# Purpose

Provide a consistent messaging protocol across all of Wisconsin Robotics’ computing solutions, from fully-fledged computers running Windows or Linux to microcontrollers with no operating system or MMU.

*Note that this specification is intentionally left vague. Implementers* ***must*** *come up with a specification for each and every robot using this specification as a starting point.*

# Packets

A packet encapsulates a single command, and its structure can be seen below.

All metadata should be treated as unsigned. Data should be transmitted in network order (big endian).

|  |  |  |
| --- | --- | --- |
| **Field** | **Length** | **Description** |
| BCL Start | 2 bytes | Start bytes, values must be 0xBA 0xAD in that order. |
| Opcode | 1 byte | Defines the actual packet purpose. |
| Source Robot ID | 1 byte | The ID of the robot from which the packet originates. |
| Source Service | 1 byte | The ID of the service from the packet originates. |
| Destination Robot ID | 1 byte | The ID of the destination robot. 0xFF denotes a broadcast to all robots. |
| Destination Service | 1 byte | The ID of the destination service. 0xFF denotes a broadcast to all active services. |
| Payload Size | 1 byte | Payload size. |
| CRC | 1 byte | The packet data is located here. |
| Payload | 0 ≤ n ≤ 246 bytes | CRC-8 checksum computed over the entire packet from the BCL Start byte through the payload. |
| BCL End | 1 byte | End byte, value must be 0xFE. |

## Optional Field Presence

Some packets may require marking certain defined fields as “valid” or “invalid” depending on circumstances. To implement this, the first field must be a presence bitset, where each bit indicates if the field is valid or not.

# Services

A BCL service is able to receive, parse, and execute packets periodically or on a event-driven basis. Each service is identified by its service ID. Each service shall be into two states: active and inactive. A service cannot be run if it is inactive.

Each service must define:

* How often to run, in milliseconds. The value 0 can be used to mark that the service is to be only run when a packet is available for that service, provided that it is marked as active.
* What to run when the service is selected. This periodically returning code can optionally return a response packet to send.
* A packet handler which runs regardless of whether the service is active or inactive.

# Service Master

When a packet is sent, it is then received by the service master. The service master is responsible for registering, initializing, managing, and scheduling each service on the robot. It will receive all incoming packets and route them to the appropriate service. Additionally, should the service return a response packet, the service master must be able to send that response packet over any interface, be it serial or network. The service master itself is a service, and handles all messages with category code 0 (see section “Defined Categories, Services, and Packets”).

# Defined Categories, Services, and Packets

Below are specification-defined categories, services, and packets. Each robot must implement them. However, additional categories, services, and packets can and should be created to fit the design of each robot.

## Service Management & Access Control

### Packets

For all Service Management & Access Control packets, the destination service ID is ignored.

#### Activate Service [Opcode 0xAC]

Activates the specified service with the given category and service ID.

Payload: *None*

Response: A ReportServiceStatus packet containing the updated status of the specified service.

#### Deactivate Service [Opcode 0xDC]

Deactivates the specified service with the given category and service ID.

Payload: *None*

Response: A ReportServiceStatus packet containing the updated status of the specified service.

#### Query Service List [Opcode 0x70]

Queries the robot for a list of all services and a friendly name for each service.

Payload: *None*

Response: A ReportServiceList packet.

#### Query Service Status [Opcode 0x73]

Queries the robot of a service's status.

Payload: *None*

Response: A ReportServiceStatus packet.

#### Query Heartbeat [Opcode 0x90]

Checks if the robot is still alive. In other words, the "Marco" part to "Marco Polo".

Payload: *None*

Response: A ReportHeartbeat packet.

#### Report Service List [Opcode 0x71]

Contains a list of all services present on the robot, with friendly names.

Payload

* Number of services (1 byte)
* Service 1
  + Service ID (1 byte)
  + Status [0 for inactive, 1 for active] (1 byte)
  + Friendly name (10 bytes)
* Service 2 …

#### Report Heartbeat [Opcode 0x91]

Sent by the robot when it is still connected to the host. The "Polo" to "Marco Polo".

Payload: *None*

#### Report Service Status [Opcode 0x74]

Sends the status of the specified service in the QueryServiceStatus packet.

Payload:

* Status (0 for inactive, 1 for active) (1 byte)

Direct Drive (Service ID 0) & Remote Drive (Service ID 1) are required services.