# OpenDCRE Documentation

***Release 1.3.0***

**Vapor IO**

**Mar 24, 2017**

# Contents

Vapor IO Software and Hardware documentation, reference, and downloads. OpenDCRE is licensed under GPL-2.0 and can be found on GitHub.

# CHAPTER 1

# OpenDCRE

## Introduction

OpenDCRE provides a securable RESTful API for monitoring and out-of-band management of data center and IT equipment. It can be configured to use power line communications (PLC) over a DC bus bar, IPMI over LAN, or Redfish over LAN. The OpenDCRE API is easy to integrate into third-party monitoring, management, and orchestration providers while providing a simple `curl`-able interface for common and custom devops tasks.

**Note:** Redfish support in OpenDCRE (v1.3.0) is still under development and testing, so it should be treated as a beta feature.

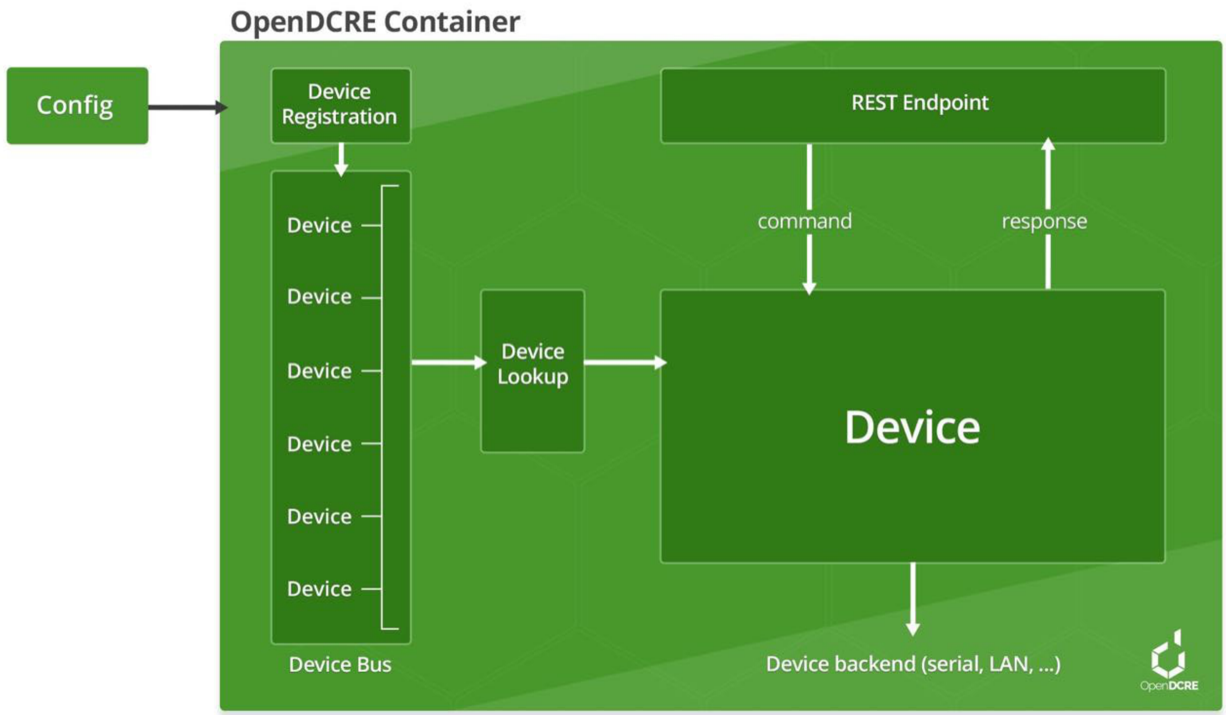
### Features

- Simple `curl`-able RESTful API
- Analog and digital sensor support (temperature, thermistor, humidity, fan speed, pressure).
- Power control and status, including power consumption and power supply status.
- Asset information for servers.
- Physical and chassis location awareness.
- Fan speed control and status.
- Chassis "identify" LED control and status.
- System boot target selection (hdd, pxe).
- Securable via TLS/SSL.
- Integration with existing Auth providers (OAuth, LDAP, AD, etc.).
- Power line communications (PLC) - all OpenDCRE commands can use PLC over a DC bus bar as transport layer.

- IPMI Bridge - all OpenDCRE commands can use IPMI 2.0 over LAN as transport layer.
- Redfish support (beta) - all OpenDCRE commands can use Redfish over LAN as transport layer.

## Architecture

OpenDCRE is a Dockerized service designed to run in a microservice architecture. It exposes a RESTful API via an HTTP endpoint in the OpenDCRE container. The HTTP endpoint is comprised of Nginx as the front-end with uwsgi as a reverse proxy for a Python Flask application. Within the Flask application, OpenDCRE a modular "device bus" definitions to define their own protocol-specific backends. The OpenDCRE endpoint routes and dispatches incoming commands to the appropriate device bus for handling.




The OpenDCRE device bus is comprised of a set of boards and devices, individually addressable, and globally scannable for a real-time inventory of addressable devices. The OpenDCRE device bus allows devices to be read and written, and for various actions to be carried out, such as power control (on/off/cycle/status). Additionally, when a physical OpenDCRE device bus is not present, a software emulator can be used to simulate OpenDCRE API commands and functionality.

All included components of OpenDCRE can be customized, integrated and secured via configuration file (nginx, uwsgi), and output their logs to a common location (/logs).

## Applications

OpenDCRE can be used as an open platform for monitoring and managing data center hardware, software and environmental characteristics. Since OpenDCRE is Dockerized, there are a wide variety of options for deployments, integrations, network connectivity, etc. Community support helps OpenDCRE grow, and enables new functionality.

## Requirements

- Docker **(preferably version 1.12 or later)**
    - used to containerize OpenDCRE, LAN based emulators, and tests.
- docker compose **(preferably version 1.10 or later)**
    - used for orchestrating and running OpenDCRE tests.
    - can be used to run OpenDCRE and LAN based emulators.
- make
    - used as the primary means for building and running OpenDCRE, emulator, and test images.

## API Reference

The examples below assume OpenDCRE is running on a given <*ipaddress*> and <*port*>. The default port for OpenDCRE is TCP port 5000. Currently, all commands are GET requests; a future version will expose these commands via POST as well.

---

**Note:** In the API reference information below, there are many references to `board_id` and `device_id` parameters. For IPMI Devices, the values in a board's `hostnames` and `ip_addresses` fields (e.g. from a *scan*) can be used in place of the `board_id`. Additionally, the `device_info` field for a given device, where specified, can be used in place of its `device_id`.

As an example, a device *read* for a system temperature device on a board whose (abridged) scan information provides us with:

```
{
  "board_id": "40000039",
  "devices": [
    {
      "device_id": "0011",
      "device_info": "System Temp",
      "device_type": "temperature"
    }
  ],
  "hostnames": [
    "kafka001.vapor.io"
  ],
  "ip_addresses": [
    "192.168.1.10"
  ]
}
```

We could formulate a temperature read call in a variety of ways:

```
GET /opendcre/1.3/read/temperature/rack_9/40000039/0011
GET /opendcre/1.3/read/temperature/rack_9/40000039/System%20Temp
GET /opendcre/1.3/read/temperature/rack_9/kafka001.vapor.io/0011
GET /opendcre/1.3/read/temperature/rack_9/kafka001.vapor.io/System%20Temp
GET /opendcre/1.3/read/temperature/rack_9/192.168.1.10/0011
GET /opendcre/1.3/read/temperature/rack_9/192.168.1.10/System%20Temp
```

As of version 1.3.0, this only works for IPMI Devices, but this functionality will come later for all other devices.

## asset information

Get asset information for a given device. The device's `device_type` must be of type `system` (as reported by the *scan* command). Asset information can consist of board information, chassis information, and product information.

### Request

### Format

```
GET /opendcre/<version>/asset/<rack_id>/<board_id>/<device_id>
```

### Parameters

**rack_id** The id of the rack on which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to read asset information for on the specified board. Hexadecimal string representation of a 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is of type `system` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

### Example

```
http://opendcre:5000/opendcre/1.3/asset/00000001/0004
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-asset-information",
  "title": "OpenDCRE Asset Information",
  "type": "object",
  "properties": {
    "bmc_ip": {
      "type": "string"
    },
    "board_info": {
```

```
        "type": "object",
        "properties": {
          "manufacturer": {
            "type": "string"
          },
          "part_number": {
            "type": "string"
          },
          "product_name": {
            "type": "string"
          },
          "serial_number": {
            "type": "string"
          }
        }
      },
      "chassis_info": {
        "type": "object",
        "properties": {
          "chassis_type": {
            "type": "string"
          },
          "part_number": {
            "type": "string"
          },
          "serial_number": {
            "type": "string"
          }
        }
      },
      "product_info": {
        "type": "object",
        "properties": {
          "asset_tag": {
            "type": "string"
          },
          "manufacturer": {
            "type": "string"
          },
          "part_number": {
            "type": "string"
          },
          "product_name": {
            "type": "string"
          },
          "serial_number": {
            "type": "string"
          },
          "version": {
            "type": "string"
          }
        }
      }
    }
  }
}
```

**Note:** Note that the `bmc_ip` field is only present for IPMI device asset info.

#### Example

```
{
  "bmc_ip": "192.168.1.118",
  "board_info": {
    "manufacturer": "Quanta",
    "part_number": "0001",
    "product_name": "Winterfell",
    "serial_number": "S1234567"
  },
  "chassis_info": {
    "chassis_type": "rack mount chassis",
    "part_number": "P1234567",
    "serial_number": "S1234567"
  },
  "product_info": {
    "asset_tag": "A1234567",
    "manufacturer": "Quanta",
    "part_number": "P1234567",
    "product_name": "Winterfell",
    "serial_number": "S1234567",
    "version": "v1.2.0"
  }
}
```

#### Errors

**500**

- asset info is unavailable or does not exist
- specified device is not of type `system`
- invalid/nonexistent `board_id` or `device_id`

---

### boot target

The boot target command may be used to get or set the boot target for a given device (whose device_type must be `system`). The boot_target command takes two required parameters - `board_id` and `device_id`, to identify the device to direct the boot_target command to. Additionally, a third, optional parameter, `target` may be used to set the boot target.

#### Request

#### Format

```
GET /opendcre/<version>/boot_target/<rack_id>/<board_id>/<device_id>[/<target>]
```

#### Parameters

**rack_id** The id of the rack upon which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to issue boot target command to on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is `system` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

**target** *(optional)* Valid target for the boot_target command include:

- `hdd` : boot to hard disk
- `pxe` : boot to network
- `no_override` : use the system default boot target

If a target is not specified, boot_target makes no changes, and simply retrieves and returns the system boot target. If `target` is specified and valid, the boot_target command will return the updated boot target value, as provided by the remote device.

#### Example

```
http://opendcre:5000/opendcre/1.3/boot_target/00000001/0004
```

#### Response

#### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-boot-target",
  "title": "OpenDCRE Boot Target",
  "type": "object",
  "properties": {
    "target": {
      "type": "string",
      "enum": [
        "no_override",
        "hdd",
        "pxe"
      ]
    }
```

```
  }
}
```

### Example

```
{
  "target": "no_override"
}
```

### Errors

**500**

- boot target action fails
- specified device is not of type `system`
- invalid/nonexistent `board_id` or `device_id`

---

## fan

The fan control command is used to get the fan speed (in RPM) for a given fan.

### Request

### Format

```
GET /opendcre/<version>/fan/<rack_id>/<board_id>/<device_id>[/<speed_rpm>]
```

### Parameters

**rack_id** The id of the rack upon which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to issue fan control command to on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is `fan_speed` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

**speed_rpm** *(optional)* Numeric decimal value to set fan speed to, in range of 0-10000.

> **Note:** IPMI devices do not yet support the setting of fan speed, so this parameter can be ignored for all IPMI setups.

#### Example

```
http://opendcre:5000/opendcre/1.3/fan/00000001/0002
```

#### Response

##### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-fan-speed",
  "title": "OpenDCRE Fan Speed",
  "type": "object",
  "properties": {
    "health": {
      "type": "string"
    },
    "states": {
      "type": "array"
    },
    "speed_rpm": {
      "type": "number"
    }
  }
}
```

##### Example

```
{
  "health": "ok",
  "speed_rpm": 4100.0,
  "states": []
}
```

#### Errors

**500**

- fan speed action fails
- specified device is not a fan device
- invalid/nonexistent `board_id` or `device_id`

## host information

Get the hostname(s) and IP address(es) for a given host. The device's `device_type` should be of type `system` (as reported by the *scan* command).

