Instruction

Z-Ware SDK 7.18.x Web Developer Guide

Document No.:	INS 14430
Version:	14
Description:	The Web API exported by Z-Ware Web Server, a Z/IP client that acts as a Z-Wave Web Gateway
Written By:	KAJAROSZ;ADGIELNI;MIKOZIK;JFR;MASZPIEC
Date:	2022-07-29
Reviewed By:	ABUENDIA;SCBROWNI;JFR;KAJAROSZ;MDUMBARE;SVRADEMA
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2022-07-29	05:26:34	NTJ	Niels Johansen	

REVISION RECORD

Doc. Ver.	Date	By	Pages affected	Brief description of changes
1	20180806	AYY	 2, 11-12, 15-16, 18, 36, 40, 43, 54, 56-59, 62, 66, 70, 82-83, 84-92, 97-99, 123	Cloned for SDK 7.00.00 from INS14167 Update supported CC list and versions. Update network information API with timestamp. Updated Library version info. Updated Command queue list. Added 3-part-versioning. Added CCs Version v3, Multilevel Switch v3, Meter v4-5. Configuration v3-4, Sound Switch v1,Doorlock v4, Indicator v1-3,Time v1-2. Updated Alarm Report Event for Scene/Security Scene event. Added State number for Binary Switch, Mutilevel Switch, Doorlock, Thermostat Fan mode, Thermostat Mode, Thermostat Setpoint, Indicator and Barrier Operator CCs.
2	20181011	AYY	7, 8, 19-20, 22 - 25, 31, 33, 39, 61-62	Updated registration API error code. Removed v1 Web API response for S2 inclusion Web APIs. Updated current network operation response to include affected Node ID info. Corrected parameter info for remove/replace failed node. Clarified error response format for all the CCs in CMD_BASIC_SETUP. Added NODE_RESET network operation and info notes. Added Firmware update network operation. Made user code param to be optional for deleting a user code. Added parameter information for network health check get info API. Update Web API error response format.
3	20190305	AYY	16-17, 19-20, 23-25, 39-40, 42-43, 46-48, 63-68, 86-88, 95-97, 108-109, 129-130, 132	Added State number for Basic, Color Switch and Sound Switch CCs. Added Window Covering CC API. Added Firmware backup function Web API. Added Firmware backup operation and completed bytes in Get Current Network Operation API. Remove ‘wkup_intv’ attribute in List Nodes API. Added Multilevel Switch Level Change Set as Scene Action. Updated Color Switch Set Action with optional color_dur parameter. Update Central Scene Command Event with additional key attribute (Value 1 to 6). Added User Code CC v2 API. Added Binary Switch CC v2 Duration support. Added S2 inclusion set grant key API with ‘accept’ support.
4	20190531	AYY	111 103 102 120-121, 123 137 51, 52 118-119 112	Added new completion code for Firmware update command. Added comment for Node association in Asoociation/Multichannel association. Added missing param_id in Window Covering CC stop level change API. Added ‘stateless’ scene status and scene action status. Added Sound Switch Tone play set for Scene action. Added functionality for Binary Sensor CC Get command to yield all cached sensor reports from different types in server. Added documentation for Z/IP Portal Interface. Added Firmware Update CC v6 API.
5	20190912	SNA AYY	 14, 86, 94, 119 130, 148, 149,150, 145 49	v1.39 for SDK v7.11.2 APIs: zw_feature, Battery v2, Sound Switch v2, Firmware Update v7 Scene: ‘active + stateless’ status. Scene: Scene Sound Switch Tone Play Set Action volume description Scene: Window Covering action. Notification clear event trigger Scene: feature web api version Added missing Color Switch CC command numbering
6	20191113	AYY	 2 86-87 62-64, 17 37 106-111	V1.41 for SDK v7.12.0 Update Table for supported Command Classes. Added ‘size’ param in Battery CC Health Get API. Added Door Lock Logging CC v1 API. Remove Command Queue API Added new node property in Node List API. Added ‘Node Identify’ API Added Protection CC v1-2 API.
7	20200110	SNA		V1.41.03 for v7.13.1 - no changes
8	20200325	SNA		Removed Portal & non-Eng UI support; removed Command Queueing
8	20200416	ADGIELNI	2, 86	Added Battery v3 API

9	20200702	SCBROWNI	All	Tech Pubs basic review
9	20200703	MIKOZIK	All	Changed title to 7.14.x
10	20201123	ADGIELNI	All 106 28, 30, 31, 31 17 33, 34	Version 7.15.0 Added Anti-theft Unlock v1 API Updated parameters of Provisioning List get/Device/Add/Inif Get Updated response parameters of List Nodes Added Network Management Installation and Maintenance Command Class v4 API
11	20210518	MIKOZIK	Cover	Update version to 7.16.x
11	20210518	KAJAROSZ	1, 128 2	Explanations regarding portal removal Add antitheft unlock to Command Classes table
12	20211122	KAJAROSZ	Cover	Bump version to 7.17.x
13	20220525	KAJAROSZ	1	Change version to 7.18.x
14	20220727	MASZPIEC	119	Update Configuration Interface Info Get API to allow to read all supported parameters with one request.

Table of Contents

1 Introduction

1.1 Purpose

Z-Ware is a Z-Wave Web Gateway which runs over Z-Wave for IP (Z/IP) Application Programming Interface (API) and exports an abstracted object-based Web API to applications. For more information on Z-Ware, see [1]. This document covers the Web API.

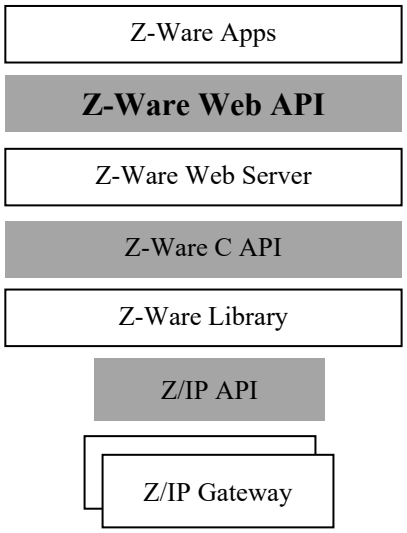


Figure 1. Z-Ware Architecture

The API provides a well-defined interface to command classes, which allow the control and monitoring various attributes of various devices in a Z-Wave network. The API also provides support for Scenes, which are a macro-control element that allows a set of actions to be executed based on user action, Z-Wave events, or schedule.

Some of the APIs are applicable only to the Portal version, and some to the Consumer Electronic (CE) version and are marked accordingly in their respective section headers, while most are applicable to both. Despite the fact, the Portal mode has been removed from Z-Ware, the Web API still has functions to configure gateway for the Portal mode (see 38 Z/IP Portal Interface API).

1.2 Audience and Requirements

Web developers

2 Overview

2.1 Command Classes

Command Class	Versions
ALARM	8
ANTITHEFT_UNLOCK	1
ASSOCIATION	2
ASSOCIATION_GRP_INFO	3
BARRIER_OPERATOR	1
BASIC	2
BATTERY	3
CENTRAL_SCENE	3
CONFIGURATION	4
DOOR_LOCK	4
DOOR_LOCK_LOGGING	1
FIRMWARE_UPDATE_MD	5
INDICATOR	3
METER	5
MULTI_CHANNEL_ASSOCIATION	3
NODE_NAMING	1
PROTECTION	2
SENSOR_BINARY	2
SENSOR_MULTILEVEL	11
SOUND_SWITCH	1
SWITCH_BINARY	2
SWITCH_COLOR	3
SWITCH_MULTILEVEL	4
THERMOSTAT_FAN_MODE	4
THERMOSTAT_FAN_STATE	2
THERMOSTAT_MODE	3
THERMOSTAT_OPERATING_STATE	2
THERMOSTAT_SETPOINT	3
TIME	2
USER_CODE	2
VERSION	3
WAKE_UP	2
WINDOW_COVERING	1
ZIP_GATEWAY	1
ZIP_PORTAL	1
ZWAVEPLUS_INFO	2

The following command classes are handled automatically internally and not exposed to the user

- Security
- CRC-16 Encapsulation
- Multi-Channel

- Multi-Command
- Manufacturer-Specific
- Device Reset Locally
- Some Z/IP-related

2.2 Objects

The API abstracts Z-Wave through objects for networks, nodes, endpoints, and interfaces.

- A Z-Wave Home Area Network (HAN), uniquely identified by a Home ID, is encapsulated in a zwnet object.
- A device in this HAN, uniquely identified by a Node ID, is encapsulated in a zwnode object.
- Channels in a multichannel device are encapsulated in a zwep object, and the node level virtual channel is given an endpoint ID of 0.
- Command classes (CCs) are either handled internally or encapsulated within specific interface objects (zwif_xxx). Interface descriptors are subclassed according to their specific capabilities. CC and their corresponding versions supported are listed in Appendix A.
- The System module (zw) provides access to the Web server and connectivity to the local controller.
- The Scenes module (zwscene) allows the execution of a set of actions triggered by a user action, schedule, or a Z-Wave event.

The diagram below shows the relationship between and the attributes and methods within these objects.

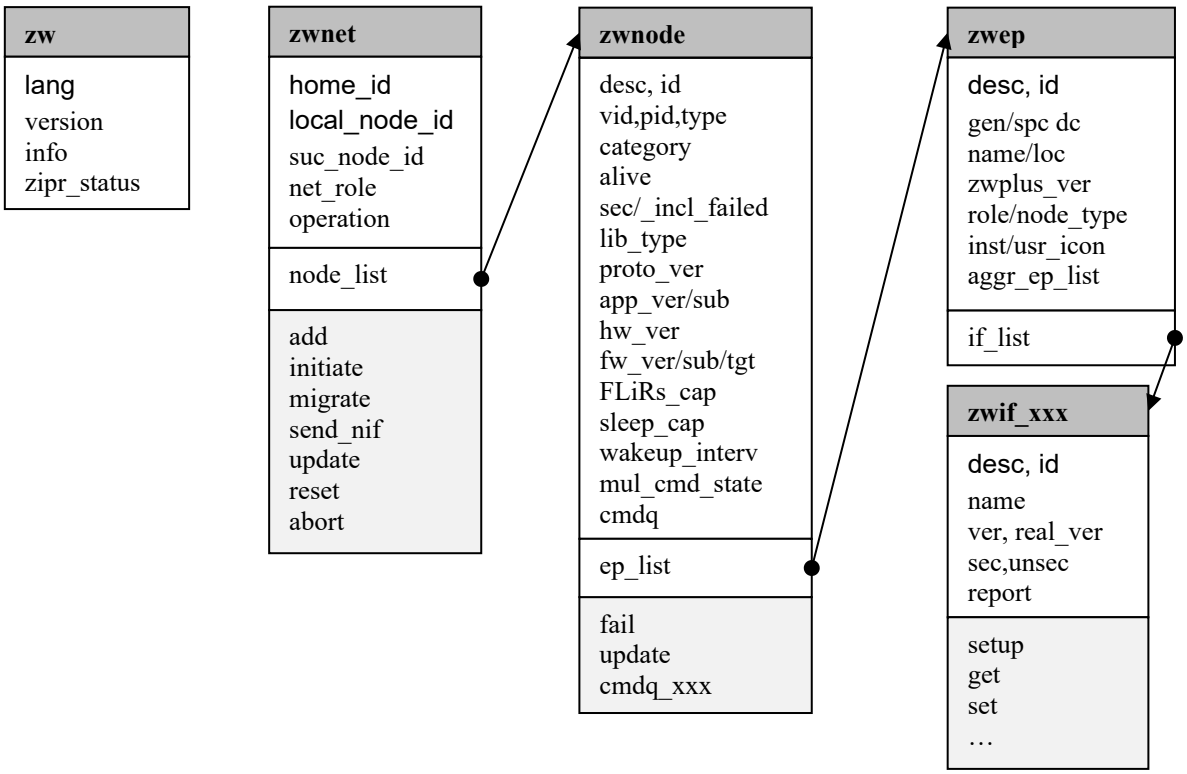


Figure 2. Z-Ware Web API Objects

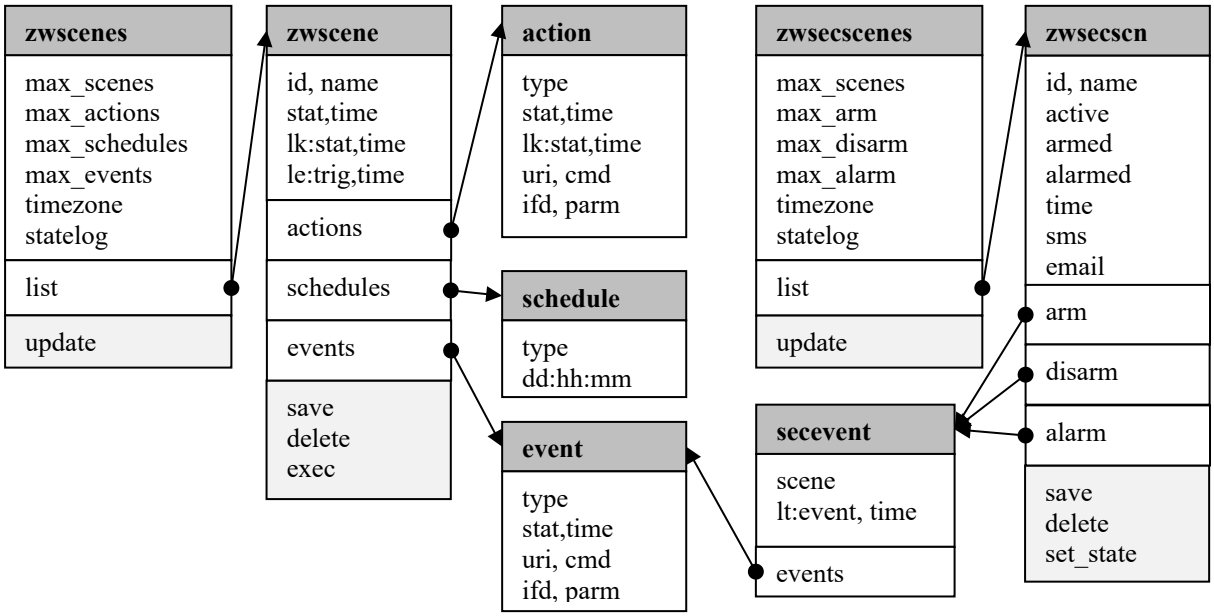


Figure 3. Z-Ware Scene API Objects

2.3 HTTP Mechanism

The Z-Ware API generally uses HTTP POST to the base URL of the Web server. The HTTP Connection timeout is 15s while maximum number of simultaneous connections is 40. The Common Gateway Interface (CGI) is used to communicate between the Web server and the Z-Wave-specific middleware, the timeout for which is 11s. Unless specified otherwise, for all API functions:

- Request Type is POST
- All numeric values sent/received in Request/Reponse are decimal values and their valid ranges are as defined by their corresponding fields in Z-Wave command class (CC) specification.
- Order of Request parameters must be followed.
- All list values received in Response are delimited by commas and terminate with a comma.
- All references to application constants in Request/Response (defined in Appendix B & C), must be replaced with their corresponding values.
- In Response, if {updt_time} is 0, it indicates that no Z-Wave reports were received since the server started up. So, the rest of the values in the response are undefined.

Except for registration APIs, prefix all other APIs with "/cgi/zcgi/networks/", for example the level interface API would be "/cgi/zcgi/networks//zwif_level". The double slash indicates the default network for the client.

2.4 Sample Usage

The following is a jQuery example of enumerating the network up to setting a dimmer's level to 50%.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" type="text/javascript"></script>
<body>
```

```
<script>

var _local;

function zwif_level_setup(ifd) {
	$.ajax({
	type: “POST”,
	url: “cgi/zcgi/networks//zwif_level”,
	data:”cmd=1&ifd=” + ifd
	});
}

function zwif_level_set(ifd, level) {
	$.ajax({
	type: “POST”,
	url: “zwif_level”,
	data:”cmd=4&ifd=” + ifd + “&value=” + level,
	});
}

function zwep_get_if_list(ep) {
	$.ajax({
	type: “POST”,
	url: “cgi/zcgi/networks//zwep_get_if_list”,
	data:”epd=” + ep,
	dataType: “xml”,
	async: false,
	success: function(xml) {
		$(xml).find(‘zwif’).each(function(){
			document.write(“zwif: (desc, id, name, ver, sec, unsec) = (“);
			document.write($(this).attr(‘desc’) + “, “);
			document.write($(this).attr(‘id’) + “, “);
			document.write($(this).attr(‘name’) + “, “);
			document.write($(this).attr(‘ver’) + “, “);
			document.write($(this).attr(‘sec’) + “, “);
			document.write($(this).attr(‘unsec’) + “)<br>”);
			/* set mlv switch to 50% */
			if ($(this).attr(‘id’) == “38”)
			{
				document.write(“sending ...<br>”);
				zwif_level_setup($(this).attr(‘desc’));
				zwif_level_set($(this).attr(‘desc’), “50”);
			}
		});
	},
	});
}

function zwnode_get_ep_list(node) {
```

```
	$.ajax({
	type: “POST”,
	url: “cgi/zcgi/networks//zwnode_get_ep_list”,
	data:”noded=” + node,
	dataType: “xml”,
	async: false,
	success: function(xml) {
		$(xml).find(‘zwep’).each(function(){
			document.write(“zwep: (desc, id, gen, spc, pid, name, loc) = (“);
			document.write($(this).attr(‘desc’) + “, “);
			document.write($(this).attr(‘id’) + “, “);
			document.write($(this).attr(‘generic’) + “, “);
			document.write($(this).attr(‘specific’) + “, “);
			document.write($(this).attr(‘name’) + “, “);
			document.write($(this).attr(‘loc’) + “)<br>”);
			zwep_get_if_list($(this).attr(‘desc’) );
		});
	},
	});
}

function zwnet_get_node_list() {
	$.ajax({
	type: ”POST”,
	url: ” cgi/zcgi/networks//zwnet_get_node_list”,
	dataType: ”xml”,
	async: false,
	success: function(xml) {
		$(xml).find(‘zwnode’).each(function(){
			id = $(this).attr(‘id’);
			if (_local != id)
			/* do not list our controller */
			{
				desc = $(this).attr(‘desc’);
				document.write(“<br>zwnode: (desc, id, vid, pid) = (“);
				document.write(desc + “, “);
				document.write(id + “, “);
				document.write($(this).attr(‘vid’) + “, “);
				document.write($(this).attr(‘pid’) + “)<br>”);
				zwnode_get_ep_list(desc);
			}
		});
	},
	});
}

function zwnet_get_desc() {
	$.ajax({
	type: “POST”,
```

```
        url: “cgi/zcgi/networks//zwnet_get_desc”,
        dataType: “xml”,
        async: false,
        success: function(xml) {
                a = $(xml).find(‘zwnet’);
                document.write(“zwnet: (home, local, suc, role) = (“);
                document.write(a.attr(‘home’) + “, “);
                _local = a.attr(‘local’);
                document.write(_local + “, “);
                document.write(a.attr(‘suc’) + “, “);
                document.write(a.attr(‘role’) + “)<br>”);
                zwnet_get_node_list();
        },
        });
}

zwnet_get_desc();

