

Programming Guide

XML 1.0 Interface

AK-SM800A / AK-SM800/ AK-SC series



Table of Content

Table of Content.....	2
Revision history.....	4
Disclaimer	5
Purpose of this document	6
Purpose of the XML interface	6
TCP Connection Requirements	7
Communication Requirements	7
Unsupported characters	7
Best practices for using the Danfoss XML open interface.....	8
Guidelines for use of XML open interface with Danfoss AK SM800 series front end	10
Guidelines for use of XML open interface with Danfoss AK SC355 series front end	16
Guidelines for use of XML open interface with Danfoss AK SC255 series front end	17
Command Overview	18
Data Types.....	20
Read Commands	21
Commands and Responses	22
1.1 read_generic_alarms.....	22
1.2 read_device_alarms	24
1.3 read_val	27
1.4 read_units	32
1.5 write_unit	34
1.6 read_devices	35
1.7 read_device.....	40
1.8 read_meters	44
1.9 read_meter.....	47
1.10 read_date_time	50
1.11 read_parm_versions	51
1.12 read_parm_info	52
1.13 read_dyn_list_info.....	55
1.14 read_menu_groups	58
1.15 read_device_summary.....	59
1.16 schedule_summary	61
1.17 schedule_detail.....	63
1.18 read_store_schedule	66
1.19 write_store_schedule	72
1.20 set_store_time.....	73
1.21 read_hvac_service	76
1.22 set_hvac_service.....	78
1.23 read_hvacs	80
1.24 read_hvac_unit	81
1.25 write_hvac_unit.....	83
1.26 write_hvac_setback	85
1.27 read_lighting.....	87
1.28 read_lighting_zone	89
1.29 set_zone_override.....	92
1.30 write_lighting_zone.....	94
1.31 read_holidays	96

1.32	write_holiday_sch	98
1.33	write_schedule_times	100
1.34	set_offset	102
1.35	read_suction_group	103
1.36	set_suction_group	105
1.37	read_circuit	106
1.38	set_circuit	111
1.39	read_condenser	113
1.40	set_condenser	116
1.41	read_inputs	118
1.42	read_relays	120
1.43	read_alarm_relays	122
1.44	read_sensors	124
1.45	read_var_outs	126
1.46	read_input	127
1.47	read_relay	129
1.48	read_sensor	131
1.49	read_var_out	133
1.50	read_monitor_summary	135
1.51	read_monitor_detail	138
1.52	set_monitor_point	143
1.53	read_history	145
1.54	read_history_cfg	152
1.55	read_device_history_cfg	156
1.56	write_digi_op	158
1.57	write_val	161
1.58	write_alarm_ack	164
1.59	write_alarm_clear	166
1.60	set_defrost	167
1.61	set_light	168
1.62	set_main_switch	169
1.63	set_cleaning	170
1.64	set_night_setback	171
1.65	set_shutdown	172
1.66	read_system_status	173
1.67	read_license_data	175
1.68	alarm_summary	177
1.69	alarm_detail	180
1.70	write_alarm_ack	185
1.71	write_alarm_clear	187
1.72	start_history_query	188
1.73	read_query_status	189
1.74	read_query_data (binary data is returned)	193
1.75	abort_query	195
	Alarm Messages	196
2	Establishing a Trusted Browser Connection	201
	Creating a Custom Certificate Authority	202
	Encoding of Files	203
	Commands and Responses	204
2.1	gen_ssc	204

2.2	gen_csr	207
2.3	load_cert.....	209
2.4	load_key_pair.....	211
3	Reference Information.....	214
3.1	Node Types	214
3.2	Error Codes	215

Revision history

October 1, 2009 Version 2.101

Released with AK255 2.101 software release

February 26, 2010 Version 2.101a

Documented valid_only attribute in read_val, read_input, read_relay
read_sensor, read_var_out, and read_monitor detail commands.

January 11th, 2011 Version 2.151

Added the following methods:

- 1) read_suction_group, read_circuit, read_condenser
- 2) set_suction_group, set_circuit, set_condenser, set_offset

October 07th, 2011

Added the pulse/vol type of meter support in read_meter and read_meters command:

February 14th, 2012

Modified tag definition for alarm received (name_id). See Alarm Messages for detail
on name_id.

April 30th, 2012

Deprecated Methods. Will no longer be supported, but will remain in current and
future builds.

August 2nd, 2012

Changed read_generic_alarms to add index and count to request. This is to reflect fix
for reset when requesting more than 500 alarms. Total returned max is now 100, and
must loop through remaining alarms for request.

November 11, 2013

Added new function for reading license information.

11 November 2015:

Formatting

August 8, 2017

Added TCP Connection Requirements

September 13, 2017

Added 'Best Practices' and 'disclaimer' sections

May 28th, 2021

Added 'Unsupported characters' section

February 28th, 2023

Added 'Establishing a Trusted Browser Connection' section

July 27th, 2023

Update "Guidelines for use of XML open interface with Danfoss AK-SM800A series
front end" to include new header authentication.

April 2023

Added Session Control- Security Enforcing level (SM800A software package R4.0.x and
above)

Disclaimer

Danfoss accepts no responsibility for possible errors in this document containing the XML 1.0 messaging protocol (the “Application”). Danfoss reserves the right to alter the Application without notice. All trademarks in this material are property of the respective companies. Danfoss and Danfoss logotype are trademarks of Danfoss A/S. All rights reserved.

The use of the Application with approved Danfoss products is provided on an “AS IS” and “AS AVAILABLE” basis for the intended purposes as determined by Danfoss only and any use is at the user’s sole risk. Danfoss does not warrant that the application will meet your requirements or that the operation hereof will be uninterrupted or error-free.

DANFOSS DISCLAIMS ALL WARRANTIES AND CONDITIONS REGARDING THE APPLICATION, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE, ACCURACY AND NON-INFRINGEMENT OF THIRD PARTIES’ RIGHTS. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT SHALL DANFOSS BE LIABLE FOR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, WHATSOEVER, INCLUDING, WITHOUT LIMITATION, DAMAGE TO PROPERTY, DAMAGES FOR LOSS OF SAVINGS OR PROFIT, OR LOSS OF DATA ARISING OUT OF ANY USE OF THE APPLICATION.

Purpose of this document

Describe the System Manager XML data interface and the specific requests and responses.

Purpose of the XML interface

Provides communication to a System Manager network using XML to facilitate access by non-Danfoss clients such as higher level data management systems. This includes read access for current values such as temperatures and pressures, configuration data, alarms, and history and limited write access and alarm acknowledgement. XML provides a means of describing data that is easy to understand and process. It also facilitates access from web browser clients.

TCP Connection Requirements

To ensure proper communication, TCP/IP protocols must be adhered to avoid disruption in socket availability.

The System Manager only allows for one request per connection. The flow of communication is as follows:

- 1) Establish socket connection on configured web server port (typically port 80).
- 2) Send xml/http request.
- 3) Receive xml/http response from web server.
- 4) Close socket (both client/server send FIN on socket connection).

The System Manager does not accept “keep-alive” type socket connections, so every request must be on a new socket. If the client does not close the socket, then there will be delays before the System Manager can force close the socket (sends RST). During this delay, there can be no more sockets accepted, and will stop communication until sockets become available.

Communication Requirements

Prior to Version VG08_0733, the maximum rate of XML requests to the SM8xx should not exceed 1 message / 5 seconds.

Unsupported characters

There are 2 different characters that cannot be used in the System Manager XML interface, or on the system manager local display, because of URL encoding and decoding, these are plus (+) and percentage (%).

There are 5 different characters that need to be escaped when communicating using XML these are double quotes ("), single quotes ('), less than (<), greater than (>) and ampersand (&).

Double quotes (") needs to be escaped to "

Single quotes (') needs to be escaped to '

Less than (<) needs to be escaped to <

Greater than (>) needs to be escaped to >

Ampersand (&) needs to be escaped to &

Best practices for using the Danfoss XML open interface

Since the Danfoss SC 255,355 and SM800/A series front ends are designed to be able to support many different applications and configurations globally, the XML open interface is designed to be flexible to accommodate the various needs for 3rd party systems to interface with the Danfoss system front ends to obtain the required parameters.

Using the recommendations in conjunction with the relevant XML command documents, should help the third-party developer provide a more robust and resilient solution.

1. All responses are default returned in the units of measurement that the local display is set to in the front end.

Recommendation: To ensure consistency, the “units” attribute is included in all requests.

2. All responses are default returned in the language that the local display is set to in the front end.

Recommendation: To ensure consistency, the “lang” attribute is included in all requests.

3. Many commands provide both a numeric and a text string response for values, where this is the case:

Recommendation

a): String responses are used for displaying data in a user interface (HMI).

b): Numeric values are used when a 3rd party system is interpreting this response to make a decision in their own platform.

4. When using enumerated lists from case and pack controllers obtained with the command *read_dyn_list_info*, it must be understood that the enumerated list is specific to the *device_id* being requested.

Recommendation: This must be maintained separately for each *device_id* that is connected to the front end and computed by the 3rd party system, the list consists of text strings and their corresponding numerical values for decision making.

5. When requesting lists of read/write parameters from case and pack controllers with the command *read_parm_info*, it must be understood that the parameter list is specific to the *device_id* being requested.

Recommendation: This must be maintained separately for each *device_id* that is connected to the front end and computed by the 3rd party system, the list consists of text strings and their corresponding numerical values for decision making.

6. There are several methods to request parameter data from the front end. Variables may be read using tags instead of *cid*, *vid* identifiers. The *read_parm_info* command provides tag, *cid*, *vid*, information.

Recommendation

a): Request the data using the *cid*/*vid* as the numerical value. The “tag” attribute should not be included if the “*cid*” and “*vid*” attributes are included.

b): The “field” element should not be used when requesting the variable’s actual data.

7. Read commands for devices, device, sensor, var_outs, inputs, and relays will contain attributes describing the numerical value and associated units of measurement. <parval=” int|signed decimal”> <units=”string”> <units_index=”int”> The attribute <parval> contains the parameter’s value which can be null if the device or point in question is offline, the attribute <units> contains the type of value, and the <units_index> contains the integer value for the internal location of the unit description

Recommendation

a): The attribute <parval> should be used to obtain the raw value of the parameter.

b): The attribute <units_index> is an integer value that can be used for quick lookup, instead of a string comparison.

