

|  |  |
| --- | --- |
|  | **imotion** |
|  |  |
|  | UDCP - Unit Discovery and Configuration Protocol |
|  |  |
|  | Protocol Description |
|  |  |
|  | Technical Description |
|  |  |
|  | Version: 0.1 |
|  | State: Draft |
|  | Classification: Internal use only |
|  | Author: WES |
|  | Creation date: 2015-06-16 |
|  | Repository: $/Gorba/Main/Common/Protocols/Documents/TD\_UDCP.docx |
|  | Gorba AG  Sandackerstrasse  9245 Oberbüren  Switzerland |

**Table of contents**

[1 Introduction 4](#_Toc423070502)

[1.1 Scope 4](#_Toc423070503)

[1.2 Intended Audience 4](#_Toc423070504)

[2 Product Overview 5](#_Toc423070505)

[2.1 Purpose 5](#_Toc423070506)

[2.2 Supported Devices 5](#_Toc423070507)

[3 Communication 6](#_Toc423070508)

[3.1 UDP/IP Parameters 6](#_Toc423070509)

[3.2 Addressing 6](#_Toc423070510)

[3.3 Requests and Responses 6](#_Toc423070511)

[3.4 Datagram Structure 6](#_Toc423070512)

[3.5 Endian-ness 6](#_Toc423070513)

[3.6 String Encoding 6](#_Toc423070514)

[3.7 Error Handling 6](#_Toc423070515)

[4 Datagrams 7](#_Toc423070516)

[4.1 UDP Header 7](#_Toc423070517)

[4.1.1 Source Port Number 7](#_Toc423070518)

[4.1.2 Destination Port Number 7](#_Toc423070519)

[4.1.3 Length 7](#_Toc423070520)

[4.1.4 Checksum 7](#_Toc423070521)

[4.2 UDCP Header 7](#_Toc423070522)

[4.2.1 Header Flags 7](#_Toc423070523)

[4.2.2 Datagram Types 8](#_Toc423070524)

[4.3 UDCP Fields 8](#_Toc423070525)

[4.3.1 Field Types 8](#_Toc423070526)

[4.3.2 Error Codes 9](#_Toc423070527)

[5 Supported Datagram Types 10](#_Toc423070528)

[5.1 Get Information Request 10](#_Toc423070529)

[5.2 Get Information Response 10](#_Toc423070530)

[5.3 Set Configuration Request 10](#_Toc423070531)

[5.4 Set Configuration Response 11](#_Toc423070532)

[5.5 Announce Request 11](#_Toc423070533)

[5.6 Announce Response 11](#_Toc423070534)

[5.7 Reboot Request 11](#_Toc423070535)

[5.8 Reboot Response 11](#_Toc423070536)

**Modification management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Modifications** | **State** |
| 0.1 | 2015-06-26 | WES | SW Dev | Initial version | draft |

**Review**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Release**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

## Scope

This document is a technical description of the Unit Discovery and Configuration Protocol. It gives all information to Gorba developers to be able to develop with and for this protocol.

This document is not intended to be a user manual.

The goal is to give a precise description of the internals of the protocol as well as its usage in real-world scenarios.

## Intended Audience

This document is written to be understood by Gorba developers. Advanced technical skills are required.

# Product Overview

## Purpose

The Unit Discovery and Configuration Protocol (UDCP) was created to provide a way to find and (temporarily) configure Gorba devices (Units) without having to access the device physically.

UDCP uses UDP datagrams over IP to communicate with devices. It uses the broadcast IP address and is therefore only suitable for use in the same subnet, but this allows the protocol to work between devices with incompatible IP settings.

When a user wants to access a device of which he doesn’t know the address configuration, he needs to be able to:

1. Discover the device(s) on the local network
2. Clearly identify the device if multiple devices are found
3. Reconfigure the device to match the user’s PC local IP settings

UDCP provides these functionalities through different datagrams.

## Supported Devices

UDPC was created not only with .NET applications in mind, but also for very limited embedded devices like exterior signs or control units. Currently it is implemented in the imotion TFT software (Hardware Manager) and used by icenter.diag.

# Communication

## UDP/IP Parameters

UDCP uses only UDP datagrams over IP. The default UDP port is **1600** and the IP address **255.255.255.255** is used for all communication.