</script></body></html>
```

2.5 API Error Response

The response of the Web APIs on success return will be specified under the respective API sections in the rest of the document. On fail return, the response will contain an <error> tag and be in the following format unless specified otherwise in the respective API sections.

Response	If parameter format is incorrect, e.g., Certain part of the key/value pair is missing: key1=value1&key2 <?xml version=“1.0”?><zwave><error> </error></zwave>
	if one or more parameters are missing/invalid/other failure: <?xml version=“1.0”?><zwave><error> *CommandName*Failed </error></zwave> *CommandName* is the name of the command/Web API. For example, /zwnet_provisioning_list_list_get API failure response will be: <zwave><error>tServWhitelistListGetFailed</error></zwave> /zwnet_provisioning_list_info API failure response will be: <zwave><error>tServWhitelistDeviceInfoFailed</error></zwave> /zwnet_provisioning_list_add API failure response will be: <zwave><error>tServWhitelistAddFailed</error></zwave>
	Empty, if network uninitialized

3 Registration API

3.1 Login

URI	/register/login
Request	usrname={username}&passwd={curr_password} {username} is the registered username for the application. {curr_password} is the current password for the application.
Response	On success, <?xml version=”1.0”?><zwave><login>status</login></zwave> where status is: 0 – successful
	On failure, <?xml version=”1.0”?><zwave><error>status</error></zwave> where status is: 1 – username blank error 2 – password blank error 3 – invalid username or password error 4 – internal error
Note	Cookies are used to maintain sessions.

3.2 Logout

URI	/register/logout
Response	<?xml version=”1.0”?><zwave><logout>status</logout></zwave> where status is: 0 – successful

3.3 Change Password

URI	/register/changepasswd
Request	curr={curr_passwd}&next={next_ passwd} {curr_passwd} is the current password for the application. {next_ passwd} is the next intended password for the application
Response	On success: <?xml version=“1.0”?><zwave><msg>tServPasswdChgSucessful</msg></zwave>
	if current or next password hash is not passed as input: <?xml version=“1.0”?><zwave><error> tServPasswdChgFailed<!—Parameter missing/invalid--> </error></zwave>
	On other failure: <?xml version=“1.0”?><zwave><error> tServPasswdChgFailed<!—Internal error--> </error></zwave>

3.4 Reset Password

URI	/register/resetpasswd
Request	usrname={username} {username} is the username of the account to be password reset.
Response	<?xml version=”1.0”?><zwave><resetpasswd>status</resetpasswd></zwave> where status is: 0 – successful, mail sent 1 – successful, mail send failed
	On failure, <?xml version=”1.0”?><zwave><error>status</error></zwave> where status is: 2 – invalid input error 3 – LDAP connection error 4 – LDAP bind error 5 – username not unique error 6 – invalid email or hash error
Note	If successful, the user also receives an email on his registered email ID with a link to complete the reset password process.

4 System API

4.1 Platform and Version Numbers

<table>
<tr><td>URI</td><td>/zw_version</td></tr>
<tr><td>Response</td><td><?xml version=“1.0”?><zwave>
<version platform=”{platform}” app_major=”{app_major}” app_minor=”{app_minor}” app_patch=”{app_patch}” ctl_major=”{ctl_major}” ctl_minor=”{ctl_minor}” ctl_patch=”{ctl_patch}” />
</zwave>

<table><tr><th>Values</th><th>Description</th></tr><tr><td>platform</td><td>Server platform e.g., Windows® OS, Linux® OS, and osx respectively for Microsoft Windows, GNU/Linux and OS X.</td></tr><tr><td>app_major/minor/patch</td><td>Application Major/minor/patch version</td></tr><tr><td>ctl_major/minor/patch</td><td>Internal API major/minor/patch version</td></tr></table></td></tr>
</table>

4.2 Information

<table>
<tr><td>URI</td><td>/zw_info</td></tr>
<tr><td>Response</td><td>
<?xml version=”1.0”?><zwave>
<zw_info vname=“{vendor_name}” pname=“{product_name}”
spltform=“{server_platform}” uname=“{username}” init=“{is_hcweb_init_done}”
home_id=“{zwave_home_id}” ctlr_id=“{zwave_controller_id}”
zipgw =”{zip_gateway}”
server_ip=“{server_ip}” client_ip=“{client_ip}” >
<version name=”{zw_info_version_name}” value=”{ zw_info_version_value}” />
[<version …> …]
</zw_info></zwave>

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>{vendor_name}</td><td>Vendor Name</td></tr>
<tr><td>{product_name}</td><td>Product Name</td></tr>
<tr><td>{server_platform}</td><td>OS platform of the server e.g., Windows OS or Linux OS</td></tr>
<tr><td>{is_hcweb_init_done}</td><td>Indicates whether the server is initialized or uninitialized.</td></tr>
<tr><td>{username}</td><td>Username for the current connection to the server.</td></tr>
<tr><td>{zwave_home_id}</td><td>Home ID of the Z-Wave network</td></tr>
<tr><td>{zwave_controller_id}</td><td>Z-Wave controller’s Node ID</td></tr>
<tr><td>{zip_gateway}</td><td>The hostname or IP address of the current Z/IP Gateway, only for Z/IP version</td></tr>
<tr><td>{server_ip}</td><td>The Z-Ware’s IP address</td></tr>
<tr><td>{client_ip}</td><td>The client’s IP address</td></tr>
</table>

<table>
<tr><th>zw_info_version_name</th><th>Description</th></tr>
<tr><td>zw_ctl</td><td>Internal Z-Wave Controller Library</td></tr>
<tr><td>zw_appl</td><td>Z-Ware</td></tr>
<tr><td>zw_webapi</td><td>Z-Ware Web API</td></tr>
<tr><td>zw_ui_engineering</td><td>Engineering Z-App</td></tr>
</table>

Example:
<?xml version=”1.0”?><zwave>
<zw_info vname=”Silicon Labs” pname=”Z-Ware and Z-Ware Apps”
spltform=”linux” uname=”kml” init=”1” home_id=”EE3B8842” ctlr_id=”1”
server_ip=”172.31.37.210” client_ip=”119.73.233.67” >
<version name=”zw_ctl” value=”3.39” />
<version name=”zw_appl” value=”0.24” />
<version name=”zw_webapi” value=”1.08” />
<version name=”zw_ui_engineering” value=”2.14” />
</zw_info></zwave>
Or,
……
<version name=”zw_ctl” value=”3.39 patch 2” />
<version name=”zw_appl” value=”1.30.2” />
……
</td></tr>
</table>

4.3 Z/IP Gateway

These requests serve to connect to any Z/IP Gateway for Z-Wave communication. They use the following parameters:

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>{zip_gw_name}</td><td>name or address of the Z/IP gateway, UTF-8 encoded and then URL encoded.
If “::” (IPv6 address with all zeros) during Initialize, previous Z/IP gateway is used.</td></tr>
<tr><td>{unsol_rpt_port}</td><td>port number for receiving unsolicited reports.
If 0, system will assign a random port to receive unsolicited reports.</td></tr>
<tr><td>{zip_gw_status}</td><td>0: no Z/IP gateway is initialized;
1: a Z/IP gateway is currently initialized.</td></tr>
<tr><td>{zip_gw_addr}</td><td>address of the Z/IP gateway after name resolution</td></tr>
<tr><td>{zip_gw_discvr_flag}</td><td>1: default. enable Z/IP gateway discovery before listing them
0: list cached Z/IP gateways</td></tr>
<tr><td>{zip_gw_flag}</td><td>decimal representation of the following 8-bit mask.
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Use</td><td>R</td><td>R</td><td>R</td><td>R</td><td>V</td><td>C</td><td>D</td><td>B</td></tr>
</table>
B – 1 if the Z/IP gateway is already initialized using another instance of Host Controller Web Server Application in the same machine; 0 if otherwise
D – 1 if the Z/IP gateway is in the discovery list; 0 if otherwise
C – 1 if the Z/IP gateway is in the cached list; 0 if otherwise
V – 1 if the Z/IP gateway is version 2 gateway; 0 if otherwise
R – Reserved</td></tr>
</table>

4.3.1 Initialize Z/IP Gateway

<table>
<tr><td>URI</td><td>/zw_gw_set</td></tr>
<tr><td>Request</td><td>For version 1 gateway:
zip_gw_name={zip_gw_name}&unsol_rpt_port={unsol_rpt_port}

For version 2 gateway:
zip_gw_name={zip_gw_name}&unsol_rpt_port={unsol_rpt_port}&key={preshared_key}

{preshared_key}: Z/IP gateway preshared key.</td></tr>
<tr><td rowspan="3">Response</td><td>Empty on success</td></tr>
<tr><td>if Z/IP gateway name or address is not passed as input:
<?xml version=“1.0”?><zwave><error>
tServNoAddrNum<!--Parameter missing/invalid-->
</error></zwave></td></tr>
<tr><td>On other failure:
<?xml version=“1.0”?><zwave><error>tServNwInit<!--Internal error--></error></zwave></td></tr>
</table>

4.3.2 Get Currently Used Z/IP Gateway

<table>
<tr><td>URI</td><td>/zw_gw_get</td></tr>
<tr><td>Response</td><td><?xml version=“1.0”?><zwave>
<gw_status status=“{zip_gw_status}” name=“{zip_gw_name}”
addr=”{zip_gw_addr}” unsol_rpt_port=”{unsol_rpt_port}” />
</zwave></td></tr>
</table>

4.3.3 List Available Z/IP Gateways

For cached gateway entries, a maximum of 5 entries will be returned. For discovery, a maximum of 200 entries will be returned, with 100 IP v4 gateway entries and 100 IP v6 gateway entries.

<table>
<tr><td>URI</td><td>/zw_gw_list</td></tr>
<tr><td>Request</td><td>[discvr={zip_gw_discvr_flag}]</td></tr>
<tr><td>Response</td><td>On success:
<?xml version="1.0"?><zwave>
<gw_discvr_status state="{discovery_state}" report_recvd="{received_reports}" total_reports="{total_reports}" />
[<gw name="{zip_gw_name}" addr="{zip_gw_addr}" unsol_rpt_port="{unsol_rpt_port}" flag="{zip_gw_flag}" />
[<gw … > …]]
</zwave>

<table>
<tr><th>Values</th><th>Description</th></tr>
<tr><td>discovery_state</td><td>Network discovery type that is currently in progress

0 - No discovery in progress
1 - IPv4 discovery in progress
2 - IPv4 mdns discovery in progress
3 - IPv6 discovery in progress
4 - IPv6 mdns discovery in progress</td></tr>
<tr><td>received_reports</td><td>Number of reports have received for this network discovery type</td></tr>
<tr><td>total_reports</td><td>Total number of reports for this network discovery type</td></tr>
</table>
</td></tr>
</table>

4.4 Feature

This returns the features supported in terms of CCs, interfaces, and versions.

<table>
<tr><td>URI</td><td>/zw_feature</td></tr>
<tr><td>Response</td><td><?xml version=“1.0”?><zwave>
[<feature id=”{intf_ID}” cc_ver=”{intf_ver}” api_ver=”{api_ver}”/>]
[<feature ….. />]

[<scene_action id=”{intf_ID}” cc_ver=”{intf_ver}”
web_cmd_list=”{cmd_list}” api_ver=”{api_ver}” />]
[<scene_action …… />]

[<scene_event id=”{intf_ID}” cc_ver=”{intf_ver}” web_cmd_list=”{cmd_list}”
api_ver=”{api_ver}” />]
[<scene_event …… />]

[<secscene_arm id=”{intf_ID}” cc_ver=”{intf_ver}” web_cmd_list=”{cmd_list}”
api_ver=”{api_ver}” />]
[<secscene_arm …… />]

[<secscene_disarm id=”{intf_ID}” cc_ver=”{intf_ver}” web_cmd_list=”{cmd_list}”
api_ver=”{api_ver}” />]
[<secscene_disarm …… />]

[<secscene_alarm id=”{intf_ID}” cc_ver=”{intf_ver}” web_cmd_list=”{cmd_list}”
api_ver=”{api_ver}” />]
[<secscene_alarm …… />]

</zwave>

<table><tr><th>Values</th><th>Description</th></tr><tr><td>intf_ID</td><td>Supported/Controllable Interface ID in Server (Z-Wave CC ID)</td></tr><tr><td>intf_ver</td><td>Supported/Controllable Interface version in Server.</td></tr><tr><td>api_ver</td><td>The Server API version of the interface.</td></tr><tr><td>cmd_list</td><td>List of supported/controllable web api commands per interfaces (Web api commands (intf_cmd) defined in individual interface API).</td></tr></table></td></tr>
<tr><td>Note</td><td>Client should use this API to determine what feature Server supports and the respective feature version.

{intf_ver} follows the Command Class version. For every new Command Class version, {api_ver} always start from 1. Any additional web api change within the same Command Class version, will increase the {api_ver} of the particular Command Class.

Usage example:
Server supports Binary Switch (ID = 37) command class v1. Response should be:
<feature id=”37” cc_ver=”1” api_ver=”1”/>]

Some time later server begins to support Binary Switch command class v2, but the web api doesn’t support the ‘duration’ param in the report command. Response should be:
<feature id=”37” cc_ver=”2” api_ver=”1”/>]

After some time server adds the support for ‘duration’ param in the Binary Switch report</td></tr>
</table>

	command, and/or other param which may or may not listed in CC spec. <feature id=”37” cc_ver=”2” api_ver=”2”/>] Scene module supports Multilevel Switch CC (ID = 38) as Scene action: <scene_action id=”38” cc_ver=”4” web_cmd_list=”4” api_ver=”1”/>] Later Scene module includes Multilevel Switch Start/Stop level change as Scene action: <scene_action id=”38” cc_ver=”4” web_cmd_list=”4,6” api_ver=”2”/>]

5 Network API

The requests are for managing the Z-Wave network.

5.1 Get Information

<table>
<tr><td>URI</td><td>/zwnet_get_desc</td></tr>
<tr><td>Response</td><td>
<?xml version="1.0"?><zwave>
<zwnet utime="{updt_time}" home="{home_id}" local="{local_node_id}"
[suc="{suc_node_id}"] role="{net_role}"
capability="{ctrl_capability}" zw_role="{ctrl_zwave_role}" />
</zwave>
<table>
<tr><th>Values</th><th>Description</th></tr>
<tr><td>updt_time</td><td>Last update time of the network composition</td></tr>
<tr><td>home_id</td><td>home ID of the Z-Wave network</td></tr>
<tr><td>local_node_id</td><td>local controller's node ID</td></tr>
<tr><td>net_role</td><td>decimal representation of the following 8-bit mask
7 6 5 4 3 2 1 0
R P I X
P – 1 if primary; 0 if otherwise.
U – 1 if SUC; 0 if otherwise.
R – Reserved
X – 1 if proxy; 0 if otherwise.
I – 1 if inclusion controller; 0 if otherwise.</td></tr>
<tr><td>ctrl_capability</td><td>Bitmask representation of the controller capability
1 – supports security 2 protocol
2 - Support inclusion on-behalf</td></tr>
<tr><td>ctrl_zwave_role</td><td>0 – Unknown
1 – SIS
2 – Inclusion
3 – Primary
4 - Secondary</td></tr>
</table>
Empty, if network uninitialized
</td></tr>
<tr><td>Note</td><td>
To determine whether current network is a S2 capable network:
if (ctrl_zwave_role == ZW_ROLE_SIS)
{
if(ctrl_capability & ZWNET_CTLR_CAP_S2)
{
//S2 capable network
}
else
{
//Non-S2 capable network
}
}
else
{
//Indeterministic whether it is a S2 capable network
</td></tr>
</table>

	}

5.2 List Nodes

URI	/zwnet_get_node_list
Response	<?xml version="1.0"?><zwave><zwnet> [<zwnode desc="{node_desc}" id="{node_ID}" property="{node_property}" vid="{vendor_ID}" pid="{prod_ID}" type="{prod_type}" category="{dev_category}" alive="{node_alive_state}" sec="{is_secure}" sec_incl_failed="{is_secure_incl_failed}" lib_type="{lib_type}" proto_ver="{proto_ver}" app_ver="{app_ver}" app_sub_ver="{app_sub_ver}" sensor="{FLiRS_cap}" sleep_cap="{sleep_capable}" s2_keys_valid="{s2_keys_valid}" s2_grant_key="{s2_grant_key}" s2_dsk="{s2_dsk} restricted="{restricted}" " > [<zwnode_sw_ver type="{sw_type}" major="{major_ver}" minor="{minor_ver}" patch="{patch_ver}" build="{build_ver}" />] [<zwnode_sw_ver … />] [<zwnode_ext_ver hw_ver="{hw_ver}"> [<zwnode_fw target="{fw_target}" ver="{fw_ver}" sub_ver="{fw_sub_ver}" /> [<zwnode_fw … > …]] </zwnode_ext_ver>] </zwnode> [<zwnode … > …]] </zwnet></zwave>

Values	Description
node_desc	Node descriptor
node_ID	Node ID
node_property	Bit mask node property for security Bit 0 (0x01) - Node capable to be included securely S0 Bit 1 (0x02) - Node capable to be included securely S2 Bit 2 (0x04) - Node is included securely Bit 3 (0x08) - Node is included insecurely Bit 4 (0x10) - Node is capable to identify itself (e.g. blinking LED 3 times)
vid, pid, type	From manufacturer-specific CC
dev_category	Device category

DEV_XXX	#	Description
SENSOR_ALARM	1	Sensor alarm
ON_OFF_SWITCH	2	On/off switch
POWER_STRIP	3	Power strip
SIREN	4	Siren
VALVE	5	Valve
SIMPLE_DISPLAY	6	Simple display

		DOORLOCK_KEYPAD	7	Door lock with keypad
		SUB_ENERGY_METER	8	Sub energy meter
		ADV_WHL_HOME_ENER_METER	9	Advanced whole home energy meter
		SIM_WHL_HOME_ENER_METER	10	Simple whole home energy meter
		SENSOR	11	Sensor
		LIGHT_DIMMER	12	Light dimmer switch
		WIN_COVERING_NO_POS	13	Window covering no position/endpoint
		WIN_COVERING_EP	14	Window covering end point aware
		WIN_COVERING_POS_EP	15	Window covering position/end point aware
		FAN_SWITCH	16	Fan switch
		RMT_CTL_MULTIPURPOSE	17	Remote control – multipurpose
		DEV_RMT_CTL_AV	18	Remote control – AV
		DEV_RMT_CTL_SIMPLE	19	Remote control – simple
		DEV_UNRECOG_GATEWAY	20	Gateway (unrecognized by client)
		DEV_CENTRAL_CTLR	21	Central controller
		DEV_SET_TOP_BOX	22	Set top box
		DEV_TV	23	TV
		DEV_SUB_SYS_CTLR	24	Sub system controller
		DEV_GATEWAY	25	Gateway
		DEV_THERMOSTAT_HVAC	26	Thermostat – HVAC
		DEV_THERMOSTAT_SETBACK	27	Thermostat – setback
		DEV_WALL_CTLR	28	Wall controller
	node_alive_state	3: “alive” state 4: “down” state 5: “sleeping” state		
	is_secure	1 if the node has secure interfaces; 0 otherwise		
	is_secure_incl_failed	1 if secure inclusion had failed for the node; 0 otherwise		
	lib_type	Z-Wave library type of the node		
	proto_ver	Z-Wave protocol version of the node		
	app_ver	Application version of the node		
	app_sub_ver	Application sub version of the node		
	FLiRS_cap	1 if the node is a FLiRS device; 0 otherwise		
	sleep_capable	1 if the node is capable to sleep (i.e., non-listening and support Wake up command class)		
	s2_keys_valid	1 means the joining node is included in new security scheme		
	s2_grant_key	the final key that is used for secure inclusion. Or 0 if the node is included non-securely		
	s2_dsk	Optional data. S2 DSK info of this node. If DSK is unavailable for this		

		node, the response will not contain this field.
	restricted	Indicates if the node currently runs in restricted mode. 0: Node is not restricted 1: Node is restricted
	sw_type	Software type. Applicable only if Version CC version >= 3. 0: SDK version 1: Application framework API version 2: Host interface version (version of the API exposed to a host CPU. 3: Z-Wave protocol version 4: Application software version
	major_ver	Major version for the software. Applicable only if Version CC version >= 3.
	minor_ver	Minor version for the software. Applicable only if Version CC version >= 3.
	patch_ver	Patch version for the software. Applicable only if Version CC version >= 3.
	build_ver	Build version for the software. Applicable only if Version CC version >= 3.
	hw_ver	Hardware version of the node (‘Hardware Version’ field of ‘Version Report Command’ under Version command class). This value is applicable only if Version command class version >= 2.
	fw_target	Target number of the firmware starting from 1
	fw_ver fw_sub_ver	Firmware version & sub version corresponding to the specific {fw_target} (‘Firmware N Version’ & ‘‘Firmware N Subversion’’ fields of ‘Version Report Command’ under Version CC), applicable only if Version CC version >= 2.
	Empty, if network uninitialized	

5.3 List Node/Endpoint Pairs

URI	/get_node_ep_list
Response	On success: <?xml version=“1.0”?><zwave> [<node_ep desc=“{ep_desc}” node_id=“{node_ID}” ep_id=“{ep_ID}” /> [<node_ep… > …]] </zwave>
	Empty, if network uninitialized
Note	Useful candidate list for multichannel association

5.4 Inclusion/Exclusion

While this API is used for normal Z-Wave Inclusion and Exclusion, it can also be used to inform the server of the intention to perform S2 Inclusion on Behalf of an Inclusion Controller (IOB). The network operation status code progression for IOB will be the same as a normal Add Node operation to ensure that only the client that initiated the IOB procedure will be prompted with the DSK and keys. Also, if the IOB API is not called, the server is unable to differentiate between an unsolicited Add Node and Replace Failed Node status and the Network operation will report Add_Node for status OP_ADD_NODE_ON_BEHALF_SEC_REQ_KEY_READY and OP_ADD_NODE_ON_BEHALF_SEC_DSK_READY for both cases.

URI	/zwnet_add

Request	cmd={net_cmd_type} dsk={dsk of the joining node} {net_cmd_type} is ZWNET_OP_ADD_NODE/RM_NODE/ADD_NODE_ON_BEHALF See 5.10 Get Current Network Operation for more info {dsk of the joining node} is for S2 security inclusion or IOB. It is optional but must be shown in the following format if it exists: “34028-23669-20938-46346-33746-07431-56821-14553”
Response	Empty on success
	if fail: <?xml version=“1.0”?> <zwave><error>tServInclFailed</error></zwave>
	Empty, if network uninitialized
Note	On successful completion, server current operation is set to ZWNET_OP_ADD/RM_NODE

For S2 security, more APIs are needed to complete the process. These same APIs are needed for Replace Failed Node API (See 6.3 Remove/Replace Failed Node) with S2 security as well.

5.4.1 Query Device Requested Keys (S2 Security Only)

A network operation status of “OP_ADD_NODE_SEC_REQ_KEY_READY” indicates that the device requested key information during an S2 secure inclusion is ready. The client that triggered the Node Add or Replace operation can now use this API to query the requested keys information from the Server.

URI	/zwnet_add_s2_get_req_keys
Response	On success: <?xml version=“1.0”?><zwave> <zwnet utime=\"%llu\" id=\"%s\"> <security req_key=”{requested keys of the joining device}” csa_pin=”{client side authentication pin for display}” /> </zwnet> </zwave> {requested keys of the joining device} is the bit mask representation of the requested keys. “csa_pin” is optional and will only be available during the client side authentication scenario.
Note	Bitmask defined as such: SEC_KEY_BITMSK_S2_K0 0x01 /**< S2: class key 0 (Ad-hoc devices)*/ SEC_KEY_BITMSK_S2_K1 0x02 /**< S2: class key 1 (Normal devices)*/ SEC_KEY_BITMSK_S2_K2 0x04 /**< S2: class key 2 (High Security devices)*/ SEC_KEY_BITMSK_S0 0x80 /**< S0: legacy security network key*/

5.4.2 Set Grant Key(s) (S2 Security Only)

URI	/zwnet_add_s2_set_grant_keys
Request	granted_keys={grant_keys} [&grant_csa={grant_csa}] [&accept={accept}]
Response	Empty on success
Note	Refer to /*zwnet_add_s2_get_req_keys* for bitmask of grant key {grant_keys } is the bit mask representation of the grant keys from user.

	{grant_csa} is optional and should only sent during the client side authentication scenario. 1 – accept. 0 – reject. In the event that {grant_csa} is absent, value 0 will be used. {accept} is optional for backward compatibility reason. It indicates whether the user wish to continue/accept the S2 inclusion bootstrapping. UI is advised to have 2 options presented on the UI, one for accepting S2 bootstrapping and one for not accepting S2 bootstrapping (Certification requirement). In the event that {accept} is absent, value 1 will be used. Eg. granted_keys=3&grant_csa=0&accept=1 OR, granted_keys=3&accept=0

5.4.3 Query Device DSK (S2 Security Only)

A network operation status of "OP_ADD_NODE_SEC_DSK_READY" indicates the device DSK information is ready during an inclusion. The client that triggered the Node Add/Replace operation can now use this API to query DSK information from Server.

URI	/ zwnet_add_s2_get_dsk
Response	On success: <?xml version="1.0"?><zwave> <zwnet utime=\"%llu\" id=\"%s\"> <security pin_required="{pin_required}" dsk="{ dsk of the joining device}"/> </zwnet> </zwave> {pin_required} will be either 1 or 0. {dsk of the joining device } may be partial dsk and need user input for the missing parts if {pin_required} is 1. For partial dsk, the format will be "-23669-20938-46346-33746-07431-56821-14553"

5.4.4 DSK Accept (S2 Security Only)

URI	/zwnet_add_s2_accept
Request	accept={ is_accept } value={full_dsk } {full_dsk } should be in this format: "34028-23669-20938-46346-33746-07431-56821-14553" {is_accept} = 1 means accept. 0 means reject.
Response	Empty on success

5.4.5 Query Smart Start Device DSK (S2 Security Only)

A network operation status of "OP_ADD_NODE_PROTOCOL_START" indicates the smart start process is in progress and joining device DSK information is ready during an inclusion. The client can now use this API to query joining node DSK information from the Server to show users which node in provisioning list is joining the network.

URI	/zwnet_add_smart_start_get_dsk

Response	On success: <?xml version="1.0"?><zwave> <zwnet utime=\"%llu\" > <smartstart_dsk dsk="{ dsk of the joining device}"/> </zwnet> </zwave> {dsk of the joining device } is a DSK of a joining device. UI can use it to match with the whitelisted device DSKs.

5.5 Initiate

This is used to Set Learn mode. Again, more APIs are required for S2 Security.

URI	/zwnet_initiate
Request	&classic={is_classic} {is_classic} is 1 for set classic learn mode, 0 or the absent of this parameter is for set NWI learn mode. The request parameter is optional and only applicable for non-S2 capable controller.
Response	Same as for Inclusion except 'tServInclFailed' is replaced with 'tServInitFailed'.
Note	On successful completion, server current operation is set to ZWNET_OP_INITIATE.

5.5.1 Query Local Node DSK (S2 Security Only)

A network operation status of "OP_INI_SEC_OWN_DSK_READY" indicates the device DSK information is ready during an intiation operation for the client to display to the user.

URI	/ zwnet_initiate_local_dsk_get
Response	On success: <?xml version="1.0"?><zwave> <zwnet utime=\"%llu\" id=\"%s\"> <security local_node_dsk="{dsk of the local node}"/> </zwnet> </zwave> {dsk of the local node} is the DSK for local node.

5.6 Send Node Information Frame (NIF)

URI	/zwnet_send_nif
Request	noded={node_desc} {node_desc} is the node descriptor of the node that is expected to receive the node information frame. {node_desc} is 0 for sending broadcast node information frame.
Response	Same as for Inclusion except 'tServInclFailed' is replaced with 'tServSendNifFailed'.
Note	On successful completion, the server current operation is set to ZWNET_OP_SEND_NIF.

5.7 Update Network

URI	/zwnet_update
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServNwUpdateFailed’.
Note	On successful completion, the server current operation is set to ZWNET_OP_UPDATE

5.8 Reset Network

URI	/zwnet_reset
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServNwReset’.
Note	On successful completion, the server current operation is set to ZWNET_OP_RESET.

5.9 Abort Network Operation

URI	/zwnet_abort
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServAbortFailed’.
Note	On successful completion, the server current operation is set to ZWNET_OP_NONE.

5.10 Get Current Network Operation

URI	/zwnet_get_operation		
Response	<?xml version=“1.0”?><zwave><zwnet> <operation op=“{net_cmd_type}” op_sts=“{net_cmd_status}” op_total_nodes=“{op_total_nodes}” op_cmplt_nodes=“{op_cmplt_nodes}” prev_op=“{net_cmd_prev}” op_node=“{op_node_id}” inif_count=”{inif_count}” op_cmplt_bytes=”{op_cmplt_bytes}” /> </zwnet></zwave>		
	Values	**Description**	
	net_cmd_type	current network operation	
		net_cmd_type/prev ZWNET_OP_XXX	**#**	**Description**
		NONE	0	No operation is executing
		INITIALIZE	1	Initialization operation
		ADD_NODE	2	Add node operation
		RM_NODE	3	Remove node operation
		RP_NODE	4	Replace failed node operation
		RM_FAILED_ID	5	Remove failed node operation
		INITIATE	6	Initiation operation by controller
		UPDATE	7	Update network topology from the SUC/SIS
		RESET	8	Restore to factory default setting

		MIGRATE_SUC	9	Create primary controller by a SUC
		MIGRATE	10	Migrate primary controller operation
		LOAD_NW_INFO	12	Load network information
		NODE_UPDATE	13	Update node info
		SEND_NIF	14	Send node information frame
		ADD_NODE_ON_BEHALF	15	Add node on-half operation (For /zwnet_add Web API only. Operation status will still be ZWNET_OP_ADD_NODE)
		RP_NODE_ON_BEHALF	16	Replace failed node on-half operation (For /zwnet_fail Web API only. Operation status will still be ZWNET_OP_RP_NODE)
		FW_UPDT	19	Firmware Update (For CMD_FIRMWARE_UPDATE_REQ_GET Web A only)
		HEALTH_CHK	20	Network health check
		NODE_RESET	21	Process node reset locally notification
		FW_DOWNLD	22	Firmware Backup (For CMD_FIRMWARE_BACKUP_REQ_GET web api only)
	net_cmd_status	Progress status of the ongoing network operation. Some are generic while others are specific to the operation.		
		net_cmd_status	**#**	**Description**
		OP_STS_NONE	0	No status to report
		OP_STS_ERROR	-1	Error
		OP_STS_NO_NET	-4	Network uninitialized
		OP_STS_ABORTED	-5	Network operation aborted
		OP_ADD_NODE_xxx	Add node status	
		PROTOCOL_DONE	1	Protocol part of adding node done
		GET_NODE_INFO	2	Getting node detailed information
		PROTOCOL_START	3	Smart Start add node Z-wave protocol started