Guidelines for use of XML open interface with Danfoss AK SM800A series front end

The following are guidelines that are required to be followed when using the open XML interface to request data from SM800A series devices

To ensure compatibility it is highly recommended that the SM 800A is upgraded to minimum firmware package version 4.0.x prior to using the XML interface. **Important note: release 4.0 includes Session Control functionality that is defaulted to 'Strict Mode' upon installation. If Strict Mode remains configured, please consult with the mandatory XML updates required, as described below**

1. All requests by default return responses in language set as the display language on local SM800A screen. If other languages are needed, they must be enabled and the *lang=* attribute is passed in each request.

Recommendation: To ensure consistency, the "lang" attribute is included in all requests.

2. The below refers to Header authentication (seen in software prior to release 4.0.x). For AK-SM800A software 4.0 and above, please consult the section about Session Control

If "Use Header Authentication" is not enabled in the comm settings, all requests should contain a valid username and password in the *user=* and *pass=* attributes of a request.

If "Use Header Authentication" is enabled, then the user authentication is stored in the HTTP header "AKSM-Auth" in the following format according to internet standard ([RFC 7617](https://tools.ietf.org/html/rfc7617)).

Listing information regarding 'Basic' HTTP Authentication Scheme:

- 1) User authentication is stored in HTTP header in the following format according to Internet Standards is as "User Name":"Password". This full text line is encoded in **base64**.
 - a. Must start with field ID: AKSM-Auth: Basic "**encoded auth string**"
- 2) No other options are needed in the HTTP header for the SM800A at this time.

Example of a header would be:

userid = *<TEXT excluding ">

password = *TEXT

user-pass = userid ":" password

If the user agent wishes to send the userid "Aladdin" and password

"open sesame", it would use the following header field:

AKSM-Auth: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

POST http://10.38.25.31/xml.cgi HTTP/1.1

Host: 10.38.25.31

Connection: keep-alive

AKSM-Auth: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

Content-Length: 139

Accept: text/xml

Accept-Charset: utf-8

Origin: http://10.38.25.31

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36

DNT: 1

Content-Type: text/xml

Referrer: http://10.38.25.31/
Accept-Encoding: gzip

Required items are in **bold**. Other items are auto-generated, if using current libraries that generate headers for HTTP.

AKSM-Auth	Authentication credentials for HTTP authentication .
Accept	Media type(s) that is/are acceptable for the response. See Content negotiation .
Content-Length	The length of the request body in octets (8-bit bytes).
Accept-Charset	Character sets that are acceptable. (utf-8, utf-16, etc...)
Content-Type	The Media type of the body of the request (used with POST and PUT requests).
Accept-Encoding	List of acceptable encodings. (HTTP Compression algorithms)

Here is the link to HTTP Authentication: Basic and Digest Access Authentication:
https://www.ietf.org/http_auth

3. Write commands will function only with a valid username and password and that the user's authorization level in the SM800A allows for configuration changes.

4. All requests should contain the *units*= attribute to identify data will be returned in US or SI units.

Recommendation: Use *units*= attribute for data requests.

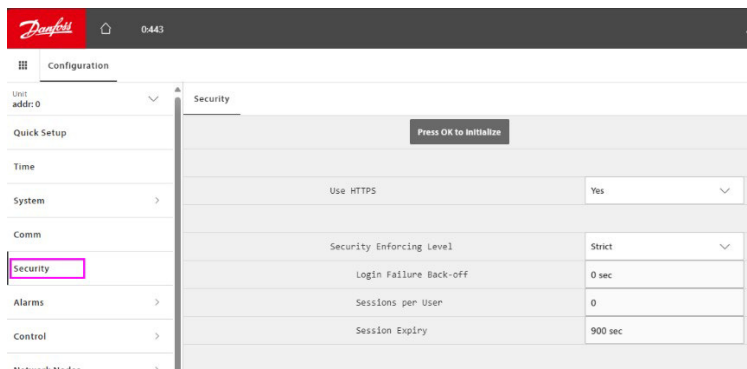
5. A minimum of 4 seconds' delay needs to be used between receiving a response of a request before another request is made.

To read multiple values in the most efficient way, it is recommended to use the *read_val* command, reading a max of 100 parameters in a single request. Pausing 4 seconds before the next request.

Note: It is possible to request parameters from differing devices in the same single *read_val* request

Session Control (Configuration>Security)

A new feature added in AK-SM800A from software package 4.0.x and above is seen under Configuration->Security menu and strengthens the System Managers security level.



What is Session Control?

Session control in the 800A is an authentication rules engine/module and is intended to improve the overall security posture of your system. Session control offers different configuration settings (Backward compatibility, Permissive and Strict). Depending on how these settings are configured, the remote interface will need to comply accordingly.

What is a remote Interface?

The term remote interface refers to StoreView Browser 5 (SvB5), StoreView Web (SvW), Alsense™ (Danfoss Digital services) and 3rd party XML1.0. SvB5, SvW and Alsense™ already support Session control. 3rd party XML1.0 users will need to update requests if Strict mode remains selected (see below)

What are these Session Control settings?

The important aspect to note is that immediately after installing System Manager 800A software package 4.0.x and above, Strict mode is set by default. These settings can always be managed via the System Manager Configuration>Security menu

Session Control setting	Description
Backward Compatible	Not recommended but when selected, http is available, and no session tokens needed
Permissive (used as a transition to strict level)	Set this level to view any error responses so that adjustments can be made in preparation for strict mode. Permissive is allowing both the old authentication and the new authentication scheme at the same time.
Strict (defaulted immediately after s/w package R4.0.x install)	Require HTTPS connection XML requests cannot contain usernames and passwords. Must provide session token in the AKSM-auth header. HTTPS becomes mandatory and authentication moves from plain-text to session-based authentication. Strict mode will sanitize all strings in the Northbound connection of XML, rejecting any commands that conflict with the sanitization

If Strict mode remains configured, ensure your XML messaging is updated according to the description of Strict mode below

Backward Compatible (lowest security level – assuming use header auth is disabled)

- XML requests contain username and password.
- No sanitizing of requests, same security as prior releases
- No changes needed on existing XML connections

Permissive (can be used as a transition to strict level)

- XML requests can contain username and password, and or a session token. The session token is received in the response of the login request.
- If the session token is provided in the AKSM-auth header, usernames and passwords are ignored
- XML request gets sanitized, a warning is returned in the response of request if any issues discovered
- Take note of these warnings in preparation for switching to strict mode
- If the session token is not used, session expiry cannot be applied
- Possible / Required changes from 3rd party XML perspective
 - No session token usage – no changes needed
 - Session token usage – remove username / password from body and observe warnings in preparation for Strict

Strict (highest security level)

- XML requests cannot contain usernames and passwords, but must provide the session token in the AKSM-auth header. If an issue is discovered in the request, it gets rejected
- Required changes from 3rd party perspective
 - username and password must be removed from the XML request payload
 - session token must be placed in a CORS header called AKSM-auth

Example below;

First send a login request:

Content-Type: text/plain

User-Agent: PostmanRuntime/7.32.3

Accept: */*

Postman-Token: dbc0f2cf-9b6e-4f33-9a6f-e007e940cfa3

Host: 10.5.240.66

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Content-Length: 50

<cmd action="getauth" user="bob" password="123" />

Then get the session token from the response:

```
<?xml version="1.0" encoding="utf-8"?>

<resp action="getauth" user="bob" password="123" error="0">

  <authorization>01-02</authorization>

  <language>English</language>

  <authtype>16777215</authtype>

  <access>01-02</access>

  <accesslevel>1</accesslevel>

  <strongpassword>0</strongpassword>

  <session_token>d9035c68fcd5aace6128e45de0dbfbd1e92351fc0ae192c2af43325f90328c4e</session_token>

</resp>
```

From now on all requests need to have this token in the header and Username and password have to be removed from the body of request:

AKSM-auth: d9035c68fcd5aace6128e45de0dbfbd1e92351fc0ae192c2af43325f90328c4e

Content-Type: text/plain

User-Agent: PostmanRuntime/7.32.3

Accept: */*

Postman-Token: 40a35551-6a41-46a0-9343-6dc12e10c58c

Host: 10.5.240.66

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Content-Length: 38

Setting the session control to either permissive or strict mode enables further configurations:

- Login Failure Back-Off: Time in seconds for how long to wait to reattempt a login after a failure. The waiting time is accumulating, means when for instance set to 5 seconds, you must wait first 5 seconds, then 10 seconds and then 15 after a failed login attempt to try again.
- Free Attempts: How many login attempts before the back-off time sets in
- Block: When in the Back-Off stage block login for the whole user account or for user account + IP address from where the login attempt was made

- Session per User: How many active sessions a user account can have, before a new try to open a session is blocked. When set to 0, there is no limit on how many sessions.
- Session Expiry: After how many seconds expires an idle session. When a session expired the user gets logged out.

Guidelines for use of XML open interface with Danfoss AK SC355 series front end

The following are guidelines that are required to be followed when using the open XML interface to request data from SC355 series devices

To ensure compatibility it is highly recommended that the SC 355 is upgraded to firmware version 3.107 prior to using the XML interface.

1. All requests by default return responses in language set as the display language on local SC 355 screen. If other languages are needed they must be enabled and the *lang=* attribute is passed in each request.

Recommendation: To ensure consistency, the “lang” attribute is included in all requests.

2. All requests should contain a valid username and password in the *user=* and *pass=* attributes of a request.

Recommendation: Use *user=* and *pass=* attributes for data requests.

3. Write commands will function only with a valid username and password and that the user’s authorization level in the SC 355 allows for configuration changes.

Recommendation: Use *user=* and *pass=* attributes for write requests.

4. All requests should contain the *units=* attribute to identify data will be returned in US or SI units.

Recommendation: Use *units=* attribute for data requests.

5. A minimum of 4 seconds’ delay needs to be used between receiving a response of a request before another request is made.

