API
Startup Manual

Table of Contents

Need Additional Technical Support?

Need additional technical support for issues or questions about the Connect Wireless ecosystem?

Scan the QR code or use the hyperlink to access our convenient web form to submit your request online at any time.

CTC's experienced support team will review your inquiry and work quickly to resolve your issues.

scan QR code or

CLICK HERE FOR SUPPORT REQUEST FORM

Summary

The Connection Technology Center wireless sensor solution communicates using websocket technology. This creates synchronous communication between layers and allows real-time bidirectional transfer of commands. A few of these commands are publicly available and documented for custom integration purposes like user interfaces or automation solutions. The system operates using a series of commands documented below.

The commands can be broken up into three different sections: Send, Return, and Notify. Along with this document, a public Postman project with sample commands is available online to help demonstrate these commands. The setup for this project is documented in the following section.

Postman Setup

The easiest way to test system functionality is through a third-party tool called Postman. This is a free powerful API platform to help create, test, and document APIs. CTC has a public project available to fork that has prepopulated examples of all commands currently available in the system. Use the following instructions to install Postman, fork CTC's project, and send/receive commands from an Access Point.

Download and Fork:

1. Download and Install the Postman desktop application here
2. Open Postman and create an account or sign in if you already have an account
3. Begin fork of the project here
4. Name and save fork to workspace as desired
5. Return to the Postman desktop app and select the newly forked project in the collections panel

Send / Receive Commands:

1. Click the “Variables” tab
2. Modify the “Current value” column to match your devices and needs
3. In the Collections tab expand the collection “CTC WebSocket Test Suite”
4. Select “Websocket”
5. Press the blue Connect button in the top right to connect to Access Point
6. Select messages on the right-hand side a press “Send” to send commands
7. Responses to commands and notifications (when subscribed) will appear in the “Response” panel at the bottom

Send Commands

The Send Commands are externally triggerable commands that will trigger an operation within the system. These are the primary ways to interact with the system as a system integrator. The execution of these commands can result in a Return Command or a Notify Command which are documented below. All Send Commands can receive an RTN_ERR message if a problem occurs that identifies the issue.

Subscribe to Changes	

This command should be run after connecting to the access point’s websocket. It will mark this connection as wanting to be updated on new readings and events.
 	
Returns: RTN_ERR
Notifies: NA
	<pre>{
	  “Type”: “POST_SUB_CHANGES”,
	  “From”: “UI”,
	  “To”: “SERV”,
	  “SessionID”: “str”,
	  “Data”: {
	  }
	}</pre>

Unsubscribe to Changes	

This command will mark this connection as not wanting to be updated on new readings and events.
 	
Returns: RTN_ERR
Notifies: NA
	<pre>{
	  “Type”: “POST_UNSUB_CHANGES”,
	  “From”: “UI”,
	  “To”: “SERV”,
	  “SessionID”: “str”,
	  “Data”: {
	  }
	}</pre>

Get Dynamic Sensors

This command will retrieve information on all serial numbers contained in the “Serials” array. An empty array will return all known sensors that have ever been connected to the system.

Returns: RTN_DYN, RTN_ERR
Notifies: NA

```
{
    "Type": "GET_DYN",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serials": [int,]
    }
}
```

Get Connected Dynamic Sensors

This command will retrieve information on all serial numbers currently connected to the system.

Returns: RTN_DYN, RTN_ERR
Notifies: NA

```
{
    "Type": "GET_DYN_CONNECTED",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
    }
}
```

Take Dynamic Vibration Reading

This command will trigger a vibration and temperature reading on the sensor with the given serial number. Returns an error if a test was unable to be triggered. Notifies subscribed clients when the test starts and when the test ends.

Returns: RTN_ERR
Notifies: NOT_DYN_READING_STARTED, NOT_DYN_READING, NOT_DYN_TEMP

```
{
    "Type": "TAKE_DYN_READING",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serial": int
    }
}
```

Take Dynamic Temperature Reading

This command will trigger a temperature reading on the sensor with the given serial number. Returns an error if a test was unable to be triggered. Notifies subscribed clients when the test ends.

Returns: RTN_ERR
Notifies: NOT_DYN_TEMP

```
{
    "Type": "TAKE_DYN_TEMP",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serial": int
    }
}
```

Take Dynamic Battery Level Reading

This command will trigger a battery level reading on the sensor with the given serial number. Returns an error if a test was unable to be triggered. Notifies subscribed clients when the test ends.

Returns: RTN_ERR
Notifies: NOT_DYN_BATT

```
{
    "Type": "TAKE_DYN_BATT",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serial": int
    }
}
```

Get Dynamic Vibration Records

This command will retrieve vibration reading records for given serials in the "Serials" list between the "Start" and "End" dates with a limit set by "Max". All parameters are optional. "Serials" defaults to all serial numbers. "Start" and "End" defaults to the most recent. "Max" defaults to 25.