### Request

#### Format

```
GET /opendcre/<version>/host_info/<rack_id>/<board_id>/<device_id>
```

#### Parameters

**rack_id** The id of the rack upon which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to issue host info control command to on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is `system` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

#### Example

```
http://opendcre:5000/opendcre/1.3/host_info/00000001/0001
```

### Response

#### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-host-information",
  "title": "OpenDCRE Host Information",
  "type": "object",
  "properties": {
    "hostnames": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "ip_addresses": {
      "type": "array",
      "items": {
        "type": "string"
```

```
      }
    }
  }
}
```

### Example

```
{
  "hostnames": [
    "cassandra000"
  ],
  "ip_addresses": [
    "10.10.1.16"
  ]
}
```

### Errors

**500**

- host info action fails
- specified device is not of type `system`
- invalid/nonexistent `board_id` or `device_id`

---

## LED

The LED control command is used to get and set the chassis "identify" LED state. `led` devices known to OpenDCRE allow LED state to be set and retrieved.

### Request

### Format

```
GET /opendcre/<version>/led/<rack_id>/<board_id>/<device_id>[/<led_state>]
```

### Parameters

**rack_id** The id of the rack upon which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to issue LED control command to on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is `led` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

**led_state** *(optional)* The LED state to set. Valid values include:

- `on` : Turn on the chassis identify LED
- `off` : Turn off the chassis identify LED

### Example

```
http://opendcre:5000/opendcre/1.3/led/00000001/0005
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-led-control",
  "title": "OpenDCRE LED Control",
  "type": "object",
  "properties": {
    "led_state": {
      "type": "string",
      "enum": [
        "on",
        "off"
      ]
    }
  }
}
```

### Example

```
{
  "led_state": "on"
}
```

### Errors

**500**

- LED control action fails
- specified device is not of type `led`
- invalid `board_id` or `device_id`

## location

The location command returns the physical location of a given board in the rack, if known, and may also include a given device's position within a chassis (when the `device_id` parameter is specified). IPMI boards return `unknown` for all fields of `physical_location` as location information is not provided by IPMI.

### Request

#### Format

```
GET /opendcre/<version>/location/<rack_id>/<board_id>[/<device_id>]
```

#### Parameters

**rack_id** The id of the rack upon which the specified board and device reside.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** *(optional)* The device to get location for on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device known to OpenDCRE - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

### Response

#### Schema

#### Device Location

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-device-
↪location",
  "title": "OpenDCRE Device Location",
  "type": "object",
  "properties": {
    "chassis_location": {
      "type": "object",
      "properties": {
        "depth": {
          "type": "string",
          "enum": [
            "unknown",
            "front",
            "middle",
            "rear"
          ]
        },
        "horiz_pos": {
```

```
          "type": "string",
          "enum": [
            "unknown",
            "left",
            "middle",
            "right"
          ]
        },
        "vert_pos": {
          "type": "string",
          "enum": [
            "unknown",
            "top",
            "middle",
            "bottom"
          ]
        },
        "server_node": {
          "type": "string"
        }
      }
    },
    "physical_location": {
      "type": "object",
      "properties": {
        "depth": {
          "type": "string",
          "enum": [
            "unknown",
            "front",
            "middle",
            "rear"
          ]
        },
        "horizontal": {
          "type": "string",
          "enum": [
            "unknown",
            "left",
            "middle",
            "right"
          ]
        },
        "vertical": {
          "type": "string",
          "enum": [
            "unknown",
            "top",
            "middle",
            "bottom"
          ]
        }
      }
    }
  }
}
```

### Board Location

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-board-
↪location",
  "title": "OpenDCRE BoardLocation",
  "type": "object",
  "properties": {
    "physical_location": {
      "type": "object",
      "properties": {
        "depth": {
          "type": "string",
          "enum": [
            "unknown",
            "front",
            "middle",
            "rear"
          ]
        },
        "horizontal": {
          "type": "string",
          "enum": [
            "unknown",
            "left",
            "middle",
            "right"
          ]
        },
        "vertical": {
          "type": "string",
          "enum": [
            "unknown",
            "top",
            "middle",
            "bottom"
          ]
        }
      }
    }
  }
}
```

### Example

### Device Location

```
{
  "chassis_location": {
    "depth": "unknown",
    "horiz_pos": "unknown",
    "server_node": "unknown",
    "vert_pos": "unknown"
  },
  "physical_location": {
```

```
    "depth": "unknown",
    "horizontal": "unknown",
    "vertical": "unknown"
  }
}
```

### Board Location

```
{
  "physical_location": {
    "depth": "unknown",
    "horizontal": "unknown",
    "vertical": "unknown"
  }
}
```

### Errors

**500**

- location command fails
- invalid/nonexistent `board_id` or `device_id`

---

## power

Control device power, and/or retrieve its power supply status. The specified device's `device_type` must be of type `power` (as reported by the *scan* command).

### Request

### Format

```
GET /opendcre/<version>/power/<rack_id>/<board_id>/<device_id>[/<command>]
```

### Parameters

**rack_id** The id of the rack which the specified board and device reside on.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN, where NNNNNN corresponds to the hex string id of each configured BMC. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to issue power command to on the specified board. Hexadecimal string representation of 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE is `power` - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

**command** *(optional)* The power command to issue. Valid commands are:

- `on` : Turn power on to specified device
- `off` : Turn power off to specified device
- `cycle` : Power-cycle the specified device
- `status` : Get power status for the specified device

For all commands, power status is returned as the command's response.

### Example

```
http://opendcre:5000/opendcre/1.3/power/00000001/000d/on
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-power-status",
  "title": "OpenDCRE Power Status",
  "type": "object",
  "properties": {
    "input_power": {
      "type": "number"
    },
    "input_voltage": {
      "type": "number"
    },
    "output_current": {
      "type": "number"
    },
    "over_current": {
      "type": "boolean"
    },
    "pmbus_raw": {
      "type": "string"
    },
    "power_ok": {
      "type": "boolean"
    },
    "power_status": {
      "type": "string",
      "enum": [
        "on",
        "off"
      ]
    },
```

```
    "under_voltage": {
      "type": "boolean"
    }
  }
}
```

#### Example



```
{
  "input_power": 198.57686579513486,
  "input_voltage": 12.500651075576853,
  "output_current": 15.879801734820322,
  "over_current": false,
  "pmbus_raw": "0,12000,2400,3356",
  "power_ok": true,
  "power_status": "on",
  "under_voltage": false
}
```



#### Errors

**500**

- power action fails
- the specified device is not of type `power`
- invalid/nonexistent `board_id` or `device_id`

---

### read

Read a value from the given `board_id` and `device_id` for a specific `device_type`. The specified `device_type` must match the actual physical device type (as reported by the *scan* command), and is used to return a translated raw reading value (e.g. temperature in C for a thermistor) based on the existing algorithm for a given sensor type.

#### Request

#### Format

```
GET /opendcre/<version>/read/<device_type>/<rack_id>/<board_id>/<device_id>
```

#### Parameters

**device_type** String value (lower-case) indicating what type of device to read: `thermistor`, `temperature`, `humidity`, `led`, `fan_speed`, `pressure`, `voltage`

**rack_id** The id of the rack which the board and device reside on.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40NNNNNN (where NNNNNN is the hex string id of each individual BMC configured with the IPMI Bridge). For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

**device_id** The device to read on the specified board. Hexadecimal string representation of a 2-byte integer value - range 0000..FFFF. Must be a valid, existing device, where the `device_type` known to OpenDCRE matches the `device_type` specified in the command for the given device - else, a 500 error is returned. For IPMI, the `device_id` can also be the value of the `device_info` field associated with the given device, if present.

### Example

```
http://opendcre:5000/opendcre/1.3/read/thermistor/00000001/0001
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-sensor-reading",
  "title": "OpenDCRE Sensor Reading",
  "type": "object",
  "oneOf": [
    {
      "description": "Temperature Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "temperature_c": {
          "type": "number"
        }
      }
    },
    {
      "description": "Thermistor Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "temperature_c": {
          "type": "number"
        }
      }
    },
    {
```

```
      "description": "Fan Speed Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "speed_rpm": {
          "type": "number"
        }
      }
    },
    {
      "description": "LED Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "led_state": {
          "type": "string",
          "enum": [
            "on",
            "off"
          ]
        }
      }
    },
    {
      "description": "Pressure Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "pressure_kpa": {
          "type": "number"
        }
      }
    },
    {
      "description": "Voltage Readings",
      "properties": {
        "health": {
          "type": "string"
        },
        "states": {
          "type": "array"
        },
        "voltage": {
          "type": "number"
        }
      }
```

```
    }
  ]
}
```

### Example

```
{
  "health": "ok",
  "states": [],
  "temperature_c": 19.73
}
```

### Errors

**500**

- the device is not readable or does not exist
- specified device is not of the specified device type
- invalid/nonexistent `board_id` or `device_id`

---

## scan

The `scan` command polls boards and devices attached to the board. There are primarily two ways that scan operates – "scan all", and "scan entity", where an entity could be a rack or a board. When performing a "scan all" command, if a scan cache does not exist, it will scan the physical boards and devices. If a scan cache does exist, it will use the results from the cache. To refresh the scan cache, the "force scan" command should be used.

**Note:** It is likely a good idea for applications to scan for all boards on startup, to ensure a proper map of boards and devices is available to the application. Mismatches of board and device types and identifiers will result in 500 errors being returned for various commands that rely on these values mapping to actual hardware.

### Request

### Format

**scan all**

```
GET /opendcre/<version>/scan
```

**force scan**

```
GET /opendcre/<version>/scan/force
```

**scan rack**

```
GET /opendcre/<version>/<rack_id>
```

**scan board**

```
GET /opendcre/<version>/<rack_id></board_id>
```

### Parameters

**force** *(optional)* A flag which, when present, will force the re-scan of all racks, boards, and devices. This parameter is only associated with "scan all" behavior and does not apply to scans at a rack or board level. Forcing a scan re-scans the devices and updates the scan cache.

**rack_id** *(optional)* The id of the rack to scan, if only the rack is specified. If the rack is specified with a board id, this is the rack where the target board resides.

**board_id** *(optional)* Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge boards have a special `board_id` of 40NNNNNN (where NNNNNN is the hex string id of each configured BMC). For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

### Example

```
GET http://opendcre:5000/opendcre/1.3/scan
GET http://opendcre:5000/opendcre/1.3/scan/force
GET http://opendcre:5000/opendcre/1.3/scan/rack_1
GET http://opendcre:5000/opendcre/1.3/scan/rack_1/00000001
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-boards-devices",
  "title": "OpenDCRE Boards and Devices",
  "type": "object",
  "properties": {
    "racks": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "rack_id": {
            "type": "string"
          },
          "boards": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "board_id": {
                  "type": "string"
```

```
                },
                "devices": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "properties": {
                      "device_id": {
                        "type": "string"
                      },
                      "device_type": {
                        "type": "string",
                        "enum": [
                          "temperature",
                          "thermistor",
                          "humidity",
                          "led",
                          "system",
                          "power",
                          "fan_speed",
                          "pressure",
                          "voltage",
                          "power_supply"
                        ]
                      }
                    }
                  }
                },
                "hostnames": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  }
                },
                "ip_addresses": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  }
                },
              }
            }
          }
        }
      }
    }
  }
}
```

### Example

```
{
  "racks": [
    {
      "boards": [
        {
          "board_id": "00000001",
```

```
          "devices": [
            {
              "device_id": "0001",
              "device_type": "system"
            },
            {
              "device_id": "0002",
              "device_type": "fan_speed"
            },
            {
              "device_id": "0003",
              "device_type": "fan_speed"
            },
            {
              "device_id": "0004",
              "device_type": "power"
            },
            {
              "device_id": "0005",
              "device_type": "led"
            },
            {
              "device_id": "2000",
              "device_type": "temperature"
            },
            {
              "device_id": "4000",
              "device_type": "temperature"
            }
          ],
          "hostnames": [
            "kafka001.vapor.io"
          ],
          "ip_addresses": [
            "192.168.1.10"
          ]
        }
      ],
      "rack_id": "rack_1"
    }
  ]
}
```


### Errors

**500**

- the scan command fails
- invalid/nonexistent `board_id`

## service version

The OpenDCRE service version command identifies the version of OpenDCRE running on the OpenDCRE instance being queried. Since OpenDCRE's REST API commands include the API version in the URI, this endpoint provides a means of getting that version if it is otherwise unknown.

### Request

### Format

```
GET /opendcre/version
```

### Response

### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-service-version",
  "title": "OpenDCRE Service Version",
  "type": "object",
  "properties": {
    "version": {
      "type": "string"
    }
  }
}
```

### Example

```
{
  "version": "1.3"
}
```

### Errors

**500**

- the endpoint is not running

---

## test

The test command provides a way to verify that the OpenDCRE service is running. It has no dependencies on any of the configured devicebus interfaces, so it serves only to test that the service is up and reachable - not that it is configured correctly. The command takes no arguments and, if successful, returns a status message of "ok".

#### Request

##### Format

```
GET  /opendcre/<version>/test
POST /opendcre/<version>/test
```

#### Response

##### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-test-status",
  "title": "OpenDCRE Test Status",
  "type": "object",
  "properties": {
    "status": {
      "type": "string"
    }
  }
}
```

##### Example

```
{
  "status": "ok"
}
```

##### Errors

**500**

- the endpoint is not running

---

### version

Get version information about a board.

#### Request

##### Format

```
GET /opendcre/<version>/version/<rack_id>/<board_id>
```

#### Parameters

**rack_id** The rack id associated with the specified board.

**board_id** Hexadecimal string representation of 4-byte integer value - range 00000000..FFFFFFFF. Upper byte of `board_id` reserved for future use in OpenDCRE. IPMI Bridge board has a special `board_id` of 40000000. For IPMI, the `board_id` can also be a hostname/ip_address that is associated with the given board.

#### Example

```
http://opendcre:5000/opendcre/1.3/version/00000001
```

#### Response

#### Schema

```
{
  "$schema": "http://schemas.vapor.io/opendcre/v1.3/opendcre-1.3-version",
  "title": "OpenDCRE Board Version",
  "type": "object",
  "properties": {
    "api_version": {
      "type": "string"
    },
    "firmware_version": {
      "type": "string"
    },
    "opendcre_version": {
      "type": "string"
    }
  }
}
```

#### Example

```
{
  "api_version": "1.3",
  "firmware_version": "OpenDCRE Emulator v1.3.0",
  "opendcre_version": "1.3.0"
}
```

#### Errors

**500**

- version retrieval does not work
- `board_id` specifies a nonexistent board

# Getting OpenDCRE

OpenDCRE's source can be found on GitHub, where the OpenDCRE Docker image will need to be built. Alternatively, a pre-built OpenDCRE image can be downloaded from DockerHub.

## Building from Source

Once the OpenDCRE source code is downloaded (either via git clone, or downloaded zip), a Docker image can be built. No additional changes are required to the source for a complete, functioning image, but customizations can be included in the image, e.g. the inclusion of site-specific TLS certificates, nginx configurations for authn/authz, etc.

The included dockerfile can be used to package up the distribution:

```
docker build -t opendcre:custom-v1.3.0 -f dockerfile/Dockerfile.x64 .
```

A Makefile recipe also exists to build the OpenDCRE image and tag it as `vaporio/opendcre-<arch>:1.3`, where *<arch>* specifies the architecture (e.g., x64). For the *x64* architecture, this recipe is:

```
make x64
```

If building a custom image, apply whatever tag is most descriptive for that image.

At this point, OpenDCRE can be tested (see *Testing*) and run (see *Running OpenDCRE*) to ensure the build was successful.

## Downloading from DockerHub

If no changes are needed to the source, the pre-packed version can be used. It can be downloaded from DockerHub simply with

```
docker pull vaporio/opendcre:1.3.0
```

## Updating

Updating OpenDCRE is as simple as building a new image from source, or pulling a new image down from Docker-Hub.

# Configuring OpenDCRE

OpenDCRE may be customized in a variety of ways, most commonly through the OpenDCRE configuration file, but also through supporting configurations (e.g. Nginx configurations).

## Configuration Options

OpenDCRE configurations are made up of two files – default configurations, and override configurations. As the name suggests, the default configurations are what come pre-built into OpenDCRE and what OpenDCRE will fall back to if no override is specified. The default configurations can be found at `/opendcre/default/default.json` in the OpenDCRE container.