To read multiple values in the most efficient way, it is recommended to use the *read_val* command, reading a max of 40 parameters in a single request. Pausing 4 seconds before the next request.

Note: It is possible to request parameters from differing devices in the same single *read_val* request

Guidelines for use of XML open interface with Danfoss AK SC255 series front end

The following are guidelines that are required to be followed when using the open XML interface to request data from SC255 series devices

To ensure compatibility it is highly recommended that the SC 255 is upgraded to firmware version 2.246 prior to using the XML interface.

1. All requests by default return responses in language set as the display language on local SC255 screen. If other languages are needed, they must be enabled and the *lang=* attribute is passed in each request.

Recommendation: To ensure consistency, the “lang” attribute is included in all requests.

2. All request should contain a valid auth code and account code in the *auth=* and *acct=* attributes of a request.

Recommendation: Use *auth=* and *acct=* attributes for data requests.

3. Write commands will function only with a valid username and password and that the user’s authorization level in the SC255 allows for configuration changes.

Recommendation: Use *auth=* and *acct=* attributes for write requests.

4. All requests should contain the *units=* attribute to identify data will be returned in US or SI units.

Recommendation: Use *units=* attribute for data requests.

5. A minimum of 4 seconds’ delay needs to be used between receiving a response of a request before another request is made.

To read multiple values in the most efficient way, it is recommended to use the *read_val* command, reading a max of 15 parameters in a single request. Pausing 4 seconds before the next request.

Note: It is possible to request parameters from differing devices in the same single *read_val* request

Command Overview

Both requests and responses are coded in XML. HTTP is the protocol. Requests are made via the POST command. Both request parameters and the responses are carried in the body of the HTTP packet which are sent to and received from URL <http://999.999.999.999/html/xml.cgi> where 999.999.999.999 is the System Manager's ip address

Commands have the format:

```
<cmd action="command" ... possibly other attributes>
...
possibly other elements
...
</cmd>
```

Responses have the format:

```
<resp action="command" ...echo command attributes if any...error="int">
...
response elements
...
</resp>
```

Responses that are greater than 1024 characters are compressed using a common compression algorithm (gzip). When receiving responses via a browser, such as when using Flex or Java Script, the browser detects that a response is compressed by testing the HTTP headers and performs the decompression. Therefore, the compression is transparent to browser based applications. If the application is not browser based, then it must examine the HTTP headers itself and when necessary, decompress responses by calling a gzip decompression function. An attribute can be added to each command, `compress="0"`, to force compression off.

All commands accept the following elements:

`compress="int"`

1 = force data to be compressed.

0 = force data to be uncompressed.

If this attribute is unspecified then returned data is uncompressed if its size is less than or equal to 1024 and compressed if its size is greater than 1024

`lang="string"`

Languages:

"English"	- e
"Spanish (Americas)"	- a
"Portuguese (Brazil)"	- b
"German"	- g
"Chinese"	- c
"Dutch"	- d
"French"	- f
"Italian"	- i
"Russian"	- r

"Spanish (Spain)"	- s
"Portuguese (Portugal)"	- p
"Czech"	- z
"Hungarian"	- h
"Polish"	- l
"Turkish"	- t

SM8XX Series ONLY languages:

"Danish"	- n
"Swedish"	- w

units="string"

"u" or "U" U.S. units

"s" or "S" SI units

date_format="int"

"0" - month, day, year order with a two digit year.

"2" - day, month year order with a two digit year.

If this attribute is unspecified its value is defaulted to month, day, year order for the AK255R, AK355, SM8XX version or day, month, year order for the ak255E version.

time_format="int"

"0" - 12 hour clock (e.g. 10:30PM)

"1" - 24 hour clock (e.g. 22:30)

If this attribute is not specified it is defaulted to 12 hour clock

All responses include an "error" attribute whose value is 0 if a command was successful. All error codes are listed in the "Error Codes" section of this document.

Data Types

In the following sections references to data types are in italics. These are defined as follows.

<i>int</i>	Signed integer
<i>uint</i>	Unsigned integer
<i>time</i>	Time in either 24 or 12 hour format
<i>date</i>	Date in either MM/DD/YY or DD/MM/YY format
<i>string</i>	Any sequence of numeric or alpha characters
<i>temp</i>	Temperature value followed by SI or US units symbol <i>e.g.</i> -80.0°C
<i>pres</i>	Pressure value followed by SI or US units symbol
<i>ipaddr</i>	IP address <i>e.g.</i> 10.199.5.74
<i>signed decimal</i>	<i>e.g.</i> -100.0
<i>DDMM</i>	Two digit day followed by two digit month
<i>MMDD</i>	Two digit month followed by two digit day
<i>HOURL2</i>	Two digit hour in range 00 through 12 followed by two digit minute followed by AM or PM <i>e.g.</i> 10:00 PM
<i>HOURL24</i>	Two digit hour in range 00 through 23 followed by two digit minute <i>e.g.</i> 22:00

Read Commands

Read commands for devices, device, sensor, var_outs, inputs, and relays will contain attributes describing the numerical value and associated units of measurement.

`<parval=" int|signed decimal"> <units="string"> <units_index="int">`

The attribute `<parval>` contains the parameter's value which can be null if the device or point in question is offline, the attribute `<units>` contains the type of value, and the `<unittype>` contains the integer value for the internal location of the unit description (NOTE: this integer value will not change and can be used for quick lookup, instead of a string comparison). The list of unit types and units is:

Units Index Units

0	no units (used for on/off inputs, relay status)
1	psi
2	bar
3	degf
4	degc
5	percent
6	ppm
7	v
8	amp
9	kw
10	kwh
11	hz
12	gpm
13	fps
14	ph
15	degfd
16	defcd
17	min
18	hr
19	sec
20	fc
21	lpm
22	lps
23	hp
24	rpm

Commands and Responses

1.1 read_generic_alarms

Returns the list of alarm descriptions associated with a specific generic device. These appear in the device's edf file in a section demarcated by <ALARM_SECTION_START> and <ALARM_SECTION_END>.

UPDATE: The total max alarms that can be returned are 100. If more than 100 alarms are requested, the total alarms returned will still be 100.

Command

```
<cmd action="read_generic_alarms" nodetype="16" node="int" index="int" count="int"/>
```

Command Attribute	Data Type	Attribute value definition
nodetype	<i>int</i>	Always "16"
node	<i>int</i>	Device node number (within generic node type)
index	<i>int</i>	When sending this command, there is a max of 100 alarms that can be sent due to size limitations of text buffering. Send the <index> attribute to help with returning all alarms if more than 100. See examples. The <index> attribute is a "0" based integer, but the returning alarm_id is "1" based. So if requests start at index="50", the response will be starting alarm_id="51"
count	<i>int</i>	The requested total of alarms to get.

Response

```
<resp node="int" ...other attributes from command...error="int">
```

```
<count>int</count>
```

```
<alarm>string</alarm>
```

```
<alarm>string</alarm>
```

```
.....
</resp>
```

Response Element or Attribute	Data Type	Definition
count	<i>int</i>	Total number of alarm descriptions for device.
alarm	<i>string</i>	Description of an alarm that the addressed device may generate.
alarm_id	<i>int</i>	Attribute for the alarm element that identifies the alarm message. This is a "one-based" id always starting at 1.

EXAMPLE 1:

```
<cmd action="read_generic_alarms" nodetype="16" node="5"/>
<resp node="5" action="read_generic_alarms" error="0">
  <count>13</count>
  <alarm alarm_id="1">--- Contr. fault</alarm>
  <alarm alarm_id="2">--- S1 error</alarm>
  <alarm alarm_id="3">--- S2 error</alarm>
  <alarm alarm_id="4">--- S3 error</alarm>
  <alarm alarm_id="5">--- S4 error</alarm>
  <alarm alarm_id="6">--- S5 error</alarm>
  <alarm alarm_id="7">--- HighTemp air</alarm>
  <alarm alarm_id="8">--- Low temp air</alarm>
  <alarm alarm_id="9">--- DI1 alarm</alarm>
  <alarm alarm_id="10">--- Amb. mode</alarm>
  <alarm alarm_id="11">--- Max HoldTime</alarm>
  <alarm alarm_id="12">--- Inject prob.</alarm>
  <alarm alarm_id="13">--- Max Def.Time</alarm>
</resp>
```

EXAMPLE 2:

```
<cmd action="read_generic_alarms" nodetype="16" node="5" index="4" count="2"/>
<resp node="5" action="read_generic_alarms" error="0">
  <count>13</count>
  <alarm alarm_id="5">--- S4 error</alarm>
  <alarm alarm_id="6">--- S5 error</alarm>
</resp>
```

EXAMPLE 3:

```
<cmd action="read_generic_alarms" nodetype="16" node="5" index="0"/>
<resp node="5" action="read_generic_alarms" error="0">
  <count>189</count>
  <alarm alarm_id="1">--- Contr. fault</alarm>
  ...
  ...
  <alarm alarm_id="100">--- Max Def.Time</alarm>
</resp>
```

EXAMPLE 4:

```
<cmd action="read_generic_alarms" nodetype="16" node="5" index="100"/>
<resp node="5" action="read_generic_alarms" error="0">
  <count>189</count>
  <alarm alarm_id="101">--- Contr. fault</alarm>
  ...
  ...
  <alarm alarm_id="189">--- Max Def.Time</alarm>
</resp>
```

1.2 read_device_alarms

Returns the names and reference numbers of all active, acked, and cleared alarms that are associated with a particular device.