Returns: RTN_DYN_READINGS, RTN_ERR
Notifies: NA

```
{
    "Type": "GET_DYN_READINGS",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serials": [int,],
        "Start": "yyyy-mm-dd",
        "End": "yyyy-mm-dd"
        "Max": int
    }
}
```

Get Dynamic Temperature Records

This command will retrieve temperature reading records for given serials in the "Serials" list between the "Start" and "End" dates with a limit set by "Max". All parameters are optional. "Serials" defaults to all serial numbers. "Start" and "End" defaults to the most recent. "Max" defaults to 25.

Returns: RTN_DYN_TEMPS, RTN_ERR
Notifies: NA

```
{
    "Type": "GET_DYN_TEMPS",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serials": [int,],
        "Start": "yyyy-mm-dd",
        "End": "yyyy-mm-dd",
        "Max": int
    }
}
```

Get Dynamic Battery Records

This command will retrieve battery-level reading records for given serials in the "Serials" list between the "Start" and "End" dates with a limit set by "Max". All parameters are optional. "Serials" defaults to all serial numbers. "Start" and "End" defaults to the most recent. "Max" defaults to 25.

Returns: RTN_DYN_BATTS, RTN_ERR
Notifies: NA

```
{
    "Type": "GET_DYN_BATTS",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serials": [int,],
        "Start": "yyyy-mm-dd",
        "End": "yyyy-mm-dd",
        "Max": int
    }
}
```

POST_LOGIN

This command logs a user in and gets their access level. This command must be done before send POST_SUB_CHANGES.

Returns: RTN_LOGIN
Notifies: NA

```
{
    "Type": "POST_LOGIN",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Email": "str",
        "Password": "str"
}
```

POST_LOGOUT

This command logs the current user out of the system.

Returns: RTN_LOGOUT
Notifies: NA

```
{
    "Type": "POST_LOGOUT",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {}
}
```

POST_DYN_WRITE_UPDATE

This command changes the sensors programmed configuration. This config will be retained on power cycle.

Returns: NA
Notifies: NOT_DYN_CONFIG_UPDATE

```
{
    "Type": "POST_DYN_WRITE_UPDATE",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serial": "int",
        "ReadInterval": "int",
        "GMode":"str - [+/-8g, +/-16g, +/-32g, +/-64g]",
        "FreqMode":"int - [400, 800, 1600, 3200, 6400, 12800, 25600]",
        "Coupling": "bool",
        "Samples": "int - [1600, 3200, 6400, 12800, 25600]"
    }
}
```

POST_DYN_CONFIG

This command changes non-programmed configuration of a sensor. This data is stored in the gateway.

Returns: RTN_DYN
Notifies: NA

```
{
    "Type": "POST_DYN_CONFIG"",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Serial": "int",
        "Favorite": "bool",
        "Nickname": "str",
        "Crit": "float",
        "CritUnit": "str [rms, p, pp]",
        "Early": "float",
        "EarlyUnit": "str [rms, p, pp]",
        "Machine": "int",
    }
}
```

Return Commands

The Return Commands are returned to a client after a Send Command is triggered. These typically carry requested data or in the case of an error an error message. These commands only occur in response to a Send Command and only go to the connected client that sent the Send Command.

Return Dynamic Vibration Records

This command returns a list of data objects representing a vibration reading. Ordered by the most recent "Time".

Triggered: GET_DYN_READINGS

```
{
    "Type": "RTN_DYN_READINGS",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "0": {
            "ID": int,
            "Serial": str,
            "Time": "yyyy-mm-dd",
            "X": str,
            "Y": str,
            "Z": str
        },
        "1": ...
    }
}
```

Return Dynamic Temperature Records

This command returns a list of data objects representing a temperature reading. Ordered by the most recent "Time".

Triggered: GET_DYN_TEMPS

```
{
    "Type": "RTN_DYN_TEMPS",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "0": {
            "ID": int,
            "Serial": str,
            "Time": "yyyy-mm-dd",
            "Temp": int
        },
        "1": ...
    }
}
```

Return Dynamic Battery Level Records

This command returns a list of data objects representing a battery-level reading. Ordered by most recent “Time”.

Triggered: GET_DYN_BATTS

```
{
    "Type": "RTN_DYN_BATTS",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "0": {
            "ID": int,
            "Serial": str,
            "Time": "yyyy-mm-dd",
            "Batt": int,
        },
        "1": ...
    }
}
```

Return Dynamic Sensors

This command returns a list of data objects representing dynamic sensors with a key of the dynamic sensor’s serial number.

Triggered: GET_DYN, GET_DYN_CONNECTED

```
{
    "Type": "RTN_DYN",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "{Device Serial}": {
            "Serial": int,
            "Connected": int,
            "AccessPoint": int,
            "PartNum": str,
            "ReadRate": int,
            "GMode": str,
            "FreqMode": str,
            "ReadPeriod": int,
            "Samples": int,
            "HwVer": str,
            "FmVer": str
        }
    }
}
```

Return Error

This command returns a data object representing an error that occurred. “Attempt” identifies what command experienced the error and “Error” provides details on what failed.