```
{
  "scan_cache_file": "/tmp/opendcre/cache.json",
  "cache_timeout": 600,
  "cache_threshold": 500,

  "devices": {
    "ipmi": {
      "from_config": "bmc_config.json"
    }
  }
}
```

The override configurations are user-specified at run time. These configurations can be mounted into the container as a volume at `/opendcre/override/config.json`. The configuration override file can be any JSON file with "config" in the filename. Of course, configurations do not need to be volume-mounted into the container, though it is convenient in various situations. In cases where it is easier to just replace the default configuration altogether, do so and rebuild the OpenDCRE docker image (see *Building from Source*).

As an example, if we wanted to override the *cache_timeout* configuration, we could mount in `config.json` which contains:

```
{
  "cache_timeout": 1000
}
```

In this case, the default values will be used for everything but *cache_timeout*.

Below are descriptions for each of the supported configuration file fields.

**scan_cache_file** The path and filename of the file used to cache OpenDCRE data, such as board records used by the "scan" command. The default value of "/tmp/opendcre/cache.json" typically is suitable and does not need to be changed.

**cache_timeout** The scan cache's time-to-live, in seconds, after which cache records will be invalidated.

**cache_threshold** The maximum number of entries to store in the scan cache.

**devices** The devices parameter is used to describe the various bus types and devices available to OpenDCRE. It accepts keys of "plc" (*PLC Device*), "ipmi" (*IPMI Device*), and "redfish" (*Redfish Device*), though none are required. See the linked sections for each devicebus type for examples of the configuration for each.

Continuing with the example above, we can see how actual devices are configured within the referenced IPMI config file, `bmc_config.json`:

```
{
  "racks": [
    {
      "rack_id": "rack_1",
      "bmcs": [
        {
          "bmc_ip": "192.168.1.110",
          "username": "ADMIN",
          "password": "ADMIN"
        },
        {
          "bmc_ip": "192.168.1.111",
          "bmc_port": 622,
          "username": "ADMIN",
```

```
          "password": "ADMIN",
          "hostnames": ["atom"],
          "ip_addresses": ["192.169.1.111"]
        }
      ]
    }
  ]
}
```

Here, we are configuring two BMCs, both on a single rack – "rack_1". The first BMC is at IP 192.168.1.110 with username ADMIN and password ADMIN. No port is specified, so it uses the default port of 623. The second BMC is at IP 192.168.1.111 with username ADMIN and password ADMIN. It has a non-standard port specified which will be used to communicate with that BMC.

See the *IPMI Device* section for greater detail on these configuration options.

### Port

By default, OpenDCRE listens on port 5000. To change the port OpenDCRE listens on, edit the `opendcre_nginx.conf` file, and the port exposed in the Dockerfile, then rebuild the OpenDCRE docker image (see *Building from Source*).

```
server {
    listen 5000;
    server_name localhost;
    charset utf-8;
    access_log /logs/opendcre.net_access.log;
    error_log /logs/opendcre.net_error.log;

    location / {
        add_header 'Access-Control-Allow-Origin' '*';
        uwsgi_pass unix://var/uwsgi/opendcre.sock;
        include /etc/nginx/uwsgi_params;
    }
}
```

### TLS/SSL

TLS/SSL certificates may be added to OpenDCRE via Nginx configuration. Refer to the Nginx documentation for instructions on how to enable TLS.

### Authentication

As OpenDCRE uses Nginx as its reverse proxy, authentication may be enabled via Nginx configuration – see the Nginx documentation for instructions on how to enable authentication.

## Running OpenDCRE

OpenDCRE is easy to start up - it just requires that the container be run. In the simplest case, OpenDCRE can be started with

```
docker run -p 5000:5000 vaporio/opendcre
```

But this wont do very much, since it will have no devices configured to issue commands to. To get OpenDCRE working with real or even emulated hardware, there is a bit of configuration that needs to be done. Below, there is a "Quick Start" approach to getting start with using OpenDCRE. After, the "Detailed Start" covers a bit more detail all the ways in which OpenDCRE can be configured for running.

## Quick Start

For the quick start, we will assume that we will be using an emulator, though the principles here should naturally and clearly extend to real hardware as well. For simplicity, This will use a single emulator (for IPMI), but nothing prevents you from configuring it with multiple emulators/devices as you see fit.

See the *Emulators* section for more complete documentation on how to configure and run the emulators.

For this example, lets say our IPMI emulator is running on 192.168.1.10, with username 'admin' and password 'admin'.

A configuration file should be created which specifies this emulator, which we will save locally as `bmc_config.json`:

```json
{
  "racks": {
    "rack_id": "rack_1",
    "bmcs": [
      {
        "bmc_ip": "192.168.1.10",
        "username": "admin",
        "password": "admin"
      }
    ]
  }
}
```

Next, we just need to run OpenDCRE and mount in the IPMI configuration file to the appropriate location.

```
docker run -p 5000:5000 -v `pwd`/bmc_config.json:/opendcre/bmc_config.json vaporio/
↪opendcre:1.3.0
```

This will start OpenDCRE and reach out to 192.168.1.10 to register the IPMI Device and scan that BMC. To use your own BMCs, you would simply provide the appropriate IP, username, and password for each BMC configured.

### Volume Mount Locations

Above, we say "mount the configuration to the appropriate location", but what are the appropriate locations?

| Devicebus Type | Container Configuration File |
|---|---|
| plc | /opendcre/plc_config.json |
| ipmi | /opendcre/bmc_config.json |
| redfish | /opendcre/redfish_config.json |

## Detailed Start

OpenDCRE devices are all set up by configuration. These configurations can either be built directly into the image (if building a custom OpenDCRE image) or mounted into the container, which is the preferred approach as it is more flexible.

When OpenDCRE starts, it reads in is configuration files to determine which devices are configured. By default, OpenDCRE points to empty configurations for PLC, IPMI, and Redfish

```
{
  "devices": {
    "plc": {
      "from_config": "plc_config.json"
    },
    "ipmi": {
      "from_config": "bmc_config.json"
    },
    "redfish": {
      "from_config": "redfish_config.json"
    }
  }
}
```

Where each of those files specifies a single rack with no devices on it, e.g. for IPMI

```
{
  "racks": [
    {
      "rack_id": "rack_1",
      "bmcs": []
    }
  ]
}
```

So, when OpenDCRE starts up with no additional configurations provided, no devices will be registered with it, so it really won't be able to perform any actions.

In the Quick Start example, we overwrite the existing "blank" IPMI BMC configuration file with one that has an actual configuration in it (via the volume mount). With that, OpenDCRE will see that there is a device specified, and will attempt to register it so that it can be used to issue commands to.

This same pattern applies to the other devicebus types, so if you want to configure OpenDCRE to work with a PLC device and a Redfish device, you need only create the appropriate configuration files for them and volume-mount them to the OpenDCRE container on startup.

It helps to familiarize yourself with the *Configuring OpenDCRE* section as well as the configurations for the *PLC Device*, *IPMI Device*, and *Redfish Device* to know what configurations are required.

Below is an example (dummy) OpenDCRE run command followed by an explanation of what each part does.

```
docker run -d \
    -p 5000:5000 \
    -e VAPOR_DEBUG=true \
    -v `pwd`/plc_config.json:/opendcre/plc_config.json \
    -v `pwd`/ipmi_config.json:/opendcre/bmc_config.json \
    -v `pwd`/config_override.json:/opendcre/override/config.json \
    vaporio/opendcre \
    ./start_opendcre.sh
```

**`-d`** the `-d` flag is used to run OpenDCRE in "detached" mode - this means Docker will not attach to the console, so OpenDCRE will run in the background.

**`-p 5000:5000`** this maps the host's port 5000 to the OpenDCRE container's port 5000 - with this, you can use the OpenDCRE REST API on port 5000 of the host.

**-e VAPOR_DEBUG=true** this sets the `VAPOR_DEBUG` envirnment variable to `true`, enabling debug logging. For more on this, see the *Debugging* section.

**-v `pwd`/plc_config.json:/opendcre/plc_config.json** this mounts in the "plc_config.json" file from the host to the "/opendcre/plc_config.json" location in the container. this will override the default (empty) PLC configurations.

**-v `pwd`/ipmi_config.json:/opendcre/bmc_config.json** this mounts in the "ipmi_config.json" file from the host to the "/opendcre/bmc_config.json" location in the container. this will override the default (empty) IPMI configurations.

**-v `pwd`/config_override.json:/opendcre/override/config.json** this mounts in the "config_override.json" file from the host to the "/opendcre/override/config.json" location in the container. this is used to override defaualt OpenDCRE configurations (including but not limited to device configurations). See the *Configuring OpenDCRE* section for more on this.

**vaporio/opendcre** this is the image to run – in this case the OpenDCRE image hosted on the Vapor IO Docker-Hub.

**./start_opendcre.sh** the command to run in the container. this particular command is superfluous as it is the default command that is run by OpenDCRE, but was included here for completeness of the example.

## DeviceBus Interfaces

At the heart of OpenDCRE is the notion of "device bus" (or "devicebus"), which is so-named due to the original single-purpose PLC bus, and can now be said to describe "devices and busses" supported by OpenDCRE. Currently, there are three devicebus interfaces supported by OpenDCRE: PLC, IPMI, and Redfish (beta).

### PLC Device

New in version 1.0.0.

PLC (Power Line Communications) support was the first (and only) mode available in early (pre v1.0) versions of OpenDCRE. The PLC Device provides support for serial-based communication between OpenDCRE and the Vapor Chamber.

#### Supported Commands

The *test* and *service version* are supported by all devicebus types. Below are the commands which are supported for PLC Devices. See the *API Reference* for details on all commands.

| Command | Supported |
|---|---|
| Version | True |
| Scan | True |
| Scan All | True |
| Read | True |
| Power | True |
| Asset | True |
| Boot Target | True |
| Chamber LED | True |
| LED | True |
| Fan | True |
| Host Info | True |

### Configuration

PLC Device configurations are specified under the `devices` field of the OpenDCRE *Configuration Options*. A simple example configuration is given below, followed by descriptions of the fields presented in the example.

```
{
  "devices": {
    "plc": {
      "from_config": "plc_config.json"
    }
  }
}
```

**from_config** The file which specifies the rack and device configurations for devices managed through OpenDCRE. See below for an example of these configurations.

**config** An alternative to `from_config` - this field allows one to specify the rack and device configurations in this configuration file as opposed to a separate one. Generally, it is recommended to use `from_config` over this, as it keeps things cleaner, but if only a few devices are being specified, it may be easier to define their configurations under this field.

As mentioned above, the `from_config` and `config` fields specify the device-specific configurations. The JSON example below could either be specified under the `config` field, or in the file specified by the `from_config` field.

```
{
  "racks": [
    {
      "rack_id": "rack_1",
      "lockfile": "/tmp/OpenDCRE.lock",
      "hardware_type": "emulator",
      "devices": [
        {
          "device_name": "/dev/ttyAMA0",
          "retry_limit": 3,
          "timeout": 0.25,
          "time_slice": 75,
          "bps": 115200
        }
      ]
    }
  ]
}
```

**racks** OpenDCRE is capable of managing multiple racks' worth of BMCs, so the top-level configuration parameter "racks" consists of a list of rack definitions (in the above example, only a single rack with rack_id of "rack_1" is specified).

**rack_id** For each rack configured with OpenDCRE, a "rack_id" must be specified to identify that rack. In the example above, "rack_1" is the rack_id. This is the same rack_id specified in OpenDCRE REST API commands. When multiple devicebus types are defined for an OpenDCRE configuration, devices in common rack_ids are merged together into the rack record in scan results for that rack. In other words, devices from multiple devicebus types may be assigned to the same rack in OpenDCRE, assuming the same rack_id is used in each of their configurations.

**lockfile** At the rack-level, a lockfile path and filename may be defined such that all devices belonging to that rack share a common lockfile, ensuring serial and exclusive access to the bus. *(This lockfile may also be shared with other racks and bus types when shared bus/hardware access must be serial across racks and bus types.)*

**hardware_type** Indicates whether hardware is emulated ("emulator") or "real". In the case of "emulator", device interface implementations may use an alternate code path (e.g. for testing or demonstration purposes) routed to an emulator, as opposed to taking physical hardware actions. When using OpenDCRE with emulator backing, "emulator" should be specified here, otherwise, when OpenDCRE is used with real hardware, "real" should be specified for hardware_type.

**devices** Within a given rack, one or more PLC devices may be specified for brokering bus access to the PLC bus. In most cases involving PLC, only a single device is present, corresponding to the PLC modem serial device and its configuration, however multiple devices can be supported (e.g. in the case of multiple PLC buses or modems in a single rack).

**device_name** The path and file name to the serial TTY device for PLC communications. When `hardware_type` is "emulator", this typically corresponds to the OpenDCRE-side of a socat-paired virtual serial connection (e.g. /dev/ttyVapor001). When `hardware_type` is "real", this corresponds to the physical serial device mapped into the OpenDCRE container for use with PLC for reading and writing.

**retry_limit** Configures the number of retries permitted (in case of line noise or bus errors) before an error is returned. The default should be sufficient in most cases. **(default: 3)**

**timeout** A decimal value indicating the time, in seconds, to wait for a response to an OpenDCRE PLC bus command before timing out. The default value is typically sufficient in physical hardware cases as well as with the OpenDCRE PLC emulator. **(default: 0.25)**

**time_slice** The time slice used during a scan command to enumerate all PLC devices on the PLC bus. This value is used to allow devices to use their internal board_id and the time slice value to determine which window to use in responding to the scan command. Users generally should not alter this value. **(default: 75)**

**bps** The bits per second configuration value to use for PLC communications on the PLC bus. This generally should not be modified by users. **(default: 115200)**

If a field is missing, or the PLC configuration file is improperly formatted, OpenDCRE PLC capabilities will not be available.

## IPMI Device

New in version 1.1.0.

IPMI Devices allow users of OpenDCRE to issue LAN-based IPMI commands using the OpenDCRE REST API.

### Supported Commands

The *test* and *service version* are supported by all devicebus types. Below are the commands which are supported for IPMI Devices. See the *API Reference* for details on all commands.

| Command | Supported |
|---|---|
| Version | True |
| Scan | True |
| Scan All | True |
| Read | True |
| Power | True |
| Asset | True |
| Boot Target | True |
| Chamber LED | False |
| LED | True |
| Fan | True |
| Host Info | True |

### Requirements

- OpenDCRE must be connected to a wired LAN network that can reach all BMCs configured to be managed over OpenDCRE.
- Knowledge of BMC IP addresses, ports, usernames, and passwords (where applicable) required.

Changed in version 1.3.0: Previously, a custom IPMI interface was used which required the specification of authentication type, integrity type, and encryption type. Now, pyghmi is used as the IPMI interface, which does not expose customization for those parameters, thus they need no longer be specified in the configuration file.

### Configuration

IPMI Device configurations are specified under the `devices` field of the OpenDCRE *Configuration Options*. A simple example configuration is given below, followed by descriptions of the fields presented in the example.