Command

```
<cmd action="read_device_alarms" nodetype="int" node="int" mod="int" point="int"/>
```

Command Attribute	Data Type	Attribute value definition										
nodetype	int	Node type. See list of node types in the “Reference Information” section										
node	int	Device node number										
mod	int	Module number										
point	int	Point number. Range depends upon nodetype; <table><tr><th>Nodetype</th><th>Range</th></tr><tr><td>0 (digital input)</td><td>1 through 8</td></tr><tr><td>1 (digital output)</td><td>9 through 16 (offset of 8 + point as displayed)</td></tr><tr><td>2 (analog input)</td><td>17 through 24 (offset of 16 + point as displayed)</td></tr><tr><td>3 (analog output)</td><td>25 through 28 (offset of 24 + point as displayed)</td></tr></table>	Nodetype	Range	0 (digital input)	1 through 8	1 (digital output)	9 through 16 (offset of 8 + point as displayed)	2 (analog input)	17 through 24 (offset of 16 + point as displayed)	3 (analog output)	25 through 28 (offset of 24 + point as displayed)
Nodetype	Range											
0 (digital input)	1 through 8											
1 (digital output)	9 through 16 (offset of 8 + point as displayed)											
2 (analog input)	17 through 24 (offset of 16 + point as displayed)											
3 (analog output)	25 through 28 (offset of 24 + point as displayed)											

Use the read_devices command to obtain the nodetype , node, mod, and point values for all devices configured in a System Manager.

Response

```
<resp action="read_device_alarms" nodetype="int" node="int" mod="int" point="int"
  error="int">
  <active>
    <ref name="string">int</ref>
    ...
    <ref name="string">int</ref>
    <total_active>int</total_active>
  </active>
  <acked>
    <ref name="string">int</ref>
    ...
    <ref name="string">int</ref>
    <total_acked>int</total_acked>
  </acked>
  <cleared>
    <ref name="string">int</ref>
    ...
    <ref name="string">int</ref>
    <total_cleared>int</total_cleared>
  </cleared>
```

```

<oldest>
  <time>time date</time>
  <ref>int</ref>
</oldest>
<newest>
  <time>time date</time>
  <ref>int</ref>
</newest>
<total>int</total>
</resp>

```

Response Element or Attribute	Data Type	Definition
ref	<i>int</i>	Alarm reference number that is unique within an System Manager unit Range 1 to 2147483647 Starts at 1 after alarm log is cleared
name	<i>string</i>	Name of alarm e.g. "IO Comm error". This is an attribute of the ref element.
active		Active alarms associated with this device. Element that contains the list of ref elements and the total_active element for active alarms.
acked		Acked alarms associated with this device. Element that contains the list of ref elements and the total_acked element for this device.
cleared		Cleared alarms associated with this device. Element that contains the list of ref elements and the total_cleared element for this device.
total_active	<i>int</i>	Number of ref elements in the active alarm list for this device.
total_acked	<i>int</i>	Number of ref elements in the acked alarm list for this device.
total_cleared	<i>int</i>	Number of ref elements in the cleared alarm list for this device.
oldest		Element that contains elements "time" and "ref". This provides the time, date, and reference number of the oldest alarm in the System Manager's alarm list for this device.
newest		Element that contains elements "time" and "ref". This provides the time, date, and reference number of the newest alarm in the System Manager's alarm list for this device.
total	<i>int</i>	Total of all alarms in all three lists for this device, total_active + total_acked + total_cleared

Example

```
<cmd action="read_device_alarms" nodetype="16" node="6" mod="0" point="0"/>
```

```
<resp nodetype="16" mod="0" node="6" action="read_device_alarms" point="0" error="0">
  <active>
    <ref name="I/O Comm error">1119</ref>
    <total_active>1</total_active>
  </active>
  <acked>
    <total_acked>0</total_acked>
  </acked>
  <cleared>
    <ref name="I/O Comm error">1103</ref>
    <ref name="I/O Comm error">1050</ref>
    <total_cleared>2</total_cleared>
  </cleared>
  <oldest>
    <time>10:36PM 04/04/95</time>
    <ref>1050</ref>
  </oldest>
  <newest>
    <time>05:17PM 05/04/95</time>
    <ref>1119</ref>
  </newest>
  <total>3</total>
</resp>
```

1.3 read_val

This command returns the value or field for each of a list of variables that are identified by node type, node, and either cid and vid, or tag

In an EKC device the vid is the PNU number and the cid is zero. A variable in an AK2 device is identified using both the cid and vid.

Use the read_parm_info command to get cid and vid values.

Variables may be read using tags instead of cid,vid identifiers. The read_parm_info command provides tag, cid, vid, information.

Command

```
<cmd action="read_val" num_only="int" valid_only="int" units="string">
  <val nodetype="int" node="int" cid="int" vid="int" tag="string" field="string"/>
  //return value of the variable
  <val nodetype="int" node="int" cid="int" vid="int" tag="string" field="string"/>
  <val nodetype="int" node="int" cid="int" vid="int" tag="string" field="string"/>
  ...
  ...
  <val nodetype=" int" node=" int" cid="int" vid=" int" tag="string" field="string"/>
  <val nodetype=" int" node=" int" cid="int" vid=" int" tag="string" field="string"/>
  <val nodetype=" int" node=" int" cid="int" vid=" int" tag="string" field="string"/>
</cmd>
```

Command Attribute	Data Type	Attribute Value Definition
num_only	int	If this attribute is set equal to "1" then a numeric code will be returned for enumeration values. If this attribute is not included or if it is set equal to "0" then the string value will be returned for an enumeration.
valid_only	int	If this attribute is set equal to "1" then: 1. A temperature value that is outside of the valid range appears as the string NaN instead of the most recent good value read. 2. A value that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.
Units	string	"u" or "U" U.S. units "s" or "S" SI units

Command Element	Data Type	Definition
val	complex	Element whose attributes identify the value that is to be read

The cmd element contains multiple val elements, each of which requests the value of a variable or of a field such as max or min that is associated with a variable.

Attribute of cmd's val element	Data Type	Definition
nodetype	int	Node type. Always set to "16" (generic device).

node	<i>int</i>	Node number (unique within nodetype) of the device that contains the variable .
cid	<i>int</i>	Component ID of the component (unique within a device) that contains the variable. the cid is always zero for variables within an EKC device. The “cid” attribute should not be included if the “tag” attribute is included.
vid	<i>int</i>	Variable ID of the variable This is unique within component if cid is non-zero, else unique within device. The vid is the PNU number if the device is an EKC. The “vid” attribute should not be included if the “tag” attribute is included.
tag	<i>string</i>	Tag that identifies a variable. This is the xml tag value within edf file (<i>see</i> <code><XMLNAME_SECTION_START></code>). The read_parm_info command provides tag, cid, vid, information. The “tag” attribute should not be included if the “cid” and “vid” attributes are included
field	<i>string</i>	Identifies data associated with the variable. If “field” is not specified, the value of the variable itself is returned. The “field” attribute may be any of the following values: “default” Factory default. “min” Minimum “max” Maximum “all” All of the above attributes are to be returned.

Response

The resp element contains multiple val elements, each of which contains a requested value of a variable or of a field such as max or min that is associated with a variable..

```

<resp action="read_val" ... echo of command attributes ...error="int">
  <val node=" int " vid=" int " cid=" int " tag="string" field="string" nodetype=" int "
    display="string" name="string" stat="string" statcode="int" pending="true/false">
    <value>signed decimal|int|string<value>
    <min> signed decimal|int|string <min>
    <max> signed decimal|int|string <max>
    <def> signed decimal|int|string <def>
  </val>
  ....
  ....
  other val elements
  ....
  ....
</resp>

```

Note that in the above response description all possible attributes and elements appear even though an actual response may not include all of them.

Response Element	Data Type	Definition
val		Element whose attributes identify the value that has been read

Attribute of resp's val element	Data Type	Definition
nodetype	<i>int</i>	Node type of the device that contains the variable
node	<i>int</i>	Node number (unique within nodetype) of the device that contains the variable .
cid	<i>int</i>	Component ID of the component (unique within a device) that contains the variable. The cid is always zero for variables within an EKC device.
vid	<i>int</i>	Variable ID of the variable This is unique within component if cid is non-zero, else unique within device. The vid is the PNU number if the device is an EKC.
field	<i>string</i>	Identifies data associated with the variable. If “field” does not appear, the value of the variable itself has been returned. The “field” attribute may have any of the following values: “default” Factory default value of variable “min” Minimum value of variable “max” Maximum value of variable “all” All of the above attributes are to be returned.
display	<i>string</i>	XML display name of the variable from the XMLNAME_SECTION section of edf file if an XML tag exists for this variable, else it's the same as “name” described below.
name	<i>string</i>	Parameter name of variable. See PARAMETER_SECTION of edf file.
pending	<i>true</i>	If the pending attribute appears, its value is always true. It means that the variable's database value has been returned but it is not a regularly polled variable and the database value has not been refreshed by reading the device for at least one minute. This attribute does not appear if the variable is regularly polled by the SYSTEM MANAGER or if its value has been read from the device and written to the database within the past one minute or if a “field” such as max, min, or default is being requested rather than the value itself. When the SYSTEM MANAGER returns pending = “true” it also initiates a read of the device that will result in the database value being refreshed. The client may repeatedly issue read variable commands until no val elements have a pending attribute.
stat	<i>string</i>	Online status. May have values: Offline, Online, Startup
statcode	<i>int</i>	Online status numeric code. 1 = Offline 2 = Online 3 = Startup
error	<i>int</i>	If an error occurs within returning the <val> tag, this attribute is shown, and no other results will be received.

The following table describes elements that appear within the val element when the command's "field" attribute = "all".

Element (within resp's val)	Data Type	Definition
value	<i>signed decimal int string</i>	Value of the variable
min	<i>signed decimal int string</i>	Minimum value of the variable .
max	<i>signed decimal int string</i>	Maximum value of the variable.
def	<i>signed decimal int string</i>	Default value of the variable.