Triggered: ALL

```
{
    "Type": "RTN_ERR",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "Attempt": str
        "Error": str
     }
}
```

RTN_LOGIN

This command is the response to a login. A session ID is passed back and needs to be used in all following commands.

Triggered: POST_LOGIN

```
{
    "Type": "RTN_LOGIN",
    "From": "UI",
    "To": "SERV",
    "SessionID": "str",
    "Data": {
        "Email": "str",
        "First": "str",
        "Last": "str",
        "Success": "bool",
        "AccessLevel": "int",
        "Verified": "bool",
        "SessionID": "str"
     }
}
```

RTN_LOGOUT

This command identifies a successful logout.

Triggered: POST_LOGOUT

```
{
    "Type": "RTN_LOGOUT",
    "From": "SERV",
    "Target": "UI",
    "SessionID": "str",
    "Data": {}
}
```

Notify Commands

The Notify Commands notify all connected clients currently subscribed to changes that occurred within the system. When a client connects and sends a "Subscribe to Changes" command it will receive these commands. A client stops receiving these commands after it sends a "Unsubscribe to Changes" command. Notify Commands can be triggered as a response to Send Commands or automatically by internal events happening within the system.

Notify Access Point Connected	

This command notifies all subscribed clients when a new access point is connected or disconnected. With the connection status being represented with a 1 or 0.
 	
Triggered: INTERNAL
	<pre>{
	"Type": "NOT_AP_CONN",
	"From": "SERV",
	"Target": "UI",
	"Data": {
	"Serial": int,
	"Connected": int
	}
	}</pre>

Notify Dynamic Sensor Connected	

This command notifies all subscribed clients when a new dynamic sensor is connected or disconnected. With the connection status being represented with a 1 or 0.
 	
Triggered: INTERNAL
	<pre>{
	"Type": "NOT_DYN_CONN",
	"From": "SERV",
	"Target": "UI",
	"Data": {
	"Serial": int,
	"Connected": int,
	"AccessPoint": int,
	"PartNum": str,
	"ReadRate": int,
	"GMode": str,
	"FreqMode": str,
	"ReadPeriod": int,
	"Samples": int,
	"HwVer": str,
	"FmVer": str
	}
	}</pre>

Notify Vibration Reading Started

This command notifies all subscribed clients when a vibration reading triggered by an external command has started.

Triggered: TAKE_DYN_READING

```
{
    "Type": "NOT_DYN_READING_STARTED",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "Serial": int,
        "Success": bool
    }
}
```

Notify Vibration Reading Complete

This command notifies all subscribed clients when a vibration reading has been completed. Can be triggered by an external command or by the sensor's configured read interval.

Triggered: TAKE_DYN_READING, INTERNAL

```
{
    "Type": "NOT_DYN_READING",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "ID": int,
        "Serial": str,
        "Time": "yyyy-mm-dd",
        "X": str,
        "Y": str,
        "Z": str
    }
}
```

Notify Temperature Reading Complete

This command notifies all subscribed clients when a temperature reading has been completed. Can be triggered by an external command or by the sensor's configured read interval.

Triggered: TAKE_DYN_TEMP, INTERNAL

```
{
    "Type": "NOT_DYN_TEMP",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "ID": int,
        "Serial": str,
        "Time": "yyyy-mm-dd",
        "Temp": int
    }
}
```

Notify Battery Level Reading Complete

This command notifies all subscribed clients when a battery-level reading has been completed. Can be triggered by an external command or by the sensor when it reaches a low battery state.

Triggered: TAKE_DYN_BATT, INTERNAL

```
{
    "Type": "NOT_DYN_BATT",
    "From": "SERV",
    "Target": "UI",
    "Data": {
        "ID": int,
        "Serial": str,
        "Time": "yyyy-mm-dd",
        "Batt": int
     }
}
```

NOT_DYN_CONFIG_UPDATE

This command notifies all subscribed clients a sensor programmed config changed.

Triggered: POST_DYN_WRITE_UPDATE

```
{
    "Type": "NOT_DYN_CONFIG_UPDATE",
    "From": "SERV",
    "Target": "UI",
    "SessionID": "str",
    "Data": {
        "Serial": "int",
        "ReadInterval": "int",
        "GMode":"str - [+/-8g, +/-16g, +/-32g, +/-64g]",
        "FreqMode":"int - [400, 800, 1600, 3200, 6400, 12800, 25600]",
        "Coupling": "bool",
        "Samples": "int - [1600, 3200, 6400, 12800, 25600]",
        "ReadPeriod": "int",
        "Fs": "int",
        "Fmax": "int"
    }
}
```