Modular Information Display and Control Protocol (MIDaC Protocol)

MIDaC Protocol is an open standard for data exchange between a client device[1] and a robotics system over TCP/IP that is based on the JSON format for data serialization.

The MIDaC Protocol is part of the MIDaC toolkit which also includes proposals for MIDaC Protocol Client and Server implementations as well as a standard implementation known as MissionControl. The MIDaC toolkit is designed so that any part can be replaced with a third-party implementation as long as they comply with the requirements that are defined within the MIDaC toolkit.

The protocol includes two message types for data exchange (data messages and control messages) and two defined handshakes (one for connecting a client to the server and one for connecting a multi-platform server with a platform specific part for data readout and control).

Client-Server-Handshake

After establishing a TCP/IP socket connection to the server a client initiates a handshake consisting of 4 parts:

1. Client -> Server: ConnREQ (Connection Request containing information about the client device)
2. Server -> Client: ConnACK or ConnREJ (Connection Acknowledgement or Connection Rejection)
3. Server -> Client: ConnLAO (A message including a description of all data properties and controls that are available)
4. Client -> Server: ConnSTT (Acknowledgment for receiving ConnLAO)

These messages are, as well as any other message, JSON formatted. The ConnACK and ConnLAO messages are split because there is theoretical support for encryption in the MIDaC protocol and if used every message after the ConnACK has to be encrypted.

If successful the server starts to periodically transmit ‚Information Messages' containing the current values for every given data property. The client can send based on user input ‚Control Messages' which contain commands for described actuators that are part of the robotics system.

Server-RSAL-Handshake

The Robotics System Abstraction Layer (RSAL) is part of the MIDaC toolkit and represents an abstraction layer that makes it possible for a MIDaC server to be multi-platform. In it all the control and data-inquiry routines are defined and wrapped so that the routines itself are exchangeable for every system. The RSAL is connected to the server over TCP/IP sockets.

When the RSAL is connecting to the server a different handshake that consist out of 3 parts is used:

1. RSAL -> Server: B-Connect (A message containing an identifier that the server passed to the RSAL while calling it)

2. Server -> RSAL: ConnACK or ConnREJ (Same as with the Client-Server handshake but empty)
3. RSAL -> Server: ConnLAO (The ConnLAO message which is passed to every connecting client)

Property data types

A defined set of data types is used within MIDaC Information Messages:
* Bool - Graph[Integer, Optional]
* Integer - Maximum[Integer], Minimum[Integer], Graph[Integer, Optional]
* Float - Maximum[Float], Minimum[Float], Graph[Integer, Optional]
* String - Maximum length[Integer]

These data types feature unique properties which are transmitted with the ConnLAO message that makes it easier to enhance them for presentation.

Controls

There are two types of controls supported by MIDaC, each with unique properties transmitted with the ConnLAO message:
* Sliders - Maximum[Integer], Minimum[Integer], AutoUpdate[String representation of bool]
* Buttons - Descriptor[String]

Accessing properties and controls

Within MIDaC messages properties and controls are accessed via unique names. These names can also be used to display them on client devices. The names are specified within the controller specific part and are up to the implementing party. The names are used as identifiers in the information and control messages.

Handshake messages

A listing of all the properties of the handshake messages is now given. When encryption is used all messages from and including the ConnLAO message are encrypted.

ConnREQ: HardwareType (String)[Type of hardware, can be: Web, PC or Smartphone], SupportedCrypto (String Array)[List of supported asymmetrical encryptions], PreferredCrypto (String)[the preferred encryption], SupportedDT (String Array)[List of supported data types and controls], PublicKeys(String Array)[Publik keys for used encryption]

ConnACK: SegmentSize (Integer)[Size of the segments used by the sockets], ChosenCrypto (String)[the chosen encryption out of the ones supported by the client. None if empty string], PublicKey (String)[Public key for used encryption]

ConnREJ: Error (String)[Error message]

ConnLAO: Information (JSON Objects)[Containing all the properties that are transmitted to the clients and their description], Controller (JSON Objects)[Containing all the control types and their descriptions]

ConnSTT: This message just includes an empty object.

B-Connect: Identifier (String)[identifier given as parameter when called]

Information messages

Information messages are the messages that are used to perform the monitoring. They transmit properties like sensor values, status messages, etc.

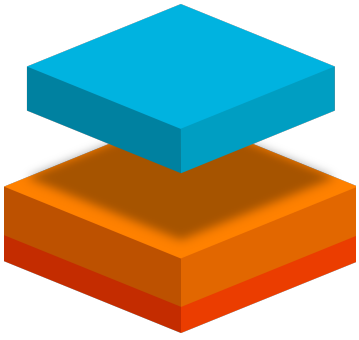
Control messages

Control messages are the messages that are used, if supported, to perform commands on the controller. They contain a value for a property that is defined in the ConnLAO message. A processing routine for the command has to be implemented on the controller side. While for sliders the value is sent, **for buttons it's just a string ‚click'**.
When a slider is set to AutoUpdate for every interaction the updated value has to be sent.

ConnLAO message layout

The ConnLAO message consists of two parts, the definition of the data properties („Information") and the definition of the controls („Controller").
Each data property has a unique identifier, is of a certain type and can have attributes specific to its datatype (see „Property data types"). They must be grouped by their data type.
For any occurring data type an object containing the descriptors of all fields of that type exists.
Controls can be grouped into control groups for simplified layouting on the client side. A group consists of a button and a slider. The group is accessed by a unique name. In it the controls have to have globally unique identifiers. Controls cannot be grouped by their type.
The button descriptor contains the string displayed on the button.

[1] Might be any device running a MIDaC Protocol Client Application



Example Messages

```
{
	"ConnREQ" : {
		"HardwareType" : "Smartphone",
		"SupportedCrypto" : [
			"RSA512"
		],
		"PreferredCrypto" : "None",
		"SupportedDT" : [
			"Bool", "String", "Integer", "Slider", "Button"
		],
		"PublicKeys" : [
			"key"
		]
	}
}
```

```
{
	"ConnACK" : {
		"SegmentSize" : 2048
		"ChosenCrypto" : ""
	}
}
```

```
{
	"ConnREJ" : {
		"Error" : "Some error because reasons"
	}
}
```

```
{
	"Data" : {
		"SomeFloatNumber" : 512.13,
		"SomeIntegerNumber" : 12,
		"SomeBool" : "true";
		"SomeStatusString" : "Some String"
	}
}
```

```
{
	"Control" : {
		"SomeSlider" : 76,
		"SomeButton" : "click"
	}
}
```

```
{
	"B-Connect" : {
		"Identifier" : "thisIsTheIdentifier"
	}
}
```

```
{
	"ConnLAO" : {
		"Information“ : {
			"Float" : {
				"SomeFloatNumber" : {
					"MinBound" : -1023.9999999,
					"MaxBound" : 1023.9999999,
					"Graph"  : 20
				}

			},
			"Integer" : {
				"SomeIntegerNumber" : {
					"MinBound" : 0,
					"MaxBound" : 1023
				}
			}
		},
		"Controller" : {
			"SomeGroup" : {
				"SomeSlider" : {
					"ControlType" : "Slider",
					"MinBound" : 0,
					"MaxBound" : 100,
					"AutoUpdate" : "true"
				},
				"SomeButton" : {
					"ControlType" : "Button",
					"Descriptor" : "Set"
				}
			},
			"SomeOtherButton" : {
				"ControlType" : "Button",
				"Descriptor" : "Switch"
			}
		}
	}
}
```