Example 1

```
<cmd action="read_val"><val nodetype="16" node="5" cid="43" vid="21" /> </cmd>
<resp action="read_val" error="0">
  <val node="5" vid="21" cid="43" nodetype="16" display="Cutout Temp" name="Cutout
  Temp" stat="Online" statcode="2">-80.0 °C </val>
</resp>
```

Example 2

```
<cmd action="read_val">
  <val nodetype="16" node="5" cid="43" vid="21" />
  <val nodetype="16" node="5" cid="43" vid="21" field="min" />
  <val nodetype="16" node="5" cid="43" vid="21" field="max" />
</cmd>

<resp action="read_val" error="0">
  <val node="5" vid="21" cid="43" nodetype="16" display="Cutout Temp"
    name="CutoutTemp" stat="Online" statcode="2">-80.0°C </val>
  <val node="5" vid="21" cid="43" field="min" nodetype="16"
    display="Cutout Temp" name="Cutout Temp" stat="Online"
    statcode="2">-80.0°C </val>
  <val node="5" vid="21" cid="43" field="max" nodetype="16"
    display="Cutout Temp" name="Cutout Temp" stat="Online" statcode="2">50.0°C
</val>
</resp>
```

Example 3

```
<cmd action="read_val"><val nodetype="16" node="5" cid="21" vid="65"/></cmd>
<resp action="read_val" error="0">
  <val cid="21" nodetype="16" vid="65" node="5" display="Def term sensor A"
    name="Def term sensor A" stat="Online" statcode="2">-100.0°C </val>
</resp>
```

Example 4

```
<cmd action="read_val"><val nodetype="16" node="5" cid="21" vid="65"
field="all"/></cmd>
<resp action="read_val" error="0">
  <val cid="21" field="all" nodetype="16" vid="65" node="5"
    display="Def term sensor A" name="Def term sensor A" stat="Online" statcode="2">
    <value>-100.0°C </value>
    <min>-100.0°C </min>
    <max>200.0°C </max>
    <def>-100.0°C </def>
  </val>
</resp>
```

1.4 read_units

Read list of System Manager unit numbers and IP addresses.

Command

```
<cmd action="read_units"/>
```

Response

```
<resp action="read_units" error="int">
  <total>int</total>
  <store_name>string</store_name>
  <unit_name>string</unit_name>
  <StoreId1>string</StoreId1>
  <StoreId2>string</StoreId2>
  <memsize>int</memsize>
  <dateformat>int</dateformat>
  <software> string </software>
  <macaddr>string</macaddr>
  <sitevizfile> string </sitevizfile>
  <unit>
    <addr>int</addr>
    <ip>ipaddr</ip>
    <port>int</port>
  </unit>
  <unit_internet >
    <addr> int </addr>
    <ip> ipaddr </ip>
    <port>int</port>
  </unit_internet >
  <unit>
    .....
  </unit>
  < unit_internet >
    .....
  </unit_internet >

</resp>
```

Response Element	Data Type	Definition
total	int	Number of System Manager units
store_name	string	Configured name of store. Max size 16 characters.
unit_name	string	Configured name of unit.
storeId1	string	Store identification code. Max eight characters.
storeId2	string	Store identification code. Max eight characters.
memsize	int	32 for 32 meg or 64 for 64 Meg
dateformat	int	0 = MMDDYY 1 = MMDDYYYY

		2 = DDMMYY 3 = DDMMYYYY
software	<i>string</i>	System Manager software version, e.g. E02.080
macaddr	<i>string</i>	MAC address in format XX-XX-XX-XX-XX-XX In a multi-unit system, this is the MAC address of the unit that is executing this XML command.
sitevizfile	<i>string</i>	No = no site jpeg file Yes = site jpeg file present This element has meaning only for the AKCS.
unit		Element that contains addr, ip and port elements.
addr (within unit)	<i>int</i>	System Manager address. Range is 0 through 9
ip (within unit)	<i>ipaddr</i>	unit's ip address.
port (within unit)	<i>int</i>	unit's port.
unit_internet		The internet address that can be accessed via the public internet. Every unit can have one public internet address. Contains addr, ip and port elements.

Example

```
<cmd action="read_units"/>
```

```
<resp action="read_units" error="0">
  <total>1</total>
  <store_name>BIG FOOD STORE</store_name>
  <unit_name>AK255 Unit 0</unit_name>
  <storeId1>AREA0010</storeId1>
  <storeId2>STOR0125</storeId2>
  <memsize>32</memsize>
  <dateformat>0</dateformat>
  <software>S02.090</software>
  <macAddr>00-0B-2D-00-29-32</macAddr>
  <unit>
    <addr>0</addr>
    <ip>10.35.36.8</ip>
    <port>80</port>
  </unit>
  <unit_internet >
    <addr>0</addr>
    <ip>10.35.36.8</ip>
    <port>80</port>
  </unit_internet >
</resp>
```

1.5 write_unit

Set the store name, store id spots, or the unit name. Can change all or one of the elements.

Command

```
<cmd action="write_unit" auth="string" acct="string" user="string" password="string">
  <store_name>string</store_name>
  <storeId1>string</storeId1>
  <storeId2>string</storeId2>
  <unit_name>string</unit_name>
</cmd>
```

Cmd Element	Data Type	Definition
store_name	string	Configured name of store. Max size 16 characters.
storeId1	string	Store identification code. Max eight characters.
storeId2	string	Store identification code. Max eight characters.
unit_name	string	Name of Unit.
error	int	Error codes described at end of document.

Example

```
<cmd action="write_unit" auth="12345" acct="50" user="Supervisor" password="12345"
  error="0">
  <store_name>My New Store</store_name>
  <storeId1>Region1</storeId1>
  <storeId2>LOC005</storeId2>
  <unit_name>Controller 2</unit_name>
</cmd>
```

OR

```
<cmd action="write_unit" auth="12345" acct="50" user="Supervisor" password="12345"
  error="0">
  <store_name>My New Store</store_name>
  <unit_name>Controller 2</unit_name>
</cmd>
```

OR

```
<cmd action="write_unit" auth="12345" acct="50" user="Supervisor" password="12345"
  error="0">
  <storeId1>Region1</storeId1>
  <storeId2>LOC005</storeId2>
</cmd>
```

1.6 read_devices

Read list of devices.

Command

```
<cmd action="read_devices"/>
```

Response

```
<resp action="read_devices" error="int">
  <unit_name>string</unit_name>
  <software>string</software>
  <store_name>string</store_name>
  <device alarm="int" ctrl_val="string" indent="int" mod="int"
    defrost="int" modelname="string" multicasename="string"
    state="int" node="int" nodetype="int" online="int"
    point="int" status="string" value="string" rack_id="int"
    suction_id="int" condenser="int">
    <name>string</name>
    <device_id>string</device_id>
    <type>string</type>
    <rack_id>int</rack_id>
    <num_suction>int</num_suction>
  </device>
  <total>int</total>
</resp>
```

root elements

Response Element	Data Type	Definition	SC 255 Support	SC355 Support	SM800 Support
unit_name	string	Name of the SC/SM front end	2.101	3.051	8.011
store_name	string	Name of the site	2.231	N/A	N/A
software	string	Version of SC/SM firmware installed	2.101	3.051	8.011
device			2.101	3.051	8.011
total	int	Total number of devices configured on SC/SM front end	2.101	3.051	8.011

device child elements

Response Element	Data Type	Definition	SC 255 Support	SC355 Support	SM800 Support
name	<i>string</i>	Name of device	2.101	3.051	8.011
device_id	<i>string</i>	Device id, which is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124 012x	2.101	3.051	8.011
type (within device element)	<i>string</i>	PACK_ONLY RACK_ONLY PACK RACK DRIVE EVAP EVAPIO CONDENSER COMPRESSOR	2.101	3.051	8.011
num_suction	<i>int</i>	Number of suction groups configured for the pack/rack group (PACK_ONLY or RACK_ONLY devices)	N/A	N/A	8.045
rack_id	<i>int</i>	ID to link suction group back to the pack/rack group device	2.231	N/A	8.031

Device element attributes

Response attribute	Data Type	Definition	SC 255 Support	SC355 Support	SM800 Support
addr	<i>string</i>	Text description of device attribute <i>node</i> if nodetype < 16 then description is in format of <i>node-mod.point</i>	2.101	3.051	8.011 Removed 8.045
alarm	<i>int</i>	Indicates if device is currently in an alarm state 0 = no active alarm 1 = active alarm	2.151	N/A	8.045
condenser	<i>string</i>	Indicates if condenser is associated with the rack device 0=No 1=Yes	2.151	N/A	8.047
ctrl_val	<i>String</i>	sepoint device is controlling to	N/A	N/A	8.045
defrost	<i>Int</i>	Indicates id the device is currently in a defrost condition 0=Not in Defrost 1=In Defrost	2.151	N/A	8.045
indent	<i>Int</i>	Danfoss use only	2.151	N/A	8.045
mod	<i>Int</i>	Module address. Will = 0 when nodetype=16	2.101	3.051	8.011
modelname	<i>String</i>	Returns device model and software version for devices with nodetype =16	2.151	N/A	8.045
multicase_name	<i>String</i>	User defined name of device	N/A	N/A	8.045
node	<i>int</i>	Address of the device on field bus	2.101	3.051	8.011
nodetype	<i>int</i>	Nodetype of device. See “nodetype” in Reference Information section	2.101	3.051	8.011
online	<i>int</i>	Communication status of device on	2.151	N/A	8.045

		field bus 0=Offline 1=Online			
point	<i>int</i>	point address. Will =0 when nodetype=16	2.101	3.051	8.011
rack_id	<i>int</i>	Index to link devices to the SC/SM grouping PACK_ONLY RACK_ONLY	N/A	N/A	8.031
state	<i>int</i>	For nodetype = 16 0 = Case not in refrigeration nor defrost 1 = running or in defrost	N/A	N/A	8.045
status	<i>string</i>	Current status message of the device	N/A	N/A	8.045
suction_id	<i>int</i>	Index to link devices to the SC/SM grouping PACK RACK	N/A	N/A	8.045
value	<i>string</i>	current reading of device	2.151	N/A	8.045

Example

```
<resp action="read_devices" error="0">
  <unit_name>SM800</unit_name>
  <software>G08.047</software>
  <device indent="0" nodetype="255" rack_id="1">
    <name>CENTRAL LT</name>
    <type>PACK_ONLY</type>
    <num_suction>1</num_suction>
  </device>
  <device alarm="0" ctrl_val="-10.5 °C" indent="0" mod="0" modelname="EKC531D1(1)-013x"
node="80" nodetype="16" online="1" point="0" status="s8-Normal" value="-7.6 °C">
    <name>CENTRAL LT</name>
    <device_id>084B8007_013x</device_id>
    <type>PACK</type>
    <rack_id>1</rack_id>
  </device>
  <device alarm="0" ctrl_val="2.0 °C" defrost="0" indent="4" mod="0" modelname="AK-CC550-
B-015B" multicasename="ASL LACTEOS 1" node="1" nodetype="16" online="1" point="0" rack_id="1"
state="0" status="(s23) Normal" suction_id="1" value="5.3 °C">
    <name>Dairy 1</name>
```

```
<device_id>084B8020_015B</device_id>
<type>EVAP</type>

<device indent="0" nodetype="255" rack_id="2">
  <name> Rack C  </name>
  <type>RACK_ONLY</type>
  <num_suction>1</num_suction>
</device>

<device alarm="0" ctrl_val="86.5 °F" indent="2" mod="1" node="30" nodetype="2"
nvalue="0.0" online="2" point="18" rack_id="2" status="Start Up" value="0.0 °F">
  <name>Condenser C</name>
  <type>COND</type>
</device>

<device alarm="0" ctrl_val="0.0 psi" indent="2" mod="1" modelname="" node="30"
nodetype="2" online="1" point="17" rack_id="2" status="System Satisfied" suction_id="2"
value="46.5 psi">
  <name>+22 Suct CA</name>
  <type>RACK</type>
  <rack_id>2</rack_id>
  <suction_id>2</suction_id>
</device>
<device condenser="1" indent="2" mod="2" modelname="" node="1" nodetype="0" online="1"
point="9" rack_id="1" suction_id="1" value="On">
  <name>Compressor A 1</name>
  <type>COMPRESSOR</type>

</device>
```

1.7 read_device

Read detailed information about a specific device.