```
{
  "devices": {
    "ipmi": {
      "scan_on_init": true,
      "device_initializer_threads": 4,
      "from_config": "bmc_config.json"
    }
  }
}
```

**from_config** The file which specifies the rack and BMC configurations for BMCs managed through OpenDCRE. See below for an example of these configurations.

**config** An alternative to `from_config` - this field allows one to specify the rack and BMC configurations in this configuration file as opposed to a separate one. Generally, it is recommended to use `from_config` over this, as it keeps things cleaner, but if only a few BMCs are being specified, it may be easier to define their configurations under this field.

**scan_on_init** *(optional)* A flag which determines whether or not the IPMI Devices will perform a scan operation on device initialization, or if it will be deferred for later. Typically, it is a good idea to scan on initialization, as that is how the board record is created and how the devices off of the BMC are found. Deferring scan to a time post-initialization can be useful in testing or if there is high network latency and one does not want the slow initialization process to delay OpenDCRE startup. **(default: true)**

**device_initializer_threads** *(optional)* The number of threads to use when initializing IPMI Devices. Since IPMI devices use LAN communication, initializing multiple devices can be done in parallel. **(default: 1)**

As mentioned above, the `from_config` and `config` fields specify the BMC-specific configurations. The JSON example below could either be specified under the `config` field, or in the file specified by the `from_config` field.

```
{
  "racks": [
    {
      "rack_id": "rack_1",
      "bmcs": [
        {
          "bmc_ip": "192.168.1.110",
          "username": "ADMIN",
          "password": "ADMIN"
        },
        {
          "bmc_ip": "192.168.1.111",
          "bmc_port": 623,
          "username": "ADMIN",
          "password": "ADMIN",
          "hostnames": ["atom"],
          "ip_addresses": ["192.169.1.111"]
        }
      ]
    }
  ]
}
```

**racks** OpenDCRE is capable of managing multiple racks' worth of BMCs, so the top-level configuration parameter "racks" consists of a list of rack definitions (in the above example, only a single rack with rack_id of "rack_1" is specified).

**rack_id** For each rack configured with OpenDCRE, a "rack_id" must be specified to identify that rack. In the example above, "rack_1" is the rack_id. This is the same rack_id specified in OpenDCRE REST API commands. When multiple devicebus types are defined for an OpenDCRE configuration, devices in common rack_ids are merged together into the rack record in scan results for that rack. In other words, devices from multiple devicebus types may be assigned to the same rack in OpenDCRE, assuming the same rack_id is used in each of their configurations.

**bmcs** The "bmcs" field consists of a list of zero or more BMC configuration records. Each BMC configuration record corresponds to an individual BMC situated in the configured rack.

**bmc_ip** The IP address (or hostname) of the BMC being configured. It must be a string value and the BMC IP must also be accessible over LAN by the OpenDCRE service.

**username** The username used to connect to the BMC. For OpenDCRE to be able to fully control a remote server, the username should have sufficient permissions on the remote BMC.

**password** The password used to connect to the BMC for the given username.

**bmc_port** *(optional)* The UDP port number of the BMC. Must be specified as an integer. **(default: 623)**

**hostnames** *(optional)* A list of known hostnames for the remote system that may be used in place of the board_id of the BMC for OpenDCRE REST API requests. This list may be augmented by OpenDCRE in case of DCMI support, where DCMI may be used to get host identification as well. At minimum, the contents of the "hostnames" list are returned in scan and host_info responses related to the given system.

**ip_addresses** *(optional)* A list of known IP addresses for the remote system that may be used in place of the board_id of the BMC for OpenDCRE REST API requests. This list may be augmented by OpenDCRE to include the bmc_ip (if not already included in this list), allowing access to any IPMI device via OpenDCRE REST API by using the BMC IP or known IP addresses in place of board_id. Contents of the "ip_addresses" list are returned in scan and host_info responses related to the given system.

If a field is missing, or the IPMI configuration file is improperly formatted, OpenDCRE IPMI capabilities will not be available.

### Supported Devices

Currently, the supported devices for IPMI include:

- power
- system
- LED
- fan
- power supply
- temperature
- voltage

### Tested BMCs

OpenDCRE v1.3 has been tested and verified to be compatible with IPMI 2.0 connections and commands for the following BMCs:

- ASpeed AST2400 (via HPE CL7100)
- Nuvoton WPCM450RA0BK (via SuperMicro X7SPA-HF)
- ASpeed AST2050 (via Tyan S8812)
- ASpeed AST1250 (via Freedom)

The OpenDCRE community welcomes testing and bug reports against other BMCs and system types.

## Redfish Device

New in version 1.3.0.

> **Warning:** Redfish support is in beta as of OpenDCRE v1.3.0

Redfish Devices map Redfish schema into OpenDCRE, allowing for LAN-based Redfish commands using the OpenDCRE REST API.

### Supported Commands

The *test* and *service version* are supported by all devicebus types. Below are the commands which are supported for Redfish Devices. See the *API Reference* for details on all commands.

| Command | Supported |
|---|---|
| Version | True |
| Scan | True |
| Scan All | True |
| Read | True |
| Power | True |
| Asset | True |
| Boot Target | True |
| Chamber LED | False |
| LED | True |
| Fan | True |
| Host Info | True |

### Configuration

Redfish Device configurations are specified under the `devices` field of the OpenDCRE *Configuration Options*. A simple example configuration is given below, followed by descriptions of the fields presented in the example.

```
{
  "devices": {
    "redfish": {
      "scan_on_init": true,
      "device_initializer_threads": 4,
      "from_config": "redfish_config.json"
    }
  }
}
```

**from_config** The file which specifies the rack and device configurations for devices managed through OpenDCRE. See below for an example of these configurations.

**config** An alternative to `from_config` - this field allows one to specify the rack and device configurations in this configuration file as opposed to a separate one. Generally, it is recommended to use `from_config` over this, as it keeps things cleaner, but if only a few devices are being specified, it may be easier to define their configurations under this field.

**scan_on_init** *(optional)* A flag which determines whether or not the Redfish Devices will perform a scan operation on device initialization, or if it will be deferred for later. Typically, it is a good idea to scan on initialization, as that is how the board record is created and how the devices are found. Deferring scan to a time post-initialization can be useful in testing or if there is high network latency and one does not want the slow initialization process to delay OpenDCRE startup. **(default: true)**

**device_initializer_threads** *(optional)* The number of threads to use when initializing Redfish Devices. Since Redfish devices use LAN communication, initializing multiple devices can be done in parallel. **(default: 1)**

As mentioned above, the `from_config` and `config` fields specify the device-specific configurations. The JSON example below could either be specified under the `config` field, or in the file specified by the `from_config` field.

```
{
  "racks": [
```

```
    {
      "rack_id": "rack_1",
      "servers": [
        {
          "redfish_ip": "192.168.1.110",
          "redfish_port": "5040",
          "timeout": 5,
          "username": "ADMIN",
          "password": "ADMIN",
          "hostnames": ["redfish-server-1"],
          "ip_addresses": ["192.168.1.110"]
        }
      ]
    }
  ]
}
```

**racks** OpenDCRE is capable of managing multiple racks' worth of servers, so the top-level configuration parameter "racks" consists of a list of rack definitions (in the above example, only a single rack with rack_id of "rack_1" is specified).

**rack_id** For each rack configured with OpenDCRE, a "rack_id" must be specified to identify that rack. In the example above, "rack_1" is the rack_id. This is the same rack_id specified in OpenDCRE REST API commands. When multiple devicebus types are defined for an OpenDCRE configuration, devices in common rack_ids are merged together into the rack record in scan results for that rack. In other words, devices from multiple devicebus types may be assigned to the same rack in OpenDCRE, assuming the same rack_id is used in each of their configurations.

**servers** The servers field consists of a list of zero or more Redfish server configuration records. Each Redfish configuration record corresponds to an individual Redfish server situated in the configured rack.

**redfish_ip** The IP address (or hostname) of the Redfish server being configured. The Redfish IP must also be accessible over LAN by the OpenDCRE service.

**redfish_port** The port which the Redfish server is listening on.

**timeout** The timeout, in seconds, for the HTTP request being made to the Redfish server before an error is raised.

**username** The username used to connect to the Redfish server.

**password** The password used to connect to the Redfish server for the given username.

**hostnames** A list of known hostnames for the remote system that may be used in place of the board_id for the Redfish server for OpenDCRE REST API requests.

**ip_addresses** A list of known IP addresses for the remote system that may be used in place of the board_id for the Redfish server for OpenDCRE REST API requests.

If a field is missing, or the Redfish configuration file is improperly formatted, OpenDCRE Redfish capabilities will not be available.

## Emulators

OpenDCRE comes with pre-built emulators for each supported devicebus type so that it can emulate API commands and functionality in the absence of supported hardware. This is especially useful for testing and experimenting with OpenDCRE.

## PLC Emulator

The PLC Emulator runs as a background process in the OpenDCRE container itself. It uses socat to create a connected pair of virtual TTY devices that can be used to simulate serial communications without hardware. The main difference between the emulated PLC comms and hardware PLC comms is that the emulator does not require the additional steps of configuring the serial device and initializing the SIG60 PLC modem.

The emulator itself is a primitive Python process that is provided a configuration file on startup to map board/device ids to raw packet readings. These packet readings are returned to the device on the other end of the virtual serial connection. Faults, repeated values, cycling values, or no value returns are supported by the PLC emulator through its config file. State can also be preserved in cases where an incoming command mutates state (e.g. turns on an LED) - state honoring may be enabled/disabled in the emulator configuration for a board/device.

### Configuration Example

```
{
  "boards": [
    {
      "board_id": "00000001",
      "firmware_version": "OpenDCRE Emulator v1.0.0 - Standalone Server",
      "devices": [
        {
          "host_info": {
            "repeatable": true,
            "responses": [
              "i10.10.1.16,htest-server0"
            ]
          },
          "has_state": true,
          "boot_target": "B0",
          "pxe" : "B1",
          "no_override": "B2",
          "hdd": "B0",
          "asset_info": {
            "repeatable": true,
            "responses": [
              "Quanta,0001,Winterfell,S1234567,rack mount chassis,P1234567,S1234567,
↪A1234567,Quanta,P1234567,Winterfell,S1234567,v1.2.0"
            ]
          },
          "device_type": "system",
          "device_id": "0001"
        },
        {
          "read": {
            "repeatable": true,
            "responses": [
              4100, 4100, 4000, 4000, 3900, 3900, 3800, 3800, 3700, 3700,
              3800, 3800, 3900, 3900, 4000, 4000, 4100, 4100, 4200, 4200
            ]
          },
          "write": {
            "repeatable": true,
            "responses": [
              "W1"
            ]
          },
```

```
        "device_type": "fan_speed",
        "device_id": "0002"
      },
      {
        "read": {
          "repeatable": true,
          "responses": [
            4100, 4100, 4000, 4000, 3900, 3900, 3800, 3800, 3700, 3700,
            3800, 3800, 3900, 3900, 4000, 4000, 4100, 4100, 4200, 4200
          ]
        },
        "write": {
          "repeatable": true,
          "responses": [
            "W1"
          ]
        },
        "device_type": "fan_speed",
        "device_id": "0003"
      },
      {
        "device_id": "0004",
        "device_type": "power",
        "has_state": true,
        "on": [
          "0,10000,0,0", "0,11000,0,0", "0,12000,0,0", "0,13000,0,0",
          "0,14000,0,0", "0,15000,0,0", "0,14000,0,0", "0,13000,0,0",
          "0,12000,0,0", "0,11000,0,0"
        ],
        "off": "64,0,0,0",
        "power": [
          "0,10000,0,0", "0,11000,0,0", "0,12000,0,0", "0,13000,0,0",
          "0,14000,0,0", "0,15000,0,0", "0,14000,0,0", "0,13000,0,0",
          "0,12000,0,0", "0,11000,0,0"
        ]
      },
      {
        "device_id": "0005",
        "device_type": "led",
        "has_state": true,
        "read": 0,
        "write": 0,
        "on": 1,
        "off": 0
      },
      {
        "read": {
          "repeatable": true,
          "responses": [
            28.78, 29.77, 30.75, 31.84, 32.82, 33.81, 34.89, 35.88, 36.96, 37.94,
            38.93, 40.21, 41.27, 42.33, 43.39, 44.45, 45.61, 46.57, 47.63, 48.69,
            49.75, 48.69, 47.63, 46.57, 45.61, 44.45, 43,39, 42.33, 41.27, 40.21,
            38.93, 37.94, 36.96, 35.88, 34.89, 33.81, 32.82, 31.84, 30.75, 29.77
          ]
        },
        "device_type": "temperature",
        "device_id": "2000"
      },
```