The latter means datagrams will almost never leave a local network. This limitation is given by the fact that routers block broadcast traffic for the obvious reason of preventing packet floods – imagine every device in the internet sending a broadcast message once in a second and sending it to all devices in the internet.

## Addressing

Since all datagrams are always sent as broadcast messages, the header of every datagram contains the UDCP address of a device. See also chapter 4 for more details about the structure of datagrams.

The UDCP address can be the MAC address (6 bytes) of any network interface, but this is not absolutely required; it should just be ensured that a UDCP address is as unique as possible.

When a datagram is meant to be read by all devices, its address must be set to FF-FF-FF-FF-FF-FF. This means a device must only react to messages sent to its own address or this broadcast address.

## Requests and Responses

Datagrams are split into two distinct categories: requests and responses. For every request there is also a response. Both share the same type identifier, but have different header flags (see chapter 4.2.1).

## Datagram Structure

Every UDCP datagram has the same basic structure:

1. Header
2. One or more fields consisting of:
   1. Type
   2. Length of content
   3. Content

## Endian-ness

All multi-byte integers in UDCP are sent as little-endian (contrary to the UDP header which uses big-endian). This means for example:

1543 as a 16-bit integer is represented by 07 06.

## String Encoding

All strings are encoded as non-null-terminated UTF-8 strings with little-endian characters. Example:

“Hello Wörld” is encoded as  
48 65 6C 6C 6F 20 57 C3 B6 72 6C 64

The strings don’t contain any header (BOM or length prefix) or footer (null-terminator).

## Error Handling

The protocol supports a very simple error reporting where a response can contain information about what went wrong when reading or handling a request. The possible error codes can be found in chapter 4.3.2. Any response datagram can contain error information.

# Datagrams

## UDP Header

The UDP header is defined in RFC768.

### Source Port Number

The source port number should be 1600.

### Destination Port Number

The destination port number always has to be 1600.

### Length

The length field has to be set to the correct value (including the UDP header).

### Checksum

The UDP checksum field should be calculated correctly by the transmitter and must be verified by the receiver.

## UDCP Header

Immediately following the UDP header is the UDCP header. From a UDP point of view, the UDCP header is the beginning of the UDP payload.

The UDCP header consists of eight bytes:

* 1 byte header flags
* 1 byte datagram type
* 6 bytes UDCP address

### Header Flags

The following flags are currently defined:

|  |  |  |  |
| --- | --- | --- | --- |
| Bit | Name | Description | Remarks |
| 0 | Direction | 0: Request, 1: Response |  |
| 1 | Reserved 1 | Unused | Has to be “0” |
| 2 | Reserved 2 | Unused | Has to be “0” |
| 3 | Reserved 3 | Unused | Has to be “0” |
| 4 | Reserved 4 | Unused | Has to be “0” |
| 5 | Reserved 5 | Unused | Has to be “0” |
| 6 | Reserved 6 | Unused | Has to be “0” |
| 7 | Reserved 7 | Unused | Has to be “0” |

Table 1 Supported header flags

The header flags may at a later point also be used to distinguish different versions of the protocol.

### Datagram Types

Currently the following datagram types are supported:

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| 0 | Get Information | Requests information about a device or provides that information. |
| 1 | Set Configuration | Temporarily sets the device configuration or acknowledges it. |
| 2 | Announce | Makes the unit announce itself or acknowledges it. |
| 3 | Reboot | Requests the unit to reboot or acknowledges it. |

Table 2 Supported datagram types

The datagram types are the same for requests and responses. See chapter 5 for more information about the different types.

## UDCP Fields

Immediately following the UDCP header are the UDCP fields. The UDCP fields fill the remainder of the UDP payload.

The fields are one or more tag-length-value (TLV) triplets (see below). The number of fields can only be figured out by reading one field after the other. When the end of the UDP payload is reached, one knows there are no more triplets available.

The receiver must be able to skip over unknown triplets. This means that if a tag is unknown, the receiver still has to read the length and then simply skip over the given number of bytes to read the next tag (or get to the end of the packet).

Each field has the following structure:

* 1 byte field type
* 1 byte length of content
* n bytes content

### Field Types

The field types are the same for all datagram types, but not all datagram types support or require all fields. See chapter 5 for more information about the different datagram types and their fields.