Command

```
<cmd action="read_device" nodetype="int" node="int" mod="int" point="int" sect="int"/>
```

Command Attribute	Data Type	Definition
nodetype	int	Nodetype of device. Can be 2 (monitor point) or 0 (digital input associated with a monitor point) or 16 (generic controller)
node	int	Node address of device within Nodetype
mod	int	Module number. Always 0 for generic devices.
point	int	Point number. Range depends upon nodetype; Nodetype Range 0 (digital input) 1 through 8 1 (digital output) 9 through 16 (offset of 8) 2 (analog input) 17 through 24 (offset of 16) 3 (analog output) 25 through 28 (offset of 24) Always 0 for generic devices. The System Manager displays point values on its screen in the range 1 through 8. The XML “point” parameter is the point number as would be displayed + an offset as noted above
sect	int	Section number. Applies to mult-section case controller.

Response

```
<resp action="read_device" ...echo attributes from command ...error="int">
<name>string</name>
<type> string </type>
<filename>string</filename>
<model>string</model>
<device_id>string</device_id>
<val> temp|pres </val>
<stat>string</stat>
<arg1>int</arg1>
<arg2>int</arg2>
<arg3>int</arg3>
<arg4>int</arg4>
<arg5>int</arg5>
<alarm> int </alarm>
<mainswitch> int </mainswitch>
<lights> int </lights>
<night> int </night>
<defrost> int </defrost>
<cleaning> int </cleaning>
<shutdown> int </shutdown>
```

```

<host>int</host>
<nodetype> int </nodetype>
<multi> int </multi>
<addr> int </addr>
<node> int </node>
<mod> int </mod>
<point> int </point>
</resp>

```

Response Element	Data Type	Definition
name	<i>string</i>	Name assigned to device
type	<i>string</i>	EVAP for evaporator, PACK for pack controller Other values: RACK_ONLY Rack information RACK - Label for Suction Group Monitoring MoniClean MoniDefrost MoniDigi OI RO SI VO
rack_id	<i>int</i>	1 based index of rack. Will only be returned if (RACK_ONLY) comes back from type. Contains num_suction attribute.
num_suction	<i>int</i>	Number of Suction Groups. Will only be returned if (RACK_ONLY) comes back from type.
filename	<i>string</i>	Name of .edf file associated with this device
model	<i>string</i>	Device model e.g. AK2CC-303US-012x For node type 2 (monitor point) this is the name of an analog sensor (e.g. PT1000). For type 0 (OI) it is type of input (e.g. Voltage)
device_id	<i>string</i>	Device id, which is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x
val	<i>temp/pres</i>	A value associated with this device. If the type is EVAP then this is the case temperature. If type is EVAP and multi = 1 for multi-section controller then it is one of several case temperatures. If the type is a PACK controller then this value is suction pressure. If type is OI then val is Off or On.
stat	<i>string</i>	Device status. For both EVAP and PACK types the status may be:

		Offline, Startup, Error, or a string from an enumerated list that is specific to the model. The possible strings for a particular device model may be found in the read_dyn_list_info output or in the associated edf file in the DYN_LIST entry “Case Status” if the device is EVAP or “Status” if it is a PACK
arg1	<i>string</i>	Future use (ignore)
arg2	<i>string</i>	Future use (ignore)
arg3	<i>string</i>	Future use (ignore)
arg4	<i>string</i>	Future use (ignore)
arg5	<i>string</i>	Future use (ignore)
alarm	<i>int</i>	Indicates if the device has any alarms 0 = no alarms, non-zero = at least one alarm condition
mainswitch	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
lights	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
night	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
defrost	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
cleaning	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
shutdown	<i>int</i>	0 = variable not available for this device, 1 = variable is available for this device
host	<i>int</i>	System Manager unit number
nodetype	<i>int</i>	nodetype of this device
multi	<i>int</i>	0 = not multi evap device, 1 = multi evap device
addr	<i>int</i>	Device’s node address within nodetype. Formatted for points that the System Manager addresses as devices. Same as the “node” attribute below for generic devices.
node	<i>int</i>	Device’s node address within nodetype. Same as addr above for generic devices.
mod	<i>int</i>	module number (0 for generic devices)
point	<i>int</i>	point number (0 for generic devices)

Example

```

<cmd action="read_device" nodetype="16" node="6" mod="0" point="0" sect="1"/>
<resp point="0" nodetype="16" node="6" mod="0" action="read_device" sect="1"
  error="0">
  <name>6 AK2-CC303A</name>
  <type>EVAP</type>
  <filename>080Z0124.edf</filename>
  <model>AK2CC-303US-012x</model>
  <device_id>080Z0124_012x</device_id>
  <val>25.5°C</val>

```

```
<stat>Injection</stat>
<arg1>4280</arg1>
<arg2>1</arg2>
<arg3>1</arg3>
<arg4>3</arg4>
<arg5>1</arg5>
<alarm>0</alarm>
<mainswitch>1</mainswitch>
<lights>1</lights>
<night>1</night>
<defrost>1</defrost>
<cleaning>0</cleaning>
<shutdown>1</shutdown>
<host>0</host>
<nodetype>16</nodetype>
<multi>1</multi>
<addr>6</addr>
<node>6</node>
<mod>0</mod>
<point>0</point>
</resp>
```

1.8 read_meters

Read list of power monitoring meters.

Command

```
<cmd action="read_meters"/>
```

Response

```
<resp action="read_meters" read_meters ="int" error="int">
  <meter id="int" >
    <name>string</name>
    <nodetype>int</nodetype>
    <node>int</node>
    <kwh>signed decimal</kwh>
    <kw>signed decimal</kw>
    <pk>signed decimal</pk>
    <acc_kwh_hourly>signed decimal</acc_kwh_hourly>
    <acc_kwh_daily>signed decimal</acc_kwh_daily>
    <pkdt>date<pkdt>
    <pk_epoch>int</pk_epoch>
    <pktm>time</pktm>
    <pk_reset_dt>date</pk_reset_dt>
    <pk_reset_tm>time</pk_reset_tm>
    <pk_reset_epoch>int</pk_reset_epoch>
    <kwh_reset_dt>date</kwh_reset_dt>
    <kwh_reset_tm>time</kwh_reset_tm>
    <kwh_reset_epoch>int</kwh_reset_epoch>
  </meter>
  <meter id="int" >                                     // If the meter type is Pulse/Vol
    <name>string</name>
    <nodetype>int</nodetype>
    <node>int</node>
    <mod>int</mod>
    <point>int</point>
    <current_hourly>int</current_hourly>
    <last_hourly>int</last_hourly>
    <current_daily >int</current_daily >
    <last_daily >int</last_daily >
    <current_weekly>int</current_weekly >
    <last_weekly >int</last_weekly>
    <current_monthly>int</current_monthly>
    <last_monthly>int</last_monthly>
    <current_yearly>int</current_yearly>
    <last_yearly>int</last_yearly>
    <total> int</total>
    <reset_epoch> int</reset_epoch>
    <units >string</units >
  </meter>
.
.
.
```

```

<meter id="int" >
  <name>string</name>
  <nodetype>int</nodetype>
  <node>int</node>
  <kwh>signed decimal</kwh>
  <kw>signed decimal</kw>
  <pk>signed decimal</pk>
  <acc_kwh_hourly>signed decimal</acc_kwh_hourly>
  <acc_kwh_daily>signed decimal</acc_kwh_daily>
  <pkdt>date</pkdt>
  <pk_epoch>int</pk_epoch>
  <pktm>time</pktm>
  <pk_reset_dt>date</pk_reset_dt>
  <pk_reset_tm>time</pk_reset_tm>
  <pk_reset_epoch>int</pk_reset_epoch>
  <kwh_reset_dt>date</kwh_reset_dt>
  <kwh_reset_tm>time</kwh_reset_tm>
  <kwh_reset_epoch>int</kwh_reset_epoch>
</meter>
</resp>

```

Response Attribute	Data Type	Definition
read_meters	<i>int</i>	The number of meters