```
        {
          "read": {
            "repeatable": true,
            "responses": [
              28.78, 29.77, 30.75, 31.84, 32.82, 33.81, 34.89, 35.88, 36.96, 37.94,
              38.93, 40.21, 41.27, 42.33, 43.39, 44.45, 45.61, 46.57, 47.63, 48.69,
              49.75, 48.69, 47.63, 46.57, 45.61, 44.45, 43,39, 42.33, 41.27, 40.21,
              38.93, 37.94, 36.96, 35.88, 34.89, 33.81, 32.82, 31.84, 30.75, 29.77
            ]
          },
          "device_type": "temperature",
          "device_id": "4000"
        }
      ]
    }
  ]
}
```

### Configuration Fields

**boards** A list of boards configured for the emulator, where a single board configuration would represent a single hardware board that should exist.

**board_id** The internal id for a single board. Board Ids in OpenDCRE have a 4 byte width and should be expressed in the config as a 4-byte hex string. Board Ids should be unique across all device instances. For PLC, the board id range starts at 0x00000000.

**firmware_version** The version string for the board, which would be returned by the OpenDCRE "version" command.

**devices** A list of all device configurations which are associated with that given board.

**repeatable** A flag which denotes that the given responses should repeat. This means that when the emulator has cycled through all of the responses in the responses list, it will return to the beginning of the list. If this is set to `false`, upon reaching the end of the responses list, the emulator will not return data, causing an error to be raised in OpenDCRE (as the emulator will not respond).

**responses** A list of canned responses for the emulator to return.

**host_info** The response(s) to return on a system "host info" command.

**asset_info** The response(s) to return on a system "asset info" command.

**read** The response(s) to return on a "read" command.

**write** The response(s) to return on a "write" command.

**on** The response(s) to return on a "power on" command.

**off** The response(s) to return on a "power off" command.

**power** The response(s) to return on a "power status" command.

**has_state** If true, then state information is preserved relative to the command (e.g. "on" or "off" for power), in which case subsequent reads retrieve a response relative to the persisted state. When state is undefined, the default response (e.g. "power") for the command is returned. When `has_state` is false, the response relative to an incoming command is returned (e.g. the response for "on" for a power "on" command).

**boot_target** Indicates the response sent back for a "get" of boot target (B0 == no_override, B1 == pxe, B2 == hdd).

**pxe** The response sent when boot target of PXE is set.

**no_override** The response sent when boot target of no_override is set.

**hdd** The response sent when boot target of HDD is set.

**device_type** Indicates the type of device that a device entry represents. This is also the `device_type` reported back in OpenDCRE REST API scan results. Valid device types include:

- `thermistor`
- `power`
- `humidity`
- `pressure`
- `led`
- `system`
- `fan_speed`
- `temperature`

Changed in version 1.2: In previous releases, a device type of `none` indicated that no device is present at a given `device_id` on the given board, and may be ignored. In OpenDCRE v1.2 the `none` device type has been removed.

**device_id** The internal device id of the device being configured, expressed as a 2-byte numeric value as a hex string. In most cases, a device id of "0001" is sufficient.

### Additional Information on Configuration Fields

### Device Type

A field corresponding to the action supported for a given device type is required. A map of device types to supported actions is below:

| Device Type | Action Supported |
|---|---|
| `thermistor` | `read` |
| `temperature` | `read` |
| `power` | `power` |
| `humidity` | `read` |
| `pressure` | `read` |
| `led` | `read`, `write` |
| `fan_speed` | `read`, `write` |
| `system` | `asset info`, `boot target` |

### Read

For the `read` action's field in the OpenDCRE emulator configuration, two fields may be configured relating to the responses returned from a read command for the given device.

First, the `repeatable` field may be set to true or false, depending on whether it is desirable for the list of responses set in the responses field to repeat in a round-robin fashion, or if a device should stop returning data after its response list has been exhausted.

The `responses` field is a list of zero or more values that may be returned for a given read command. The raw values are converted (where necessary) by the built-in OpenDCRE conversion functions, based on the given `device_type`.

When a list of values is provided for responses, the emulator iterates sequentially through the items in that list, until the list is exhausted (if repeatable is set to "true", then the emulator returns to the beginning of the list).

An empty responses list means the device returns no data, which translates to a 500 error for the read command at the OpenDCRE REST API level (useful for simulating errors). To always return the same single value, a responses list with a single element, and repeatable set to "true" will suffice.

### Read Response Format

The table below describes the response format for each device type for `read` commands to the emulator.

| Device Type | Format |
|---|---|
| `thermistor` | integer, converted by OpenDCRE |
| `temperature` | numeric, sent back as numeric value (e.g. 28.78) |
| `humidity` | numeric, converted by OpenDCRE |
| `led` | integer, `1` is `on` and `0` is `off`; all other values are errors |
| `fan_speed` | integer, sent back as integer value (e.g. 4100) |

Values that do not conform to the above formats will result in errors to `read` requests made to the emulator, as they would on the device bus.

### Write

For the `write` action's field in the OpenDCRE emulator configuration, two fields may be configured, relating to the responses returned from a write command for the given device. The fields are laid out and function in the same manner as `read` fields.

### Write Response Format

The table below describes the response format for each device type for `write` commands to the emulator.

| Device Type | Format |
|---|---|
| `led` | string - `W1` is successful, while `W0` is unsuccessful; all other values are errors. |
| `fan_speed` | string - `W1` is successful, while `W0` is unsuccessful; all other values are errors. |

Values that do not conform to the above formats will result in errors to `write` requests made to the emulator, as they would on the device bus.

Writing to a device from OpenDCRE to the emulator does not currently result in any state change for a corresponding device in the emulator. That functionality may be added in a future release.

### Power

For the `power` action's field in the OpenDCRE emulator configuration, similar fields are present - repeatable and responses.

For every power command (e.g. `on`/`off`/`cycle`/`status`) issued to a power device in the OpenDCRE emulator, a response is returned from the responses list, which may be repeatable or non-repeatable. The values in the responses list correspond to power status values returned over PMBUS from the hot swap controller on an OCP server, and are expressed as an integer value in the emulator configuration (see example above). OpenDCRE converts the raw response to a friendly power status result using its built-in conversion functions.

### Other Notes

The OpenDCRE emulator is also used for testing purposes, and additional emulator configurations may be found under the /opendcre/opendcre_southbound/tests/data directory of the OpenDCRE Docker container.

An invalid emulator configuration will cause the OpenDCRE emulator to fail to start or function properly.

Additional features of the emulator that may be used by advanced users or hardware/protocol developers include:

- Ability to send back raw bytes for responses to scan, version, read, write, and power commands. In tests, this can be seen where a list (or list of lists) of integer values is specified for a given response. Special sentinel values (999, 10xx) are used to place sequence numbers and checksums into the packet stream.
- Ability to support command retries in cases of invalid packets, line noise, etc.
- Ability to support 'scan-all' command and retries using time-division multiplexing; success and failure scenarios may be implemented for various configurations. See the test-scanall tests.

### Running the Emulator

To run the PLC emulator, simply specify the startup script for the PLC emulator.

```
docker run -p 5000:5000 vaporio/opendcre ./start_opendcre_plc_emulator.sh
```

or, if using docker-compose:

```
opendcre:
  image: vaporio/opendcre
  command: ./start_opendcre_plc_emulator.sh
  ports:
    - 5000:5000
```

The examples above will start the emulator with the default configuration file, found at /opendcre/opendcre_southbound/emulator/plc/data/example.json. To specify different emulator configurations, simply pass that file as an argument to the emulator start script. Note that if the non-default emulator configuration is not built into the OpenDCRE image, it will need to be volume-mounted in, e.g.

```
docker run \
    -p 5000:5000 \
    -v `pwd`/emulator_config:/opendcre/new_emulator_config.json \
    vaporio/opendcre \
    ./start_opendcre_plc_emulator.sh /opendcre/new_emulator_config.json
```

## IPMI Emulator

For IPMI communications, there is an IPMI emulator which exists as a Dockerized Python multithreaded UDP server that accepts inbound UDP IPMI packets, processes them, and returns a response based on the emulator configuration and internal state.

The IPMI Emulator, which is perhaps better described as a BMC Emulator, is stateful where applicable. For example, one can set the boot target or LED state on the emulator and a subsequent examination of either should reveal the state to be the new values it was set to.

The IPMI Emulator is primarily designed to work with pyghmi, as that is the library used within OpenDCRE to issue IPMI commands. To accommodate pyghmi, the emulator supports: - HMAC-SHA1-96 integrity checking - RAKP_HMAC_SHA1 authentication - AES_CBC_128 encryption. The encrypted mode can be tested using pyghmi or ipmitool.

For ease of use and simplicity for debugging, it also supports no authentication/encryption which allows all bytes to be examined (e.g. with Wireshark). This unencrypted mode can be tested using ipmitool.

The IPMI emulator is largely just a framing device which unpacks incoming requests and packs outgoing responses. The actual logic to handle commands is often simple, typically just returning values either from internal state or emulator configuration.

### Configuration Example

The IPMI Emulator configuration lives in the `opendcre/opendcre_southbound/emulator/ipmi/data` directory and is built into the emulator's Docker image (at the same path, starting at root). Configurations can be changed by either - Modifying the source configurations and rebuilding the Docker image - Mounting in configuration overrides with Docker volumes.

There are four configuration files associated with the IPMI emulator

**Note:** All of the raw bytes specified in these config files were taken off the wire (using Wireshark) when communicating with a real BMC.

### bmc.json

*bmc.json* contains the configurations for the mock BMC that is the IPMI emulator. It allows the specification of device info, chassis info, channel authentication capabilities, and dcmi configurations. Generally, the configurations specified in this file are the raw bytes that make up the IPMI responses.


```
{
  "device": {
    "device_id": "20",
    "device_revision": "01",
    "device_availability": "03",
    "minor_firmware_revision": "16",
    "ipmi_version": "02",
    "additional_device_support": "bf",
    "manufacturer_id": 47488,
    "product_id": 2566
  },
  "chassis": {
    "current_power_state": "01",
    "last_power_event": "00",
    "misc_state": "40",
    "bootdev": "no_override"
  },
  "channel_auth_capabilities": {
    "channel": "01",
    "version_compatibility": "96",
    "user_capabilities": "06",
    "supported_connections": "03",
    "oem_id": 21317,
    "oem_auxiliary_data": "00"
  },
  "capabilities": {
    "hpm": ["81", "b4", "cb", "20", "08", "3e", "c1", "d9"],
    "picmg": ["81", "b4", "cb", "20", "10", "00", "c1", "0f"],
    "vita": ["81", "b4", "cb", "20", "14", "00", "c1", "0b"]
```

```
  },
  "dcmi": {
    "power": {
      "current_watts": [185, 188, 186, 189, 188, 192, 195, 199, 210, 211, 213, 211,␣
↪212],
      "min_watts": 150,
      "max_watts": 250,
      "avg_watts": 200,
      "reporting_interval_ms": 305000
    },
    "capabilities": {
      "1": ["dc", "01", "05", "02", "00", "01", "07"],
      "2": ["dc", "01", "05", "02", "00", "00", "00", "00", "00"],
      "3": ["dc", "01", "05", "02", "20", "00"],
      "4": ["dc", "01", "05", "02", "ff", "ff", "ff"],
      "5": ["dc", "01", "05", "02", "01", "00"]
    }
  }
}
```

### fru.json

*fru.json* contains the raw configuration data for the mock BMC's FRU. The config file specifies the FRU inventory area and the raw data that makes up the FRU.

```
{
  "inventory_area": 1024,
  "device_access": 0,
  "data": [
    "01", "00", "00", "01", "06", "00", "00", "f8", "01", "05", "00",
    "00", "00", "00", "ca", "53", "75", "70", "65", "72", "6d", "69",
    "63", "72", "6f", "c0", "ca", "20", "20", "20", "20", "20", "20",
    "20", "20", "20", "20", "c0", "c0", "c1", "00", "00", "00", "00",
    "00", "00", "fc", "00", "01", "03", "00", "c0", "c0", "c0", "c0",
    "ca", "20", "20", "20", "20", "20", "20", "20", "20", "20", "20",
    "c0", "c0", "c1", "00", "00", "b1"
  ]
}
```

### sdr.json

*sdr.json* contains the raw configuration data for the mock BMC's SDR. This includes the version, record count, free space, latest addition, latest erase, and operation support. The configuration for the actual SDR records is not specified in this file, but in *sdr_entries.json*.

```
{
  "sdr_version": 1.5,
  "record_count": 21,
  "free_space": 1663,
  "latest_addition_ts": 0,
  "latest_erase_ts": 0,
  "operation_support": "2f"
}
```

### sdr_entries.json

*sdr_entries.json* is the config file where all device records belonging to the SDR are defined. The number of devices defined in this config should match the device count specified in *sdr.json*. For each record, an id, sensor type, data, readings, event messages, and threshold comparison field should be specified. The sensor type field is not used by the IPMI emulator, but is used as a convenient means of labeling the record with a human-readable description.

