

---

# REDSCAN mini Web API Tutorials (For Basic Integration) Ver 1.1.0

## 1. Introduction

In this document, we will explain how to receive alarm information from REDSCAN mini via HTTP (HTTPS). Using this function, software such as VMS can integrate such as starting recording by alarm information.

If you want to know details about commands, please see the “REDSCAN mini Web API Command List”.

### 1.1. Compatible firmware version

- Version 2.1.0 or later.

### 1.2. Setting value etc.

In this document, unless otherwise noted, it is assumed that REDSCAN mini is set under the following conditions.

- The protocol is HTTP  
To use HTTPS, you need to change the unit setting. Also, the URLs are changed as follows.
  - http:// → https://
  - ws:// → wss://
- Assume that the IP address of REDSCAN mini is "192.168.0.126".
- For ease of explanation, the examples are shown with HTML and JavaScript, but as long as HTTP communication including WebSocket connection can be made, it can be used from any programming language.
  - Sample of WebSocket using "C #".  
[https://www.example-code.com/csharp/websocket\\_connect.asp](https://www.example-code.com/csharp/websocket_connect.asp)

## 2. Tutorial

This chapter shows how to create a simple program to receive alarm information from REDSCAN mini. There are two ways to receive alarms: "basic integration" and "advanced integration".

With "Basic Integration" you can easily receive detected alarm information by event code. This method

is suitable for integration such as starting recording by triggering an alarm. Also, in this mode, it is possible to receive abnormal state such as tampering, rotation, etc.

"Advanced Integration" is suitable for advanced integration such as receiving detailed information of the detected object including position. You can overlay them on the map.

## 2.1. Tutorial of “Basic Integration”

This section shows an example of displaying alarm information on the GUI with “Basic Integration”. First, an example of HTML source code is shown below, and the content will be explained later.

An example of implementation is attached file of "ecxml.html".

Line No.	
1	<code>&lt;!DOCTYPE html&gt;</code>
2	<code>&lt;html&gt;</code>
3	<code>&lt;head&gt;</code>
4	<code>    &lt;title&gt;Event Code Client&lt;/title&gt;</code>
5	<code>&lt;/head&gt;</code>
6	<code>&lt;script&gt;</code>
7	<code>    window.onload = function(){</code>
8	<code>        url = "ws://192.168.0.126/getEventCode";</code>
9	<code>        ws = new WebSocket( url );</code>
10	<code>        ws.onopen = function(e) {</code>
11	<code>            ws_enabled = true;</code>
12	<code>            ws.send( "Format=xml¥n" );</code>
13	<code>        };</code>
14	<code>        ws.onmessage = function ( e ) {</code>
15	<code>            var text = document.createTextNode( e.data );</code>
16	<code>            document.getElementById( "xml" ).textContent = e.data;</code>
17	<code>        };</code>
18	<code>    }</code>
19	<code>&lt;/script&gt;</code>
20	<code>&lt;body&gt;</code>
21	<code>&lt;h3&gt;Event Code as XML&lt;/h3&gt;</code>
22	<code>    &lt;textarea id="xml" rows=20 cols=50&gt;&lt;/textarea&gt;</code>
23	<code>&lt;/body&gt;</code>
24	<code>&lt;/html&gt;</code>

- Opening WebSocket (line 9) in window open event (line 7) .
- The URL to WebSocket is "ws: // 192.168.0.126 / getrec" (line 8).
- The XML format is specified in the “onopen” event is called back when the connection is successful. (line 12).
  - “Plain text” can be selected (Format=plain), but XML is recommended.
- After format specification, data is transmitted each time object is detected, and the “onmessage” function is called back. (line 14)
- In the call back function, the received data is displayed in the text area. (lines 15 to 16)

An example of the execution result is shown below.

### Event Code as XML

```
<?xml version="1.0" encoding="utf-8"?><EventCode>
<MasterAlarm>true</MasterAlarm>
<LatestArea>A2</LatestArea>
<DetectedArea>A1,A2</DetectedArea>
<Disqualification>true</Disqualification>
<AntiRotation>true</AntiRotation>
<AntiMasking>true</AntiMasking>
<InternalError>>false</InternalError>
<Soiling>>false</Soiling><Tamper>true</Tamper>
<output1>>false</output1><output2>>false</output2>
<output3>true</output3><input1>>false</input1>
</EventCode>
```

Received data is in XML format.

REDSAN mini sometimes sends alarm information if there is no detected object. <MasterAlarm> shows whether detected objects exist or not. The above example shows objects are detected. An example of response when there is no detected object is shown below.

### Event Code as XML

```
<?xml version="1.0" encoding="utf-8"?><EventCode>
<MasterAlarm>>false</MasterAlarm></EventCode>
```

#### <MasterAlarm>:

When some objects are detected, the text in this tag will be “true”.

When the objects disappear, this tag will be “false”.

When the preset time (default: 10 seconds) past after master alarm was cleared, this tag will be “cleared”.

#### <LatestArea>:

Shows the area where the newest object is detected.

Possible Code: A1/A2/B1/B2

The areas are able to specified using RSMA.

**<DetectedArea>:**

Shows an area where object has detected as A1, A2, B1 or B2. Multiple areas can be shown.

**<Disqualification >:**

When the unit finds bad condition like fog or hard rain etc., the text in this tag will be “true”.

**<AntiRotation >:**

When the unit finds it was rotated, the text in this tag will be “true”.

**<AntiMasking >:**

When the unit finds the laser window was covered, the text in this tag will be “true”.

**<InternalError >:**

When internal error has occurred, the text in this tag will be “true”.

**<Soiling >:**

When laser window gets dirt, the text in this tag will be “true”.

**<Tamper>:**

When the cover is opened, or the unit is removed from the wall, the text in this tag will be “true”.

**<DeviceMonitoring>:**

The unit send data on a regular basis even if no object is detected, when "device monitoring" function is enabled. When its function is enabled, the text in this tag will be "true". The “device monitoring” function can be switched on and off using RSMA.

**<Output1>:**

This tag shows the status of output No.1.

**<Output2>:**

This tag shows the status of output No.2.

**<Output3>:**

This tag shows the status of output No.3.

**<Input1>:**

This tag shows the status of input No.1.