Response Element	Data Type	Definition
meter		Element that contains other elements that describe a meter
name	<i>string</i>	Name of meter
id	<i>int</i>	Id of meter (1 based) 1 -> number of emons. Returned as an attribute in the meter element.
nodetype	<i>int</i>	Node type
node	<i>int</i>	Device node number
mod	<i>int</i>	Module number (for Pulse/Vol meter type only)
point	<i>int</i>	Point number (for Pulse/Vol meter type only)
current_hourly	<i>int</i>	Current hourly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_hourly	<i>int</i>	Last hourly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_daily	<i>int</i>	Current daily value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_daily	<i>int</i>	Last daily value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_weekly	<i>int</i>	Current weekly value of Pulse/Vol meter type (for Pulse/Vol meter type only)

last_weekly	<i>int</i>	Last weekly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_monthly	<i>int</i>	Current monthly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_monthly	<i>int</i>	Last monthly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_yearly	<i>int</i>	Current yearly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_yearly	<i>int</i>	Last yearly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
total	<i>int</i>	Total current value of the Pulse/Vol meter type (for Pulse/Vol meter type only)
reset_epoch	<i>int</i>	Epoch time of most recent Pulse/Vol meter. Epoch time is seconds since 12:00 am 1970 System Manager local time. (for Pulse/Vol meter type only)
units	<i>string</i>	The units used for the Pulse/Vol meter. (for Pulse/Vol meter type only)
kwh	<i>signed decimal</i>	Kilowatt hours accumulated since most recent kwh reset.
kw	<i>signed decimal</i>	Current kilowatt reading
pk	<i>signed decimal</i>	Peak kw since most recent peak kw reset
acc_kwh_hourly	<i>signed decimal</i>	Accumulated Hourly
acc_kwh_daily	<i>signed decimal</i>	Accumulated Daily
pkdt	<i>date</i>	Date of peak kw.
pktm	<i>time</i>	Time of peak kw
pk_epoch	<i>int</i>	Epoch time of peak kw. Epoch time is seconds since 12:00 am 1970 System Manager local time
pk_reset_dt	<i>date</i>	Date of most recent peak kw reset
pk_reset_tm	<i>time</i>	Time of most recent peak kw reset
pk_reset_epoch	<i>int</i>	Epoch time of most recent peak kw reset. . Epoch time is seconds since 12:00 am 1970 System Manager local time.
kwh_reset_dt	<i>date</i>	Date of most recent kwh reset
kwh_reset_tm	<i>time</i>	Time of most recent kwh reset
kwh_reset_epoch	<i>int</i>	Epoch time of most recent kwh reset. Epoch time is seconds since 12:00 am 1970 System Manager local time.

1.9 read_meter

Read one or many meter values.

This function will return the values associated with the specific meter ID returned from read meters. There is also an attribute that will tell the function to only return the current kw from each meter requested. The id attribute is the order in which the energy meters have been setup within the AK-255.

Command

```
<cmd action="read_meter">
  <meter id="int" type="int"/>
  ...
  ...
</cmd>
```

Response

```
<resp action="read_meter">
  <meter id="int" type="int" error="int">
    <name>string</name>
    <nodetype>int</nodetype>
    <node>int</node>
    <kwh>signed decimal</kwh>
    <kw>signed decimal</kw>
    <pk>signed decimal</pk>
    <acc_kwh_hourly>signed decimal</acc_kwh_hourly>
    <acc_kwh_daily>signed decimal</acc_kwh_daily>
    <pkdt>date</pkdt>
    <pk_epoch>int</pk_epoch>
    <pktm>time</pktm>
    <pk_reset_dt>date</pk_reset_dt>
    <pk_reset_tm>time</pk_reset_tm>
    <pk_reset_epoch>int</pk_reset_epoch>
    <kwh_reset_dt>date</kwh_reset_dt>
    <kwh_reset_tm>time</kwh_reset_tm>
    <kwh_reset_epoch>int</kwh_reset_epoch>
  </meter>
  <meter id="int" type="int" error="int">    // If the meter type is Pulse/Vol
    <name>string</name>
    <nodetype>int</nodetype>
    <node>int</node>
    <mod>int</mod>
    <point>int</point>
    <current_hourly>int</current_hourly>
    <last_hourly>int</last_hourly>
    <current_daily>int</current_daily>
    <last_daily>int</last_daily>
    <current_weekly>int</current_weekly>
    <last_weekly>int</last_weekly>
    <current_monthly>int</current_monthly>
    <last_monthly>int</last_monthly>
```

```

<current_yearly>int</current_yearly>
<last_yearly>int</last_yearly>
<total> int</total>
<reset_epoch> int</reset_epoch>
<units >string</units >
</meter>
.
.
.
<meter id="int" type="int" error="int">
  <name>string</name>
  <nodetype>int</nodetype>
  <node>int</node>
  <kw>signed decimal</kw>
</meter>
</resp>

```

Response Attribute	Data Type	Definition
read meter		Complex element
id	<i>int</i>	Id of meter (1 based) 1 -> number of emons
type	<i>int</i>	If specified then only return the current kw reading (kw). If zero (0) or not specified, then all parameters will be returned for the (id) requested.

Response Element	Data Type	Definition
meter		Element that contains other elements that describe a meter
name	<i>string</i>	Name of meter
nodetype	<i>int</i>	Node type
node	<i>int</i>	node
mod	<i>int</i>	Module number (for Pulse/Vol meter type only)
point	<i>int</i>	Point number (for Pulse/Vol meter type only)
current_hourly	<i>int</i>	Current hourly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_hourly	<i>int</i>	Last hourly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_daily	<i>int</i>	Current daily value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_daily	<i>int</i>	Last daily value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_weekly	<i>int</i>	Current weekly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_weekly	<i>int</i>	Last weekly value of Pulse/Vol meter type (for Pulse/Vol meter type only)

current_monthly	<i>int</i>	Current monthly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_monthly	<i>int</i>	Last monthly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
current_yearly	<i>int</i>	Current yearly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
last_yearly	<i>int</i>	Last yearly value of Pulse/Vol meter type (for Pulse/Vol meter type only)
total	<i>int</i>	Total current value of the Pulse/Vol meter type (for Pulse/Vol meter type only)
reset_epoch	<i>int</i>	Epoch time of most recent Pulse/Vol meter. Epoch time is seconds since 12:00 am 1970 System Manager local time. (for Pulse/Vol meter type only)
units	<i>string</i>	The units used for the Pulse/Vol meter. (for Pulse/Vol meter type only)
kwh	<i>signed decimal</i>	Kilowatt hours accumulated since most recent kwh reset.
kw	<i>signed decimal</i>	Current kilowatt reading
pk	<i>signed decimal</i>	Peak kw since most recent peak kw reset
acc_kwh_hourly	<i>signed decimal</i>	Accumulated Hourly
acc_kwh_daily	<i>signed decimal</i>	Accumulated Daily
pkdt	<i>date</i>	Date of peak kw.
pktm	<i>time</i>	Time of peak kw
pk_epoch	<i>int</i>	Epoch time of peak kw. Epoch time is seconds since 12:00 am 1970 System Manager local time
pk_reset_dt	<i>date</i>	Date of most recent peak kw reset
pk_reset_tm	<i>time</i>	Time of most recent peak kw reset
pk_reset_epoch	<i>int</i>	Epoch time of most recent peak kw reset. . Epoch time is seconds since 12:00 am 1970 System Manager local time.
kwh_reset_dt	<i>date</i>	Date of most recent kwh reset
kwh_reset_tm	<i>time</i>	Time of most recent kwh reset
kwh_reset_epoch	<i>int</i>	Epoch time of most recent kwh reset. Epoch time is seconds since 12:00 am 1970 System Manager local time.

1.10 read_date_time

Command

```
<cmd action="read_date_time"/>
```

Response

```
<resp action="read_date_time">
  <year>int</year>
  <month>int</month>
  <day>int</day>
  <hour>int</hour>
  <minute>int</minute>
  <second>int</second>
  <epoch>uint</epoch>
  <timezone>int</timezone>
  <daylightsavings>int</daylightsavings>
</resp>
```

Response Element	Data Type	Definition
year	int	Two digit year
month	int	Month 01 through 12
day	int	Day of month 01 through 31
hour	int	00 through 23
minute	int	00 through 59
second	int	00 through 59
epoch	uint	Date and time expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.
timezone	int	UTC time zone offset expressed as 100*hours. e.g. Eastern US time has an offset of -500
daylightsavings	int	0 = daylight savings not in effect 1 = daylight savings is in effect

1.11 read_parm_versions

Command

```
cmd action="read_parm_versions"/>
```

Response

```
<resp action="read_parm_versions">
  <count>int</count>
  <parm_file>
    <device_id>string</device_id>
    <rev>string</rev>
  </parm_file>
  <parm_file>
    <device_id>string</device_id>
    <rev>string</rev>
  </parm_file>
  ---
  ---
  ---
  <parm_file>
    <device_id>string</device_id>
    <rev>string</rev>
  </parm_file>
</resp>
```

Response Elements	Data Type	Definition
count	int	Number of parm_file elements to follow
parm_file		Element that contains device_id and rev elements
device_id	string	Device id, which is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x
rev	string	Device's edf file revision. This is maintained in the file by PVCS. e.g. 1.46.

Example

```
<cmd action="read_parm_versions"/>

<resp action="read_parm_versions" error="0">
  <count>1</count>
  <parm_file>
    <device_id>080Z0124_012x</device_id>
    <rev>1.46</rev>
  </parm_file>
</resp>
```

1.12 read_parm_info

Command

```
<cmd action="read_parm_info" device_id="string" num_only="int" tag="1"/>
```

Command Attribute	Data Type	Definition
device_id	string	Id of the device whose parameter information is to be read. It is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x
tag	int	OPTIONAL: If supplied value always = 1. Used to only list parameters that have associated xml tags in the edf.
num_only	int	If this attribute is defined and = "1" then display numeric values for enumerations

Response

```
<resp action="read_parm_info" ...echo attributes from command ...error="int">
<parms count="int">
  <parm tag="string" dyn_list_name="string" unit="string" cid="int" vid="int" meas="true"
  setting="true" name="string" min="int" max="int" default="int" group="int" type="string"/>
  <parm tag="string" dyn_list_name="string" unit="string" cid="int" vid="int" meas="true"
  setting="true" name="string" min="int" max="int" default="int" group="int" type="string"/>
  ...
  <parm tag="string" dyn_list_name="string" unit="string" cid="int" vid="int" meas="true"
  setting="true" name="string" min="int" max="int" default="int" group="int" type="string"/>
</parms>
</resp>
```

Response Element or Attribute	Data Type	Definition
parms		Element that contains parm elements, one for each parameter described in the edf file.
count (attribute of parms)	int	Attribute of parms element. This is the number of parm elements contained in the parms element
parm		Element whose attributes describe a parameter.
tag	string	XML name of parameter. Not all parameters have a tag. These appear in the edf file between <XMLNAME_SECTION_START> and <XMLNAME_SECTION_END> "tag" is an attribute of the "parm" element.