		SEC_REQ_KEY_READY	11	(S2 inclusion only) Device requested key info is ready. Client may use */zwnet_add_s2_get_req_keys* to query the requested key information.

		SEC_DSK_READY	12	(S2 inclusion only) Device DSK info is ready. Client may use */zwnet_add_s2_get_dsk* to query for the DSK information.
		ON_BEHALF_SEC_REQ_KEY_READY	21	(S2 inclusion only) Device requested key info is ready. Client may use */zwnet_add_s2_get_req_keys* to query the requested key information.
		ON_BEHALF_SEC_DSK_READY	22	(S2 inclusion only) Device DSK info is ready. Client may use */zwnet_add_s2_get_dsk* to query for the DSK information.
		OP_RM_NODE_XXX	Remove node status	
		LEARN_READY	1	Ready to remove a node
		FOUND	2	Found a node
		REMOVING	3	Removing the node
		OP_RP_NODE_XXX		Replace node status
		READY	1	Ready to replace a node
		PROTOCOL_DONE	2	Protocol part of replacing node done
		SEC_INCD	3	Adding node securely
		GET_NODE_INFO	4	Getting node detailed information
		OP_INI_XXX	Initiate status	
		STARTED	1	Initiating started, ready to be added/removed to/fromnetwork
		PROTOCOL_DONE	2	Protocol part of initiating done
		SEC_INCD	3	Trying to be included securely
		GET_NODE_INFO	4	Getting node detailed information
		SEC_OWN_DSK_READY	11	(S2 inclusion only) Local node DSK info is ready. Client may use */zwnet_initiate_local_dsk_get* to query for the DSK information.
		OP_NU_XXX	Network update status	
		TOPOLOGY	1	Network topology update started

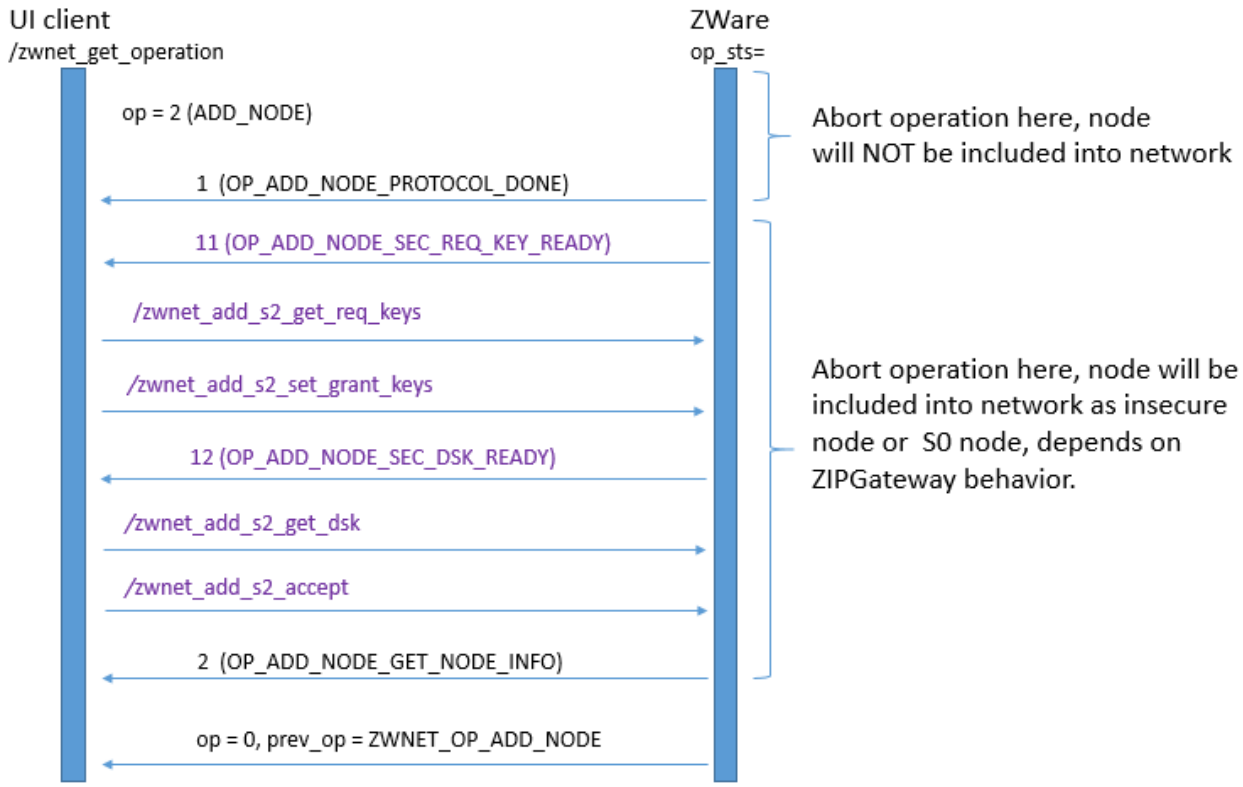
		NEIGHBOR	2	Node neighbor update started
		GET_NODE_INFO	3	Node information update started
		OP_FW_XXX	Firmware update status	
		UPLOAD_STARTED	1	Uploading firmware to device started
		UPLOADING	2	Uploading firmware to device in progress with additional percentage info in op_total_nodes and op_cmplt_nodes
		OP_FW_XXX	Firmware download/backup status	
		DOWNLOAD_STARTED	1	Download/backup firmware from device started
		DOWNLOADING	2	Download/backup firmware from device in progress with additional percentage info in op_total_nodes and op_cmplt_nodes
		OP_HEALTH_CHK_XXX	Network health check status	
		STARTED	1	Network health check started
		PROGRESS	2	Network health check in progress
		CMPLT	3	Network health check completed
	op_total_nodes	Total number of nodes involved in the current operation. The value is applicable for operations that may involve multiple nodes and may be used as a progress indicator. For other operations, the value is 0.		
	op_cmplt_nodes	Number of nodes on which the current operation is completed. The value is applicable for operations that may involve multiple nodes and may be used as a progress indicator. For other operations, the value is 0		
	net_cmd_prev	The previous network operation. This state gets updated every time the state of current operation changes		
	op_node_id	Node ID of the node which is affected by current network operation (e.g., Update node, Add/Remove node operations, and so on). Different network operations may display the affected Node ID at different stage of the operations.		
	inif_count	Number of inif in the network. If the value > 0, client should use /zwnet_provisioning_list_inif_get to query the INIF device information and display to user.		
	op_cmplt_bytes	Number of bytes for which the current operation is completed. Currently it is only used for Firmware download/backup ZWNET_OP_FW_DOWNLD operation		

Note	This command is polled to wait for completion of zwnet_add, initiate, migrate, send_nif, update, reset, abort, fail, node_reset and zwnodew_update operations by checking net_cmd_prev and net_cmd_status fields. {op_node_id} will be given for the following network operations (may not be applied for all the status of the operation): ADD_NODE, RM_NODE, RP_NODE, RM_FAILED_ID, NODE_UPDATE, ADD_NODE_ON_BEHALF, RP_NODE_ON_BEHALF, NODE_RESET. {net_cmd_type} = NODE_RESET indicates controller received the reset locally notification from node ID {op_node_id}. {net_cmd_prev} = NODE_RESET and {net_cmd_status} = OP_STS_NONE means the node has been reset and left the network. Client should use /zwnet_get_node_list to get the refreshed node list. {net_cmd_prev} = NODE_RESET and {net_cmd_status} = OP_STS_ERROR means the node's reset operation fails and it is (likely) to be still in the network. The client should use /zwnet_get_node_list to get the refreshed node list and node status.

5.10.1 S2 Inclusion Flow Diagram

During a node inclusion, the sequence flow diagram for /zwnet_get_operation is shown below. The purple color response is optional depending on whether the including node has COMMAND_CLASS_SECURITY. If it's an on-behalf inclusion, the op_sts 11/12 (OP_ADD_NODE_SEC_REQ_KEY_READY/ OP_ADD_NODE_SEC_DSK_READY) will be replaced by 21/22 (OP_ADD_NODE_ON_BEHALF_SEC_REQ_KEY_READY/ OP_ADD_NODE_ON_BEHALF_SEC_DSK_READY).

As indicated by the diagram, if /zwnet_abort is called prior to receiving OP_ADD_NODE_PROTOCOL_DONE operation status, the new node will NOT be included into the network. If /zwnet_abort or /zwnet_add_s2_accept with "accept=0" is called after receiving OP_ADD_NODE_PROTOCOL_DONE operation status, the new node will still be included in the network as an insecure node or S0 node depending on the ZIPGateway behavior.



5.11 Provisioning List get

URI	/zwnet_provisioning_list_list_get
Response	<?xml version="1.0"?><zwave><zwnet> <pl_list complete="{complete}"> [<pl_device_info dsk ="{dsk of a device}" name="{naming_name}" loc="{naming_loc}" ptype_generic="{generic_class}" ptype_specific="{specific_class}" ptype_icon="{icon_type}" pid_manufacturer_id="{manufacturer_id}" pid_product_type="{product_type}" pid_product_id="{product_id}" pid_app_version="{application_version}" pid_app_sub_version="{application_sub_version}" interval="{interval} " uuid_format="{uuid_format} " uuid_data="{uuid_data_string} " pl_status="{pl_status}" grant_keys="{grant_keys} " boot_mode="{boot_mode} " node_id="{node_id}" node_status="{node_status}" sup_proto="{sup_proto}" />] </pl_list> </zwnet></zwave>

Values	Description
complete	Flag to indicate whether the list is complete. 1: complete 0: not complete
dsk of a device	Device-Specific Key (DSK). The format of the DSK must be 8 groups of 5 digits separated by '-' as shown in the example: “34028-23669-20938-46346-33746-07431-56821-14553”
naming_name	It is the device name in UTF-8 encoding. It must not contain a period character '.' and an underscore character '_', and must not end with the dash character '-'.
naming_loc	It is a device location in UTF-8 encoding. It must not contain a period character '.' and an underscore character '_', and must not end with the dash character '-'.
generic_class	Generic device class
specific_class	Specific device class
icon_type	Installer icon type
manufacturer_id	Manufacturer ID
product_type	Product type
product_id	Product ID
application_version	Application version
application_sub_version	Application sub version
interval	Smart Start inclusion request interval in unit of 128 seconds. This field must have value ranging from 5 to 99.
uuid_format	UUID presentation format: 0: 32 hex digits, no delimiters 1: 16 ASCII chars, no delimiters 2: "sn:" followed by 32 hex digits, no delimiters 3: "sn:" followed by 16 ASCII chars, no delimiters 4: "UUID:" followed by 32 hex digits, no delimiters 5: "UUID:" followed by 16 ASCII chars, no delimiters 6: RFC4122-compliant presentation (e.g., “58D5E212-165B-4CA0-909B-C86B9CEE0111”)
uuid_data_string	Uuid data string in ASCII to display. UI should display the uuid with the prefix as described by ‘uuid_format’. No other processing/manipulation required for uuid_data_string.
pl_status	Status of the provisioning list entry 0: Pending. Node has not been included in the network yet. 2: Passive. The node is in the Provisioning List but the supporting or controlling node decided that the node is unlikely to issue SmartStart inclusion requests in the near future. SmartStart Inclusion requests will be ignored by the Z/IP Gateway. All entries with this status MUST be updated to the “Pending” status (PL_INCL_STS_PENDING) when a Provisioning List Iteration Get Command is received. 3: Ignored. SmartStart inclusion requests sent by the node in the Provisioning List entry will be ignored until the status is changed again by a Z/IP Client or controlling node.
grant_keys	Bit mask of S2 keys to be granted.

	boot_mode	Bootstrapping mode. 0: Bootstrapping mode S2. The node MUST manually be set to Learn Mode and follow the S2 bootstrapping instructions (if any). 1: Bootstrapping mode Smart Start. The node will be included and S2 bootstrapped automatically using the Smart Start functionality. 2: Bootstrapping mode Long Range Smart Start. The node will be included and S2 bootstrapped automatically using the Long Range Smart Start functionality
	node_id	Assigned node ID after inclusion. 0 indicates that a Node ID is not assigned.
	node_status	Network status for the provisioning list entry node. 0: Not included. The node in the Provisioning List is not currently included (added) in the network. 1: Added. The node in the Provisioning List is included in the network and is functional. 2: The node in the Provisioning List has been included in the Z-Wave network but is now marked as failing.
	sup_proto	Comma separated list of protocols supported by the node. 1: Z-Wave 2: Z-Wave Long Range
Note	Other than DSK, all other information is optional.	

5.12 Provisioning List Device Info

URI	/zwnet_provisioning_list_info
Request	dsk={dsk of the joining node}
Response	<?xml version="1.0"?><zwave><zwnet> <pl_device_info dsk ="{dsk of a device}" name="{naming_name}" loc="{naming_loc}" ptype_generic="{generic_class}" ptype_specific="{specific_class}" ptype_icon="{icon_type}" pid_manufacturer_id="{manufacturer_id}" pid_product_type="{product_type}" pid_product_id="{product_id}" pid_app_version="{application_version}" pid_app_sub_version="{application_sub_version}" interval="{interval} " uuid_format="{uuid_format} " uuid_data="{uuid_data_string} " pl_status="{pl_status}" grant_keys="{grant_keys} " boot_mode="{boot_mode} " node_id="{node_id}" node_status="{node_status}" sup_proto="{sup_proto}" /> </zwnet></zwave>
Note	Other than DSK, all other information is optional. Refer to /zwnet_provisioning_list_list_get for the meaning of the various fields.

5.13 Provisioning List Add

URI	/zwnet_provisioning_list_add
Request	dsk={dsk of the joining node} name={naming_name} loc={naming_loc} ptype_generic={generic_class}&ptype_specific={specific_class}&ptype_icon={icon_type} pid_manufacturer_id={manufacturer_id}&pid_product_type={product_type}& pid_product_id={product_id}&pid_app_version={application_version}& pid_app_sub_version={application_sub_version} interval={interval} uuid_format={uuid_format} uuid_data={uuid_data_string} pl_status={pl_status} grant_keys={grant_keys} boot_mode={boot_mode} sup_proto="{sup_proto}"
Response	Empty on success
Note	Other than dsk, all other information are optional. ptype_xxx information are associated together. If any of the ptype_xxx information is given, all the ptype_xxx information must be given as well. pid_xxx information are associated together. If any of the pid_xxx information is given, all the pid_xxx information must be given as well. Refer to /zwnet_provisioning_list_list_get for meaning of the various fields.

5.14 Provisioning List Remove

URI	/zwnet_provisioning_list_remove
Request	dsk={dsk of the joining node} {dsk of the joining node} is Device-Specific Key (DSK). The format of the DSK must be 8 groups of 5 digits separated by '-' as shown in the example: "34028-23669-20938-46346-33746-07431-56821-14553"
Response	Empty on success

5.15 Provisioning List Remove All

URI	/zwnet_provisioning_list_remove_all
Response	Empty on success

5.16 Provisioning List Inif Get

URI	/zwnet_provisioning_list_inif_get

Response	<?xml version="1.0"?><zwave><zwnet> [<pl_device_info dsk ="{dsk of a device}" name="{naming_name}" loc="{naming_loc}" ptype_generic="{generic_class}" ptype_specific="{specific_class}" ptype_icon="{icon_type}" pid_manufacturer_id="{manufacturer_id}" pid_product_type="{product_type}" pid_product_id="{product_id}" pid_app_version="{application_version}" pid_app_sub_version="{application_sub_version}" interval="{interval} " uuid_format="{uuid_format} " uuid_data="{uuid_data_string} " pl_status="{pl_status}" grant_keys="{grant_keys} " boot_mode="{boot_mode} " node_id="{node_id}" node_status="{node_status}" sup_proto="{sup_proto}" />] </zwnet></zwave>
	If failed, return tServWhitelistInifReportFailed
Note	Because the other client may have acknowledged (/zwnet_provisioning_list_inif_ack) on a DSK, the number of provisioning device returned by this API may not correspond to the inif_count that obtained from /zwnet_get_operation.

5.17 Provisioning List Inif Acknowledge

URI	/zwnet_provisioning_list_inif_ack
Request	dsk={dsk of the joining node}
Response	Empty on success If failed, return "tServWhitelistInifAckFailed"
Note	After a DSK is acknowledged, a subsequent query of /zwnet_provisioning_list_inif_get will not return the DSK.

5.18 Network Health Check

URI	/zwnet_health_check
Request	cmd={net_cmd_type} {net_cmd_type} is ZWNET_OP_HEALTH_CHK See 5.10 Get Current Network Operation for more info
Response	Empty on success
	On failure: <?xml version="1.0"?><zwave><error> tServNwHealthCheckFailed</error></zwave>
Note	On successful completion, the server current operation is set to ZWNET_OP_HEALTH_CHK.

5.19 Network Health Check Get Information

URI	/zwnet_health_check_get_info
Response	On success: <?xml version="1.0"?><zwave><zwnet> <nw_health_chk progress="{progress}" prg_node_cnt="{prg_node_cnt}" prg_total="{prg_total}" report="{report}" sts_cnt="{sts_cnt}" /> [<nw_health_info node_id="{node_id}" sts_cat="{sts_cat}" value="{value}" /> [<nw_health_info...>...]] </zwnet></zwave>
	On failure: <?xml version="1.0"?><zwave><error> tServNwHealthCheckGetInfoFailed</error></zwave>
Note	This API provides information when Health check is in progress and the final report when health check is completed. {progress} is 1 indicates health check is still in progress. {report} is 1 indicates health check is completed and report is available. When health check is in progress: {progress} is 1. {prg_node_cnt} is number of health check completed nodes. {prg_total} is total number of nodes scheduled for health check. {report} is 0. {sts_cnt} is 0. When health check is complete and report info is available: {progress} is 0. {prg_node_cnt} is 0. {prg_total} is 0. {report} is 1. {sts_cnt} is total number of node health status report available in <nw_health_info> tags. In <nw_health_info> tags: {node_id} is node ID. {sts_cat} is status category which is derived from network health value: 0: Network health green (network health is good). 1: Network health yellow (Network health is acceptable, but latency can be observed occasionally). 2: Network health red (Network health is insufficient because frames are dropped). 3: Network health critical (Network health is critical because Z-Wave node is not responding at all). {value} is calculated network health value ranges from 0 – 10.

5.20 Z-Wave Long Range Channel Configuration Get

URI	/zwnet_ima_lr_channel_get
Response	On success: <?xml version="1.0"?><zwave><zwnet>

<utime="{utime}" channel="{channel}" />
</zwnet></zwave>

Values	Description
channel	Z-Wave Long Range Channel: 1: Primary 2: Secondary

On failure:
<?xml version="1.0"?><zwave><error></error></zwave>

5.21 Z-Wave Long Range Channel Configuration Set

URI	/zwnet_ima_lr_channel_set
Request	channel={channel}

Values	Description
channel	Z-Wave Long Range Channel: 1: Primary 2: Secondary

Response	Empty on success
	On failure: <?xml version="1.0"?><zwave><error></error></zwave>

6 Node/Endpoint API

6.1 List Endpoints

<table>
<tr><td>URI</td><td>/zwnode_get_ep_list</td></tr>
<tr><td>Request</td><td>noded={node_desc}</td></tr>
<tr><td>Response</td><td>On success:
<?xml version=“1.0”?><zwave><zwnode desc=“{node_desc}”>
[<zwep desc=“{ep_desc}” id=“{ep_ID}”
generic=“{generic_dev_cls}” specific=“{specific_dev_cls}”
name=”{ep_name}” loc=”{ep_loc}”
zwplus_ver=”{zwplus_ver}” role_type=”{role_type}” node_type =”{node_type}”
instr_icon=”{instr_icon}” usr_icon=”{usr_icon}”
aggregated_ep_list=”{aggregated_ep_list}”
/>
[<zwep … > …]]
</zwnode></zwave>

<table>
<tr><th>Values</th><th>Description</th></tr>
<tr><td>generic_dev_cls</td><td>Generic device class number</td></tr>
<tr><td>specific_dev_cls</td><td>Specific device class number</td></tr>
<tr><td>ep_name</td><td>Endpoint Name, UTF-8 encoded and then URL encoded</td></tr>
<tr><td>ep_loc</td><td>Endpoint Location, UTF-8 encoded and then URL encoded</td></tr>
<tr><td>zwplus_ver</td><td>‘Z-Wave+ version’ field of ‘Z-Wave+ Info Report Command’ under Z-Wave+ Info command class, 0 if the node doesn’t support Z-Wave+</td></tr>
<tr><td>role_type
node_type</td><td>‘Role Type’ & ‘Node Type’ fields of ‘Z-Wave+ Info Report Command’ under Z-Wave+ Info command class, applicable only if {zwplus_ver} is not 0</td></tr>
<tr><td>instr_icon</td><td>ZWave plus installer icon type</td></tr>
<tr><td>usr_icon</td><td>ZWave plus user icon type</td></tr>
<tr><td>aggregated_ep_list</td><td>List of aggregated ep members. If list is empty, this is not an aggregated endpoint. Otherwise it is an aggregated endpoint and the list contains the aggregated members for this endpoint.</td></tr>
</table>
</td></tr>
<tr><td></td><td>if noded is not passed as input
<?xml version=“1.0”?><zwave><error>
tServEpListFailed<!–Node id is missing-->
</error></zwave></td></tr>
<tr><td></td><td>if {node_desc} is invalid:
<?xml version=“1.0”?><zwave><error>
tServEpListFailed<!–Invalid node id-->
</error></zwave></td></tr>
<tr><td></td><td>Empty, if network uninitialized</td></tr>
</table>

6.2 List Interfaces in Endpoint

<table>
<tr><td>URI</td><td>/zwep_get_if_list</td></tr>
<tr><td>Request</td><td>epd={ep_desc}</td></tr>
<tr><td>Response</td><td>On success:
<?xml version="1.0"?><zwave><zwep desc="{ep_desc}">
[<zwif desc="{intf_desc}" id="{intf_ID}"
name="{intf_name}" ver="{intf_ver}" real_ver="{real_intf_ver}"
sec="{intf_access_secure}" unsec="{intf_access_unsecure}" />
[<zwif …> …]]
</zwep></zwave>

<table>
<tr><th>Values</th><th>Description</th></tr>
<tr><td>intf_ID</td><td>Interface ID. (Z-Wave CC ID)</td></tr>
<tr><td>intf_name</td><td>Interface name (Z-Wave CC Name)</td></tr>
<tr><td>intf_ver</td><td>Interface virtual version (Z-Wave CC Version). Sometimes a CC can be upgraded or even virtualized by the Z-Ware Library Device Database. An Alarm Sensor CC version 1 virtualized as an Alarm (Notification) CC version 4 to support deprecated CCs in a unified current fashion is an example.</td></tr>
<tr><td>real_intf_ver</td><td>The actual Z-Wave CC version from the device. For a simulated interface, this version will be 0. The client should not perform unsupported commands on this interface.</td></tr>
<tr><td>intf_access_secure</td><td>1 if the interface can be accessed securely; 0 otherwise</td></tr>
<tr><td>intf_access_unsecure</td><td>1 if the interface can be accessed unsecurely; 0 otherwise</td></tr>
</table>
</td></tr>
<tr><td></td><td>if epd is not passed as input
<?xml version="1.0"?><zwave><error>
tServIntfListFailed<!–Endpoint id is missing-->
</error></zwave>,</td></tr>
<tr><td></td><td>if {ep_desc} is invalid:
<?xml version="1.0"?><zwave><error>
tServIntfListFailed<!–Invalid endpoint id-->
</error></zwave></td></tr>
<tr><td></td><td>Empty, if network uninitialized</td></tr>
</table>

6.3 Remove/Replace Failed Node

While Remove or Replace Failed Node is a Z-Wave Network Operation, it is exported as a Z-Ware Node API. This API can also be used to inform the server that the Inclusion Controller (ROB) plans to perform the S2 Replace Node, similarly as IOB. For S2 security, more APIs are needed to complete the process as in the Include API. See 5.4 Inclusion/Exclusion.

URI	/zwnet_fail
Request	cmd={net_cmd_type}&noded={node_desc} }&dsk={dsk of the joining node} {net_cmd_type} is RP_NODE / RM_FAILED_ID / RP_NODE_ON_BEHALF See 5.10 Get Current Network Operation for more info {node_desc} is the node descriptor of the failed node. {dsk of the joining node} is for replace failed node in S2 security only and optional but must be shown in the following format if exists:

	“34028-23669-20938-46346-33746-07431-56821-14553”
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServRemRepFailed’
Note	On successful completion, server current operation is set to ZWNET_OP_RM/RP_NODE

6.4 Update Node

URI	/zwnode_update
Request	noded={node_desc} {node_desc} is the node descriptor of the node to be updated
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServNodeUpdateFailed’
Note	On successful completion, server current operation is set to ZWNET_OP_NODE_UPDATE

6.5 Name/Location

Because get is done when a node is added or updated, is not required in this case. For endpoints that do not support the corresponding CC, the information is stored locally on the server.

6.5.1 Set

URI	/zwep_nameloc
Request	cmd={ep_cmd}&epd={ep_desc}&name={naming_name}&loc={naming_loc}
	{ep_cmd} is CMD_NAMING_SET which has a value of 1 {ep_desc} is the endpoint descriptor {naming_name} & {naming_loc} are the name/location to be set for the endpoint (‘Node name/Location’ field of ‘Node Name Set Command’ under Node Naming and Location CC). The name is UTF-8 encoded and then URL encoded. The maximum length of UTF-8 encoded text is 32 bytes.
Response	Same for Basic Select except - {intf_cmd} is replaced with CMD_NAMING_SET - tServBasic is replaced with tServNaming

6.6 Node Identify

If the node supports ‘identify’ function and indicates it in their node_property field in zwnet_get_node_list API, client can instruct the node to identify itself through the following API. Note that controller will not have ‘Node identify’ (control) function if it is a secondary controller.

URI	/zwnode_identify
Request	noded={node_desc}
Response	Same as for Inclusion except ‘tServInclFailed’ is replaced with ‘tServNodeIdentify’

7 Basic Interface API

These HTTP request parameters are used in this section:

URI	/zwif_basic
Request	cmd={intf_cmd}&ifd={intf_desc}

Additional request parameters are available per request. Most of the other interfaces will have similar requests and will follow the same behavior except where specified differently.

Passive requests will not initiate sending a Z-Wave message and will rely on information already available on the server. Active requests trigger Z-Wave messages and, if the report is received within the CGI timeout, the response will contain the newly received values.

The following request/responses parameters are used in this and other interface sections:

Parameter	Description	
{intf_cmd}	Commands below	
	CMD_BASIC_SETUP	1
	CMD_BASIC_GET	2
	CMD_BASIC_REPORT	3
	CMD_BASIC_SET	4
{intf_desc}	Interface descriptor	
{basic_state}	'Value' field of 'Basic CC Report Command'	
{updt_time}	last updated POSIX format time (receive timestamp of the last Z-Wave report)	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

7.1 Select

This request is used to set up any special report callback and must be called once before any other calls are made on the interface.

Request	{intf_cmd} is CMD_BASIC_SETUP
Response	Empty, if success
	if parameter format is incorrect, e.g., if the value is missing: cmd=1&ifd <?xml version="1.0"?><zwave><error> </error></zwave>
	if one or more parameters are missing/invalid/other failure: <?xml version="1.0"?><zwave><error> tServBasic[CMD_BASIC_SETUP] </error></zwave>
	Empty, if network uninitialized

7.2 Get (Active)

This request gets the state of the node from the node itself.

Request	{intf_cmd} is CMD_BASIC_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <basic utime="{updt_time}" state="{current_state}" target_state="{target_state}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave> "{current_state}": current value of the device hardware. "{target_state}": target value of an ongoing transition or the most recent transition. "{duration}": the time needed to reach the Target Value. 0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFD (128~253) : 1~126 mins. 0xFE (254): Unknown duration 0xFF (255) : Reserved
	Failure is the same as for Select except (intf_cmd} is replaced accordingly.
Note	"{state_num}" will be updated when {current_state} is updated.

7.3 Get (Passive)

This request returns the last known state of the node at the server.

Request	{intf_cmd} is CMD_BASIC_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <basic utime="{updt_time}" state="{current_state}" target_state="{target_state}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave>
	Failure is the same as for Select except (intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when {current_state} is updated.

7.4 Set

This request sets the state of the node.

Request	&value={basic_state} – additional line
	{intf_cmd} is CMD_BASIC_SET
Response	Empty, if success
	Failure is the same as for Select except (intf_cmd} is replaced accordingly

8 Binary Switch Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_switch
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Binary Switch CC commands:

Parameter	Binary Switch CC reference	
{intf_cmd}	Commands below	
	CMD_BSWITCH_SETUP	1
	CMD_BSWITCH_GET	2
	CMD_BSWITCH_REPORT	3
	CMD_BSWITCH_SET	4
{bin_sw_state}	'Value' field in Report and Set commands	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	
{duration}	The time needed to reach the Target Value. 0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFE (128~254) : 1~127 mins. 0xFF (255) : Factory default duration	

8.1 Select

Request	{intf_cmd} is CMD_BSWITCH_SETUP
Response	Behavior is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServBinSw' in the response

8.2 Get (Active)

Request	{intf_cmd} is CMD_BSWITCH_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <switch utime="{updt_time}" state="{current_state}" target_state="{target_state}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave> "{current_state}": current ON/OFF value of the device hardware. "{target_state}": target value of an ongoing transition or the most recent transition. "{duration}": the time needed to reach the Target Value.

	0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFD (128~253) : 1~126 mins. 0xFE (254) : Unknown duration 0xFF (255) : Reserved
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	“{state_num}” will be updated when {current_state} is updated.

8.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_BSWITCH_REPORT
Response	Same as Get (Active)

8.4 Set

Request	&value={bin_sw_state}[&dur={duration}]
	{intf_cmd} is CMD_BSWITCH_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	{duration} is optional and applicable for v2 or above device only. In the event that this parameter is absent, value 0 (change instantly) will be used.

9 Multilevel Switch Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_level
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Multilevel Switch CC Reference	
{intf_cmd}	Commands below	
	CMD_MSWITCH_SETUP	1
	CMD_MSWITCH_GET	2
	CMD_MSWITCH_REPORT	3
	CMD_MSWITCH_SET	4
	CMD_MSWITCH_LVL_CHG_GET	5
	CMD_MSWITCH_LVL_CHG_SET	6
	CMD_MSWITCH_SUP_GET	7
	CMD_MSWITCH_SUP_REPORT	8
{multi_lvl_sw_state}	'Value' field	
{multi_lvl_sw_dur}	'Dimming Duration' field for CC version >= 2. Defaults to 255	
{multi_lvl_sw_lvl_chg_sts}	level change status of the multilevel switch setting. 1 if level change has started and is proceeding; 0 if level change is not proceeding	
{multi_lvl_sw_dir}	'Up/Down' field	
{multi_lvl_sw_ignore_start}	'Ignore Start Level' field	
{multi_lvl_sw_start_lvl}	'Start Level' field. This parameter can be omitted when {multi_lvl_sw_ignore_start} is 1.	
{multi_lvl_sw_dur}	'Dimming Duration' field for CC version >= 2. Defaults to 255.	
{multi_lvl_sw_sec}	'Inc/Dec' field for CC version >= 3. Defaults to 3.	
{multi_lvl_sw_step}	'Step Size' field for CC version >= 3. Defaults to 0.	
{multi_lvl_sw_pri_type}	'Primary Switch Type' field of Multilevel Switch Supported Report command version >= 3. Otherwise undefined.	
