic flow Example

A Client on 192.168.0.10 broadcasts a UDP packet containing the DISCOVERY MESSAGE on the DISCOVERY PORT. Devices listening for broadcasts on this port respond directly to the Client with the response message. Clients pull the Alpaca API port from the message and then can query the management API for details about the Device.

[2] On Microsoft operating systems a UUID is known as a GUID.

# Appendix 3 - Detailed Discovery Protocol Examples