Currently the following field types are supported:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Type | Description |
| 0 | None | String | Not a valid field type, only used for “Error Field” (see below) |
| 1 | Unit Name | String | Name of the unit |
| 2 | Software Version | String | Version of the software, this can be also just the version of the component responsible for UDCP (e.g. Hardware Manager) |
| 3 | IP Address | 4 bytes | IPv4 address of the unit |
| 4 | Network Mask | 4 bytes | IPv4 network mask of the unit |
| 5 | Gateway | 4 bytes | IPv4 gateway address of the unit |
| 6 | DHCP Enabled | 1 byte | If set to 1, the IPv4 settings should be obtained using DHCP |
| 100 | Error Code | 1 byte | Kind of error that occurred |
| 101 | Error Field | 1 byte | Field type of the field that contained an error |
| 102 | Error Message | String | Human readable error message |

Table 3 Supported field types

### Error Codes

The “Error Code” can contain one of the following error codes:

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| 0 | OK | Everything was OK. |
| -1 | Bad Datagram | The datagram could not be read. |
| -2 | Bad Field | One of the fields could not be read. If the datagram contains this error code, it should also contain a field of type “Error Field” telling which field was bad. |
| -3 | Could Not Process | The datagram could not be processed by the device (e.g. a functionality is not supported on a given device). |

Table 4 Supported error codes

# Supported Datagram Types

Every datagram has a list of required and optional fields, this chapter lists all types with their respective fields.

## Get Information Request

This request is usually sent from a PC to all devices using the address FF-FF-FF-FF-FF-FF to discover all devices available on the local network. It can also be sent to a single device to get information if its UDCP address is known.

This request does not contain any fields.

A device should always answer with a “Get Information Response” upon reception of this request.

## Get Information Response

This response is sent from a device to a PC upon reception of a “Get Information Request”. It reports all known information about the device.

This response must contain the following fields:

* Unit Name (1)
* Software Version (2)
* IP Address (3)
* Network Mask (4)

It may also contain the following fields, if their value is known:

* Gateway (5)
* DHCP Enabled (6)
* Any error fields (100, 101, 102)

## Set Configuration Request

This request is sent from a PC to a device to temporarily reconfigure it. This allows to change the IP settings of a device if they don’t match the local network. The changed IP settings should be discarded upon reboot of the device, they should never be written to a persistent storage.

If the device is to be configured using DHCP, it must only contain the following field:

* DHCP Enabled (6) set to “1”

If the device is to be configured with static IP settings, it must contain the following fields:

* IP Address (3)
* Network Mask (4)

In this case it may also contain the following fields:

* Gateway (5) if it is to be reconfigured
* DHCP Enabled (6) set to “0”

A device should always answer with a “Set Configuration Response” upon reception of this request.

## Set Configuration Response

This response is sent from a device to a PC upon reception of a “Set Configuration Request”. It is used to report the success or failure of said request.

This response may contain the following fields:

* Error Code (100)
* Error Field (101)
* Error Message (102)

## Announce Request

This request is sent from a PC to a device to make it “announce” itself. This functionality can be used to distinguish multiple devices. A user can request one device to announce itself when he has multiple devices in front of him and does not know which device is which.

An announcement can be different depending on the kind of device, possible examples are:

* The display showing a special message (as it is the case on the TFT)
* A LED blinking
* A sound being played by the device

This request does not contain any fields.

A device should always answer with an “Announce Response” upon reception of this request.

## Announce Response

This response is sent from a device to a PC upon reception of an “Announce Request”. It is used to report the success or failure of said request.

This response may contain the following fields:

* Error Code (100)
* Error Field (101)
* Error Message (102)

## Reboot Request

This request is sent from a PC to a device to force it to reboot. This can be used if the device is in an unknown state and is in many cases better (and easier) than just powering the device off and on again.

This request does not contain any fields.

A device should always answer with a “Reboot Response” upon reception of this request before rebooting.

## Reboot Response

This response is sent from a device to a PC upon reception of an “Reboot Request”. It is used to report the success or failure of said request.

This response may contain the following fields:

* Error Code (100)
* Error Field (101)
* Error Message (102)