{multi_lvl_sw_sec_type}	'Secondary Switch Type' field of Multilevel Switch Supported Report command version >= 3. Otherwise undefined.	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

9.1 Select

Request	{intf_cmd} is CMD_MULTILEVEL_SWITCH_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServMultiLvlSw'

9.2 Get (Active)

Request	{intf_cmd} is CMD_ MULTILEVEL_SWITCH_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <level utime="{updt_time}" state="{multi_lvl_sw_state}" target_state="{target_state}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave> </zwave> "{target_state}": target value of an ongoing transition or the most recent transition. "{duration}": the time needed to reach the Target Value. 0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFD (128~253) : 1~126 mins. 0xFE (254) : Unknown duration 0xFF (255) : Reserved
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when {multi_lvl_sw_state} is updated.

9.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_ MULTILEVEL_SWITCH_REPORT
Response	Same as Get (Active)

9.4 Set

Request	&value={multi_lvl_sw_state}[&dur={multi_lvl_sw_dur}]
	{intf_cmd} is CMD_ MULTILEVEL_SWITCH_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

9.5 Get Level Change (Passive)

Request	{intf_cmd} is CMD_MSWITCH_LVL_CHG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <level_change status= "{multi_lvl_sw_lvl_chg_sts}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

9.6 Start Level Change

Request	&start_stop=1&dir={multi_lvl_sw_dir}&ignore_start_lvl={multi_lvl_sw_ignore_start} &start_lvl={multi_lvl_sw_start_lvl} [&dur={multi_lvl_sw_dur}][&sec={multi_lvl_sw_sec}][&step={multi_lvl_sw_step}]
	{intf_cmd} is CMD_MSWITCH_LVL_CHG_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly

9.7 Stop Level Change

Request	&start_stop=0
	{intf_cmd} is CMD_MSWITCH_LVL_CHG_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly

9.8 Get Capabilities (Active)

Request	{intf_cmd} is CMD_MSWITCH_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <level_sup utime="{updt_time}" pri_type="{multi_lvl_sw_pri_type}" sec_type="{multi_lvl_sw_sec_type}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Multilevel Switch CC version >= 3, otherwise behavior undefined

9.9 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_MSWITCH_SUP_REPORT
Response	Same as Get Capabilities (Active)
Note	Applicable for Multilevel Switch CC version >= 3, otherwise behavior undefined

10 Color Switch Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_color
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Color Switch CC Reference	
{intf_cmd}	Commands below	
	CMD_SWITCH_COLOR_SETUP	1
	CMD_SWITCH_COLOR_GET	2
	CMD_SWITCH_COLOR_REPORT	3
	CMD_SWITCH_COLOR_SET	4
	CMD_SWITCH_COLOR_ LVL_CHG_GET	5
	CMD_SWITCH_COLOR_ LVL_CHG_SET	6
	CMD_SWITCH_COLOR_SUP_GET	7
	CMD_SWITCH_COLOR_SUP_REPORT	8
H{ color_sw_id}	'Color Component ID' field	
{color_sw_cvalue}	'Current Value' field	
{color_sw_tvalue}	'Target Value' field	
{color_sw_dur}	'Duration' field for CC version >= 2. Defaults to 255	
{color_sw_lvl_chg_sts}	level change status of the color switch setting. 1 if level change has started and is proceeding; 0 if level change is not proceeding	
{color_sw_dir}	'Up/Down' field	
{color_sw_ignore_start}	'Ignore Start Level' field	
{color_sw_start_lvl}	'Start Level' field. This parameter can be omitted when {color_sw_ignore_start} is 1.	
{ color_sw_id_list}	Comma separated list of color components	
{color_sw_value_list}	Comma separated list of color values	
{ color_sw_sup_list}	Comma separated list of supported color components – Bit Mask field of Supported Report command	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

10.1 Select

Request	{intf_cmd} is CMD_COLOR_SWITCH_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServColorSw'

10.2 Get (Active)

Request	[&color_id={color_sw_id}]
	{intf_cmd} is CMD_COLOR_SWITCH_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<color utime="{updt_time}" color_id="{color_sw_id}" color_cvalue="{color_sw_cvalue}" [color_tvalue="{color_sw_tvalue}"] [color_dur="{color_sw_dur}"] state_num="{state_num}" /> [<color...>...]] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	All supported color component values are returned when color_id_list is not provided. If there is no {color_sw_tvalue} or {color_sw_dur} (Ver 1-2 Color Switch CC), the response will be: <zwif desc="{intf_desc}"> <color utime="516516581" color_id="3" color_cvalue="20" state_num="4" /> </zwif> Otherwise (Ver 3 or above Color Switch CC), the response will be: <zwif desc="{intf_desc}"> <color utime="516516581" color_id="3" color_cvalue="20" color_tvalue="30" color_dur="10" state_num="4" /> </zwif> "{state_num}" is based on color switch id. Different id has its own {state_num}. {state_num} will be updated when {color_sw_cvalue} is updated.

10.3 Get (Passive)

Request	[&color_id_list={color_sw_id_list}]
	{intf_cmd} is CMD_COLOR_SWITCH_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<color utime="{updt_time}" color_id="{color_sw_id}" color_cvalue="{color_sw_cvalue}" [color_tvalue="{color_sw_tvalue}"] [color_dur="{color_sw_dur}"] state_num="{state_num}" /> [<color...>...]] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	All supported color component values are returned when color_id_list is not provided. If there is no {color_sw_tvalue} or {color_sw_dur} (Ver 1-2 Color Switch CC), the response will be: <zwif desc="{intf_desc}"> <color utime="516516581" color_id="3" color_cvalue="20" state_num="4" /> <color utime="516516582" color_id="4" color_cvalue="50" state_num="1" />

	… </zwif> Otherwise (Ver 3 or above Color Switch CC), the response will be: <zwif desc="{intf_desc}"> <color utime="516516581" color_id="3" color_cvalue="20" color_tvalue="30" color_dur="10" state_num="4" /> <color utime="516516582" color_id="4" color_cvalue="50" color_tvalue="60" color_dur="5" state_num="1" /> … </zwif> "{state_num}" is based on color switch id. Different id has its own {state_num}. {state_num} will be updated when {color_sw_cvalue} is updated.

10.4 Set

Request	&color_id_list={color_sw_id_list}&color_value_list={color_sw_value_list} [&color_dur={color_sw_dur}]
	{intf_cmd} is CMD_COLOR_SWITCH_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

10.5 Get Level Change (Passive)

Request	[&color_id_list={color_sw_id_list}]
	{intf_cmd} is CMD_COLOR_SWITCH_LVL_CHG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <color_change color_id="{color_sw_id}" status= "{color_sw_lvl_chg_sts}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	All supported color component level change statuses are returned when color_id_list is not provided.

10.6 Start Level Change

Request	&color_id={color_sw_id}&color_start_stop=1&color_dir={color_sw_dir}&color_ignore_start={color_sw_ignore_start}[&color_start_lvl={color_sw_start_lvl}][&color_dur={color_sw_dur}]
	{intf_cmd} is CMD_COLOR_SWITCH_LVL_CHG_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly

10.7 Stop Level Change

Request	&color_id={color_sw_id}&color_start_stop=0
	{intf_cmd} is CMD_COLOR_SWITCH_LVL_CHG_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly

10.8 Get Capabilities (Active)

Request	{intf_cmd} is CMD_CSWITCH_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <color_sup utime="{updt_time}" color_sup_list="{color_sw_sup_list}"/> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

10.9 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_CSWITCH_SUP_REPORT
Response	Same as Get Capabilities (Active)

11 Binary Sensor Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_bsensor
Request	Same as for Basic Interface

The following request/responses parameters are used in this section, with references to fields in the Binary Sensor CC commands:

Parameter	Description	
{intf_cmd}	Commands below	
	CMD_BSENSOR_SETUP	1
	CMD_BSENSOR_GET	2
	CMD_BSENSOR_REPORT	3
	CMD_BSENSOR_SUP_GET	4
	CMD_BSENSOR_SUP_REPORT	5
{bsensor_state}	'Sensor Value' field in Binary Sensor CC Report	
{bsensor_type}	'Sensor Type' field in Binary Sensor CC Report	
{idle_time}	The last updated time of idle state formatted in POSIX time (receive timestamp of the last Z-Wave report stating idle condition of the sensor)	
{event_time}	The last updated time of event detected state, formatted in POSIX time	

11.1 Select

Request	{intf_cmd} is CMD_BSENSOR_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServBinSnsr'

11.2 Get (Active)

Request	[&type={bsensor_type}] {intf_cmd} is CMD_BSENSOR_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<bsensor state="{bsensor_state}" idle="{idle_time}" event="{event_time}" type="{bsensor_type}" />] [<bsensor … />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	For ver1 device, {bsensor_type} param in the request should be omitted or value should equal to 0. For ver2 or above device, if {bsensor_type} is 0 or omitted, Server will send 0xFF (first sensor type on supported list) type to device instead. For response, please refer to Get (Passive) API.

11.3 Get (Passive)

Request	Same as Get (Active) except {intf_cmd} is CMD_BSENSOR_REPORT
Response	Same as Get (Active)
Note	When request consists of a specific {bsensor_type} (not 0 and not 0xFF), the request will only consist of 1 sensor report with the specific {bsensor_type}. If no {bsensor_type} is specified, or value 0 is used in the request, the response will contain all the cached reports that Server is holding on at the moment. The list of cached reports may or may not contain the full set of sensor types for individual sensor types and should be treated as such. If {bsensor_type} is 0xFF (255) in the request, the response will contain only one sensor report. The recommended, “version agnostic”, way for the client is to get the supported sensor type first. Then, issue active gets for individual type sensor value (just like any other Command Classes that have supported types). Subsequently, issue passive get without any type to obtain any changes on individual values.

11.4 Get Support (Active)

Request	{intf_cmd} is CMD_BSENSOR_SUP_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <bsensor utime=“{updt_time}” list=“{sensor_type_list}” /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Binary Sensor CC version >= 2, otherwise behavior undefined

11.5 Get Support (Passive)

Request	Same as Get Support (Active) except {intf_cmd} is CMD_BSENSOR_SUP_REPORT
Response	Same as Get Support (Active)

12 Multilevel Sensor Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_sensor
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Multilevel Sensor CC Report field	
{intf_cmd}	Commands below	
	CMD_MSENSOR_SETUP	1
	CMD_MSENSOR_GET	2
	CMD_MSENSOR_REPORT	3
	CMD_MULTILEVEL_SENSOR_SUP_GET	4
	CMD_MULTILEVEL_SENSOR_SUP_REPORT	5
	CMD_MULTILEVEL_SENSOR_SCALE_SUP_GET	6
	CMD_MULTILEVEL_SENSOR_SCALE_SUP_REPORT	7
{snsr_type}	Sensor Type	
{snsr_value}	Sensor Value	
{snsr_precision}	Precision	
{snsr_unit}	Scale	

12.1 Select

Request	{intf_cmd} is CMD_MSENSOR_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServMultiLvlSnsr'

12.2 Get (Active)

Request	type={snsr_type}&unit={snsr_unit}
	{intf_cmd} is CMD_MSENSOR_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<sensor utime="{updt_time}" type="{snsr_type}" value="{snsr_value}" precision="{snsr_precision}" unit="{snsr_unit}" /> [<sensor…>…]] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	When the request consists of a specific {snsr_type} and {snsr_unit}, the request will only consist of 1 sensor report with the specified {snsr_type} and {snsr_unit}. If no {snsr_type} and {snsr_unit} is specified in the request, it is treated as querying for default sensor type value only. The response will contain all the cached reports that Server holding on the moment. The list of cached reports may or may not contain the full Set of sensor values for individual

	sensor type and unit and should not be treated as such. If the {snsr_type} is 0 (invalid), it will be treated as no specific {snsr_type} and {snsr_unit} scenario. The recommended, “version agnostic”, way for the client is to get the supported type and unit first. Then, issue active gets for individual type/unit sensor value (just like any other Command Classes that have supported types). Subsequently, issue passive get without any type/unit to obtain any changes on individual values.

12.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_MSENSOR_REPORT
Response	Same as Get (Active)

12.4 Get Support (Active)

Request	{intf_cmd} is CMD_MSENSOR_SUP_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <sensor utime=“{updt_time}” list=“{sensor_type_list}” /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Multilevel CC version >= 5. For Multilevel CC version < 5, it still works if the device responds to Multilevel sensor Get command during inclusion.

12.5 Get Support (Passive)

Request	Same as Get Support (Active) except {intf_cmd} is CMD_MSENSOR_SUP_REPORT
Response	Same as Get Support (Active)

12.6 Get Supported Scales (Active)

Request	&type={snsr_type} {intf_cmd} is CMD_MULTILEVEL_SENSOR_SCALE_SUP_REPORT (Appendix B.4)
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <sensor utime=“{updt_time}” type=“{snsr_type}” list=“{sensor_scale_list}”/> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Multilevel CC version >= 5. For Multilevel CC version < 5, it still works if the device responds to Multilevel sensor Get command during inclusion.

12.7 Get Supported Scales (Passive)

Request	Same as Get Supported Scales (Active) except {intf_cmd} is CMD_MULTILEVEL_SENSOR_SCALE_SUP_REPORT

Response	Same as Get Supported Scales (Active)
Note	Applicable for Multilevel CC version >= 5. For Multilevel CC version < 5, it still works if the device responds to Multilevel sensor Get command during inclusion.

13 Meter Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_meter
Request	Same as for Basic Interface

Note that Monitor/Config requests use different URLs as they belong to different CCs.

The following request/responses parameters are used in this section:

Parameter	Meter CC Command field	
{intf_cmd}	Commands below	
	CMD_METER_SETUP	1
	CMD_METER_GET	2
	CMD_METER_REPORT	3
	CMD_METER_SUP_GET	4
	CMD_METER_SUP_REPORT	5
	CMD_METER_RESET	6
{meter_unit}	Scale field of Get Command version >= 2. Default to 0	
{meter_type}	Meter Type field of Report Command	
{meter_value}	Meter Value field of Report Command	
{meter_precision}	Precision field of Report Command	
{meter_unit}	Scale field of Report Command	
{meter_rtype}	Rate Type field of Report Command v2	
{meter_delta}	Delta Time field of Report Command v2	
{meter_prev}	Previous Meter Value field of Report Command v2	
{meter_type}	Meter Type field of Supported Report Command	
{meter_sup_units}	‘Scale Supported field of Supported Report Command	
{meter_is_reset}	Meter Reset field of Supported Report Command	
{meter_adm_no}	Meter Point Adm. Number’ field of Table Point Adm. Number Set Command. The admin number text is UTF-8 encoded and then URL encoded.	
{meter_id}	Meter ID field of Table ID Report Command. The meter ID text is UTF-8 encoded and then URL encoded.	

13.1 Select

Request	{intf_cmd} is CMD_METER_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - ‘tServBasic’ is replaced with ‘tServMeter’

13.2 Get (Active)

Request	[&unit={meter_unit}][&rtype={meter_rtype}] {intf_cmd} is CMD_METER_GET meter_rtype: 0 or no 'rtype' parameter – No preference. 1 – Import 2 - Export
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<meter utime="{updt_time}" type="{meter_type}" value="{meter_value}" precision="{meter_precision}" unit="{meter_unit}" rtype="{meter_rtype}" delta="{meter_delta}" prev="{meter_prev}" /> [<meter…>…]] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	The CC spec does not clearly indicate the values for the new units in Electric meter, we will define the new value to be 8 and 9, for unit "KVar" and 'KVarh'.

13.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_METER_REPORT
Response	Same as Get (Active)

13.4 Get Support (Active)

Request	{intf_cmd} is CMD_METER_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <meter_sup utime="{updt_time}" type="{meter_type}" units="{meter_sup_units}" reset="{meter_is_reset}" [rtype={meter_rtype}] /> </zwif></zwave>,
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Meter CC version >= 2, otherwise behavior undefined

13.5 Get Support (Passive)

Request	Same as Get Support (Active) except {intf_cmd} is CMD_METER_SUP_REPORT
Response	Same as Get Support (Active)
Note	Applicable for Meter CC version >= 2, otherwise behavior undefined

13.6 Reset

Request	{intf_cmd} is CMD_METER_RESET
Response	Response is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Meter CC version >= 2, otherwise behavior undefined.

14 Door Lock Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_dlck
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Door Lock CC Command field	
{intf_cmd}	Commands below	
	CMD_DLOCK_SETUP	1
	CMD_DLOCK_OP_GET	2
	CMD_DLOCK_OP_REPORT	3
	CMD_DLOCK_OP_SET	4
	CMD_DLOCK_CFG_GET	5
	CMD_DLOCK_CFG_REPORT	6
	CMD_DLOCK_CFG_SET	7
	CMD_DLOCK_CAP_GET	8
	CMD_DLOCK_CAP_REPORT	9
{dlock_mode}	Door Lock Mode field of Operation Report Command	
{dlock_out_mode}	Outside Door Handles Mode field of Operation Report Command	
{dlock_in_mode}	Inside Door Handles Mode field of Operation Report Command	
{dlock_cond}	Door Condition field of Operation Report Command	
{dlock_to_min}	Lock Timeout Minutes field of Operation/Configuration Report Command	
{dlock_to_sec}	Lock Timeout Seconds field of Operation/Configuration Report Command	
{dlock_type}	is the operation type of the door lock configuration (‘Operation Type’ field of ‘Door Lock Configuration Report Command’ under Door Lock CC)	
{dlock_out_sta}	Outside Door Handles State field of Configuration Report Command	
{dlock_in_sta}	Inside Door Handles State field of Configuration Report Command	
{dlock_blk_to_blk}	Block-to-block (BTB) field of Configuration Report Command. 0 means disable. Non-0 means enable.	
{dlock_ta}	Twist assist (TA) field of Configuration Report Command. 0 means disable. Non-0 means enable.	
{dlock_auto_rlck_tm}	Auto-relock time field of Configuration Report Command. Range in 0..65535.	
{dlock_hold_rel_tm}	Hold and release time field of Configuration Report Command. Range in 0..65535.	
{dlock_op_type_list}	Comma separated list of Operation Types – Bit Mask field of Supported Operation Types in Capability Report Command	
{dlock_mode_list}	Comma separated list of door lock modes – Bit Mask field of Supported door lock modes in Capability Report Command	
{dlock_out_mask}	Supported Outside Handle Modes Bitmask in Capability Report Command	
{dlock_in_mask}	Supported Inside Handle Modes Bitmask in Capability Report Command	
{dlock_comp_mask}	Supported Door components Bitmask in Capability Report Command	
{dlock_cap_mask}	Additional Door lock capability Bitmask: 1: Block-to-block (BTB) capability 2: Twist assist (TA) capability	

	4: Hold and release capability 8: Auto relock capability
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.

14.1 Select

Request	{intf_cmd} is CMD_DLOCK_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServDLock'

14.2 Get (Active)

Request	{intf_cmd} is CMD_DLOCK_OP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <dlck_op utime="{updt_time}" mode="{dlock_mode}" out_mode="{dlock_out_mode}" in_mode="{dlock_in_mode}" cond="{dlock_cond}" tmout_min="{dlock_to_min}" tmout_sec="{dlock_to_sec}" target_mode="{target_mode}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave> "{target_mode}": target door lock mode of an ongoing transition or the most recent transition. "{duration}": the time needed to reach the Target Mode. 0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFD (128~253) : 1~126 mins. 0xFE (254): Unknown duration 0xFF (255): Reserved
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when {dlock_mode} is updated.

14.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_DLOCK_OP_REPORT
Response	Same as Get (Active)

14.4 Set

Request	&mode={dlock_mode}
	{intf_cmd} is CMD_DLOCK_OP_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

14.5 Get Configuration (Active)

Request	{intf_cmd} is CMD_DLOCK_CFG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <dlck_cfg utime="{updt_time}" type="{dlock_type}" out_sta="{dlock_out_sta}" in_sta="{dlock_in_sta}" tmout_min="{dlock_to_min}" tmout_sec="{dlock_to_sec}" blk_to_blk="{dlock_blk_to_blk}" twist_asst="{dlock_ta}" auto_rlck_tm="{dlock_auto_rlck_tm}" hold_rel_tm="{dlock_hold_rel_tm}" state_num="{state_num}" /> </zwif></zwave >
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	{dlock_blk_to_blk}, {dlock_ta}, {dlock_auto_rlck_tm}, {dlock_hold_rel_tm} values are Door Lock CC version >= 4. "{state_num}" will be updated when any of value in Doorlock Config is changed. For example, type, in/out state, tmout_min, blk_to_blk, and so on.

14.6 Get Configuration (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_DLOCK_CFG_REPORT
Response	Same as Get (Active)

14.7 Set Configuration

Request	&type={dlock_type}&out_sta={dlock_out_sta}&in_sta={dlock_in_sta} &tmout_min={dlock_to_min}&tmout_sec={dlock_to_sec} [&blk_to_blk={dlock_blk_to_blk}&twist_asst={dlock_ta} &auto_rlck_tm={dlock_auto_rlck_tm}&hold_rel_tm={dlock_hold_rel_tm}]
	{intf_cmd} is CMD_DLOCK_CFG_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	{dlock_blk_to_blk}, {dlock_ta}, {dlock_auto_rlck_tm}, {dlock_hold_rel_tm} values are optional and only applicable for Door Lock CC version >= 4. Any value missing in the request will be treated as value 0.

14.8 Get Capability (Active)

Request	{intf_cmd} is CMD_DLOCK_CAP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <dlck_cap utime="{updt_time}" op_type_list="{dlock_op_type_list}" mode_list="{dlock_mode_list}" out_mask="{dlock_out_mask}" in_mask="{dlock_in_mask}" comp_mask="{dlock_comp_mask}" cap_mask="{dlock_cap_mask}" /> </zwif></zwave >
	Failure is the same for Select except {intf_cmd} is replaced accordingly

14.9 Get Capability (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_DLOCK_CAP_REPORT
Response	Same as Get (Active)

15 Door Lock Logging Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_dlck_log
Request	Same as for Basic Interface

The following request/responses parameters are used in this section with references to fields in the Door lock logging CC commands:

Parameter	Door Lock Logging CC Command field	
{intf_cmd}	Commands below	
	CMD_DOOR_LOCK_LOGGING_SETUP	1
	CMD_DOOR_LOCK_LOGGING_GET	2
	CMD_DOOR_LOCK_LOGGING_REPORT	3
	CMD_DOOR_LOCK_LOGGING_SUP_GET	4
	CMD_DOOR_LOCK_LOGGING_SUP_REPORT	5
{record_id}	Record number in Record Get command. Value 0 is used to get the most recent record entry.	
{record_count}	Number of record reports to get from Sever cache, starting from {record_id}. Applicable for Passive Get only.	
{record_status}	Record status. 0: Record is empty. 1: Record is valid.	
{event}	Event Type field in Record Report command.	
{user_id}	User Identifier field in Record Report command.	
{user_code}	User Code field in Record Report command.	
{year}	Year field in Record Report command. Eg. 2019	
{month}	Month field in Record Report command: 1-12	
{day}	Day field in Record Report command: 1-31	
{hour}	Hour field in Record Report command: 0-23	
{min}	Minute field in Record Report command: 0-59	
{sec}	Second field in Record Report command: 0-59	
{max_rec_cnt}	Maximum number of door lock logging records supported by the device.	

15.1 Select

Request	{intf_cmd} is CMD_DOOR_LOCK_LOGGING_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - ‘tServBasic’ is replaced with ‘tServDLockLogging’

15.2 Get (Active)

Request	&rec_id={record_id}
	{intf_cmd} is CMD_DOOR_LOCK_LOGGING_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<dlck_log utime="{updt_time}" rec_id="{record_id}" rec_status="{record_status}" event="{event}" user_id="{user_id}" user_code="{user_code}" year="{year}" month="{month}" day="{day}" hour="{hour}" min="{min}" sec="{sec}" />] [<dlck_log … />] </zwif></zwave>
	Failure same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServDLockLogging'

Note	{record_count} is optional and only applicable for Passive Get. Absence of this param will be treated as {record_count} is 1. If this param is present in Active Get with value other than 0 or 1, an error will be returned.
	
	If the UI needs to show multiple records information, it is advised to use Active Get to query all the values then use Passive Get to retrieve any additional updates.
	
	Eg. UI needs to show 10 records with their information start from User ID 1:
	
	for (i = 0; i < 10; i++)
	{
	Active Get (rec_id = i);
	}
	
	Passive Get (rec_id = 1 & rec_cnt = 10)

15.3 Get (Passive)

Request	&rec_id={record_id}[&rec_cnt={record_count}]
	{intf_cmd} is CMD_DOOR_LOCK_LOGGING_REPORT
Response	Same as Get (Active) except {intf_cmd} is replaced accordingly
Note	See Get (Active)

15.4 Get Capabilities (Active)

Request	{intf_cmd} is CMD_DOOR_LOCK_LOGGING_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <dlck_log_sup utime="{updt_time}" max_rec_cnt="{max_rec_cnt}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

15.5 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_DOOR_LOCK_LOGGING_SUP_REPORT
Response	Same as Get Capabilities (Active)

16 User Code Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_usrcod
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	User Code CC Command field	
{intf_cmd}	Commands below	
	CMD_USER_CODE_GET	1
	CMD_USER_CODE_REPORT	2
	CMD_USER_CODE_SET	3
	CMD_USER_CODE_USERS_GET	4
	CMD_USER_CODE_USERS_REPORT	5
	CMD_USER_CODE_CAP_GET	6
	CMD_USER_CODE_CAP_REPORT	7
	CMD_USER_CODE_KEYPAD_MODE_GET	8
	CMD_USER_CODE_KEYPAD_MODE_REPORT	9
	CMD_USER_CODE_KEYPAD_MODE_SET	10
	CMD_USER_CODE_MASTER_CODE_GET	11
	CMD_USER_CODE_MASTER_CODE_REPORT	12
	CMD_USER_CODE_MASTER_CODE_SET	13
	CMD_USER_CODE_CHECKSUM_GET	14
	CMD_USER_CODE_CHECKSUM_REPORT	15
{code_user_id}	User Identifier field of Get Command	
{code_user_status}	User ID Status field of Report Command	
{code_user_code}	USER CODE field of Report Command	
{code_user_cnt}	range of {code_user_id} that can be used with Get/Set operations	
{report_more}	Report More field in Extended Get Command. Only for device ver2 or above, and only for Active Get.	
{user_count}	Number of user code reports to get from Sever cache, starting from {code_user_id}. For all versions of device but only applicable for Passive Get.	
{next_user_id}	Next User ID field in Extended Report Command. Only meaningful for device ver2 or above.	
{user_id_list}	Comma separated list of User IDs	
{user_status_list}	Comma separated list of User ID Status.	
{user_code_list}	Comma separated list of User Codes.	
{cap_mask}	Capability support bitmask. 0x01: Bitmask to indicate support of master code functionality 0x02: Bitmask to indicate support of master code de-activation 0x04: Bitmask to indicate support of user code checksum functionality 0x08: Bitmask to indicate support of reporting multiple user codes at once in a single Extended User Code Report Command 0x10: Bitmask to indicate support of setting multiple user codes at once in a single Extended User Code Set Command	

{user_id_status_list}	Comma separated list of supported User ID status – Supported User ID Status Bit Mask field of Capability Report command
{keypad_mode_list}	Comma separated list of supported Keypad Modes – Supported Keypad Modes Bit Mask field of Capability Report command
{ascii_key_list}	Comma separated list of supported ascii keys – Supported Keys Bit Mask field of Capability Report command
{keypad_mode}	Keypad Mode in Keypad Mode Report Command.
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.
{master_code}	Master Code field in Master Code Report Command. Master Code in empty string "" indicates Master Code is deactivated or to be deactivated.
{checksum}	Checksum field in Checksum Report Command.

16.1 Get (Active)

Request	&id={code_user_id}[&report_more={report_more}]