```
{
  "records": [
    {
      "id": "0000",
      "sensor_type": "System Temp",
      "data": [
        "04", "00", "51", "01", "36", "20", "00", "11", "07", "01", "7d", "68",
        "01", "01", "80", "7a", "80", "7a", "3f", "3f", "80", "01", "00", "00",
        "01", "00", "00", "00", "00", "00", "07", "2d", "4a", "fc", "7f", "80",
        "4f", "4d", "4b", "f7", "f9", "fb", "02", "02", "00", "00", "00", "cb",
        "53", "79", "73", "74", "65", "6d", "20", "54", "65", "6d", "70"
      ],
      "readings": [
        49, 49, 48, 47, 48
      ],
      "event_messages": "c0",
      "threshold_comparison": ["c0"]
    },
    {
      "id": "0047",
      "sensor_type": "CPU Temp",
      "data": [
        "47", "00", "51", "01", "33", "20", "00", "12", "03", "01", "7f", "68",
        "01", "01", "80", "7a", "80", "7a", "3f", "3f", "80", "01", "00", "00",
        "01", "00", "00", "00", "00", "00", "07", "1e", "59", "fc", "7f", "80",
        "5f", "5a", "55", "f5", "f8", "fb", "02", "02", "00", "00", "00", "c8",
        "43", "50", "55", "20", "54", "65", "6d", "70"
      ],
      "readings": [
        41, 40, 41, 41
      ],
      "event_messages": "c0",
      "threshold_comparison": ["c0"]
    },
    {
      "id": "008a",
      "sensor_type": "CPU FAN",
      "data": [
        "8a", "00", "51", "01", "32", "20", "00", "41", "1d", "01", "7d", "68",
        "04", "01", "95", "7a", "95", "7a", "3f", "3f", "00", "12", "00", "00",
        "b9", "00", "00", "c0", "00", "01", "07", "80", "aa", "14", "ff", "00",
        "b2", "af", "ac", "10", "11", "12", "01", "01", "00", "00", "00", "c7",
        "43", "50", "55", "20", "46", "41", "4e"
      ],
      "readings": [
        34, 34, 35, 34, 33
      ],
      "event_messages": "c0",
      "threshold_comparison": ["c0"]
    },
    ...
```

```
    (abridged for brevity)
  ]
}
```

### Getting the Emulator

Since the IPMI Emulator is a standalone image, it needs to either be pulled from DockerHub, or built from Dockerfile.

From DockerHub,

```
docker pull vaporio/ipmi-emulator-x64
```

From Dockerfile, first navigate to `opendcre_southbound/emulator/ipmi`. Then, you can build the IPMI emulator image with

```
make build-x64
```

### Running / Using the Emulator

Running the emulator in isolation is straightforward enough. Once you have the image, you can run it either with docker:

```
docker run --name ipmi-emulator -p 623:623/udp vaporio/ipmi-emulator-x64
```

or with docker-compose

```
docker-compose -f ipmi-emulator.yml up --build -d ipmi-emulator
```

where `ipmi-emulator.yml` contains

```
ipmi-emulator:
  container_name: ipmi-emulator
  image: vaporio/ipmi-emulator-x64
  command: ./start_ipmi_emulator.sh
  ports:
    - 623:623/udp
```

While other emulators (e.g. the PLC emulator) are built in to OpenDCRE and can be run from the same container, the IPMI emulator must be run from a separate container, as shown above.

This is done in part for emulator isolation, but also because it allows for more flexible test setups. For instance, with the emulator running in a separate container it is possible to spin up multiple emulator instances, each with their own configuration, to emulate OpenDCRE performance against different BMC models. Additionally, with docker-compose, OpenDCRE can have multiple proxies to the same emulator to simulate a full rack, cluster, or multi-cluster of BMCs.

**Warning:** When using the IPMI emulator in a proxied fashion, where multiple composefile links point to the same emulator, the number of requests issued against the emulator can become an issue, especially under high network latency, where the requests back up and time out.

When run locally, this had caused the emulator to freeze up and communications between OpenDCRE and the emulator fail. One solution to this is to run the IPMI Emulator on a separate instance/machine when there will be heavy load placed upon it. This will ensure that it is given enough machine resources to operate at full capacity - although network latency can then become an issue.

Running the emulator on a separate instance/machine is recommended even without heavy load, for stability and performance.

Above, we describe how to run an IPMI emulator. Some additional configuration will need to happen with OpenDCRE in order for it to register the IPMI emulator as a usable interface.

The networking between the emulator and OpenDCRE is determined by the BMC config used by OpenDCRE. For example, if there were a BMC config containing the record:

```
{
  "bmc_ip": "localhost",
  "username": "ADMIN",
  "password": "ADMIN"
}
```

we would want the emulator to be running on the same machine as OpenDCRE, as localhost should resolve to the emulator.

The containers can also be linked in the composefile, if running on the same machine, so we can reference the emulator using the container name as a hostname:

```
{
  "bmc_ip": "ipmi-emulator",
  "username": "ADMIN",
  "password": "ADMIN"
}
```

Of course, a plain IP for the machine running the emulator can be supplied as the bmc_ip without any need to create container links.

An example (abridged) composefile with the two containers linked is as follows:

```
opendcre:
  image: vaporio/opendcre-core-x64
  command: ./start_opendcre.sh
  ports:
    - 5000:5000
  links:
    - ipmi-emulator

ipmi-emulator:
  image: vaporio/ipmi-emulator-x64
  ports:
    - 623:623/udp
```

Note that here, the OpenDCRE instance was started without running any other emulator. While it is possible (and fine) to run the IPMI emulator alongside any of the serial emulators, keeping things isolated to IPMI-only for testing is usually prudent.

## Redfish Emulator

**Warning:** Redfish support is in beta as of OpenDCRE v1.3.0

Like the IPMI emulator, the Redfish Emulator is a standalone Dockerized python application. It runs a simple Flask webserver that serves up statically defined configuration data. It supports the basic Redfish commands and is stateful, for operations where state can be preserved (e.g. turning an LED on). By default, the Redfish emulator runs on port 5040. This can be changed by updating the Dockerfile and specifying the correct port mapping at run time.

### Configuration

The configuration files which make up the Redfish emulator backend are too numerous to include here - instead, see the Redfish mockups which the configuration hierarchy is based off of.

To re-configure the Redfish emulator, either a new emulator image can be built with the new configuration placed in the emulator's *Resources* directory, or it can be volume mounted in over the emulator's *Resources* directory.

### Getting the Emulator

Since the Redfish Emulator is a standalone image, it needs to either be pulled from DockerHub, or built from Dockerfile.

From DockerHub,

```
docker pull vaporio/redfish-emulator-x64
```

From Dockerfile, first navigate to `opendcre_southbound/emulator/redfish`. Then, you can build the Redfish emulator image with

```
make build-x64
```

### Running the Emulator

Running the Redfish emulator is simple, given that the desired configurations (whether they be the default or custom built-in/volume-mounted configurations) are correctly placed in the image:

```
docker run -p 5040:5040 vaporio/redfish-emulator-x64
```

# Testing

OpenDCRE is well-tested with hundreds of test cases that can be run. OpenDCRE tests must be run from the source code (see *Getting OpenDCRE*). All of the tests can be found under the `opendcre_southbound/tests` directory.

**OpenDCRE tests are made up of four somewhat distinct parts:**

1. The test runner
2. The test configuration
3. The test suite
4. The test cases

## Test Runner

In `opendcre_southbound/tests`, there is a Makefile – this is the test runner. The Makefile contains recipes to run all of the tests OpenDCRE has. Tests can be run at a suite-level, or in groups of suites (e.g. all tests, tests for a given devicebus type, etc). Running all OpenDCRE tests is as simple as

```
make test-x64
```

Testing for only a single devicebus type can be done by adding the device type suffix, e.g. for PLC

```
make test-x64-plc
```

for IPMI

```
make test-x64-ipmi
```

and for Redfish

```
make test-x64-redfish
```

See the Makefile for more recipes and more information about how to run the tests.

## Test Configuration

Each test in OpenDCRE is run out of a Docker container – fitting, since OpenDCRE runs out of a Docker container. This gives us an easy and uniform way to do unit testing and integration testing. The tests are orchestrated using docker-compose, and as such, each test suite will need to have its own compose file configuration. These compose files can be found in the `opendcre_southbound/tests/_composefiles` directory. Below is an example composefile configuration, followed by a brief explanation.

```
test-container-x64:
  container_name: test-container-x64
  build: ../../../..
  dockerfile: dockerfile/Dockerfile.x64
  command: bash -c "sleep 15 && python ./opendcre_southbound/tests/test-plc-endpoints.
↪py"
  links:
    - opendcre-southbound-test-container

opendcre-southbound-test-container:
  build: ../../../..
  dockerfile: dockerfile/Dockerfile.x64
  command: ./start_opendcre_plc_emulator.sh ./opendcre_southbound/emulator/plc/data/
↪test_bus.json
  expose:
    - 5000
  volumes:
    - ../../data/plc_override_config.json:/opendcre/override/config.json
  environment:
    - VAPOR_DEBUG=true
```

In the example above, we are performing an integration test on the OpenDCRE REST API with PLC backing (via the PLC emulator). In it, we have two containers that will run, "test-container-x64", the container that will run the actual test suite, and "opendcre-southbound-test-container", the OpenDCRE instance running with the PLC emulator backing.

---

**Note:** All containers which run the actual test cases should be named "test-container-x64", since this name is used in the Makefile runner to attach to that container, allowing us to see the test results in the console.

---

Both test container and OpenDCRE container are built from the same Dockerfile - this is just for convenience since all of the test dependencies already exist in the OpenDCRE image.

The test container runs a test suite after a sleep. Not all tests need a sleep period, but they are often included to allow OpenDCRE to come up and fully configure before the tests start running against it.

Finally, we set the `VAPOR_DEBUG` environment variable to `true` - this enables debug logging in OpenDCRE. This isn't necessary, but if a test does fail, it makes it eaiser to find the root of the failure.

### Test Suite

As seen above in the compose file, the test suite is defined in the `opendcre_southbound/tests` directory. In the same directory, there should be a subdirectory with a name corresponding to the name of the python file that is the test suite. The subdirectory contains the test cases which make up the suite. The suite acts only to aggregate and run the test cases.

### Test Cases

The test cases are the actual test code that is run. It uses Python's unittest package to define the tests in the test cases. As mentioned in the previous section, these are aggregated into a suite for running, so the test cases need not be contained to a single file, and are in fact often broken up into multiple files for clarity and organization.

## Debugging

In the case of OpenDCRE failures or errors, it is helpful to know how some basic debugging steps.

### Startup Errors

As covered in the subsequent sections here, OpenDCRE logs out to files within the container. One exception to this is for errors on container startup. In addition to logging out to file, it logs out to stderr of the container (pid 1), so it is accessible via `docker logs`. If a startup error should occur, the OpenDCRE container should terminate. In this case, it is often prudent to check the docker logs first to see if there is any information pertaining to a startup error.

```
docker logs opendcre
```

A startup error likely will only happen if there is a serious misconfiguration, or if building a custom OpenDCRE image, something was changed to break the initialization process.

### Enabling Debug Logging

By default, production logging is enabled (logging at an ERROR level). To run things in "debug" mode, where DEBUG level logs are collected, simply run the container with the environment variable `VAPOR_DEBUG` set to `true`. This can be done in the docker command

```
docker run -p 5000:5000 -e VAPOR_DEBUG=true vaporio/opendcre
```

or via composefile

```
opendcre:
  image: vaporio/opendcre
  ports:
    - 5000:5000
  environment:
    - VAPOR_DEBUG=true
```

With debug logging enabled, there will be two sets of logs captured – the error logs (same as the production logs), and the debug logs.

## Examining Log Files

There are two primary methods for accessing a container's logs: `docker exec` and `docker cp`.

### docker exec

Using `docker exec` requires the container to be running. If the container has terminated and you need to get at the logs, this is not the method that should be used. To exec into a container interactively,

```
docker exec -ti <name or id of container> /bin/bash
```

This will drop you into the working directory of the container, `/opendcre`. From there, navigate to `/logs`, where all of the container logs are kept.

### docker cp

Using `docker cp` can be done when the container is running or when it has terminated. It is used to copy the logs from the container file system to the host file system.

```
docker cp <name or id of container>:/logs <host copy destination>
```

This will place the log files on the host system at the specified location.

# Troubleshooting

This section contains information on troubleshooting, common gotchas, and generally things to watch out for.

## Testing the Service

A good practice is to hit the *test* endpoint after starting up a service. It can also be useful in ensuring that a service is still running. It will only return "ok" if the service is up and running, but note that an "ok" response from the test endpoint does **not** indicate the lack of errors elsewhere - it only means that the Flask application within the Docker container is running and that it is reachable.

## Raising an Issue

If you come across an error or issue with OpenDCRE or just have a question/suggestion about it, please raise an issue for it!

If raising an issue around an error, please include as much context information around it as possible.

## Gotchas

### scan results unavailable on init

Particularly in cases when there are many devices configured, or where there is high network latency when configuring LAN Devices, one may observe that the initial startup takes some time and scan results are not available. OpenDCRE attempts to enumerate remote devices (e.g. BMCs for IPMI) on init by default, so that it can cache the results for subsequent commands. As such, the initialization process may take a few minutes.

To help mitigate this, some configuration options have been added around LAN-based devices, both for multi-threaded device initialization, and deferring scan to a time post-initialization.

If can results are unavailable for an extended amount of time, or just to validate that there are no errors while the initial scan takes place, one can examine the OpenDCRE log file, as described in the *Debugging* section.

# Release Notes

## v1.3.0

*February 21, 2017*

The OpenDCRE 1.3.0 release is another significant release which brings many changes to the OpenDCRE code base. Along with significant code refactoring, cleanup, documentation, and testing, this release brings in Redfish support.

**Note:** Redfish support should be considered a beta feature as of v1.3.0

### Changelog

- Generalization of the underlying devicebus model. This makes it easier to add support for new protocols / devices moving forward.
- Added command dispatching from OpenDCRE endpoints to the new underlying devicebus model.
- Improvements to OpenDCRE configuration which allow for default and override configs.
- Switched internal IPMI back-end from custom C implementation to OpenStack's pyghmi
- [BETA] Redfish support in OpenDCRE
  - Standalone Redfish Emulator
  - Redfish Device integration into OpenDCRE
  - Test cases for the emulator as well as OpenDCRE configured under Redfish
- Various bug fixes and performance fixes.
- Improvements to code organization, formatting, and documentation.

- Dockerfile optimizations.
- Additional test cases added and existing test cases expanded.

### Contributors

- Andrew Cencini, Vapor IO
- Erick Daniszewski, Vapor IO
- Matthew Hink, Vapor IO
- Klemente Gilbert-Espada, Vapor IO
- Kyler Burke, Vapor IO
- Morgan Mills, Vapor IO / Bennington College
- Linh Hoang, Vapor IO / Bennington College

## v1.2.0

*March 5, 2016*

The OpenDCRE v1.2.0 release is a significant release, adding a large set of new features to OpenDCRE.

This release provides a long-awaited major update to the original OpenDCRE v1.x codebase, and has excellent feature, test, and usability enhancements.

### Changelog

- IPMI 2.0 support added to IPMI bridge
- Added support via PLC and IPMI for
  - *fan* command (fan control)
  - *led* command (chassis "identify" LED control)
  - *location* command (physical and intra-chassis location)
  - *asset* command (asset information)
  - *boot_target* command (boot target selection)
  - *temperature*, *humidity*, *fan_speed* sensor support added
- Normalization of OpenDCRE API command layout for consistency and future functionality
- PLC communications (devicebus_interfaces) for v1 RPI HAT finalized
- Emulator enhancements for new functionality
- Improvements to code organization, PEP8 compliance, documentation
- Testing via docker-compose (supported on RPI and Linux/MacOS)

### Contributors

- **Andrew Cencini, Vapor IO (maintainer)**
  - IPMI, PLC, code, test enhancements
- **Klemente Gilbert-Espada, Vapor IO (docs, contributor)**
  - RPI `pyserial` bugfix, Sphinx documentation
- **Erick Daniszewski, Vapor IO (contributor)**
  - Test enhancements, bugfixes

Special thanks to early adopters and testers of OpenDCRE.

## Glossary

**OpenDCRE** The Open Data Center Runtime Environment

**devicebus** A term which describes devices (e.g. IPMI device) and busses (e.g. PLC bus), synonymous with “device bus”.

**PLC** Power line communications

## License

OpenDCRE is licensed under the GNU General Public License v2.0. The license, and that of OpenDCRE’s dependencies are in full, below.