	{intf_cmd} is CMD_USER_CODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <usrcod_rpt utime="{updt_time}" next_id={next_user_id} > [<usrcod utime="{updt_time}" id="{code_user_id}" status="{code_user_status}" code="{code_user_code}" />] [<usrcod… />] </usrcod_rpt> </zwif></zwave>
	Failure same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServUserCode'

Note	{report_more} is optional and only applicable for device ver2 or above, and only for Active Get. Absence of this param will be treated as {report_more} is 0. If this param is present in Passive Get and the value is not 0, an error will be returned.
	
	{user_count} is optional and only applicable for Passive Get. Absence of this param will be treated as {user_count} is 1. If this param is present in Active Get with value other than 0 or 1, an error will be returned.
	
	If the UI needs to show multiple user code information, it is advised to use Active Get with {report_more} set to 1 to query all the values then use Passive Get to retrieve any additional updates.
	
	Eg. UI needs to show 10 user IDs with their status and codes start from User ID 1:
	
	start_index = 1;
	
	While(1)
	{
	Active Get (id = start_index & report_more = 1)

	: response
	{
	If(next_id != 0 & next_id <= 10)
	{
	start_index = next_id;
	Continue;
	}
	Else
	Break;
	}
	}
	
	Passive Get (id = 1 & user_cnt = 10)

16.2 Get (Passive)

Request	&id={code_user_id}[&report_more={report_more}][&user_cnt={user_count}]
	{intf_cmd} is CMD_USER_CODE_REPORT
Response	Same as Get (Active) except {intf_cmd} is replaced accordingly
Note	See Get (Active)

16.3 Set

Request	&id={user_id_list}&status={user_status_list}[&code={user_code_list}]
	{intf_cmd} is CMD_USER_CODE_SET
Response	Empty on success
	Failure same as for Get except {intf_cmd} is replaced accordingly
Note	{user_code_list} is optional and will be ignored when {user_status_list} has only 1 value and it is 0 (delete user code). UI can set multiple User codes with 1 API if the device advertises Multiple User Code Set support. The number of components in {user_id_list}, {user_status_list} and {user_code_list} should match. Note that the whole request should be able to fit into 1 Z-Wave packet. If too many number User IDs are sent and not able to fit into 1 Z-Wave packet due to different encapsulations or other reason, an error will be returned. Eg. For ver 1 device, the request should be: &id=3 &status=1 &code=”123456” For ver 2 device that supports Multiple User Code Set, the request can be:

	&id="3,4,5" &status="1,2,1" &code="123456,234567,345678"

16.4 Get Number of Users (Active)

Request	{intf_cmd} is CMD_USER_CODE_USERS_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <usrcod_sup utime="{updt_time}" user_cnt="{code_user_cnt}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

16.5 Get Number of Users (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_USER_CODE_USERS_REPORT
Response	Same as Get Capabilities (Active)

16.6 Get Capabilities (Active)

Request	{intf_cmd} is CMD_USER_CODE_CAP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <usrcod_cap utime="{updt_time}" cap_mask="{cap_mask}" user_id_status_list="{user_id_status_list}" keypad_mode_list="{keypad_mode_list}" ascii_key_list="{ascii_key_list}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

16.7 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_USER_CODE_CAP_REPORT
Response	Same as Get Capabilities (Active)

16.8 Get Keypad Mode (Active)

Request	{intf_cmd} is CMD_USER_CODE_KEYPAD_MODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}">

	<usrcod_kpad_md utime="{updt_time}" keypad_mode="{keypad_mode}" state_num="{state_num}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when {keypad_mode} is updated.

16.9 Get Keypad Mode (Passive)

Request	Same as Get Keypad Mode (Active) except {intf_cmd} is CMD_USER_CODE_KEYPAD_MODE_REPORT
Response	Same as Get Keypad Mode (Active)

16.10 Set Keypad Mode

Request	&keypad_mode={keypad_mode}
	{intf_cmd} is CMD_USER_CODE_KEYPAD_MODE_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

16.11 Get Master Code (Active)

Request	{intf_cmd} is CMD_USER_CODE_MASTER_CODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <usrcod_master_code utime="{updt_time}" master_code="{master_code}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

16.12 Get Master Code (Passive)

Request	Same as Get Master Code (Active) except {intf_cmd} is CMD_USER_CODE_MASTER_CODE _REPORT
Response	Same as Get Master Code (Active)

16.13 Set Master Code

Request	&master_code={master_code}
	{intf_cmd} is CMD_USER_CODE_MASTER_CODE_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

16.14 Get Checksum (Active)

Request	{intf_cmd} is CMD_USER_CODE_CHECKSUM_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <usrcod_chksum utime="{updt_time}" checksum="{checksum}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

16.15 Get Checksum (Passive)

Request	Same as Get Checksum (Active) except {intf_cmd} is CMD_USER_CODE_CHECKSUM_REPORT
Response	Same as Get Checksum (Active)

17 Thermostat Mode Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_thrmo_md
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Thermostat Mode CC Command field	
{intf_cmd}	Commands below	
	CMD_THRMO_MODE_SETUP	1
	CMD_THRMO_MODE_GET	2
	CMD_THRMO_MODE_REPORT	3
	CMD_THRMO_MODE_SET	4
	CMD_THRMO_MODE_SUP_GET	5
	CMD_THRMO_MODE_SUP_REPORT	6
{thrmo_mode}	Mode field of Report/Set Command	
{thrmo_mode_list}	Comma separated list of supported thermostat modes – Bit Mask field of Supported Report command	
{thrmo_manf_dat_list}	Comma separated list of manufacturer data (Manufacturer Data field of Version 3 or above Set/Report command)	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

17.1 Select

Request	{intf_cmd} is CMD_THRMO_MODE_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServThrmoMode'

17.2 Get (Active)

Request	{intf_cmd} is CMD_THRMO_MODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_md utime="{updt_time}" mode="{thrmo_mode}" manf_dat_list="{thrmo_manf_dat_list}" state_num="{state_num}" /> </zwif></zwave> {thrmo_manf_dat_list} is optional and will only be available for "Manufacturer-Specific mode" for Version 3 or above Get command.
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when either {thrmo_mode} or {thrmo_manf_dat_list} is

	updated.

17.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_THRMO_MODE_REPORT
Response	Same as Get (Active)

17.4 Set

Request	&mode={thrmo_mode}&manf_dat_list="{thrmo_manf_dat_list}"
	{intf_cmd} is CMD_THRMO_MODE_SET {thrmo_manf_dat_list} is optional and only applicable for "Manufacturer-Specific mode" for Version 3 or above Set command.
Response	Same as Select except {intf_cmd} is replaced accordingly

17.5 Get Capabilities (Active)

Request	{intf_cmd} is CMD_THRMO_MODE_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_md_sup utime="{updt_time}" list="{thrmo_mode_list}" /> </zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly

17.6 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_THRMO_MODE_SUP_REPORT
Response	Same as Get Capabilities (Active)

18 Thermostat Operating State Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_thrmo_op_sta
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Thermostat Operating State CC Command field	
{intf_cmd}	Commands below	
	CMD_THRMO_OP_STATE_SETUP	1
	CMD_THRMO_OP_STATE_GET	2
	CMD_THRMO_OP_STATE_REPORT	3
	CMD_THRMO_OP_STATE_SUP_GET	4
	CMD_THRMO_OP_STATE_SUP_REPORT	5
	CMD_THRMO_OP_STATE_LOG_GET	6
	CMD_THRMO_OP_STATE_LOG_REPORT	7
{thrmo_op_state}	Operating State field of Report Command	
{thrmo_op_log_list}	Comma separated list of operating states (Bit Mask field of Logging Get command and Logging Supported Report command)	
{log_todayhours}	Usage Today (Hours) field of Logging Report Command	
{log_todayminutes}	Usage Today (Minutes) field of Logging Report Command	
{log_yesterdayhours}	Usage Yesterday (Hours) field of Logging Report Command	
{log_yesterdayminutes}	Usage Yesterday (Minutes) field of Logging Report Command	

18.1 Select

Request	{intf_cmd} is CMD_THRMO_OP_STATE_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - ‘tServBasic’ is replaced with ‘tServThrmoOpState’

18.2 Get (Active)

Request	{intf_cmd} is CMD_THRMO_OP_STATE_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <thrmo_op_sta utime=“{updt_time}” state=“{thrmo_op_state}” /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

18.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_THRMO_OP_STATE_REPORT
Response	Same as Get (Active)

18.4 Logging Get (Active)

Request	&log_list={thrmo_op_log_list}
	{intf_cmd} is CMD_THRMO_OP_STATE_LOG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<thrmo_op_sta_log utime="{updt_time}" state="{thrmo_op_state}" todayhours="{log_todayhours}" todayminutes="{log_todayminutes}" yesterdayhours="{log_yesterdayhours}" yesterdayminutes="{log_yesterdayminutes}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

18.5 Logging Get (Passive)

Request	&log_list={thrmo_op_log_list}
	{intf_cmd} is CMD_THRMO_OP_STATE_LOG_REPORT
Response	Same as Logging Get (Active)

18.6 Logging Supported Get (Active)

Request	{intf_cmd} is CMD_THRMO_OP_STATE_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_op_sta_log_sup utime="{updt_time}" log_sup_list="{thrmo_op_log_list}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

18.7 Logging Supported Get (Passive)

Request	{intf_cmd} is CMD_THRMO_OP_STATE_SUP_REPORT
Response	Same as Logging Supported Get (Active)

19 Thermostat Fan Mode Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_thrmo_fan_md
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Thermostat Fan Mode CC Command field	
{intf_cmd}	Commands below	
	CMD_THRMO_FAN_MODE_SETUP	1
	CMD_THRMO_FAN_MODE_GET	2
	CMD_THRMO_FAN_MODE_REPORT	3
	CMD_THRMO_FAN_MODE_SET	4
	CMD_THRMO_FAN_MODE_SUP_GET	5
	CMD_THRMO_FAN_MODE_SUP_REPORT	6
{thrmo_fan_mode}	Fan Mode field of Report/Set Command	
{thrmo_fan_off}	Off field of Report command version >= 3. Otherwise undefined. Off field of Set command version >= 2. Defaults to 0. The value is 1 if fan is fully off; 0 if otherwise.	
{thrmo_fan_mode_list}	Comma separated list of supported thermostat fan modes – Bit Mask field of Supported Report command	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

19.1 Select

Request	{intf_cmd} is CMD_THRMO_FAN_MODE_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServThrmoFanMode'

19.2 Get (Active)

Request	{intf_cmd} is CMD_THRMO_FAN_MODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_fan_md utime="{updt_time}" mode="{thrmo_fan_mode}" off="{thrmo_fan_off}" state_num="{state_num}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when either {thrmo_fan_mode} or {thrmo_fan_off} is updated.

19.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_THRMO_FAN_MODE_REPORT
Response	Same as Get (Active)

19.4 Set

Request	&mode={thrmo_mode}[&off={thrmo_fan_off}]
	{intf_cmd} is CMD_THRMO_FAN_MODE_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

19.5 Get Capabilities (Active)

Request	{intf_cmd} is CMD_THRMO_FAN_MODE_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_fan_md_sup utime="{updt_time}" list="{thrmo_fan_mode_list}" /> </zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly

19.6 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_THRMO_FAN_MODE_SUP_REPORT
Response	Same as Get Capabilities (Active)

20 Thermostat Fan State Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_thrmo_fan_sta
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Thermostat Fan State CC Command field	
{intf_cmd}	Commands below	
	CMD_THRMO_FAN_STATE_SETUP	1
	CMD_THRMO_FAN_STATE_GET	2
	CMD_THRMO_FAN_STATE_REPORT	3
{thrmo_fan_state}	Fan Operating State field of Report Command	

20.1 Select

Request	{intf_cmd} is CMD_THRMO_FAN_STATE_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServThrmoFanState'

20.2 Get (Active)

Request	{intf_cmd} is CMD_THRMO_FAN_STATE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_fan_sta utime="{updt_time}" state="{thrmo_fan_state}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

20.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_THRMO_FAN_STATE_REPORT
Response	Same as Get (Active)

21 Thermostat Setpoint Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_thrmo_setp
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Thermostat Setpoint CC Command field	
{intf_cmd}	Commands below	
	CMD_THRMO_SETPT_SETUP	1
	CMD_THRMO_SETPT_GET	2
	CMD_THRMO_SETPT_REPORT	3
	CMD_THRMO_SETPT_SET	4
	CMD_THRMO_SETPT_SUP_GET	5
	CMD_THRMO_SETPT_SUP_REPORT	6
	CMD_THRMO_SETPT_RANGE_GET	7
	CMD_THRMO_SETPT_RANGE_REPORT	8
{setpt_type}	Setpoint Type field of Report/Set command	
{setpt_value}	Value field of Report command after applying ‘Size’ and ‘Precision’ fields	
{setpt_value_mag}	Value field of Set command (without applying ‘Size’ and ‘Precision’ fields)	
{setpt_precision}	Precision field of Report/Set command	
{setpt_unit}	Scale field of Report/Set command	
{setpt_size}	Size field of Report/Set command	
{thrmo_setpt_list}	Comma separated list of supported thermostat setpoint types – Bit Mask field of Supported Report command	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when ’utime’ is not 0.	

21.1 Select

Request	{intf_cmd} is CMD_THRMO_SETPT_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - ‘tServBasic’ is replaced with ‘tServThrmoSetpt’

21.2 Get (Active)

Request	&type={setpt_type}
	{intf_cmd} is CMD_THRMO_SETPT_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> [<thrmo_setp utime=“{updt_time}” type=”{setpt_type}” value=”{setpt_value}” precision=”{setpt_precision}” unit=”{setpt_unit}” size=”{setpt_size}”

	state_num="{state_num}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" is based on setpoint Type. Different setpoint Type has its own {state_num}. {state_num} will be updated when any of the following is updated: {value}, {precision}, {unit}, {size}.

21.3 Get (Passive)

Request	&type={setpt_type}
	{intf_cmd} is CMD_THRMO_SETPT_REPORT
Response	Same as Get (Active)

21.4 Set

Request	&type={setpt_type}&value={setpt_value_mag} &precision={setpt_precision}&unit={setpt_unit}
	{intf_cmd} is CMD_THRMO_SETPT_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

21.5 Get Capabilities (Active)

Request	{intf_cmd} is CMD_THRMO_SETPT_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <thrmo_setp_sup utime="{updt_time}" list="{thrmo_setpt_list}" /> </zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly

21.6 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_THRMO_SETPT_SUP_REPORT
Response	Same as Get Capabilities (Active)

21.7 Range Get (Active)

Request	&type={setpt_type}
	{intf_cmd} is CMD_THRMO_SETPT_RANGE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<thrmo_setp_range utime="{updt_time}" type="{setpt_type}" min_value="{setpt_value}" min_precision="{setpt_precision}" min_unit="{setpt_unit}" min_size="{setpt_size}" max_value="{setpt_value}" max_precision="{setpt_precision}" max_unit="{setpt_unit}"

	max_size="{setpt_size}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly Request parameter "type" is mandatory

21.8 Range Get (Passive)

Request	Same as the Range Get (Active) except {intf_cmd} is CMD_THRMO_SETPT_RANGE_REPORT
Response	Same as the Range Get (Active)
Note	Same as the Passive Get command, the request parameter "type" in this API is optional. When the request consists of specific "type", the response will only contain the report for the specified "type". When no "type" is specified in the request, the response will contain all the cached range reports that server is holding. The list of cached range reports may or may not contain the full set of thermostat setpoint types. In the "version agnostic" way, the client should get the supported type first. Then issue active range gets for individual type. Subsequently just issue passive range get without any parameter to obtain any changes on individual values.

22 Alarm Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_alrm
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Alarm CC Command field	
{intf_cmd}	Commands below	
	CMD_ALARM_SETUP	1
	CMD_ALARM_GET	2
	CMD_ALARM_REPORT	3
	CMD_ALARM_SET	4
	CMD_ALARM_TYPE_SUP_GET	5
	CMD_ALARM_TYPE_SUP_REPORT	6
	CMD_ALARM_EVENT_SUP_GET	7
	CMD_ALARM_EVENT_SUP_REPORT	8
{alarm_vtype}	Alarm Type field of Get Command	
{alarm_ztype}	Z-Wave Alarm Type field of Get Command v2. Defaults to 0	
{alarm_level}	Alarm Level field of Report Command	
{alarm_ext}	if following extended information is valid. 1 if valid (v2); 0 if invalid (v1)	
{alarm_zensor}	Zensor Net Source Node ID field of Report Command v2. Otherwise undefined.	
{alarm_status}	Z-Wave Alarm Status field of Report/Set Command v2; Otherwise undefined.	
{alarm_event}	Z-Wave Alarm Event field of Report Command v2; Otherwise undefined.	
{alarm_eparam_len}	event parameter length in bytes for version >= 2; Otherwise undefined.	
{alarm_eparam_type}	1 if Node Location Report; 2 if User Identifier of User Code Report. Applicable only for class version >= 2; Otherwise, undefined.	
{alarm_eparam}	Event Parameter field of Report Command v2; Otherwise undefined. If the event parameter type is Node Location Report, the location is UTF-8 encoded and then URL encoded.	
{alarm_have_vtype}	V1 Alarm field of Type Supported Report Command	
{alarm_list}	comma separated list of Z-Wave Alarm Types – Bit Mask field of Type Supported Report Command	

22.1 Select

Request	{intf_cmd} is CMD_ALARM_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServAlarm'

22.2 Get (Active)

Request	&vtype={alarm_vtype}[&ztype={alarm_ztype}[&event={alarm_event}]]
	{intf_cmd} is CMD_ALARM_GET
Response	On success: On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<alrm utime="{updt_time}" vtype="{alarm_vtype}" level="{alarm_level}" ext="{alarm_ext}" zensor_nodeid="{alarm_zensor}" status="{alarm_status}" ztype="{alarm_ztype}" event="{alarm_event}" eparam_len="{alarm_eparam_len}" eparam_type="{alarm_eparam_type}" eparam="{alarm_eparam}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

For request,
v1 Alarm CC will be: &vtype={alarm_vtype}
v2 Alarm CC will be: &vtype={alarm_vtype}&ztype={alarm_ztype}
v3 or above Alarm CC will be: &vtype={alarm_vtype}&ztype={alarm_ztype}&event={alarm_event}

For response, if request ztype is 0xFF, only the latest alarm report will be returned.
Otherwise the server may return a list of alarm reports that are deemed to match the request.
E.g., Alarm report with the same ztype and event as specified in the request. Or,
Alarm report with the same ztype but event 0, or event 0xFE. Or,
Alarm report with status 0xFE, ztype 0 and event 0.

The list of these alarm report can not be used as a complete history log of the alarm reports received by the server, as the newer report will always overwrite the history of an older report with the same ztype and event.

22.3 Get (Passive)

Request	&vtype={alarm_vtype}[&ztype={alarm_ztype}]
	{intf_cmd} is CMD_ALARM_REPORT
Response	Same as Get (Active)

22.4 Set

Request	&ztype={alarm_ztype}&status={alarm_status}
	{intf_cmd} is CMD_ALARM_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	Applicable for Alarm CC version >= 2, otherwise behavior undefined

22.5 Get Capabilities (Active)

Request	{intf_cmd} is CMD_ALARM_TYPE_SUP_GET (Appendix B.4)
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}">

	<alrm_sup utime="{updt_time}" has_vtype="{alarm_have_vtype}" list="{alarm_list}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Alarm CC version >= 2, otherwise behavior undefined

22.6 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_ALARM_TYPE_SUP_REPORT
Response	Same as Get Capabilities (Active)
Note	Applicable for Alarm CC version >= 2, otherwise behavior undefined

22.7 Get Supported Events (Active)

Request	&ztype={alarm_ztype} {intf_cmd} is CMD_ALARM_EVENT_SUP_GET (Appendix B.4)
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <alrm_sup utime="{updt_time}" ztype="{alarm_ztype}" list="{event_list}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for Alarm CC version >= 3, otherwise behavior undefined

22.8 Get Supported Events (Passive)

Request	Same as Get Supported Events (Active) except {intf_cmd} is CMD_ALARM_EVENT_SUP_REPORT
Response	Same as Get Supported Events (Active)
Note	Applicable for Alarm CC version >= 3, otherwise behavior undefined

23 Wake Up Interface API

Note: It is not recommended that the Client use this API as the newer underlying ZIPGW had a mailbox feature that relies on these settings.

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_wakeup
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Wake Up CC Command field	
{intf_cmd}	Commands below	
	CMD_WAKE_UP_SETUP	1
	CMD_WAKE_UP_GET	2
	CMD_WAKE_UP_REPORT	3
	CMD_WAKE_UP_SET	4
{wkup_intvl}	Seconds field of Interval Report Command	
{wkup_node_id}	NodeID field of Interval Report Command, 255 for broadcast.	
{wkup_min}	Minimum Wake Up Interval Seconds field of Interval Capabilities Report Command, applicable only for version >= 2; Otherwise, undefined.	
{wkup_max}	Maximum Wake Up Interval Seconds field of Interval Capabilities Report Command, applicable only for version >= 2; Otherwise, undefined.	
{wkup_def}	Default Wake Up Interval Seconds field of Interval Capabilities Report Command, applicable only for version >= 2; Otherwise, undefined.	
{wkup_step}	Wake Up Interval Step Seconds field of Interval Capabilities Report Command, applicable only for version >= 2; Otherwise, undefined.	
{wkup_node_desc}	node descriptor to receive wake up notification: 255 for broadcast	

23.1 Select

Request	{intf_cmd} is CMD_WAKE_UP_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServWkup'

23.2 Get (Active)

Request	{intf_cmd} is CMD_WAKE_UP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <wakeup utime="{updt_time}" intvl="{wkup_intvl}" node_id="{wkup_node_id}" min="{wkup_min}" max="{wkup_max}" def="{wkup_def}" step="{wkup_step}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

23.3 Get (Passive)

Request	{intf_cmd} is CMD_WKUP_REPORT
Response	Same as Get (Active)

23.4 Set

Request	&intvl={wkup_intvl}&noded={wkup_node_desc}
	{intf_cmd} is CMD_WAKE_UP_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

24 Battery Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_battery
Request	Same as for Basic Interface

The following request/responses parameters are used in this section:

Parameter	Battery CC Command field	
{intf_cmd}	Commands below	
	CMD_BATTERY_SETUP	1
	CMD_BATTERY_GET	2
	CMD_BATTERY_REPORT	3
	CMD_BATTERY_HEALTH_GET	4
	CMD_BATTERY_HEALTH_REPORT	5
{battery_level}	Battery Level field of Battery Report Command	
{rechargeable}	Rechargeable field in Battery Report Command. Only for device ver2 or above.	
{charge_sts}	Battery Charing Status field in Battery Report Command. Only for device ver2 or above.	
{bkup_batt}	Back-up battery field in Battery Report Command. Only for device ver2 or above.	
{overheat}	Overheating field in Battery Report Command. Only for device ver2 or above.	
{low_fluid}	Low fluid field in Battery Report Command. Only for device ver2 or above.	
{rechg_req}	Replace/recharge status field in Battery Report Command. Only for device ver2 or above.	
{disconnect}	Disconnected field in Battery Report Command. Only for device ver2 or above.	
{low_temp_sts}	Low temperature Status field in Battery Report Command. Only for device ver3 or above. The value is 0 when status is unknown. Value 1 indicates that battery is not charging due to low temperature. Value 2 indicates that battery is operational.	
{max_cap}	Maximum Capacity field in Battery Health Report Command.	
{batt_temp_value}	Battery temperature values calculated based on precision and size in Battery Health Report Command.	
{precision}	Precision field in Battery Health Report Command.	
{unit}	Scale field in Battery Health Report Command.	
{size}	Size field in Battery Health Report Command.	

24.1 Select

Request	{intf_cmd} is CMD_BATTERY_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServBattery'

24.2 Get (Active)

Request	{intf_cmd} is CMD_BATTERY_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <battery utime="{updt_time}" level="{battery_level}" rechargeable="{rechargeable}" charge_sts="{charge_sts}" bkup_batt="{bkup_batt}" overheat="{overheat}" low_fluid="{low_fluid}" rechg_req="{rechg_req}" disconnect="{disconnect}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Depends on the device version, the api response may vary. For a ver 1 device, the response will be (example): <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <battery utime="1546851812" level="50" /> </zwif></zwave> For ver 2 or above device, the response will be (example): <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <battery utime="1546851812" level="50" rechargeable="1" charge_sts="1" bkup_batt="0" overheat="0" low_fluid="0" rechg_req="1" disconnect="0" /> </zwif></zwave>

24.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_BATTERY_REPORT
Response	Same as Get (Active)

24.4 Health Get (Active)

Request	{intf_cmd} is CMD_BATTERY_HEALTH_GET
Response	<?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <battery_health utime="{updt_time}" max_cap="{max_cap}" batt_temp="{batt_temp_value}" precision="{precision}" unit="{unit}" size="{size}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	When {size} is 0, client should ignore the value in {batt_temp_value}, {precision} and {unit} and treat battery temperature as 'unknown' as per Command Class Spec.

24.5 Health Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_BATTERY_HEALTH_REPORT
Response	Same as Get (Active)

25 Central Scene Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_centr_scene
Request	Same as for Basic Interface

Note that there is no Active Get available for this command class, to conform to the CC Spec.

The following request/responses parameters are used in this section:

Parameter	Central Scene CC Command field	
{intf_cmd}	Commands below	
	CMD_CENTRAL_SCENE_SETUP	1
	CMD_CENTRAL_SCENE_REPORT	3
	CMD_CENTRAL_SCENE_SUP_GET	4
	CMD_CENTRAL_SCENE_SUP_REPORT	5
	CMD_CENTRAL_SCENE_CONFIG_GET	6
	CMD_CENTRAL_SCENE_CONFIG_REPORT	7
	CMD_CENTRAL_SCENE_CONFIG_SET	8
{sequence_number}	Sequence number field of Central Scene Notification Command	
{key_attribute}	Key attribute field of Central Scene Notification Command	
{scene_number}	Scene number field of Central Scene Notification Command. Or, in Get Capabilities response, it indicates the scene/key/button number to which the supported key attribute list belongs.	
{scene_count}	Supported Scenes field of Supported Report Command	
{same_key_attribute}	Identical field of Supported Report Command v2. Defaults to 1.	
{supported_key_attribute_list}	Comma separated list of supported key attributes – Bit Mask field of Supported Report Command	
{slow_refresh}	Status for slow refresh of Key Held Down notification. Non-zero=enable; 0=disable	

25.1 Select

Request	{intf_cmd} is CMD_CENTRAL_SCENE_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly

25.2 Get (Passive)

Request	{intf_cmd} is CMD_CENTRAL_SCENE_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <centr_scene utime="{updt_time}" sequence_number="{sequence_number}" key_attribute="{key_attribute}" scene_number="{scene_number}" slow_refresh="{slow_refresh}"/> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

25.3 Get Capabilities (Active)

Request	{intf_cmd} is CMD_CENTRAL_SCENE_SUP_GET (Appendix B.4)
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <centr_scene_sup utime="{updt_time}" scene_count="{scene_count}" same_key_attribute="{same_key_attribute}" slow_refresh="{slow_refresh}"> [<key_attribute_sup scene_number="{scene_number}" list="{supported_key_attribute_list}" />] </centr_scene_sup> </zwif></zwave> "{slow_refresh}" value 1 indicates that the devices support Slow Refresh capability. Value 0 indicates otherwise.
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	<key_attribute_sup> xml section is only available for Central Scene CC version >= 2.

Sample response:

- V1 Central Scene Command Class, no <key_attribute_sup> section will be returned. The sample response is:

```
<?xml version="1.0"?><zwave><zwif desc="{intf_desc}">
    <centr_scene_sup utime="465846513" scene_count="5" same_key_attribute="1">
    </centr_scene_sup>
</zwif></zwave>
```

- V2 Central Scene Command Class. All the scenes support the same key attributes. Therefore only 1 <key_attribute_sup> section will be returned.
  Sample resonse:

```
<?xml version="1.0"?><zwave><zwif desc="{intf_desc}">
    <centr_scene_sup utime="465846513" scene_count="5" same_key_attribute="1">
            <key_attribute_sup scene_number="1" list="0,1,2,3" />
    </centr_scene_sup>
</zwif></zwave>
```

- V2 Central Scene CC. Different scenes support the different key attributes. Individual <key_attribute_sup> section for different scene will be returned.
  Sample response:

```
<?xml version="1.0"?><zwave><zwif desc="{intf_desc}">
    <centr_scene_sup utime="465846513" scene_count="3" same_key_attribute="1">
            <key_attribute_sup scene_number="1" list="0,1,2,3" />
            <key_attribute_sup scene_number="2" list="0,3" />
            <key_attribute_sup scene_number="3" list="0,3" />
    </centr_scene_sup>
</zwif></zwave>
```

25.4 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_CENTRAL_SCENE_SUP_REPORT
Response	Same as Get Capabilities (Active)

25.5 Get Configuration (Active)

Request	{intf_cmd} is CMD_CENTRAL_SCENE_CONFIG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <centr_scene_config utime="{updt_time}" slow_refresh="{slow_refresh}"> </centr_scene_config> </zwif></zwave> "{slow_refresh}" value 1 indicates that the devices uses Slow Refresh. Value 0 indicates otherwise.
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	This API is only available for Central Scene CC version >= 3

25.6 Get Configuration (Passive)

Request	Same as Get Configuration (Active) except {intf_cmd} is CMD_CENTRAL_SCENE_CONFIG_REPORT
Response	Same as Get Configuration (Active)

25.7 Set Configuration

Request	&slow_refresh={slow_refresh}
	{intf_cmd} is CMD_CENTRAL_SCENE_CONFIG_SET "{slow_refresh}" value 1 indicates that the devices uses Slow Refresh. Value 0 indicates otherwise.
Response	Same as Select except {intf_cmd} is replaced accordingly

26 Barrier Operator Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_barrier_op
Request	Same as for Basic Interface

The following request/responses parameters are used in this section with references to fields in the Barrier Operator CC commands:

Parameter	Barrier Operator CC reference	
{intf_cmd}	Commands below	
	CMD_BARRIER_OP_SETUP	1
	CMD_BARRIER_OP_GET	2
	CMD_BARRIER_OP_REPORT	3
	CMD_BARRIER_OP_SET	4
	CMD_BARRIER_OP_SUBSYS_SUP_GET	5
	CMD_BARRIER_OP_SUBSYS_SUP_REPORT	6
	CMD_BARRIER_OP_SUBSYS_GET	7
	CMD_BARRIER_OP_SUBSYS_REPORT	8
	CMD_BARRIER_OP_SUBSYS_SET	9
{target_state}	'Target Value' field in Set command	
{barrier_op_state}	'Value' field in Report commands	
{subsys_type_list}	Comma separated list of supported barrio operator subsystem types – Bit Mask field of Supported Report command	
{subsys_type}	"Subsystem Type" field in Event Signal Set, Get and Report commands	
{subsys_state}	"Subsystem State" field in Event Signal Set and Report commands	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

26.1 Select

Request	{intf_cmd} is CMD_ BARRIER_OP_SETUP
Response	Behavior is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServBarrierOp' in the response

26.2 Get (Active)

Request	{intf_cmd} is CMD_BARRIER_OP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}">

	<barrier_op utime="{updt_time}" state="{barrier_op_state}" state_num="{state_num}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when {barrier_op_state} is updated.

26.3 Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_BARRIER_OP_REPORT
Response	Same as Get (Active)

26.4 Set

Request	&state={target_state}
	{intf_cmd} is CMD_BARRIER_OP_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

26.5 Get Subsystem Capabilities (Active)

Request	{intf_cmd} is CMD_BARRIER_OP_SUBSYS_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> < barrier_op_subsys_sup utime="{updt_time}" list="{subsys_type_list}" /> </zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly

26.6 Get Subsystem Capabilities (Passive)

Request	Same as Get Subsystem Capabilities (Active) except {intf_cmd} is CMD_BARRIER_OP_SUBSYS_SUP_REPORT
Response	Same as Get Capabilities (Active)

26.7 Get Subsystem Configuration (Active)

Request	&type={subsys_type}
	{intf_cmd} is CMD_BARRIER_OP_SUBSYS_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<barrier_op_subsys utime="{updt_time}" type="{subsys_type}" state="{subsys_state}" state_num="{state_num}" />]

	</zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" is based on subsys type. Different subsys types have its own {state_num}. {state_num} will be updated when {subsys_state} is changed within a subsys type.

26.8 Get Subsystem Configuration (Passive)

Request	&type={subsys_type}
	{intf_cmd} is CMD_BARRIER_OP_SUBSYS_REPORT
Response	Same as Get Subsystem Config (Active)

26.9 Set Subsystem Configuration

Request	&type={subsys_type}&state={ subsys_state}
	{intf_cmd} is CMD_BARRIER_OP_SUBSYS_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

27 Sound Switch Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_sound_sw
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Sound Switch CC commands:

Parameter	Sound Switch CC reference	
{intf_cmd}	Commands below	
	CMD_SOUND_SWITCH_SETUP	1
	CMD_SOUND_SWITCH_TONE_PLAY_GET	2
	CMD_SOUND_SWITCH_TONE_PLAY_REPORT	3
	CMD_SOUND_SWITCH_TONE_PLAY_SET	4
	CMD_SOUND_SWITCH_TONE_INFO_GET	5
	CMD_SOUND_SWITCH_TONE_INFO_REPORT	6
	CMD_SOUND_SWITCH_TONE_CONFIG_GET	7
	CMD_SOUND_SWITCH_TONE_CONFIG_REPORT	8
	CMD_SOUND_SWITCH_TONE_CONFIG_SET	9
{tone_id}	Tone ID in the commands	
{volume}	Volume setting in percentage of the device. Ranges in 0..100.	
{def_tone_id}	Default tone ID	
{tone_duration}	Tone duration in seconds. Max 16 bit (65536 seconds).	
{name}	Name or label string of the tone. Maximum length 255.	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	
{tone_volume}	'Play command tone volume' in Sound Switch Tone Play Set or Sound Switch Tone Play Get command. 0: use the configured current volume at the node (Sound Switch Configuration Set Command) 1..100: Actual volume setting from 1% to 100%. 255: use most recent non-zero volume setting if the current volume is muted (0x00)	

27.1 Select

Request	{intf_cmd} is CMD_SOUND_SWITCH_SETUP
Response	Behaviour is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServSoundSw' in the response

27.2 Tone Play Get (Active)

Request	{intf_cmd} is CMD_SOUND_SWITCH_TONE_PLAY_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <sound_sw utime="{updt_time}" tone_id="{tone_id}" tone_volume="{tone_volume}" state_num="{state_num}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	{tone_id} indicates currently played tone. 0 means no tone is currently being played. "{state_num}" will be updated when {tone_id} is updated.

27.3 Tone Play Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_SOUND_SWITCH_TONE_PLAY_REPORT
Response	Same as Tone Play Get (Active)

27.4 Tone Play Set

Request	&tone_id={tone_id}&[tone_volume={tone_volume}]
	{intf_cmd} is CMD_SOUND_SWITCH_TONE_PLAY_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	{tone_id} value: 0 : Stop playing any tone 1-254 : Play the specified tone id. If the tone id is not supported by the device, default tone will be played. 255 : Play default tone. {tone_volume} is optional and only applicable for ver 2 or above.

27.5 Tone Info Get (Active)

Request	[&tone_id={tone_id}]
	{intf_cmd} is CMD_SOUND_SWITCH_TONE_INFO_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <sound_sw_tone_info utime="{updt_time}" > [<tone_info tone_id=={tone_id} tone_duration={tone_duration} tone_name={name} />] </sound_sw_tone_info></zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly
Note	{tone_id} in request is optional. If {tone_id} is missing in request, it will be treated as value 0, which means to retrieve all the tone information.

27.6 Tone Info Get (Passive)

Request	Same as Tone Info Get (Active) except {intf_cmd} is CMD_SOUND_SWITCH_TONE_CONFIG_REPORT
Response	Same as Tone Info Get (Active)

27.7 Tone Configuration Get (Active)

Request	{intf_cmd} is CMD_SOUND_SWITCH_TONE_CONFIG_GET
	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <sound_sw_config utime="{updt_time}" volume="{volume}" def_tone_id="{def_tone_id}" state_num="{state_num}" /> </zwif></zwave>
Response	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" will be updated when either {volume} or {def_tone_id} is updated.

27.8 Tone Configuration Get (Passive)

Request	Same as Tone Config Get (Active) except {intf_cmd} is CMD_SOUND_SWITCH_TONE_CONFIG_REPORT
Response	Same as Tone Config Get (Active)

27.9 Tone Configuration Set

Request	[&volume={volume}][&def_tone_id={def_tone_id}]
	{intf_cmd} is CMD_SOUND_SWITCH_TONE_CONFIG_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	Both *volume* and *def_tone_id* is optional. However at least 1 param should be present in the request. If *volume* is missing, it will be treated as value 255. If *def_tone_id* is missing, it will be treated as value 0. {volume} value: 0 : Off/Mute. 1..100 : Volume in percentage. 255 : Restore most recent non-zero volume setting / set default tone only without modify volume. {def_tone_id} value: 0 : No modification on default tone. Other values : Set the specified id as default tone.

28 Indicator Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_ind
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Indicator CC commands:

Parameter	Indicator CC reference	
{intf_cmd}	Commands below	
	CMD_INDICATOR_SETUP	1
	CMD_INDICATOR_GET	2
	CMD_INDICATOR_REPORT	3
	CMD_INDICATOR_SET	4
	CMD_INDICATOR_SUP_GET	5
	CMD_INDICATOR_SUP_REPORT	6
{indicator_state}	'Indicator 0 Value' field in Report and Set commands, valid only if the device is a ver1 device.	
{indicator_id}	Indicator ID in the commands. Only for device ver2 or above.	
{property_id}	Property ID in the commands. Only for device ver2 or above.	
{property_value}	Property value in the commands. Only for device ver2 or above.	
{property_id_list}	Comma separated list of property IDs. Only for device ver2 or above.	
{property_value_list}	Comma separated list of property values. Only for device ver2 or above.	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is incremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	

28.1 Select

Request	{intf_cmd} is CMD_INDICATOR_SETUP
Response	Behaviour is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServIndctr' in the response

28.2 Get (Active)

Request	[&ind_id={indicator_id}]
	{intf_cmd} is CMD_INDICATOR_GET

Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <ind utime="{updt_time}" value="{indicator_state}" ind_id="{indicator_id}" state_num="{state_num}" > [<ind_property property_id="{property_id}" property_value="{property_value}" />] [<ind_property … … />] </ind> </zwif></zwave>
	Failure is the same as for Select except (intf_cmd} is replaced accordingly
Note	{indicator_id} in the request is only mandatory for ver2 or above device. Absence of this param will be treated as ind_id is 0 (assuming the UI client only supports ver 1 Indicator CC). Depending on the device version, the API response may vary. For a ver 1 device, the response will be (example): <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <ind utime="1546851812" value="3" ind_id="0" > </ind> </zwif></zwave> For ver 2 or above device, the response will be (example): <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <ind utime="1546851812" value="0" ind_id="67" > <ind_property property_id="3" property_value="8" /> <ind_property property_id="4" property_value="100" /> <ind_property property_id="5" property_value="3" /> </ind> </zwif></zwave> 'value' in this case is only for backward compatibility and the actual value itself should be ignored. UI should use the presence of <ind_property> to determine whether 'value' field should be respected. "{state_num}" is based on Indicator ID. Different Indicator ID has its own {state_num}. {state_num} will be updated when any of the property values do not match the previous value.

28.3 Get (Passive)

Request	Same as Get (Active) except {intf_cmd} is CMD_INDICATOR_REPORT
Response	Same as Get (Active)

28.4 Set

Request	[&value={indicator_state}][&ind_id="{indicator_id}"&property_id_list="{property_id_list}"

	&property_value_list=”{property_value_list}”]
	{intf_cmd} is CMD_INDICATOR_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	{indicator_state} is only mandatory for ver 1 device. {indicator_id}, ”{property_id_list}” and ”{property_value_list}” in the request is only mandatory for ver2 or above device. The number of components in ”{property_value_list}” should match what is in ”{property_id_list}”. For example, for ver1 device, the request should be: &value=3 For ver2 device, the request should be: &ind_id=67 &property_id_list=”3,4,5,” &property_value_list=”8,100,3,” Absence of a parameter is treated as 0.

28.5 Get Capabilities (Active)

Request	{intf_cmd} is CMD_INDICATOR_SUP_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <ind_sup utime=“{updt_time}” > [<ind_property_sup ind_id=”{indicator_id} property_id_list=”{property_id_list}” />] [<ind_property_sup … … />] </ind_sup> </zwif></zwave>
	Failure is the same as for Select except (intf_cmd} is replaced accordingly

28.6 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_INDICATOR_SUP_REPORT
Response	Same as Get Capabilities (Active)

29 Time Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_time
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Time CC commands:

Parameter	Sound Switch CC reference	
{intf_cmd}	Commands below	
	CMD_TIME_SETUP	1
	CMD_TIME_GET	2
	CMD_TIME_REPORT	3
	CMD_TIME_DATE_GET	4
	CMD_TIME_DATE_REPORT	5
	CMD_TIME_TZ_DST_GET	6
	CMD_TIME_TZ_DST_REPORT	7

29.1 Select

Request	{intf_cmd} is CMD_TIME_SETUP
Response	Behaviour is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServTime' in the response

29.2 Time Get (Active)

Request	{intf_cmd} is CMD_TIME_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <time utime="{updt_time}" hour="{hour}" minute="{minute}" second="{second}" rtc_fail="{rtc_fail}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	{hour}: in 24 hours format, in the range 0..23. {minute}: in the range 0..59. {second}: in the range 0..59. {rtc_fail}: Flag to indicate if RTC oscillator has been stopped and hence the advertised time might be inaccurate. 1=stopped; 0=running or node does not support this feature

29.3 Time Get (Passive)

Request	Same as Basic Get (Passive) except {intf_cmd} is CMD_TIME_REPORT
Response	Same as Time Get (Active)

29.4 Date Get (Active)

Request	{intf_cmd} is CMD_TIME_DATE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <time_date utime="{updt_time}" year="{year}" month="{month}" day="{day}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	{year}: e.g., 2018. {month}: in the range 1..12. {day}: in the range 1..31.

29.5 Date Get (Passive)

Request	Same as Date Get (Active) except {intf_cmd} is CMD_TIME_DATE_REPORT
Response	Same as Date Get (Active)

29.6 Time Zone & Daylight Savings Time Get (Active)

Request	{intf_cmd} is CMD_TIME_TZ_DST_GET
	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <time_tz_dst utime="{updt_time}" tz_sign="{tz_sign}" tz_hour="{tz_hour}" tz_minute="{tz_minute}" dst_sign="{dst_sign}" dst_minute_offset="{dst_minute_offset}" dst_month_start="{dst_month_start}" dst_day_start="{dst_day_start}" dst_hour_start="{dst_hour_start}" dst_month_end="{dst_month_end}" dst_day_end="{dst_day_end}" dst_hour_end="{dst_hour_end}" /> </zwif></zwave>
Response	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	{tz_sign}: Sign (plus or minus) to apply to the {tz_hour} and {tz_minute} fields. 0 = Plus sign (positive offset from UTC); 1 = Minus sign (negative offset from UTC) {dst_sign}: Sign (plus or minus) for the {dst_minute_offset} field to apply to the current time while in the Daylight Savings Time. 0 = Plus sign (positive offset from current time); 1 = Minus sign (negative offset from current time) {dst_month_start}: Month of the year when Daylight Savings Time starts. Range 1..12 (representing respectively January...December). {dst_day_start}: Day of the month when DST starts. Range 1..31.

	{dst_hour_start}: Hour of the day when DST starts. Range 0..23. {dst_month_end}: Month of the year when Daylight Savings Time ends. Range 1..12 (representing respectively January...December). {dst_day_end}: Day of the month when DST ends. Range 1..31. {dst_hour_end}: Hour of the day when DST ends. Range 0..23.

29.7 Time Zone & Daylight Savings Time Get (Passive)

Request	Same as Time Zone & Daylight Savings Time Get (Active) except {intf_cmd} is CMD_TIME_TZ_DST_REPORT
Response	Same as Time Zone & Daylight Savings Time Get (Active)

30 Window Covering Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_window_cvr
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Window Covering CC commands:

Parameter	Window Covering CC reference	
{intf_cmd}	Commands below	
	CMD_WINDOW_COVERING_SETUP	1
	CMD_WINDOW_COVERING_GET	2
	CMD_WINDOW_COVERING_REPORT	3
	CMD_WINDOW_COVERING_SET	4
	CMD_WINDOW_COVERING_LEVEL_CHANGE_SET	6
	CMD_WINDOW_COVERING_SUP_GET	7
	CMD_WINDOW_COVERING_SUP_REPORT	8
{param_id}	Window Covering parameter IDs (Parm ID)	
{curr_value}	Current value in Window Covering Report Command	
{target_value}	Target value in Window Covering Report Command	
{duration}	The time needed to reach the Target Value. 0 : 0 seconds. Already at the Target Value. 0x01..0x7F (1~127) : 1~127 seconds. 0x80..0xFE (128~254) : 1~127 mins. 0xFF (255) : Factory default duration	
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	
{param_id_list}	Comma separate list of param_ids.	
{param_value_list}	Comma separate list of param values.	
{direction}	Up/Down bit in the Window Covering Start Level Change command. 0 : Level change increasing. 1 : Level change decreasing.	

30.1 Select

Request	{intf_cmd} is CMD_WINDOW_COVERING_SETUP
Response	Behaviour is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServWindowCvr' in the response

30.2 Get (Active)

Request	&param_id={param_id}
	{intf_cmd} is CMD_WINDOW_COVERING_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <window_cvr utime="{updt_time}" param_id="{param_id}" curr_value="{curr_value}" target_value="{target_value}" duration="{duration}" state_num="{state_num}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	"{state_num}" is based on param_id. Different param_id has its own {state_num}. {state_num} will be updated when {curr_value} is updated.

30.3 Get (Passive)

Request	[&param_id={param_id}]
	{intf_cmd} is CMD_WINDOW_COVERING_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<window_cvr utime="{updt_time}" param_id="{param_id}" curr_value="{curr_value}" target_value="{target_value}" duration="{duration}" state_num="{state_num}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	In Passive Get command, the Request parameter is optional. When the Request parameter is not absent, only 1 cached report with matched param_id will be returned. If the Request parameter is absent, all the cached reports for the different param_ids will be returned.

30.4 Set

Request	&param_id_list={param_id_list}&param_value_list={param_value_list}[&dur={duration}]
	{intf_cmd} is CMD_WINDOW_COVERING_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	The number of components in "{param_id_list}" should match what is in "{param_value_list}". Eg. &param_id_list="1,3,5," &param_value_list="8,50,0," {duration} is optional. In the event that this parameter is absent, factory default duration (0xFF) will be used.

30.5 Start Level Change

Request	&start_stop=1&param_id={param_id}&direction={direction}[&dur={duration}]

	{intf_cmd} is CMD_WINDOW_COVERING_LEVEL_CHANGE_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly {duration} is optional. In the event that this parameter is absent, factory default duration (0xFF) will be used.

30.6 Stop Level Change

Request	&start_stop=0&param_id={param_id}
	{intf_cmd} is CMD_WINDOW_COVERING_LEVEL_CHANGE_SET
Response	Empty, on success
	Failure is the same for Select except {intf_cmd} is replaced accordingly

30.7 Get Capabilities (Active)

Request	{intf_cmd} is CMD_WINDOW_COVERING_SUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <window_cvr_sup utime="{updt_time}" list="{param_id_list}" /> </zwif></zwave>
	Failure is the same as Select except {intf_cmd} is replaced accordingly

30.8 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_WINDOW_COVERING_SUP_REPORT
Response	Same as Get Capabilities (Active)

31 Anti-theft Unlock Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_atu
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Anti-theft Unlock CC commands:

Parameter	Window Covering CC reference	
{intf_cmd}	Commands below	
	CMD_ANTITHEFT_UNLOCK_SETUP	1
	CMD_ANTITHEFT_UNLOCK_GET	2
	CMD_ANTITHEFT_UNLOCK_REPORT	3
	CMD_ANTITHEFT_UNLOCK_SET	4
{state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when 'utime' is not 0.	
{state}	State value in Anti-theft Unlock Report Command 0: Node is unlocked 1: Node is locked	
{restricted}	Restricted value in Anti-theft Unlock Report Command 0: Node is not restricted 1: Node is restricted	
{hint}	Hint value in Anti-theft Unlock Report Command encoded as hexadecimal string, 0-10 bytes. Eg. 01AA02CC	
{manuf_id}	Manufacturer ID value in Anti-theft Unlock Report Command	
{entity_id}	Z-Wave Alliance locking entity ID value in Anti-theft Unlock Report Command	
{magic}	Magic Code value in Anti-theft Unlock Report Command encoded as hexadecimal string, 1-10 bytes. Eg. 0102FFEE	

31.1 Select

Request	{intf_cmd} is CMD_ANTITHEFT_UNLOCK_SETUP
Response	Behaviour is same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServAtu' in the response

31.2 Get (Active)

Request	{intf_cmd} is CMD_ANTITHEFT_UNLOCK_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <atu_sta utime="{updt_time}" state_num="{state_num}" state="{state}" restricted="{restricted}" hint="{hint}" manuf_id="{manuf_id}" entity_id="{entity_id}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

31.3 Get (Passive)

Request	{intf_cmd} is CMD_ANTITHEFT_UNLOCK_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <atu_sta utime="{updt_time}" state_num="{state_num}" state="{state}" restricted="{restricted}" hint="{hint}" manuf_id="{manuf_id}" entity_id="{entity_id}" /> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

31.4 Set

Request	&magic_code={magic}
	{intf_cmd} is CMD_ANTITHEFT_UNLOCK_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

32 Protection Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_prot
Request	Same as for Basic Interface

The following request/response parameters are used in this section with references to fields in the Protection commands:

Parameter	Protection CC Command field	
{intf_cmd}	Commands below	
	CMD_PROTECTION_SETUP	1
	CMD_PROTECTION_GET	2
	CMD_PROTECTION_REPORT	3
	CMD_PROTECTION_SET	4
	CMD_PROTECTION_EC_GET	5
	CMD_PROTECTION_EC_REPORT	6
	CMD_PROTECTION_EC_SET	7
	CMD_PROTECTION_TMOUT_GET	8
	CMD_PROTECTION_TMOUT_REPORT	9
	CMD_PROTECTION_TMOUT_SET	10
	CMD_PROTECTION_SUP_GET	11
	CMD_PROTECTION_SUP_REPORT	12
{local_prot}	Local protection state in Protection Set Command.	
{rf_prot}	RF protection state in Protection Set Command. Applicable for v2 or above only.	
{node_id}	Node ID that has exclusive control can override the RF protection state of the device and can control it regardless of the protection state. Node id of zero is used to reset the protection exclusive control state.	
{remain_tm}	Remaining time. 0x00 : No timer is set. All "normal operation" Commands must be accepted. 0x01 to 0x3C : 1 second (0x01) to 60 seconds (0x3C); 0x41 to 0xFE : 2 minutes (0x41) to 191 minutes (0xFE); 0xFF : No Timeout - The Device will remain in RF Protection mode infinitely.	
{ec_flag}	Flag to indicates whether the device supports Exclusive Control.	
{tmout_flag}	Flag to indicates whether the device supports timeout for RF Protection State.	
{local_list}	Comma separated list of supported local protection states.	
{rf_list}	Comma separated list of supported RF protection states.	
{state_num_local} {state_num_rf} {state_num}	An unsigned 16-bit integer which indicates the change of state. It is inremented by one whenever a state change is detected. It will loop around 0 when it reaches 0xFFFF. Valid only when ’utime’ is not 0.	

32.1 Select

Request	{intf_cmd} is CMD_PROTECTION_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServProt'

32.2 Get (Active)

Request	{intf_cmd} is CMD_PROTECTION_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <prot utime="{updt_time}" local_prot="{local_prot}" rf_prot="{rf_prot}" state_num_local="{state_num_local}" state_num_rf="{state_num_rf}" /> </zwif></zwave>
	Failure same as for Basic Select except - {intf_cmd} is replaced accordingly - 'tServBasic' is replaced with 'tServProt'
Note	{rf_prot} value is applicable for v2 or above only. For v1 device, {rf_prot} will always be 0 (unprotected). "{state_num_local}" will be updated when {local_prot} is updated. "{state_num_rf}" will be updated when {rf_prot} is updated.

32.3 Get (Passive)

Request	{intf_cmd} is CMD_PROTECTION_REPORT
Response	Same as Get (Active) except {intf_cmd} is replaced accordingly
Note	See Get (Active)

32.4 Set

Request	&local_prot={local_prot}[&rf_prot={rf_prot}]
	{intf_cmd} is CMD_PROTECTION_SET
Response	Same as Select except {intf_cmd} is replaced accordingly
Note	{rf_prot} is optional and applicable for v2 or above only.

32.5 Exclusive Control Get (Active)

Request	{intf_cmd} is CMD_PROTECTION_EC_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <prot_ec utime="{updt_time}" node_id="{node_id}" state_num="{state_num}" /> </zwif></zwave>
	Failure same as for Basic Select except - {intf_cmd} is replaced accordingly

	- ‘tServBasic’ is replaced with ‘tServProt’
Note	“{state_num}” will be updated when {node_id} is updated.

32.6 Exclusive Control Get (Passive)

Request	{intf_cmd} is CMD_PROTECTION_EC_REPORT
Response	Same as Exclusive Control Get (Active) except {intf_cmd} is replaced accordingly
Note	See Exclusive Control Get (Active)

32.7 Exclusive Control Set

Request	&node_id={node_id}
	{intf_cmd} is CMD_PROTECTION_EC_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

32.8 Timeout Get (Active)

Request	{intf_cmd} is CMD_PROTECTION_TMOUT_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <prot_tmout utime=“{updt_time}” tmout=“{remain_tm}” /> </zwif></zwave>
	Failure same as for Basic Select except - {intf_cmd} is replaced accordingly - ‘tServBasic’ is replaced with ‘tServProt’

32.9 Timeout Get (Passive)

Request	{intf_cmd} is CMD_PROTECTION_TMOUT_REPORT
Response	Same as Timeout Get (Active) except {intf_cmd} is replaced accordingly
Note	See Timeout Get (Active)

32.10 Timeout Set

Request	&tmout={remain_tm}
	{intf_cmd} is CMD_PROTECTION_TMOUT_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

32.11 Get Capabilities (Active)

Request	{intf_cmd} is CMD_PROTECTION_SUP_GET

Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <prot_sup utime="{updt_time}" ec_flag="{ec_flag}" tmout_flag="{tmout_flag}" local_list="{local_list}" rf_list="{rf_list}" /> </zwif></zwave>
	Failure same as for Get except {intf_cmd} is replaced accordingly

32.12 Get Capabilities (Passive)

Request	Same as Get Capabilities (Active) except {intf_cmd} is CMD_PROTECTION_SUP_REPORT
Response	Same as Get Capabilities (Active)

33 Group Interface API

The following is also applicable for multi instance/channel association. These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_group
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Association CC Command field	
{intf_cmd}	Commands below	
	CMD_ASSOCIATION_GET	1
	CMD_ASSOCIATION_REPORT	2
	CMD_ASSOCIATION_SET	3
	CMD_ASSOCIATION_GROUPINGS_GET	4
	CMD_ASSOCIATION_GROUPINGS_REPORT	5
	CMD_ASSOCIATION_SPECIFIC_GROUP_GET	6
	CMD_ASSOCIATION_SPECIFIC_GROUP_REPORT	7
{assoc_grp_id}	group ID	
{assoc_max_cnt}	Maximum Nodes Supported field of Report Command	
{assoc_ep_cnt}	Number of endpoints in the group	
{assoc_ep_list}	Comma separated list of endpoints, represented as Node_ID|Endpoint_ID.	
{assoc_add_del}	0 for addition; 1 for deletion.	
{assoc_member_cnt}	Number of node/endpoint pairs added/deleted in this operation. Maximum is 5.	
{assoc_node_id} {assoc_ep_id}	The node ID / endpoint ID pair to be added/deleted.	
{assoc_grp_cnt}	Number of groups supported	
{assoc_grp_actv}	Current active group ID	

33.1 Get (Active)

Request	&group_id={assoc_grp_id}
	{intf_cmd} is CMD_ASSOCIATION_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<group utime="{updt_time}" group="{assoc_grp_id}" max_cnt="{assoc_max_cnt}" ep_cnt="{assoc_ep_cnt}" ep_list="{assoc_ep_list}" />] </zwif></zwave>
	Failure is the same for Basic Select except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServAssoc
Note	In the {assoc_ep_list}, the Endpoint_ID could be 255 to denote a Node level association (before the marker) in multichannel association or association. Eg. 1|255 denotes Node 1 (Node level association) 1|0 denotes Node 1 Ep 0 (Endpoint level association)

	UI should memorize the Endpoint ID received in this API, be it 255 or 0. To remove the particular association, the Endpoint ID returned by this API needs to be used.

33.2 Get (Passive)

Request	&group_id={assoc_grp_id}
	{intf_cmd} is CMD_ASSOCIATION_REPORT
Response	Same as for Get (Active) except {intf_cmd} is replaced accordingly

33.3 Set

Request	&add_del={assoc_add_del}&group_id={assoc_grp_id}&member_cnt={assoc_member_cnt} [&node_id={assoc_node_id}&ep_id={assoc_ep_id}]
	{intf_cmd} is CMD_ASSOCIATION_SET Example 1. add_del=0 group_id=1 member_cnt=1 node_id=1 ep_id=0 Example 2. add_del=0 group_id=1 member_cnt=2 node_id=1 ep_id=0 node_id=2 ep_id=1
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServAssoc

33.4 Get Supported Groupings (Active)

Request	{intf_cmd} is CMD_ASSOCIATION_GROUPINGS_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <group_sup utime="{updt_time}" grp_cnt="{assoc_grp_cnt}" /> </zwif></zwave>
	Failure same as for Get except with {intf_cmd} replaced accordingly

33.5 Get Supported Groupings (Passive)

Request	Same as Get Supported Groupings (Active) except {intf_cmd} is CMD_ASSOCIATION_GROUPINGS_REPORT
Response	Same as Get Supported Groupings (Active)

33.6 Get Specific Group – Current Active Group (Active)

Request	{intf_cmd} is CMD_ASSOCIATION_SPECIFIC_GROUP_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <group_actv utime="{updt_time}" grp_actv="{assoc_grp_actv}" /> </zwif></zwave>
	Failure same as for Get except with {intf_cmd} replaced accordingly
Note	Applicable for Association CC version >= 2, otherwise behavior undefined

33.7 Get Specific Group – Current Active Group (Passive)

Request	Same as Get Specific Group (Active) except {intf_cmd} is CMD_ASSOCIATION_SPECIFIC_GROUP_REPORT
Response	Same as Get Specific Group (Active)
Note	Applicable for Association CC version >= 2, otherwise behavior undefined

34 Group Info interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_group_info
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Association Group Information CC fields
{intf_cmd}	Commands below
	CMD_ASSOCIATION_GRP_INFO_GET | 1
{agi_dynamic}	Flag to indicate if the group info is dynamic. ‘Dynamic Info’ field of ‘Association Group Info Report Command’.
{agi_grp_id}	‘Grouping Identifier’ field of ‘Association Group Info Report Command’.
{agi_profile}	The generic and specific device type that indicates the intended device type of target device. ‘Profile’ field of ‘Association Group Info Report Command’.
{agi_evt_code}	The simple AV control code defining the intended mechanism for controlling targets in the association group. ‘Event Code’ field of ‘Association Group Info Report Command’.
{agi_name}	The name of the association group. ‘Name’ field of ‘Association Group Name Report Command’.
{agi_intf_name}	Interface name (Z-Wave CC Name). This is “UNKNOWN” if no match.
{agi_intf_cmd_name}	Command name within the specific interface (Z-Wave Command Name). This is “UNKNOWN” if no match.

34.1 Get

This call returns immediately with values discovered as part of capability discovery during node inclusion phase.

Request	{intf_cmd} is CMD_ASSOCIATION_GRP_INFO_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <group_info_list dynamic=”{agi_dynamic}”> [<group_info group=“{agi_grp_id}” profile=”{agi_profile}” evt_code=”{agi_evt_code}” name=”{agi_name}”> [<group_cmd intf_name=”{agi_intf_name}” intf_cmd_name=”{agi_intf_cmd_name}” /> [<group_cmd…>…]] </group_info> [<group_info…>…]] </group_info_list></zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServAssocGrpInfo

35 Configuration Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_config
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Configuration CC Command field	
{intf_cmd}	Commands below	
	CMD_CONFIGURATION_SETUP	1
	CMD_CONFIGURATION_GET	2
	CMD_CONFIGURATION_REPORT	3
	CMD_CONFIGURATION_SET	4
	CMD_CONFIGURATION_INFO_GET	5
	CMD_CONFIGURATION_INFO_REPORT	6
{config_pnum}	Parameter number (Parameter Number field of Report Command or Parameter Offset field of Bulk Set Command)	
{config_pval}	Parameter value (Configuration Value field of Report Command and Bulk Report Command)	
{config_psize}	Size of parameter value (Size field of Set Command and Bulk Set Command).	
{config_use_def}	Whether to use the default factory setting (Default' field of Set Command and Bulk Set Command)	
{config_pcnt}	Number of parameters (Number of Parameters field of Bulk Set Command)	
{config_hand_shake}	Whether to use the hand shake feature (Handshake field of Bulk Set Command)	
{config_reset_all}	"1" to send out Default Reset Command to reset all configuration parameters to their default values. Applicable only for class version >= 4; Otherwise, undefined.	
{config_pformat}	Format of parameter value (Format field of Properties Report Command)	
{config_pname}	Name of parameter (Name field of Name Report Command)	
{config_pinfo}	Info of parameter (Info field of Info Report Command)	
{config_pmin}	Minimum value of parameter (Minimum Value field of Properties Report Command)	
{config_pmax}	Maximum value of parameter (Maximum Value field of Properties Report Command)	
{config_pdefault}	Default value of parameter (Default Value field of Properties Report Command)	
{config_re_incl_req}	Re-inclusion Required field of Properties Report Command	
{config_read_only}	Read-only field of Properties Report Command	
{config_bulk_support}	If device supports Bulk Commands. 1 if support; 0 if not support.	
{config_adv}	Advanced field of Properties Report Command	

35.1 Select

Request	{intf_cmd} is CMD_CONFIGURATION_SETUP
Response	Same as for Basic Select except - {intf_cmd} is replaced accordingly

	- 'tServBasic' is replaced with 'tServConfig'

35.2 Get (Active)

Request	&param_num={config_pnum}&param_cnt={config_pcnt}
	{intf_cmd} is CMD_CONFIGURATION_GET {config_pcnt} is optional and only applicable for Version 2 or above Bulk Get command.
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<config utime="{updt_time}" num="{config_pnum}" val="{config_pval}" />] </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

35.3 Get (Passive)

Request	Same as Get (Active) except {intf_cmd} is CMD_CONFIGURATION_REPORT
Response	Same as for Get (Active)

35.4 Set

Request	&reset_all_param={config_reset_all} &param_num={config_pnum}&param_cnt={config_pcnt} &hand_shake={config_hand_shake}&use_def={config_use_def} &param_val={config_pval}&param_size={config_psize} &param_format={config_pformat}
	{intf_cmd} is CMD_CONFIGURATION_SET config_reset_all} is optional and only applicable for class version >= 4. If {config_reset_all} is used, it must equal "1" and all other parameters are not required. {config_pcnt} and {config_hand_shake} are optional and only applicable for Version 2 or above Bulk Set command. {config_pformat} is optional and only applicable for class version >= 3. For Version 3 or above Configuration CC, if {config_pformat} is not provided, it's assumed to equal "0". If {config_use_def} equals "1", {config_pval}, {config_psize} and {config_pformat} are not required.
Response	Same as for Select except {intf_cmd} is replaced accordingly

35.5 Info Get (Active)

Request	&param_num={config_pnum}&param_cnt={config_pcnt}
	{intf_cmd} is CMD_CONFIGURATION_INFO_GET

	{config_pcnt} is optional and only applicable for Version 2 or above Bulk Set command. To get a list of all supported params {config_pnum} and {config_pcnt} must be equal 0.
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> [<config utime="{updt_time}" num="{config_pnum}" name="{config_pname}" info="{config_pinfo}" format="{config_pformat}" size="{config_psize}" min_val="{config_pmin}" max_val="{config_pmax}" dflt_val="{config_pdefault}" re_incl_req="{config_re_incl_req}" read_only="{config_read_only}" bulk_support="{config_bulk_support}" adv="{config_adv}" /> </zwif></zwave> {config_re_incl_req}, {config_read_only}, {config_bulk_support} and {config_adv} are optional and will only be available for class version >= 4.
	Failure is the same for Select except {intf_cmd} is replaced accordingly
Note	Applicable for class version >= 3, otherwise behavior undefined.

35.6 Info Get (Passive)

Request	Same as Info Get (Active) except {intf_cmd} is CMD_CONFIGURATION_INFO_REPORT
Response	Same as for Info Get (Active)

36 Firmware Update Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_fw
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Firmware Update Meta Data CC	
{intf_cmd}	Commands below	
	CMD_FIRMWARE_INFO_GET	1
	CMD_FIRMWARE_INFO_REPORT	2
	CMD_FIRMWARE_UPDATE_REQ_GET	3
	CMD_FIRMWARE_UPDATE_REQ_REPORT	4
	CMD_FIRMWARE_BACKUP_REQ_GET	5
	CMD_FIRMWARE_BACKUP_REQ_REPORT	6
	CMD_FIRMWARE_ACTIVATION_REQ_GET	7
	CMD_FIRMWARE_ACTIVATION_REQ_REPORT	8
{fw_vid}	'Manufacturer ID' field of 'Firmware Meta Data Report Command' or 'Firmware Update Meta Data Request Get Command'	
{zw_fw_id}	'Firmware 0 ID' field of 'Firmware Meta Data Report Command'	
{fw_chksum}	'Firmware 0 Checksum' field of 'Firmware Meta Data Report Command' or 'Checksum' field of 'Firmware Update Meta Data Request Get Command'	
{fw_max_frag_sz}	'Max Fragment Size' field of 'Firmware Meta Data Report Command'	
{fw_fixed_frag_sz}	flag to indicate if {fw_max_frag_sz} is fixed. If 1, fixed size; if 0, variable size.	
{fw_upgrade_flag}	'Firmware Upgradeable' field of 'Firmware Meta Data Report Command'	
{app_data_dir}	Location of the application data in the Server. Firmware to be upgraded needs to be placed under the 'data' folder in this location.	
{fw_target}	Target number of the firmware starting with 1.	
{fw_id}	'Firmware N ID' fields of 'Firmware Meta Data Report Command' or 'Firmware ID' field of 'Firmware Update Meta Data Request Get Command'	
{fw_file_name}	Name of the firmware file under Server's <app-data-area>/data/ folder	
{req_time}	Timestamp of the latest firmware update/backup/activation request (formatted as POSIX time)	
{req_status_updt_time}	Receive timestamp (formatted as POSIX time) of the latest 'Firmware Update Meta Data Request Report Command') or 'Firmware Update Meta Data Prepare Report Command' or,	

	'Firmware Update Activation Status Report Command'.
{req_status}	'Status' field of 'Firmware Update Meta Data Request Report Command' or 'Firmware Update Meta Data Prepare Report Command' or 'Firmware Update Activation Status Report Command'.
{req_cmplt_updt_time}	Timestamp (formatted as POSIX time) of the latest 'Firmware Update Meta Data Get/Report Command') or 'Firmware Update Meta Data Prepare Get/Report Command'. This time gets updated only when {req_status} is 0xff (that is, when update/backup request is accepted).
{cmplt_status}	Completion status of the latest firmware update request (depends on the progress of the series of 'Firmware Update Meta Data Get/Report Command'). For Firmware Update request: 0x00: Update Failed. Checksum error in requested firmware; 0x01: Update Failed. Download of the requested firmware failed; 0x02: Update Failed. The transferred image does not match the Manufacturer ID; 0x03: Update Failed. The transferred image does not match the Firmware ID; 0x04: Update Failed. The transferred image does not match the Firmware Target; 0x05: Update Failed. Invalid file header information; 0x06: Update Failed. Invalid file header format; 0x07: Update Failed. Insufficient memory; 0x08: Update Failed. Hardware version mismatched; 0x10: Battery level is low, firmware update was not initiated; 0x11: Battery level is unknown, firmware update was not initiated; 0xFD: Firmware image downloaded successfully, waiting for activation command; 0xFE: Firmware update successfully completed; 0xFF: Firmware update successfully completed pending restart. For Firmware Backup request: 0x00: Backup Failed. Checksum error in downloaded firmware; 0x01: Backup Failed. Download of the requested firmware failed; 0x02: Backup Failed. Insufficient memory; 0x03: Backup Failed. Firmware download request status failed, downloading not started; 0x04: Backup Failed. Firmware download request timeout, downloading not started;, 0x05: Backup Failed. Writing firmware file failed; 0x06: Backup Failed. The firmware size is too large;, 0x07: Backup Failed. Other error; 0xFF: Firmware successfully downloaded and saved to file. This status is updated only when {req_status} is 0xff (that is, when update request is accepted).
{hw_ver_valid}	Flag to indicate whether {hw_ver} is valid. If 1, valid; if 0, invalid.

{hw_ver}	'Hardware Version' fields of 'Firmware Meta Data Report Command' or 'Firmware Update Meta Data Request Get Command'
{fw_backup_file_full_path}	Only valid in Local mode. Full path of the generated firmware file in Server directory.
{func_normally}	Flag to indicate whether other command classes function normally during firmware update. 2: Function normally; 1: Certain command classes will not function; 0: Unknown (For Firmware CC v5 and below that doesn't support this flag).
{fw_activation_flag}	Flag to indicate whether node supports the subsequent activation of firmware after firmware update transfer. 2: Support; 1: Not support; 0: Information not available (For Firmware CC v6 and below that doesn't support this flag).
{delay_activation}	Flag to indicate whether the target device should delay firmware update until "activation" command is received. 1: Delay; (User would need to activate the firmware separately after update transfer) 0: No delay, i.e. activate the firmware immediately after update transfer.
{checksum}	'Checksum' field of 'Firmware Update Meta Data Request Get Command'.

36.1 Get Info (Active)

Request	{intf_cmd} is CMD_FIRMWARE_INFO_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <fw_info utime="{updt_time}" vid="{fw_vid}" zw_fw_id="{zw_fw_id}" chksum="{fw_chksum}" max_frag_sz="{fw_max_frag_sz}" fixed_frag_sz="{fw_fixed_frag_sz}" upgrade_flag="{fw_upgrade_flag}" hw_ver_valid="{hw_ver_valid}" hw_ver="{hw_ver}" func_normally="{func_normally}" activation_flg="{fw_activation_flag}" app_data_dir="{app_data_dir}"> [<fw_id target="{fw_target}" id="{fw_id}" /> [<fw_id…>…]] </fw_info></zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServFirmware
Note	If client needs to show 'Delay Activation' option, and the 'Activation' button along with the 'Firmware Update' button, the recommended behavior would be: - Not to show the 'Delay Activation' option or 'Activation' button for Firmware Update CC version 1-3. Reason being verson 1-3 doesn't support Activation command. - Show 'Delay Activation' option and 'Activation' button for Firmware Update CC version 4-6. Reason being verson 4 supports Activation command, however the Firmware Info report has no way

	to indicate whether the device supports ‘delay activation’. Client has to always show the ‘Delay activation’ option and button in this case. - For Firmware Update CC version 7 or above, UI will show the ‘Delay activation’ option and button, depending on the ‘activation_flag’ value in Firmware Info report.

36.2 Get Info (Passive)

Request	{intf_cmd} is CMD_FIRMWARE_INFO_REPORT
Response	Same as Get Info (Active)

36.3 Get Update Request (Active)

This call is responsible for triggering a new firmware update request.

Request	&vid={fw_vid}&fw_target={fw_target}&fw_id={fw_id} &file_name={fw_file_name}&hw_ver={hw_ver}[&delay_activation={delay_activation}]
	{intf_cmd} is CMD_FIRMWARE_UPDATE_REQ_GET
Response	On success: <?xml version=“1.0”?><zwave><zwif desc=“{intf_desc}”> <fw_update req_time=“{req_time}” req_status_utime=”{req_status_updt_time}” req_status=”{req_status}” cmplt_status_utime=”{cmplt_status_updt_time}” cmplt_status=”{cmplt_status}” /> </zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServFirmware
Note	{delay_activation} is optional and supported from Firmware Update CC version 7 onwards. If this parameter is not present for Firmware Update CC version 7 or above, it is treated as 0 (no delay).

36.4 Get Update Request (Passive)

This call is responsible for querying the status of ongoing firmware update request.

Request	{intf_cmd} is CMD_FIRMWARE_UPDATE_REQ_REPORT
Response	Same as Get Update Request (Active)

The following is the pseudo code for interpreting the different stages of firmware update.