```
OpenDCRE v.1
Copyright (c) 2015-2017 Vapor IO, Inc. and OpenDCRE contributors.

The following is courtesy of our legal counsel:

Use and transfer of OpenDCRE may be subject to certain restrictions by
the United States and other governments. It is your responsibility to
ensure that your use and/or transfer does not violate applicable laws.

For more information, please see https://www.bis.doc.gov

This product includes software developed at Vapor IO, Inc.
(http://www.vapor.io) that is licensed under the GNU General Public
License Version 2 (see LICENSE file) and other third party open
source software in unmodified binary form. Certain of these third
party open source software components have their own license as noted
and provided below.

This product contains software (https://github.com/docker/docker)
developed by Docker, Inc., licensed under the Apache License, Version
2.0. Copyright (c) 2012-2016 Docker, Inc.

This product contains software (https://github.com/certik/python-2.7)
developed by Python Software Foundation, licensed under the PSF
license, Version 2. Copyright © 2001, 2002, 2003, 2004, 2005, 2006,
2007, 2008, 2009, 2010, 2011,2012 Python Software Foundation.  All
rights reserved. Copyright (c) 2000 BeOpen.com.  All rights reserved.
```

```
Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All rights reserved.  Copyright (c) 1991-1995 Stichting Mathematisch
Centrum. All rights reserved.

This product contains software (https://github.com/gcc-mirror/gcc)
developed at the GNU Complier Collection, licensed under the GNU
General Public License, Version 3. Copyright years on GCC source files
may be listed using range notation, e.g., 1987-2012, indicating that
every year in the range, inclusive, is a copyrightable year that could
otherwise be listed individually.

This product contains software (http://nginx.org/) developed by Nginx,
Inc. and licensed under the 2-clause BSD-like license.  Copyright (c)
2002-2016 Igor Sysoev.  Copyright (c) 2011-2016 Nginx, Inc.

This product contains software (https://github.com/unbit/uwsgi)
developed at the uWSGI Project and licensed under the GNU General
Public License, Version 2, Linking Exception.  Copyright (c) 2012-2014
uWSGI.

This product contains software (http:flask.pocoo.org) developed at the
Flask Project and licensed under the BSD License.  Copyright (c) 2016
by Armin Ronacher and various contributors.  Some rights reserved.

This product contains software (https://pypi.python.org/pypi/pip)
developed by the pip developers and licensed under the MIT License.
Copyright (c) 2008-2016 The pip developers (see AUTHORS.txt file).

This product contains software (http://www.dest-unreach.org/socat/) licensed
under the GNU General Public License Version 2.0.

This product contains software (https://github.com/pyserial/pyserial)
developed by Chris Liechti and licensed under the BSD 3-clause
license.  Copyright (c) 2001-2016 Chris Liechti cliechti@gmx.net.  All
rights reserved.

This product contains software (https://github.com/kennethreitz/requests) developed
by Andrey Petrov and various contributors and licensed under the MIT License.␣
↪Copyright
2008-2011 Andrey Petrov and contributors (see CONTRIBUTORS.txt),
Modifications copyright 2012 Kenneth Reitz.

This product contains software (https://git.openstack.org/cgit/openstack/pylockfile/)
developed by Skip Montanaro and licensed under the MIT License.
Copyright (c) 2007 Skip Montanaro.

This product contains software (http://savannah.gnu.org/projects/make)
developed at the GNU Project and licensed under the GNU General Public
License Version 2 or later.

This product contains software (https://www.openssl.org/) developed by
the OpenSSL Project and licensed under the OpenSSL License.  Copyright
(c) 1999-2016 The OpenSSL Project Authors.  All Rights Reserved.

This product contains software (https://pypi.python.org/pypi/setuptools)
developed by PyPA and licensed under the Zope Public License Ver.2.0
or the PSF License.  Copyright (c) 2014, PyPA.
```

```
This product contains software (https://golang.org/) developed by the
Go Authors and licensed under the BSD License.  Copyright (c) 2012 The
Go Authors.  All rights reserved.

This product contains software (https://github.com/CumulusNetworks/jdoo/blob/master/
↪sha.c)
developed by Scott G. Miller and licensed under the GNU General Public
License Version 2.  Copyright (c) 2000, 2001, 2003 Scott G. Miller.

This product contains software (https://linuxcontainers.org/) licensed
under the terms of the GNU LGPLv2.1+ License, with some Android
compatibility bits licensed under a standard 2-clause BSD license and
some binaries and templates licensed under the GNU General Public
License Version 2.

This product contains software (https://github.com/openstack/pyghmi)
licensed under the terms of the Apache License Version 2.0.

This product contains software (https://pypi.python.org/pypi/pymodbus)
licensed under the terms of the BSD License. Copyright (c) 2011 Galen Collins.
All rights reserved.

This product contains software (https://pypi.python.org/pypi/Twisted) licensed
under the terms of the MIT License.  Copyright (c) 2001-2016 Allen Short, Amber
Hawkie Brown, Andrew Bennetts, Andy Gayton, Antoine Pitrou, Apple Computer, Inc.,
Ashwini Oruganti, Benjamin Bruheim, Bob Ippolito, Canonical Limited, Christopher
Armstrong, David Reid, Divmod Inc., Donovan Preston, Eric Mangold, Eyal Lotem,
Google Inc., Hybrid Logic Ltd., Hynek Schlawack, Itamar Turner-Trauring, James
Knight, Jason A. Mobarak, Jean-Paul Calderone, Jessica McKellar, Jonathan D. Simms,
Jonathan Jacobs, Jonathan Lange, Julian Berman, Jürgen Hermann, Kevin Horn, Kevin␣
↪Turner,
Laurens Van Houtven, Mary Gardiner, Massachusetts Institute of Technology, Matthew
Lefkowitz, Moshe Zadka, Paul Swartz, Pavel Pergamenshchik, Rackspace, US Inc., Ralph
Meijer, Richard Wall, Sean Riley, Software Freedom Conservancy, Tavendo GmbH, Thijs
Triemstra, Thomas Herve, Timothy Allen, Tom Prince, Travis B. Hartwell and others that
have contributed code to the public domain.

This product contains software (https://pypi.python.org/pypi/zope.interface) licensed
under the Zope Public License Ver. 2.1. Copyright (c) Zope Foundation and␣
↪Contributors.

----------------------------------------------------------------------
----------------------------------------------------------------------


APACHE LICENSE
                           Version 2.0, January 2004
                        https://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use,
reproduction, and distribution as defined by Sections 1 through 9 of
this document.

      "Licensor" shall mean the copyright owner or entity authorized
 by the copyright owner that is granting the License.
```

```
      "Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making
modifications, including but not limited to software source code,
documentation source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but not
limited to compiled object code, generated documentation, and
 conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work (an
example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or
Object form, that is based on (or derived from) the Work and for which
the editorial revisions, annotations, elaborations, or other
modifications represent, as a whole, an original work of authorship.
For the purposes of this License, Derivative Works shall not include
works that remain separable from, or merely link (or bind by name) to
the interfaces of, the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including the
original version of the Work and any modifications or additions to
that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control
systems, and issue tracking systems that are managed by, or on behalf
of, the Licensor for the purpose of discussing and improving the Work,
but excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal
Entity on behalf of whom a Contribution has been received by Licensor
and subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions
of this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.
```