```
Status = “”
If {req_time} != 0
{
        Status = “Sending firmware update request”
        If {req_status_updt_time} >= {req_time}
        {
```

```
                Status = Message for {req_status}
                If {cmplt_status_updt_time} >= {req_time}
                {
                        Status = Message for {cmplt_status}
                }
        }
}
```

36.5 Get Backup Request (Active)

This call is responsible for triggering a new firmware backup request.

Request	&vid={fw_vid}&fw_target={fw_target}&fw_id={fw_id}&hw_ver={hw_ver}
	{intf_cmd} is CMD_FIRMWARE_BACKUP_REQ_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <fw_backup vid={fw_vid} fw_tgt={fw_target} fw_id={fw_id} hw_ver={hw_ver} req_time="{req_time}" req_status_utime="{req_status_updt_time}" req_status="{req_status}" cmplt_status_utime="{cmplt_status_updt_time}" cmplt_status="{cmplt_status}" fw_file_fullpath="{fw_backup_file_full_path}" /> </zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServFirmware

36.6 Get Backup Request (Passive)

This call is responsible for querying the status of ongoing/previous firmware backup request.

Request	[&vid={fw_vid}&fw_target={fw_target}&fw_id={fw_id}&hw_ver={hw_ver}]
	{intf_cmd} is CMD_FIRMWARE_BACKUP_REQ_REPORT
Response	Same as Get Backup Request (Active)
Note	The request string is optional. If it is an empty string, it will query the status of ongoing/previous firmware backup request, regardless vid, fw_tgt, fw_id etc. For passive Get, if the request contains the 4 IDs (vid, fw_tgt, fw_id & hw_ver) but if any of the 4 IDs does not match the ID that is currently backup, or what was previously back up, the response will contain the IDs with all timestamp and status 0 to indicate no such record. For passive Get, if the request is empty but there is no record in the Server for any Backup process, the returned vid, fw_tgt and fw_id will be value 65535 to indicate it is a invalid entry. Client should ignore the response in this case.

36.7 Get Activation Request (Active)

This call is responsible for triggering the subsequent activation of firmware after firmware update transfer.

Request	{intf_cmd} is CMD_FIRMWARE_ACTIVATION_REQ_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <fw_actv vid="{fw_vid}" fw_tgt="{fw_target}" fw_id="{fw_id}" hw_ver="{hw_ver}" checksum="{checksum}" req_time="{req_time}" req_status_utime="{req_status_updt_time}" req_status="{req_status}" /> </zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServFirmware

36.8 Get Activation Request (Passive)

This call is responsible for querying the status of ongoing firmware activation request.

Request	{intf_cmd} is CMD_FIRMWARE_ACTIVATION_REQ_REPORT
Response	Same as Get Activation Request (Active)
Note	Similar to Firmware Update request or Firmware backup request, client should check the other attribute values in the response only if req_status_utime >= req_time.

37 Z/IP Gateway Interface API

These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_gw
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Z/IP Gateway CC	
{intf_cmd}	Commands below	
	CMD_GATEWAY_MODE_GET	1
	CMD_GATEWAY_MODE_REPORT	2
	CMD_GATEWAY_MODE_SET	3
	CMD_GATEWAY_LOCK_SET	4
	CMD_GATEWAY_UNSOL_DEST_GET	5
	CMD_GATEWAY_UNSOL_DEST_REPORT	6
	CMD_GATEWAY_UNSOL_DEST_SET	7
{gw_status}	Return status of the call 0: success; 1: timeout or the gateway is locked with its parameter hidden 2: transmit error.	
{gw_mode}	'Mode' field of 'Gateway Mode Set/Report Command'	
{gw_profile_present}	Portal profile presence – 1: present; 0: missing	
{gw_peer_addr} {gw_peer_name}	'Ipv6 Address'/'Peer Name' field of 'Gateway Mode Set/Report Command', UTF-8 encoded and then URL encoded, applicable only when {gw_mode} is 2	
{gw_peer_port}	'Port' field of 'Gateway Set/Mode Report Command', applicable only when {gw_mode} is 2.	
{gw_lock}, {gw_show}	'Lock'/'Show' field of 'Gateway Lock Set Command'	
{gw_unsol_dest_addr}	'Unsolicited Ipv6 Destination' field of 'Unsolicited Destination Set/Report Command' under Z/IP Gateway CC, UTF-8 encoded and then URL encoded	
{gw_unsol_dest_port}	'Unsolicited Destination Port' field of 'Unsolicited Destination Set/Report Command' under Z/IP Gateway CC	
{gw_local_addr}	The discovered local address (at Server) that is reachable by the gateway.	
{gw_local_port}	The local port (at Server) that is capable of handling unsolicited reports.	

37.1 Get Mode (Active)

Request	{intf_cmd} is CMD_GATEWAY_MODE_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <gw_mode utime="{updt_time}" status="{gw_status}" mode="{gw_mode}" profile_present="{gw_profile_present}" peer_addr="{gw_peer_addr}" peer_name="{gw_peer_name}" peer_port="{gw_peer_port}" /> </zwif></zwave>

	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServGateway

37.2 Get Mode (Passive)

Request	{intf_cmd} is CMD_GATEWAY_MODE_REPORT
Response	Same as Get Mode (Active)

37.3 Set Mode

Request	&mode={gw_mode}&peer_addr={gw_peer_addr} &peer_name={gw_peer_name}&peer_port={gw_peer_port}
	{intf_cmd} is CMD_GATEWAY_MODE_SET
Response	Same as for Basic Set except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServGateway

37.4 Set Lock

Request	&lock={gw_lock}&show={gw_show}
	{intf_cmd} is CMD_GATEWAY_LOCK_SET
Response	Same as for Basic Set except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServGateway

37.5 Get Unsolicited Destination (Active/Passive)

{gw_local_addr} and {gw_local_port} can be used to prompt the user to set the unsolicited destination to self (the Server).

Request	{intf_cmd} is CMD_GATEWAY_UNSOL_DEST_GET/REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <gw_unsol_dest utime="{updt_time}" unsol_dest_addr="{gw_unsol_dest_addr}" unsol_dest_port="{gw_unsol_dest_port}" local_addr="{gw_local_addr}" local_port="{gw_local_port}" /> </zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServGateway

37.6 Set Unsolicited Destination

Request	&unsol_dest_addr={gw_unsol_dest_addr}&unsol_dest_port={gw_unsol_dest_port}
	{intf_cmd} is CMD_GATEWAY_UNSOL_DEST_SET
Response	Same as for Basic Set except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServGateway

38 Z/IP Portal Interface API

Despite the fact, the Portal mode has been removed from Z-Ware, this Web API function still allows to configure gateway for the Portal mode.
These HTTP request parameters are used in this section with extensions specified separately:

URI	/zwif_gw_cfg
Request	Same as for Basic Interface

The following request/response parameters are used in this section:

Parameter	Z/IP Gateway CC	
{intf_cmd}	Commands below	
	CMD_PORTAL_GATEWAY_CONFIG_GET	1
	CMD_PORTAL_GATEWAY_CONFIG_REPORT	2
	CMD_PORTAL_GATEWAY_CONFIG_SET	3
	CMD_PORTAL_GATEWAY_CONFIG_STATUS_REPORT	4
{gw_status}	Return status of the call 0: success; 1: timeout or the gateway is locked with its parameter hidden 2: transmit error.	
{lan_ipv6_addr}	LAN IPv6 address in Gateway Configuration Report Command, in string format	
{lan_prefix_len}	LAN IPv6 Prefix Length in Gateway Configuration Report Command	
{portal_ipv6_prefix}	Portal IPv6 Prefix in Gateway Configuration Report Command, in string format	
{portal_prefix_len}	Portal IPv6 Prefix Length in Gateway Configuration Report Command	
{pan_ipv6_prefix}	PAN IPv6 Prefix in Gateway Configuration Report Command, in string format	
{default_gw_ipv6_addr}	Default Gateway IPv6 Address in Gateway Configuration Report Command, in string format	
{set_time}	Time when Config Set command is sent	
{status_updt_time}	Time when Config Status is received	
{status}	Status in Gateway Configuration Status Command	

38.1 Gateway Config Get (Active)

Request	{intf_cmd} is CMD_PORTAL_GATEWAY_CONFIG_GET
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <gw_cfg utime="{updt_time}" lan_addr="{lan_ipv6_addr}" lan_prefix_len="{lan_prefix_len}" portal_prefix="{portal_ipv6_prefix}" portal_prefix_len="{portal_prefix_len}" pan_prefix="{pan_ipv6_prefix}" default_gw_addr="{default_gw_ipv6_addr}" /> </zwif></zwave>
	Failure is the same for Basic Get except - {intf_cmd} is replaced accordingly - tServBasic is replaced with tServPortal

38.2 Gateway Config Get (Passive)

Request	{intf_cmd} is CMD_PORTAL_GATEWAY_CONFIG_REPORT
Request	Same as Gateway Config Get (Active)

38.3 Gateway Config Set

Request	&lan_addr={lan_ipv6_addr}&lan_prefix_len={lan_prefix_len}& portal_prefix={portal_ipv6_prefix}&portal_prefix_len={portal_prefix_len}& pan_prefix={pan_ipv6_prefix}&default_gw_addr={default_gw_ipv6_addr}
	{intf_cmd} is CMD_PORTAL_GATEWAY_CONFIG_SET
Response	Same as Select except {intf_cmd} is replaced accordingly

38.4 Gateway Config Status (Passive)

Request	{intf_cmd} is CMD_PORTAL_GATEWAY_CONFIG_STATUS_REPORT
Response	On success: <?xml version="1.0"?><zwave><zwif desc="{intf_desc}"> <gw_cfg_sts set_time="{set_time}" status_utime="{status_updt_time}" status="{status}"/> </zwif></zwave>
	Failure is the same for Select except {intf_cmd} is replaced accordingly

39 Scenes API

The following HTTP parameters are used always:

URI	/zw_scene
Request	cmd={scene_cmd}\r\n

The following parameters are used in selected HTTP Requests/Responses.

Parameter	Description		
{scene_cmd}	CMD_SCENE_XXX	#	Description
	LIST	1	Get the list of Scenes
	GET	2	Get the detailed information of a Scene
	SAVE	3	Update an existing Scene or create a new Scene if the Scene id is zero
	DELETE	4	Delete a Scene
	EXEC	6	Execute/Run a Scene
	STATUS_UPDATE	7	Request update for status a Scene or all Scenes
	GET_STATELOG	8	Get the statelog. This API could be used to poll for the current and the last state of the Scenes and Security Scenes.
	GET_SUP	9	Get capabilities of scene module
{scene_id}	Unique ID for a Scene generated at the Scene creation time		
{scene_name}	Scene name		
Error X	Error	Value	
	1	<?xml version=\"1.0\"?><zwave><error> tServScene[{scene_cmd}]<!--Parameter missing/invalid--> </error></zwave>	
	2	<?xml version=\"1.0\"?><zwave><error> tServScene[{scene_cmd}]<!--Internal error--> </error></zwave>	
{scene_active_triggers}	Scene level enable/disable switch and overrides *This is not used in current implementation.*		
	SCENE_ACTIVE_XXX	#	Descripton
	NONE	0	Scene is disabled
	MANUAL	1	Scene may be executed manually
	SCHEDULE	2	Scene may be executed by “Schedule”
	EVENT	4	Scene may be executed by “Event
{scene_action_type}	SCENE_ACTION_XXX	#	Descripton
	USER	1	Reserved for future use
	ZWAVE	2	Z-Wave device action
	EMAIL	3	Reserved for future use
	SMS	4	Reserved for future use
{scene_schedule_type}	SCENE_SCHEDULE_XXX	#	Description
	NORMAL	1	Schedule enabled
	DISABLED	2	Schedule disabled
{scene_event_type}	SCENE_EVENT_XXX	#	Description
	ZWAVE	1	Scene Event enabled
	DISABLED	2	Scene Event disabled
{scene_action_uri}	Scene Action URI		
{scene_action_cmd}	Appendix B.4		
{scene_action_ifd}	Scene Action device interface descriptor		
{scene_action_if_parm}	Parameters specific to the interface. See the specific interface section.		
{scene_status}	Scene status. This is derived from the status of all the actions in a scene. The		

<table>
<tr><td></td><td>Scene status can be one of the following.
“Inactive” (FALSE) = 0
“Active” (TRUE) = 1
“Unknown” = 2
“Stateless” = 3
"Active + Stateless" = 4

A ‘Stateless’ scene is neither ‘Active’ nor ‘Inactive’. It is ‘Stateless’ because it does not have a definite target to achieve.

If all the actions for a scene {scene_action_status} are equal ‘Stateless’, then that scene {scene_status} will be ‘Stateless’.

If one or more actions for a scene {scene_action_status} are equal ‘Stateless’, and all the other actions for the same scene {scene_action_status} are equal 'Active', then that scene {scene_status} will be ‘Active + Stateless’.

If all the actions for a scene {scene_action_status} are equal ‘Active’, then that scene {scene_status} will be ‘Active’.

If one of the actions for a scene {scene_action_status} is equal ‘Inactive’, then that scene {scene_status} will be ‘Inactive’.

If one of the actions for a scene {scene_action_status} is equal ‘Unknown’ but none of the actions for the same scene {scene_action_status} is equal ‘Inactive’, then that scene {scene_status} will be ‘Unknown’.

For clarity, please refer to the table ”Scene Action Status and Scene Status”.

It is recommended for the UI to use a different color to represent ‘Stateless’ scene compared to normal ‘Active/Inactive’ scene state. When the ‘Stateless’ scene is triggered, the {scene_status} will still be ‘Stateless’. The only difference is in Scene details will be {scene_last_execution_time}. UI may consider showing the last execution time stamp for the ‘Stateless’ scene and/or a small notification icon for a few seconds when the ‘Stateless’ scene is executed.</td></tr>
<tr><td>{scene_action_status}</td><td>Scene Action status. It is calculated based on the current state of a Z-Wave device and comparing it with the desired state set by the Scene Action. The Scene action status can be one of the following.
“Inactive” (FALSE) = 0
“Active” (TRUE) = 1
“Unknown” = 2
“Stateless” = 3

An action can be ‘stateless’ if it doesn’t have a definite target to achieve.
Eg.
Action for Binary Switch ON ➔ it is either “active” or “inactive”
Action for Multilelvel Switch Set level ➔ it is either “active” or “inactive”
Action for Multilevel Switch Start/Stop level change ➔ it is “stateless”</td></tr>
<tr><td>{sutime}</td><td>Last updated POSIX format time</td></tr>
<tr><td>{scene_last_known_status}</td><td>Last known status of a scene.</td></tr>
<tr><td>{scene_last_known_sutime}</td><td>Status update time for the last known status</td></tr>
<tr><td rowspan="3">{scene_has_triggers}
bitmask</td><td>SCENE_TRIGGER_XXX</td><td>#</td><td>Description</td></tr>
<tr><td>MANUAL</td><td>0</td><td>If there is no Schedule or Device event trigger included in this scene, the scene can only be triggered manually by the user. This is the default value.</td></tr>
<tr><td>SCHEDULE</td><td>1</td><td>The Scene has Schedule trigger set.</td></tr>
</table>

	EVENT	2	The Scene has Event trigger set.
	BOTH	3	The Scene has Schedule & Event triggers set
{scene_last_execution_trigger}	Trigger of the last execution of the scene		
{scene_last_execution_time}	Last execution time for a scene		
{scene_schedule_day}	Day of the week bitmask, set to enable, bits 0..6: Sunday..Saturday		
{scene_schedule_hour}	Hour of the day, 0 to 23		
{scene_schedule_minute}	Minute of the hour, 0 to 59		

39.1 Scene Action Status and Scene Status

Scene status is decided via the priority decision as descripted in the below table.

Priority (highest P1 to lowest P5)	Scene Action Status	Scene Status
P1	Any scene action status is Inactive	Inactive
P2	Any scene action status is Unknown	Unknown
P3	All scene actions status are Active	Active
P4	All scene actions status are Stateless	Stateless
P5	Some scene actions status are Stateless with others been Active	Active + Stateless

39.2 Get Capabilities

Request	{scene_cmd} is CMD_SCENE_GET_SUP
Response	<?xml version=”1.0”?><zwave> <scene max_scenes=”MAX_NUM_SCENES” max_actions=”MAX_NUM_ACTIONS” max_schedules=”MAX_NUM_SCHEDULES” max_events=”MAX_NUM_EVENTS” timezone=”TIMEZONE” /> </zwave> where MAX_NUM_SCENES = maximum number of scenes supported for this network (RAC) MAX_NUM_ACTIONS = maximum number of actions per scene for this network (RAC) MAX_NUM_SCHEDULES = maximum number of schedules per scene for this network (RAC) MAX_NUM_EVENTS = maximum number of events per scene for this network (RAC) TIMEZONE = configured timezone for this network (RAC) Example Response: <?xml version=”1.0”?><zwave> <scene max_scenes=”20” max_actions=”10” max_schedules=”1” max_events=”2” timezone=”Asia/Singapore” /> </zwave>
	On error returns Error 1

39.3 List Scenes

Request	{scene_cmd} is CMD_SCENE_LIST
Response	<?xml version=”1.0”?><zwave> <scene desc=”{scene_id}” name=”{scene_name}” active=”{scene_active_triggers}” status={scene_status} sutime={scene_status_update_time}

	lkstatus="{scene_last_known_status}" lksutime="{scene_last_known_sutime}" triggers="{scene_has_triggers}" /> [<scene …> …] </zwave> if {sutime} is 0, no Z-Wave reports were received since the server started up. So, the Scene and its action's status values in the response are undefined. The Scene status is defined as follows a) "Active" (TRUE) only if we can confirm that all the devices are in the desired state. b) "Inactive" (FALSE) if any of the device is in different state than the desired state. c) "Unknown" if none of the device is in a different state than the desired state but some devices are in unknown state (i.e., unable to get the report). d) "Stateless" if device does not have a definite target state to desire. (i.e., start/stop level change) e) "Active + Stateless" if there is a combination of "Active" and "Stateless" scene status for different devices for a specific scene. Example: <?xml version="1.0"?><zwave> <scene desc="1" name="Coming Home" active="1" status="1" sutime="1334894102" lkstatus="1" lksutime="1334893000" triggers="1" /> <scene desc="2" name="Night Mode" active="1" status="1" sutime="1334894102" lkstatus="1" lksutime="1334893000" triggers="3" /> <scene desc="16" name="NewScene" active="1" status="1" sutime="1334894102" lkstatus="1" lksutime="1334893000" triggers="3" /> </zwave>
	On error returns Error 1

39.4 Get Scene Details

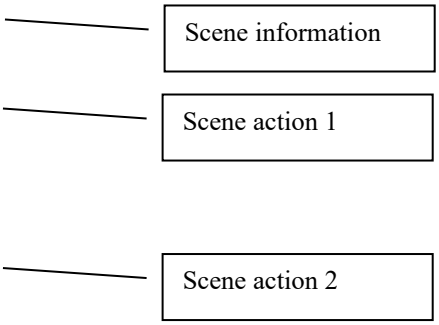
Request	scened={scene_id}\r\n
	{scene_cmd} is CMD_SCENE_GET
Response	<?xml version="1.0"?><zwave> <scene desc="{scene_id}" name="{scene_name}" active="{scene_active_triggers}" status={scene_status} sutime={scene_status_update_time} lkstatus="{scene_last_known_status}" lksutime="{scene_last_known_sutime}" triggers="{scene_has_triggers}" letrigger="{scene_last_execution_trigger}" letime="{scene_last_execution_time}" > <action actype="{scene_action_type}" status={scene_action_status} sutime="{scene_action_status_update_time}" lkstatus="{scene_action_last_known_status}" lksutime="{scene_action_last_known_sutime}" uri="{scene_action_uri}" cmd="{scene_action_cmd}" ifd="{scene_action_ifd}" {scene_action_if_parm} /> [<action …> …] <schedule sctype="{scene_schedule_type}" day="{scene_schedule_day}" hour="{scene_schedule_hour}" minute="{scene_schedule_minute}" /> [<schedule…>…]

	<event evtype="{scene_event_type}" status={scene_event_status} sutime={scene_event_status_update_time} uri="{scene_event_uri}" cmd="{scene_event_cmd}" ifd="{scene_event_desc_id}" {scene_event_device_parameters} /> [<event…>…] </scene></zwave> {scene_action_status} Scene Action status. It is calculated based on the current state of a Z-Wave device and comparing it with the desired state set by the Scene Action. The Scene action status can be one of the following. "Inactive" (FALSE) = 0 "Active" (TRUE) = 1 "Unknown" = 2 "Stateless" = 3 For example, <?xml version="1.0"?><zwave> <scene desc="16" name="NewScene2" active="1" status="0" sutime=" 1334894102" lkstatus="0" lksutime="0" triggers="0" letrigger="1" letime="1334892000"> <action actype="2" status="0" sutime="1334894102" lkstatus="0" lksutime="0" uri="zwif_bin_sw" cmd="4" ifd="8" value="255" /> <action actype="2" status="1" sutime= "1334894102" lkstatus="0" lksutime="0" uri="zwif_bin_sw" cmd="4" ifd="9" value="0" /> <action actype="2" status="0" sutime= "1334894102" lkstatus="0" lksutime="0" uri="zwif_bin_sw" cmd="4" ifd="10" value="255" /> <action actype="2" status="1" sutime= "1334894102" lkstatus="0" lksutime="0" uri="zwif_bin_sw" cmd="4" ifd="11" value="0" /> </scene></zwave>
	On error return Error 1

39.5 Save Scene

Create a new scene or modify an existing scene.

Request	scened={scene_id}\r\n name={scene_name}\r\n active={scene_active_triggers}\r\n action={scene_action_type}\r\n uri={scene_action_uri}\r\n cmd={scene_action_cmd}\r\n ifd={scene_action_ifd}\r\n {scene_action_if_parm}\r\n [action …] schedule={scene_schedule_type}\r\n day={scene_schedule_day}\r\n hour={scene_schedule_hour}\r\n minute={scene_schedule_minute}\r\n [schedule …] event={scene_event_type}\r\n uri={scene_event_uri}\r\n cmd={scene_event_cmd}\r\n ifd={scene_event_desc_id}\r\n {scene_event_device_parameters}\r\n [event …]

	{scene_cmd} is CMD_SCENE_SAVE If {scene_id} is 0, new scene is created; else existing scene is modified Valid actions can be Set Commands for Binary/Multilevel Switch, Door Lock and Thermostat Setpoint Interfaces. Valid events can be Report Commands for Binary/Multilevel Sensor, Door Lock and Alarm Interfaces. Example: cmd=3 scened=16 name=NewScene2 active=1 action=2 uri=zwif_bin_sw cmd=4 ifd=8 value=255 action=2 uri=zwif_bin_sw cmd=4 ifd=9 value=0 
Response	<?xml version="1.0"?><zwave><scene desc="{scene_id}" /></zwave> {scene_id} is the id for the new scene created, or the existing scene modified
	On error returns Error 1 or Error 2
Note	Scene status becomes Unknown after an Edit operation as Actions might have been changed. The client should request a Status Update after a successful Edit.

39.6 Delete Scene

Request	scened={scene_id}\r\n
	{scene_cmd} is CMD_SCENE_DELETE
Response	<?xml version="1.0"?><zwave><scene desc="{scene_id}" /></zwave>
	On error returns Error 1 or Error 2

39.7 Execute Scene

Request	scened={scene_id}\r\n
	{scene_cmd} is CMD_SCENE_EXEC
Response	<?xml version="1.0"?><zwave><scene desc="{scene_id}" /></zwave>
	On error returns Error 1 or Error 2

39.8 Update Scene Status

Request	scened={scene_id}\r\n all={scene_flag_update_all}\r\n

	{scene_cmd} is CMD_SCENE_STATUS_UPDATE When “scene_flag_update_all” is set to 0 the client side must provide a valid scene id but if this flag is set to 1 then field “scened” could be 0.
Response	<?xml version=”1.0”?><zwave><scene desc=”{scene_id}” /></zwave>
	On error returns Error 1 or Error 2 Error 2 if a “Scene status update” operation is already in progress. The client is responsible for requesting status update again. The client should use CMD_SCENE_GET_STATELOG command to check if any status update operation is in progress before sending the request to the server to minimize chances of this error.
Note	This command returns without waiting for the status update to complete. CMD_SCENE_GET_STATELOG can be used to check completion. On completion, CMD_SCENE_LIST/GET can be used to get the latest information. This command is automatically called internally when the local controller is aware of any action that might alter the state of the scenes module e.g., scene activation, addition, deletion or a Z-Wave report indicating an action setting has been changed. As this API results in Z-Wave polling for action states in the scene(s) indicated, it should be used sparingly, especially since some slow mechanical devices like shades and door locks may require several polls before they reach the desired action state.

39.9 Scene Get Statelog

Get current and last state information of the Scenes module. The following parameters are used:

Parameter	Description
{scene_statelog_type}	Bitmap of scene mode status 1 – status update 2 – execution 4 – add 8 – delete
{scene_statelog_xxx_op }	Operation 0 – None 1 – Status update for scene module 2 – Status update for 1 scene 3 – Add 4 – Delete 5 – Execute
{scene_statelog_xxx_op_trigger}	Trigger 0 – none 1 – manual 2 – schedule 3 – Z-Wave event 4 – System event e.g., a Scene deleted because all the Z-Wave nodes in the scene action(s) are deleted from the network

Request	type={scene_statelog_type}\r\n
	{scene_cmd} is CMD_SCENE_GET_STATELOG
Response	<?xml version=”1.0”?><zwave> <statelog type=”{scene_statelog_type}” curr_op=”{scene_statelog_curr_op}” cur_op_scened=”{scene_id}”

	cur_op_trigger="{scene_statelog_cur_op_trigger}" last_op_scened="{scene_id}" last_op_time="{scene_statelog_last_op_time}" last_op_trigger="{scene_statelog_last_op_trigger}" /> … </zwave> {scene_id} is 0 if not required by the related operation. For example, <?xml version="1.0"?><zwave> <statelog type="1" curr_op="0" cur_op_scened="0" cur_op_trigger="0" last_op_scened="0" last_op_time="0" last_op_trigger="0" /> <statelog type="2" curr_op="0" cur_op_scened="0" cur_op_trigger="0" last_op_scened="0" last_op_time="0" last_op_trigger="0" /> <statelog type="4" curr_op="0" cur_op_scened="0" cur_op_trigger="0" last_op_scened="5" last_op_time="1354774678" last_op_trigger="4" /> <statelog type="8" curr_op="0" cur_op_scened="0" cur_op_trigger="0" last_op_scened="0" last_op_time="0" last_op_trigger="0" /> </zwave>
	On error return Error 1
Note	This is a light weight command that can be used to poll the server so that, if there was any operation on the server side (e.g., Scene added by another client), the client could use CMD_SCENE_LIST or CMD_SCENE_GET to fetch the latest information from the server. The client should store the xxx_last_op_time fields in this call for future comparison to discover newly completed operations. The client should also collate multiple operations by checking xxx_curr_op before responding with an appropriate action for system efficiency.

40 Security Scenes API

The following HTTP parameters are always used:

URI	/zw_scene		
Request	cmd={scene_cmd}\r\n		
	CMD_SECURITY_SCENE_XXX	#	Description
	LIST	101	Get the list of Security Scenes
	GET	102	Get the detailed information of a Security Scene
	SAVE	103	Update an existing Security Scene or create a new Security Scene if the Scene id is zero
	DELETE	104	Delete a Security Scene
	ACTIVE_SET	105	
	SET_STATE	106	Set a Security Scene to arm or disarm state
	GET_STATELOG	8	Get the statelog. This API could be used to poll for the current and the last state of the Scenes and Security Scenes.
	GET_SUP	109	Get range capabilities of security scene module

The following commands are used.

40.1 Get Capabilities

Request	{scene_cmd} is CMD_SECURITY_SCENE_GET_SUP
Response	<?xml version="1.0"?> <zwave> <security_scene max_security_scenes="MAX_NUM_SECURITY_SCENES" max_arm_events="MAX_NUM_ARM_EVENTS" max_disarm_events="MAX_NUM_DISARM_EVENTS" max_alarm_events="MAX_NUM_ALARM_EVENTS" timezone="TIMEZONE" /> </zwave> where MAX_NUM_SECURITY_SCENES = maximum number of Security Scenes supported for this network (RAC) MAX_NUM_ARM_EVENTS = maximum number of Arm events per Security Scene for this network (RAC) MAX_NUM_DISARM_EVENTS = maximum number of disarm events per Security Scene for this network (RAC) MAX_NUM_ALARM_EVENTS = maximum number of Alarm events per Security Scene for this network (RAC) TIMEZONE = configured timezone for this network (RAC) Example Response: <?xml version="1.0"?> <zwave> <security_scene max_security_scenes="2" max_arm_events="10" max_disarm_events="10"

	max_alarm_events="16" timezone="Asia/Singapore" /> </zwave>
	On error returns Error 1

40.2 List Security Scenes

Request	{scene_cmd} is CMD_SECURITY_SCENE_LIST
Response	<?xml version="1.0"?> <zwave> <security_scene desc=”{scene_id}” name=”{scene_name}” active=”{scene_active_flag}” armed=”{scene_arm_status}” alarmed=”{scene_alarm_status}” utime={scene_last_update_time} > <lt_alarm_on type="0" time="0" > {scene_last_alarm_on_event} </<lt_alarm_on> </security_scene> [<security_scene…>…] </zwave> {scene_active_flag} is Security Scene level enable/disable but this functionality is not part of the specification at present. {scene_last_alarm_on_event} is an event that caused the Security Scene to go into Alarm state last time. Example Response: <?xml version="1.0"?> <zwave> <security_scene desc="10003" name="SecurityScene_1" active="1" armed="0" alarmed="0" utime="1435734964" > <lt_alarm_on type="0" time="0" /> </security_scene> <security_scene desc="10005" name="SecurityScene_2" active="1" armed="0" alarmed="0" utime="1435735293" > <lt_alarm_on type="0" time="0" /> </security_scene> </zwave>
	On error returns Error 1

40.3 Get Security Scene Details

Request	scened={scene_id}\r\n
	{scene_cmd} is CMD_SECURITY_SCENE_GET
Response	<?xml version="1.0"?> <zwave> <security_scene desc=”{scene_id}” name=”{scene_name}” active=”{scene_active_flag}” armed=”{scene_arm_status}” alarmed=”{scene_alarm_status}”