```
5. Submission of Contributions. Unless You explicitly state
   otherwise, any Contribution intentionally submitted for
   inclusion in the Work by You to the Licensor shall be under the
   terms and conditions of this License, without any additional
   terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or
   modify the terms of any separate license agreement you may have
   executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the
   trade names, trademarks, service marks, or product names of the
   Licensor, except as required for reasonable and customary use in
   describing the origin of the Work and reproducing the content of
   the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or
   conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or
   FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for
   determining the appropriateness of using or redistributing the
   Work and assume any risks associated with Your exercise of
   permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and
   grossly negligent acts) or agreed to in writing, shall any
   Contributor be liable to You for damages, including any direct,
   indirect, special, incidental, or consequential damages of any
   character arising as a result of this License or out of the use
   or inability to use the Work (including but not limited to
   damages for loss of goodwill, work stoppage, computer failure or
   malfunction, or any and all other commercial damages or losses),
   even if such Contributor has been advised of the possibility of
   such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty,
   indemnity, or other liability obligations and/or rights
   consistent with this License. However, in accepting such
   obligations, You may act only on Your own behalf and on Your
   sole responsibility, not on behalf of any other Contributor, and
   only if You agree to indemnify, defend, and hold each
   Contributor harmless for any liability incurred by, or claims
   asserted against, such Contributor by reason of your accepting
   any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

```
      https://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied.
   See the License for the specific language governing permissions and
   limitations under the License.



----------------------------------------------------------------------
----------------------------------------------------------------------

                           PSF LICENSE
                            VERSION 2

A. HISTORY OF THE SOFTWARE
==========================

Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC.  Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team.  In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope
Corporation, see http://www.zope.com).  In 2001, the Python Software
Foundation (PSF, see http://www.python.org/psf/) was formed, a
non-profit organization created specifically to own Python-related
Intellectual Property.  Zope Corporation is a sponsoring member of
the PSF.

All Python releases are Open Source (see http://www.opensource.org for
the Open Source Definition).  Historically, most, but not all, Python
releases have also been GPL-compatible; the table below summarizes
the various releases.

    Release         Derived     Year        Owner       GPL-
                    from                                compatible? (1)

    0.9.0 thru 1.2              1991-1995   CWI         yes
    1.3 thru 1.5.2  1.2         1995-1999   CNRI        yes
    1.6             1.5.2       2000        CNRI        no
    2.0             1.6         2000        BeOpen.com  no
    1.6.1           1.6         2001        CNRI        yes (2)
    2.1             2.0+1.6.1   2001        PSF         no
    2.0.1           2.0+1.6.1   2001        PSF         yes
    2.1.1           2.1+2.0.1   2001        PSF         yes
    2.2             2.1.1       2001        PSF         yes
    2.1.2           2.1.1       2002        PSF         yes
    2.1.3           2.1.2       2002        PSF         yes
```

```
    2.2.1           2.2         2002        PSF         yes
    2.2.2           2.2.1       2002        PSF         yes
    2.2.3           2.2.2       2003        PSF         yes
    2.3             2.2.2       2002-2003   PSF         yes
    2.3.1           2.3         2002-2003   PSF         yes
    2.3.2           2.3.1       2002-2003   PSF         yes
    2.3.3           2.3.2       2002-2003   PSF         yes
    2.3.4           2.3.3       2004        PSF         yes
    2.3.5           2.3.4       2005        PSF         yes
    2.4             2.3         2004        PSF         yes
    2.4.1           2.4         2005        PSF         yes
    2.4.2           2.4.1       2005        PSF         yes
    2.4.3           2.4.2       2006        PSF         yes
    2.5             2.4         2006        PSF         yes
    2.7             2.6         2010        PSF         yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under
    the GPL.  All Python licenses, unlike the GPL, let you distribute
    a modified version without making your changes open source.  The
    GPL-compatible licenses make it possible to combine Python with
    other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible,
    because its license has a choice of law clause.  According to
    CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1
    is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's
direction to make these releases possible.


B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON
===============================================================

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python
alone or in any derivative version, provided, however, that PSF's
License Agreement and PSF's notice of copyright, i.e., "Copyright (c)
2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation; All Rights
Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
```

```
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.


BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0
-------------------------------------------

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an
office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the
Individual or Organization ("Licensee") accessing and otherwise using
this software in source or binary form and its associated
documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License
Agreement, BeOpen hereby grants Licensee a non-exclusive,
royalty-free, world-wide license to reproduce, analyze, test, perform
and/or display publicly, prepare derivative works, distribute, and
otherwise use the Software alone or in any derivative version,
provided, however, that the BeOpen Python License is retained in the
Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS"
basis.  BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE
SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS
AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY
DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
```

```
5. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all
respects by the law of the State of California, excluding conflict of
law provisions.  Nothing in this License Agreement shall be deemed to
create any relationship of agency, partnership, or joint venture
between BeOpen and Licensee.  This License Agreement does not grant
permission to use BeOpen trademarks or trade names in a trademark
sense to endorse or promote products or services of Licensee, or any
third party.  As an exception, the "BeOpen Python" logos available at
http://www.pythonlabs.com/logos.html may be used according to the
permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.


CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1
---------------------------------------

1. This LICENSE AGREEMENT is between the Corporation for National
Research Initiatives, having an office at 1895 Preston White Drive,
Reston, VA 20191 ("CNRI"), and the Individual or Organization
("Licensee") accessing and otherwise using Python 1.6.1 software in
source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python 1.6.1
alone or in any derivative version, provided, however, that CNRI's
License Agreement and CNRI's notice of copyright, i.e., "Copyright (c)
1995-2001 Corporation for National Research Initiatives; All Rights
Reserved" are retained in Python 1.6.1 alone or in any derivative
version prepared by Licensee.  Alternately, in lieu of CNRI's License
Agreement, Licensee may substitute the following text (omitting the
quotes): "Python 1.6.1 is made available subject to the terms and
conditions in CNRI's License Agreement.  This Agreement together with
Python 1.6.1 may be located on the Internet using the following
unique, persistent identifier (known as a handle): 1895.22/1013.  This
Agreement may also be obtained from a proxy server on the Internet
using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python 1.6.1 or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS"
basis.  CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.
```

```
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. This License Agreement shall be governed by the federal
intellectual property law of the United States, including without
limitation the federal copyright law, and, to the extent such
U.S. federal law does not apply, by the law of the Commonwealth of
Virginia, excluding Virginia's conflict of law provisions.
Notwithstanding the foregoing, with regard to derivative works based
on Python 1.6.1 that incorporate non-separable material that was
previously distributed under the GNU General Public License (GPL), the
law of the Commonwealth of Virginia shall govern this License
Agreement only as to issues arising under or with respect to
Paragraphs 4, 5, and 7 of this License Agreement.  Nothing in this
License Agreement shall be deemed to create any relationship of
agency, partnership, or joint venture between CNRI and Licensee.  This
License Agreement does not grant permission to use CNRI trademarks or
trade name in a trademark sense to endorse or promote products or
services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying,
installing or otherwise using Python 1.6.1, Licensee agrees to be
bound by the terms and conditions of this License Agreement.

        ACCEPT


CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2
--------------------------------------------------

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam,
The Netherlands.  All rights reserved.

Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Stichting Mathematisch
Centrum or CWI not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.


----------------------------------------------------------------------
```

```
----------------------------------------------------------------------

GNU General Public License version 3

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. http://www.fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
the GNU General Public License is intended to guarantee your freedom
to share and change all versions of a program--to make sure it remains
free software for all its users. We, the Free Software Foundation, use
the GNU General Public License for most of our software; it applies
also to any other work released this way by its authors. You can apply
it to your programs, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights. Therefore, you
have certain responsibilities if you distribute copies of the
software, or if you modify it: responsibilities to respect the freedom
of others.

For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received. You must make sure that they, too, receive
or can get the source code. And you must show them these terms so they
know their rights.

Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software. For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the
manufacturer can do so. This is fundamentally incompatible with the
aim of protecting users' freedom to change the software. The
```

```
systematic pattern of such abuse occurs in the area of products for
individuals to use, which is precisely where it is most unacceptable.
Therefore, we have designed this version of the GPL to prohibit the
practice for those products. If such problems arise substantially in
other domains, we stand ready to extend this provision to those
domains in future versions of the GPL, as needed to protect the
freedom of users.

Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish
to avoid the special danger that patents applied to a free program
could make it effectively proprietary. To prevent this, the GPL
assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and
modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds
of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this
License. Each licensee is addressed as “you”. “Licensees” and
“recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of
an exact copy. The resulting work is called a “modified version” of
the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based
on the Program.

To “propagate” a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy. Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other
parties to make or receive copies. Mere interaction with a user
through a computer network, with no transfer of a copy, is not
conveying.

An interactive user interface displays “Appropriate Legal Notices” to
the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License. If
the interface presents a list of user commands or options, such as a
```

```
menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for
making modifications to it. “Object code” means any non-source form of
a work.

A “Standard Interface” means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form. A
“Major Component”, in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities. However, it does not include the work’s
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work. For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

The Corresponding Source need not include anything that users can
regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same
work.

2. Basic Permissions.

All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met. This License explicitly affirms your unlimited
permission to run the unmodified Program. The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work. This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey,
without conditions so long as your license otherwise remains in force.
You may convey covered works to others for the sole purpose of having
them make modifications exclusively for you, or provide you with
facilities for running those works, provided that you comply with the
```

```
terms of this License in conveying all material for which you do not
control copyright. Those thus making or running the covered works for
you must do so exclusively on your behalf, under your direction and
control, on terms that prohibit them from making any copies of your
copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the
conditions stated below. Sublicensing is not allowed; section 10 makes
it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such
circumvention is effected by exercising rights under this License with
respect to the covered work, and you disclaim any intention to limit
operation or modification of the work as a means of enforcing, against
the work's users, your or third parties' legal rights to forbid
circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any non-
permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these
conditions:

The work must carry prominent notices stating that you modified it,
and giving a relevant date.

The work must carry prominent notices stating that it is released
under this License and any conditions added under section 7. This
requirement modifies the requirement in section 4 to "keep intact all
notices".

You must license the entire work, as a whole, under this License to
anyone who comes into possession of a copy. This License will
therefore apply, along with any applicable section 7 additional terms,
to the whole of the work, and all its parts, regardless of how they
```

```
are packaged. This License gives no permission to license the work in
any other way, but it does not invalidate such permission if you have
separately received it.

If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your work
need not make them do so.

A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
“aggregate” if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation’s users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of
sections 4 and 5, provided that you also convey the machine-readable
Corresponding Source under the terms of this License, in one of these
ways:

Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium customarily
used for software interchange.

Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a written
offer, valid for at least three years and valid for as long as you
offer spare parts or customer support for that product model, to give
anyone who possesses the object code either (1) a copy of the
Corresponding Source for all the software in the product that is
covered by this License, on a durable physical medium customarily used
for software interchange, for a price no more than your reasonable
cost of physically performing this conveying of source, or (2) access
to copy the Corresponding Source from a network server at no charge.

Convey individual copies of the object code with a copy of the written
offer to provide the Corresponding Source. This alternative is allowed
only occasionally and noncommercially, and only if you received the
object code with such an offer, in accord with subsection 6b.

Convey the object code by offering access from a designated place
(gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge. You need not require recipients to copy the
Corresponding Source along with the object code. If the place to copy
the object code is a network server, the Corresponding Source may be
on a different server (operated by you or a third party) that supports
equivalent copying facilities, provided you maintain clear directions
next to the object code saying where to find the Corresponding Source.
Regardless of what server hosts the Corresponding Source, you remain
obligated to ensure that it is available for as long as needed to
```

```
satisfy these requirements.

Convey the object code using peer-to-peer transmission, provided you
inform other peers where the object code and Corresponding Source of
the work are being offered to the general public at no charge under
subsection 6d.

A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any
tangible personal property which is normally used for personal,
family, or household purposes, or (2) anything designed or sold for
incorporation into a dwelling. In determining whether a product is a
consumer product, doubtful cases shall be resolved in favor of
coverage. For a particular product received by a particular user,
“normally used” refers to a typical or common use of that class of
product, regardless of the status of the particular user or of the way
in which the particular user actually uses, or expects or is expected
to use, the product. A product is a consumer product regardless of
whether the product has substantial commercial, industrial or non-
consumer uses, unless such uses represent the only significant mode of
use of the product.

“Installation Information” for a User Product means any methods,
procedures, authorization keys, or other information required to
install and execute modified versions of a covered work in that User
Product from a modified version of its Corresponding Source. The
information must suffice to ensure that the continued functioning of
the modified object code is in no case prevented or interfered with
solely because modification has been made.

If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information. But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or
updates for a work that has been modified or installed by the
recipient, or for the User Product in which it has been modified or
installed. Access to a network may be denied when the modification
itself materially and adversely affects the operation of the network
or violates the rules and protocols for communication across the
network.

Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.
```

```
7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law. If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it. (Additional permissions may be written to require their own
removal in certain cases when you modify the work.) You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders
of that material) supplement the terms of this License with terms:

Disclaiming warranty or limiting liability differently from the terms
of sections 15 and 16 of this License; or

Requiring preservation of specified reasonable legal notices or author
attributions in that material or in the Appropriate Legal Notices
displayed by works containing it; or

Prohibiting misrepresentation of the origin of that material, or
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

Limiting the use for publicity purposes of names of licensors or
authors of the material; or

Declining to grant rights under trademark law for use of some trade
names, trademarks, or service marks; or

Requiring indemnification of licensors and authors of that material by
anyone who conveys the material (or modified versions of it) with
contractual assumptions of liability to the recipient, for any
liability that these contractual assumptions directly impose on those
licensors and authors.

All other non-permissive additional terms are considered “further
restrictions” within the meaning of section 10. If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term. If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
```

```
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions; the
above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

However, if you cease all violation of this License, then your license
from a particular copyright holder is reinstated (a) provisionally,
unless and until the copyright holder explicitly and finally
terminates your license, and (b) permanently, if the copyright holder
fails to notify you of the violation by some reasonable means prior to
60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License. If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run
a copy of the Program. Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance. However,
nothing other than this License grants you permission to propagate or
modify any covered work. These actions infringe copyright if you do
not accept this License. Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License. You are not responsible
for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations. If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
```

```
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License. For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based. The
work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned
or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version. For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement). To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients. "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
```

```
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

A patent license is “discriminatory” if it does not include within the
scope of its coverage, prohibits the exercise of, or is conditioned on
the non-exercise of one or more of the rights that are specifically
granted under this License. You may not convey a covered work if you
are a party to an arrangement with a third party that is in the
business of distributing software, under which you make payment to the
third party based on the extent of your activity of conveying the
work, and under which the third party grants, to any of the parties
who would receive the covered work from you, a discriminatory patent
license (a) in connection with copies of the covered work conveyed by
you (or copies made from those copies), or (b) primarily for and in
connection with specific products or compilations that contain the
covered work, unless you entered into that arrangement, or that patent
license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under
this License and any other pertinent obligations, then as a
consequence you may not convey it at all. For example, if you agree to
terms that obligate you to collect a royalty for further conveying
from those to whom you convey the Program, the only way you could
satisfy both those terms and this License would be to refrain entirely
from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work. The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions
of the GNU General Public License from time to time. Such new versions
will be similar in spirit to the present version, but may differ in
detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program
specifies that a certain numbered version of the GNU General Public
License “or any later version” applies to it, you have the option of
```

```
following the terms and conditions either of that numbered version or
of any later version published by the Free Software Foundation. If the
Program does not specify a version number of the GNU General Public
License, you may choose any version ever published by the Free
Software Foundation.

If the Program specifies that a proxy can decide which future versions
of the GNU General Public License can be used, that proxy's public
statement of acceptance of a version permanently authorizes you to
choose that version for the Program.

Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND
PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE
DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR
CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR
CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT
NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR
LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM
TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER
PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.


----------------------------------------------------------------------
----------------------------------------------------------------------


2 CLAUSE BSD-LIKE LICENSE

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

```
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long
 * as the following conditions are aheared to.  The following
 * conditions apply to all code found in this distribution, be it the
 * RC4, RSA, lhash, DES, etc., code; not just the SSL code.  The SSL
 * documentation included with this distribution is covered by the
 * same copyright terms except that the holder is Tim Hudson
 * (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices
 * in the code are not to be removed.
 * If this package is used in a product, Eric Young should be given
 * attribution as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above
 *    copyright notice, this list of conditions and the following
 *    disclaimer in the documentation and/or other materials provided
 *    with the distribution.
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the
 *    library being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative
 *    thereof) from the apps directory (application code) you must
 *    include an acknowledgement:
 *    "This product includes software written by Tim Hudson
```

```
  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program.  We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder.  Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License.  It also provides other free software developers Less
of an advantage over competing non-free programs.  These disadvantages
are the reason we use the ordinary General Public License for many
libraries.  However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it
becomes a de-facto standard.  To achieve this, non-free programs must
```

```
be allowed to use the library.  A more frequent case is that a free
library does the same job as widely used non-free libraries.  In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

  In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software.  For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.


                  GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.
```

```
  1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.
   2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
```

```
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library.  To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License.  (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.)  Do not make any other change in
these notices.
   Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library".  Such a
work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library".  The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

  When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library.  The
threshold for this to be true is not precisely defined by law.

  If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work.  (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

  Otherwise, if the work is a derivative of the Library, you may
```

```
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.
   6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

  You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License.  You must supply a copy of this License.  If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License.  Also, you must do one
of these things:

    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library.  (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application
    to use the modified definitions.)

    b) Use a suitable shared library mechanism for linking with the
    Library.  A suitable mechanism is one that (1) uses at run time a
    copy of the library already present on the user's computer system,
    rather than copying library functions into the executable, and (2)
    will operate properly with a modified version of the library, if
    the user installs one, as long as the modified version is
    interface-compatible with the version that the work was made with.

    c) Accompany the work with a written offer, valid for at
    least three years, to give the same user the materials
    specified in Subsection 6a, above, for a charge no more
    than the cost of performing this distribution.

    d) If distribution of the work is made by offering access to copy
    from a designated place, offer equivalent access to copy the above
    specified materials from the same place.

    e) Verify that the user has already received a copy of these
    materials or that you have already sent this user a copy.

  For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it.  However, as a special exception,
the materials to be distributed need not include anything that is
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.
```

```
  It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system.  Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.
   7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

    a) Accompany the combined library with a copy of the same work
    based on the Library, uncombined with any other library
    facilities.  This must be distributed under the terms of the
    Sections above.

    b) Give prominent notice with the combined library of the fact
    that part of it is a work based on the Library, and explaining
    where to find the accompanying uncombined form of the same work.

  8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License.  Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License.  However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with
this License.
   11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all.  For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.
```

```
If any portion of this section is held invalid or unenforceable under
Any particular circumstance, the balance of the section is intended to
apply, and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number.  If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation.  If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.
   14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission.  For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this.  Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

                            NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
```

```
LIBRARY IS WITH YOU.  SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

                     END OF TERMS AND CONDITIONS

----------------------------------------------------------------------
----------------------------------------------------------------------

Zope Public License (ZPL) Version 2.1

A copyright notice accompanies this license document that
identifies the copyright holders.

This license has been certified as open source. It has also been
designated as GPL compatible by the Free Software Foundation
(FSF).

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

1. Redistributions in source code must retain the accompanying
copyright notice, this list of conditions, and the following
disclaimer.

2. Redistributions in binary form must reproduce the
accompanying copyright notice, this list of conditions, and the
following disclaimer in the documentation and/or other materials
provided with the distribution.

3. Names of the copyright holders must not be used to endorse or
promote products derived from this software without prior
written permission from the copyright holders.

4. The right to distribute this software or to use it for any
purpose does not give you the right to use Servicemarks (sm) or
Trademarks (tm) of the copyright holders. Use of them is covered
by separate agreement with the copyright holders.

5. If any files are modified, you must cause the modified files
to carry prominent notices stating that you changed the files
and the date of any change.

Disclaimer

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND
ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
```