```
                                        utime={scene_last_update_time} >
                <arm exec_scened={scene_arm_scened} >
                        <event evtype=="{scene_event_type}"
                                        status=={scene_event_status}
                                        sutime=={scene_event_status_update_time}
                                        uri="{scene_event_uri}" cmd="{scene_event_cmd}"
                                        ifd="{scene_event_desc_id}"
                                        {scene_event_device_parameters} />
                        [<event...>...]
                        <lt time={scene_last_event_time} >
                                {scene_last_arm_event}
                        </lt>
                </arm>
                <disarm exec_scened={scene_disarm_scened} >
                        <event evtype=="{scene_event_type}"
                                        status=={scene_event_status}
                                        sutime=={scene_event_status_update_time}
                                        uri="{scene_event_uri}" cmd="{scene_event_cmd}"
                                        ifd="{scene_event_desc_id}"
                                        {scene_event_device_parameters} />
                        [<event...>...]
                        <lt time={scene_last_event_time} >
                                {scene_last_disarm_event}
                        </lt>
                </disarm>
                <alarm exec_scened={scene_alarm_scened} >
                        <event evtype=="{scene_event_type}"
                                        status=={scene_event_status}
                                        sutime=={scene_event_status_update_time}
                                        uri="{scene_event_uri}" cmd="{scene_event_cmd}"
                                        ifd="{scene_event_desc_id}"
                                        {scene_event_device_parameters} />
                        [<event...>...]
                        <lt_on time={scene_last_event_time} >
                                {scene_last_arm_on_event}
                        </lt_on>
                        <lt_off time={scene_last_event_time} >
                                {scene_last_arm_off_event}
                        </lt_off>
                </alarm>
                <notification on={scene_notification_on_flag} >
                        <sms on={scene_sms_notification_on_flag}
                                number={scene_sms_notification_number} />
                        <email on={scene_email_notification_on_flag}
                                address={scene_email_notification_address} />
                </notification>
        </security_scene>
</zwave>
```

{scene_last_arm_event} is an event that caused the Security Scene to go into Arm state last time.
{scene_last_disalarm_event} is an event that caused the Security Scene to go into Disarm state last time.
{scene_last_alarm_on_event} is an event that caused the Security Scene to go into Alarm state last time.
{scene_last_alarm_off_event} is an event that caused the Security Scene Alarm state to be reset last time. This could be a user action.

	Example Response:
	<?xml version="1.0"?>
	<zwave>
	  <security_scene desc="10005"
	  name="SecurityScene_1"
	  active="1" armed="0" alarmed="0"
	  utime="1435735293" >
	  <arm exec_scened="2" >
	  <event evtype="1" status="0"
	  sutime="0" uri="zwif_bsensor"
	  cmd="2" ifd="3145988" state="255" />
	  <lt type="0" time="0" />
	  </arm>
	  <disarm exec_scened="0" >
	  <lt type="0" time="0" />
	  </disarm>
	  <alarm exec_scened="1" >
	  <event evtype="1" status="0"
	  sutime="0" uri="zwif_bsensor"
	  cmd="2" ifd="3145988" state="0" />
	  <lt_on type="0" time="0" />
	  <lt_off type="0" time="0" />
	  </alarm>
	  <notification on="1" >
	  <sms on="0"
	  number="6512345678" />
	  <email on="1"
	  address="abcdef%40abcmail.com" />
	  </notification>
	  </security_scene>
	</zwave>

	On error return Error 1

40.4 Save Security Scene

Request	scened={scene_id}\r\n name={scene_name}\r\n active={scene_active_flag}\r\n is_armed={is_armed}\r\n scene_id_at_arm={scene_arm_scened}\r\n scene_id_at_disarm={scene_disarm_scened}\r\n scene_id_at_alarm={scene_alarm_scened}\r\n is_notification_on={scene_notification_on_flag}\r\n is_notification_by_sms_on={scene_sms_notification_on_flag}\r\n is_notification_by_email_on={scene_email_notification_on_flag}\r\n notification_sms_number={scene_sms_notification_number}\r\n notification_email={scene_email_notification_address}\r\n event_arm=={scene_event_type}\r\n uri={scene_event_uri}\r\n cmd={scene_event_cmd}\r\n ifd={scene_event_desc_id}\r\n {scene_event_device_parameters}\r\n [event_arm...]

	event_disarm=={scene_event_type}\r\n uri={scene_event_uri}\r\n cmd={scene_event_cmd}\r\n ifd={scene_event_desc_id}\r\n {scene_event_device_parameters}\r\n [event_disarm...] event_alarm=={scene_event_type}\r\n uri={scene_event_uri}\r\n cmd={scene_event_cmd}\r\n ifd={scene_event_desc_id}\r\n {scene_event_device_parameters}\r\n [event_alarm...]
	{scene_cmd} is CMD_SECURITY_SCENE_SAVE If {scene_id} is 0, new Security Scene is created; else existing Security Scene is edited. {is_armed} is used only when creating a new Security Scene. 0 means the Security Scene is created and is in disarmed state. 1 means this Security Scene is automatically armed after creation. scene_id_at_arm, scene_id_at_disarm and scene_id_at_alarm are optional parameters. Omited if there is no Scene for the specific state. Example Request: cmd=103 scened=10000 name=SecurityScene_1 active=1 scene_id_at_arm=2 scene_id_at_disarm=0 scene_id_at_alarm=1 is_notification_on=1 is_notification_by_sms_on=0 is_notification_by_email_on=1 notification_sms_number=65123456789 notification_email=abcdef%40abcmail.com event_alarm=1 uri=zwif_bsensor cmd=2 ifd=3145988 state=255 type=1
Response	<?xml version=”1.0”?><zwave><security_scene desc=”{scene_id}” /></zwave> {scene_id} is the id for the new Security Scene created, or the existing Security Scene edited.
	On error returns Error 1

40.5 Delete Security Scene

Request	scened={scene_id}\r\n
	{scene_cmd} is CMD_SECURITY_SCENE_DELETE
Response	<?xml version=”1.0”?><zwave><security_scene desc=”{scene_id}” /></zwave>
	On error returns Error 1

40.6 Security Scene Set State

Request	scened={scene_id}\r\n arm={scene_arm_state}\r\n alarm={scene_alarm_state}\r\n
	{scene_cmd} is CMD_SECURITY_SCENE_SET_STATE Both arm and alarm states can be set together or one at a time i.e., if either arm or alarm parameter is given, the other becomes optional. For arm, the valid values are 0 and 1. Other values will be ignored. For alarm, the only valid value is 0 (i.e., you can only reset Alarm state).
Response	<?xml version=”1.0”?><zwave><security_scene desc=”{scene_id}” /></zwave>
	On error returns Error 1

40.7 Security Scene Get Statelog

Request	type={security_scene_statelog_type}\r\n	
	{scene_cmd} is CMD_SCENE_GET_STATELOG	
Response	<?xml version="1.0"?> <zwave> <statelog type=”{scene_statelog_type}” counter==”{scene_change_counter}” scened=”{scene_id}” /> [<statelog... />...] </zwave> {scene_id} is 0 if not required by the related operation. Example Response: <?xml version="1.0"?> <zwave> <statelog type="256" counter="2" scened="10001" /> <statelog type="512" counter="1" scened="10001" /> <statelog type="1024" counter="25" scened="10001" /> <statelog type="2048" counter="5" scened="10000" /> <statelog type="4096" counter="1" scened="10001" /> </zwave> Where statelog type is	
	ADD	256
	DELETE	512
	EDIT	1024
	ARM	2048
	ALARM	4096
	On error return Error 1	
Note	This is a light weight command that can be used to poll the server so that if there was any operation on the server side (e.g., Security Scene added by another client), the client could use CMD_SCENE_GET_STATELOG’s response to detect this change and fetch the latest information from the server, if needed. STATELOG Command number of statelog (CMD_SCENE_GET_STATELOG) is the same as normal Scenes. It provides flexibility to either poll only normal Scenes, only Security Scenes, or both. This is done to avoid double polling by any Client which has to show normal and Security. Scenes on the same page	

41 Interface APIs Used In Scene/Security Scene

Some of the interface APIs can be used in Scene as actions or events, or as disarm/arm/alarm events in a Security scene. The following table lists the interface APIs that can be used.

	Scene		Security Scene		Web API
	As Scene action	As Scene event	As disarm/arm event	As alarm event	Version
Basic	×				1
Binary switch	×				1
Multilevel switch	×				1
Multilevel switch level change	×				1
Color switch	×				1
Door lock	×	×	×		1
Barrier Operator	×				1
Sound Switch	×				1
Thermostat Mode	×				1
Thermostat setpoint	×				1
Window Covering	×				1
Basic event		×		×	1
Binary sensor		×		×	1
Multilevel sensor		×			1
Alarm		×		×	2
Central Scene		×	×		1

When the interface APIs are used in Scene or Security Scene, the format is:

uri=<uri of the API>
cmd=<Get or Set cmd>
ifd=<desc>
<interface parameter specific to the interface API>

For example:

uri=zwif_bin_sw
cmd=4
ifd=8
value=255

The “cmd” value will be either the *Get* cmd of the interface API, if the interface is used as an *event* for Scene or Security scene, or the *Set* cmd of the API, if the interface is used as an *action* for a Scene.

The interface-specific parameters will vary depending on the individual interface APIs. For most interface APIs, parameters in *Set* command will be used for Scene action, and parameters in *report* command will be used for an event.

Most of these parameters will be consistent with the interface APIs parameters when they are not used with Scene/Security scene. However, some interfaces have a different set of parameters. The following section illustrates all parameters for individual interface APIs:

41.1 Basic Set Event

Unlike other interface APIs where the *Report* command parameter will be used for Scene/Security scene event, for Basic interface API, Basic Set parameter will be used for both action and event, with different 'cmd':

Scene action	
URI	/zwif_basic
cmd	CMD_BASIC_SET
parameter	value={basic_state}

Scene/Security Scene event	
URI	/zwif_basic
cmd	CMD_BASIC_EVENT
parameter	value_low={lower range} value_high={higher range}

41.2 Binary Switch Set Action

Scene action	
URI	/zwif_switch
cmd	CMD_BINARY_SWITCH_SET
parameter	value={bin_sw_state}

41.3 Multilevel Switch Set Action

Scene action	
URI	/zwif_level
Cmd	CMD_ MULTILEVEL_SWITCH_SET
parameter	value={multi_lvl_sw_state}

41.4 Multilevel Switch Level Change Set Action

Scene action	
URI	/zwif_level
Cmd	CMD_ MULTILEVEL_SWITCH_LVL_CHG_SET
parameter	start_stop={Level change} dir={direction} ignore_start_lvl={Ignore start level?} start_lvl={Level} dur={duration} sec={Optional: with secondary switch?}

	step={Optional: with secondary switch step size}
Note	start_stop={stop=0, start=1} dir={up=0, down=1, no-change=3} ignore_start_lvl={Yes=1 (use current level), No=0 (use start level)} start_lvl={Level from 1 to 99, (0=Off/Disable, 255=On/Enable)} dur={0=Instantly, 1..127=seconds, 128..254=(1 to 127 minutes)} sec={Optional: with secondary switch, 0=Increment, 1=Decrement, 2=Reserved, 3=No Increment/Decrement} step={Optional: with secondary swtich step size}

41.5 Color Switch Set Action

Scene action	
URI	/zwif_color
Cmd	CMD_COLOR_SWITCH_SET
parameter	valueN={value_for_color_component_id_N} color_dur={Optional: duration for color transition, default: 0=Instantly}
Note	color_dur={0=Instantly, 1..127=seconds, 128..254=(1 to 127 minutes)}

Where N is 0 to 8 corresponding to 'Color Component ID' 0 to 8 (0=Warm White to 8=Indexed Color). For more information, see [4] for Color Switch Component IDs. Sending 1 single set action command with multi parameters (example: value0=XX to value8=YY) is supported.

41.6 Door Lock Set Action

Scene action	
URI	/zwif_dlck
Cmd	CMD_DLOCK_OP_SET
parameter	mode={dlock_mode}

Scene/Security Scene event	
URI	/zwif_dlck
Cmd	CMD_DLOCK_OP_GET
parameter	mode={dlock_mode}

41.7 Barrier Operator Set Action

Scene action	
URI	/zwif_barrier_op
Cmd	CMD_BARRIER_OP_SET
parameter	target_state={target_value}

41.8 Thermostat Mode Set Action

Scene action	
URI	/zwif_thrmo_md
Cmd	CMD_THRMO_MODE_SET
parameter	mode={thrmo_mode}

41.9 Thermostat Setpoint Set Action

Scene action	
URI	/zwif_thrmo_setp
Cmd	CMD_THRMO_SETPT_SET
parameter	type={setpt_type} value={setpt_value_mag} precision={setpt_precision} unit={setpt_unit} size={setpt_octet_size}

Where setpt_octet_size = 1, 2 or 4.

For setpoint value = 25.5 deg Celsius, setpt_value_mag=255, precision=1, unit=0.

For Thermostat Setpoint version 2 and below: Parameters ‘precision’, ‘unit’ and ‘size’ SHOULD base on CMD_THRMO_SETPT_GET to get Thermostat Setpoint Report’s ‘precision’, ‘unit’ and ‘size’.

For Thermostat Setpoint version 3 and above: Parameters ‘precision’, ‘unit’ and ‘size’ SHOULD base on CMD_THRMO_SETPT_RANGE_GET to get Thermostat Setpoint Capabilities Report’s using ‘min_precision’, ‘min_unit’ and ‘min_size’.

41.10 Sound Switch Tone Play Set Action

Scene action	
URI	/zwif_sound_sw
Cmd	CMD_SOUND_SWITCH_TONE_PLAY_SET
parameter	tone_id={tone_id} volume={tone_volume}
Note	{tone_id} value: 0 : Stop playing any tone. 1-254 : Play the device supported tone id. {volume} value: 1..100 : Volume level in percentage. The parameter ‘volume’ is to ensure tone is always playback at the desired loudness when scene action is executed. Setting for ‘volume’ will not be used for scene status decision purpose. Only parameter ‘tone_id’ is used for deciding scene status for ‘Active’ or ‘Inactive’ indication. Set tone_id=0 if you want to stop (muted) any tone playing.

41.11 Window Covering Set Action

Scene action	
URI	/zwif_window_cvr
Cmd	CMD_WINDOW_COVERING_SET
parameter	param_id_list={list of param_id to be set, separated by commas} param_value_list={list of value to be set for respective param_id, separated by commas} duration={duration, factory default duration=255}
Note	Scene now only handle with duration=255 (factory default)

41.12 Window Covering Level Change Set Action

Scene action	
URI	/zwif_window_cvr
Cmd	CMD_WINDOW_COVERING_LEVEL_CHANGE_SET
parameter	start_stop={0=Stop, 1=Start} param_id={param_id to start/stop level change} direction={direction, applicable for start_stop=1} duration={duration, factory default duration=255, applicable for start_stop=1}
Note	Scene now only handle with duration=255 (factory default) Direction={0:level change increasing, 1:level change decreasing} For param_id=10, 11: Direction={0:Closing to the left, 1:Closing to the right} For param_id=22, 23: Direction={0:Closing down inside, 1:Closing up inside} For other param_id: Direction={0:Opening, 1:Closing} Parameters ‘direction’ and ‘duration’ are only applicable for start_stop=1 (start)

41.13 Binary Sensor Report Event

Scene/Security Scene event	
URI	/zwif_bsensor
Cmd	CMD_BINARY_SENSOR_GET
parameter	state={bin_snsr_state} type={bin_snsr_type}
Note	“type={bin_snsr_type}” is only applicable for binary sensor Version 2 or above

41.14 Multilevel Sensor Report Event

Scene/Security Scene event	
URI	/zwif_sensor
Cmd	CMD_MULTILEVEL_SENSOR_GET
parameter	type={snsr_type} unit={snsr_unit} value_low={lower range of snsr_value} precision_vl={lower range of snsr_precision} value_high={higher range of snsr_value } precision_vh={higher range of snsr_precision }

41.15 Alarm Report Event

Scene/Security Scene event	
URI	/zwif_alrm
Cmd	CMD_ALARM_GET
parameter	va_type={alarm_vtype} va_level={alarm_level} za_type={alarm_ztype} za_event={alarm_event_list} [za_state_idle_event={alarm_state_idle_event_list}]
Note	"va_type={alarm_vtype}" and "va_level={alarm_level}" are normally use for Alarm Version 1. "za_type={alarm_ztype}" and "za_event={alarm_event}" are only applicable for Alarm Version 2 and above. If “va_type” or “va_level” is not in-use, set the value -1 (ignore) for alarm_vtype or alarm_level respectively. (Example: va_type=-1 va_level=-1) Parameters "alarm_event_list" and "alarm_state_idle_event_list" are list of required alarm events values to be triggered and this list of values are separated by commas. When "alarm_event_list" include 0 (State Idle), the parameter za_state_idle_event={alarm_state_idle_event_list} is required, otherwise "za_state_idle_event" parameter will be ignored. Note: "za_state_idle_event_list" of value 0 is considered as "Any Event Parameter". Example 1 (Alarm v1): va_type=1&va_level=255 OR va_type=1&va_level=255&za_type=0&za_event=0 Example 2 (Notification CC v8, Smoke Alarm, smoke detected): va_type=-1&va_level=-1&za_type=1&za_event=1,2 Example 3 (Notification CC v8, Smoke Alarm, smoke detected event has cleared): va_type=-1&va_level=-1&za_type=1&za_event=0&za_state_idle_event=1,2 OR va_type=-1&va_level=-1&za_type=1&za_event=0,0&za_state_idle_event=1,2 Example 4 (Notification CC v8, Smoke Alarm, smoke detected and smoke detected event has cleared): va_type=-1&va_level=-1&za_type=1&za_event=0,1&za_state_idle_event=1
version	2: added new parameter (za_state_idle_event)

41.16 Central Scene Command Event

Scene/Security Scene event	
URI	/zwif_centr_scene

Cmd	CMD_CENTRAL_SCENE_REPORT
parameter	central_scene_key_number={scene_number} central_scene_key_attribute={key_attribute}
Note	central_scene_key_attribute supported are • 0=Key Pressed 1 time, • 1=Key Released, • 2=Key Held Down, • 3=Key Pressed 2 times, • 4=Key Pressed 3 times, • 5=Key Pressed 4 times, • 6=Key Pressed 5 times.

References

[1] Silicon Labs, INS14428, INS, Z-Ware Web User Guide
[2] Silicon Labs, APL13031, APL, Z-Wave Networking Basics
[3] Silicon Labs, SDS10242, SDS, Z-Wave Device Class Specification
[4] Silicon Labs, SDS13781, SDS, Z-Wave Application Command Class Specification
[5] Silicon Labs, SDS13782, SDS, Z-Wave Management Command Class Specification
[6] Silicon Labs, SDS13783, SDS, Z-Wave Transport-Encapsulation Command Class Specification
[7] Silicon Labs, SDS13784, SDS, Z-Wave Network-Protocol Command Class Specification