tag_desc	<i>string</i>	“tag” description. This appears in the edf file between <XMLNAME_SECTION_START> and <XMLNAME_SECTION_END> “tag” is an attribute of the “parm” element
dyn_list_name	<i>string</i>	If this attribute is present then this parameter is a dynamic list. The value of the dyn_list_name attribute matches the name of a dynamic list. The read_dyn_list_info command reads the dynamic lists. The dyn_list_name and the unit attributes are mutually exclusive, if one is there, the other is not.
unit	<i>string</i>	Engineering units. e.g. degf or psi
cid	<i>int</i>	Component ID as defined in the cid column of the edf file’s PARAMETER_SECTION
vid	<i>int</i>	Variable IDs as defined in the vid column of the edf file’s PARAMETER_SECTION
name	<i>string</i>	The parameter’s name as it appears in the “name” column of the edf file’s PARAMETER_SECTION
min	<i>int or string</i>	Minimum value that this parameter may have. It is a string when the parameter is a dyn list.
max	<i>int or string</i>	Maximum value that this parameter may have. . It is a string when the parameter is a dyn list.
default	<i>int or string</i>	Parameter’s default value. It is a string when the parameter is a dyn list.
group	<i>int</i>	Number of group to which parameter is a member
exp	<i>int</i>	Exponent – Used internally and can be ignored.
rw	<i>string</i>	“R” = read only parameter, “W”= Read and Write parameter.
meas	<i>string</i>	The meas attribute appears (and always has value “true”) if the parameter is in the edf file's SUMMARY_MSR_LIST. This marks the parameters that are displayed in the 255's "measures" summary list.
setting	<i>string</i>	The setting attribute appears (and always has value “true”) if the parameter is in the edf file's SUMMARY_SET_LIST This marks the parameters that are displayed in the 255's "settings" summary list.
type	<i>string</i>	ak2 type

Example

```

<cmd action="read_parm_info" device_id="080Z0124_012x"/>
<resp device_id="080Z0124_012x" action="read_parm_info" error="0">
  <parms count="291">
    <parm dyn_list_name="Off/On" unit="" cid="78" vid="8" name="Main switch"
min="Off" max="On" default="Off" group="16" exp="1" rw="W"/>
    <parm unit="" cid="2" vid="8" name="AK2 error" min="Off" max="On" default="Off"
group="16" exp="1" rw="R"/>
  </parms>
</resp>

```

```
<parm unit="percent" cid="74" vid="9" name="Man Valve A" min=" 0.0" max="
100.0" default=" 0.0" group="8" exp="1" rw="W"/>
<parm unit="percent" cid="75" vid="9" name="Man Valve B" min=" 0.0" max="
100.0" default=" 0.0" group="9" exp="1" rw="W"/>
<parm unit="percent" cid="76" vid="9" name="Man Valve C" min=" 0.0" max="
100.0" default=" 0.0" group="10" exp="1" rw="W"/>
<parm unit="percent" cid="77" vid="9" name="Man Valve D" min=" 0.0" max="
100.0" default=" 0.0" group="11" exp="1" rw="W"/>
.
.
.
<parm dyn_list_name="Auto_off_on" unit="" cid="70" vid="8" name="Solenoid control
A" min="Auto" max="Man Off" default="Auto" group="8" exp="1" rw="W"/>
<parm dyn_list_name="Auto_off_on" unit="" cid="71" vid="8" name="Solenoid control
A" min="Auto" max="Man Off" default="Auto" group="9" exp="1" rw="W"/>
<parm dyn_list_name="Auto_off_on" unit="" cid="72" vid="8" name="Solenoid control
A" min="Auto" max="Man Off" default="Auto" group="10" exp="1" rw="W"/>
<parm dyn_list_name="Auto_off_on" unit="" cid="73" vid="8" name="Solenoid control
A" min="Auto" max="Man Off" default="Auto" group="11" exp="1" rw="W"/>
<parm dyn_list_name="Auto_off_on" unit="" cid="66" vid="8" name="Fan control"
min="Auto" max="Man Off" default="Auto" group="2" exp="1" rw="W"/>
</parms>
</resp>
```

1.13 read_dyn_list_info

Command

```
<cmd action="read_dyn_list_info" device_id="string"/>
```

Command Attribute	Data Type	Definition
device_id	string	Id of the device whose parameter information is to be read. It is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x

Response

```
<resp action="read_dyn_list_info" ...echo attributes from command ...error="int">
```

```
<dyn_lists count="int">
```

```
  <dyn_list name="string" count="int">
```

```
    <t>string</t>
```

```
    <v>int</ v >
```

```
  < t >string</ t >
```

```
  < v >int</ v >
```

```
  ---
```

```
  ---
```

```
  ---
```

```
  < t >string</ t >
```

```
  < v >int</ v >
```

```
</dyn_list>
```

```
<dyn_list name="string" count="int">
```

```
  <t>string</t>
```

```
  <v>int</ v >
```

```
  < t >string</ t >
```

```
  < v >int</ v >
```

```
  ---
```

```
  ---
```

```
  ---
```

```
  < t >string</ t >
```

```
  < v >int</ v >
```

```
</dyn_list>
```

```
---
```

```
---
```

```
---
```

```
<dyn_list name="string" count="int">
```

```
  <t>string</t>
```

```
  <v>int</ v >
```

```
  < t >string</ t >
```

```
  < v >int</ v >
```

```
  ---
```

```
  ---
```

```
  ---
```

```
  < t >string</ t >
```

```

    < v>int</ v>
  </dyn_list>
</dyn_lists>
</resp>

```

Response Element or Attribute	Data Type	Definition
dyn_lists		Element that contains the dynamic list definitions from the DYN LIST section of the edf file.
count (attribute of dyn_lists)	int	Number of dyn_list elements contained in the dyn_lists element
dyn_list		Element that describes a dynamic list
dyn_list_name (attribute of dyn_list)	int	Name of this dynamic list
count (attribute of dyn_list)	int	Number of text/value pairs in this dynamic list
t (element within dyn_list element)	string	Text for one dynamic list entry. This element is immediately followed by the “value” element that forms this text/value pair
v (element within dyn_list element)	int	Numeric value for one dynamic list entry. This element is immediately preceded by the “text” element that forms this text/value pair

Example

```

<cmd action="read_dyn_list_info" device_id="080Z0124_012x"/>

<resp action="read_dyn_list_info" device_id="080Z0124_012x" error="0">
  <dyn_lists count="30">
    <dyn_list name="Status Chars" count="16">
      <t>N</t>
      <v>0</v>
      <t>R</t>
      <v>1</v>
      .
      .
      .
      <t>N</t>
      <v>15</v>
    </dyn_list>
    <dyn_list name="Off/On" count="2">
      <t>Off</t>
      <v>0</v>
      <t>On</t>
      <v>1</v>
    </dyn_list>
  </dyn_lists>
</resp>

```

```
.  
. .  
<dyn_list name="Auto_off_on" count="3">  
  <t>Auto</t>  
  <v>0</v>  
  <t>Man On</t>  
  <v>1</v>  
  <t>Man Off</t>  
  <v>2</v>  
</dyn_list>  
</dyn_lists>  
</resp>
```

1.14 read_menu_groups

Returns the menu group descriptions that appear in the section of the edf file demarcated by <GROUPTEXT_SECTION_START> and <GROUPTEXT_SECTION_END>

<cmd action="read_menu_groups" device_id="string"/>

Command Attribute	Data Type	Definition
device_id	string	Id of the device whose menu groups information is to be read. It is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x

Response

```
<resp action="read_menu_groups" ...other attributes from command...error="int">
  <count>int</count>
  <group>
    <number>int</number>
    <text>string</text>
  </group>
  <group>
    <number>int</number>
    <text>string</text>
  </group>
  .....
  <group>
    <number>int</number>
    <text>string</text>
  </group>
</resp>
```

Response Element	Data Type	Definition
count	int	Number of menu group descriptors
group		Element that describes a group.
number	int	Element within group that contains the number of the group
text	string	Element within group that contains group's text

1.15 read_device_summary

Command

```
<cmd action="read_device_summary" node="int"/>
```

This command lists the points that appear on the System Manager's "measures" summary list given a device's node number.

Another way to get this same information is to use the read_parm_info command. The parameters that appear in the read_parm_info response and whose "meas" attribute is true also appear in the read_device_summary response.

Command Attribute	Data Type	Definition
node	int	Node number of the generic device whose device summary is to be read

Response

```
<resp node="int" action="read_device_summary" error="0">
```

```
<summary_count>int</summary_count>
```

```
<summary_list>
```

```
<cid>int</cid>
```

```
<vid>int</vid>
```

```
<name>string</name>
```

```
</summary_list>
```

```
-----
```

```
-----
```

```
summary_list elements
```

```
-----
```

```
-----
```

```
</resp>
```

Response Element	Data Type	Definition
summary_count		Number of summary_list elements.
summary_list	int	Element that contains cid, vid, and name of one parameter that appears in the System Manager's "measures" summary list
cid		Component Id of parameter. Contained in summary_list element.
vid	int	Variable Id of parameter. Contained in summary_list element.

Example

```
<cmd action="read_device_summary" node="113"/>
```

```
<resp node="113" action="read_device_summary" error="0">
```

```
<summary_count>8</summary_count>
<summary_list>
  <cid>0</cid>
  <vid>2007</vid>
  <name>--- EKC State</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>133</vid>
  <name>--- Po b</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>134</vid>
  <name>--- Pc b</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>516</vid>
  <name>--- Comp.Cap %</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>530</vid>
  <name>--- Cond.Cap %</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>120</vid>
  <name>r24 Comp. ref.b</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>2562</vid>
  <name>u44 Sc3 temp</name>
</summary_list>
<summary_list>
  <cid>0</cid>
  <vid>2563</vid>
  <name>u45 Sc4 temp</name>
</summary_list>
</resp>
```

1.16 schedule_summary

Lists schedules and their status.

Command

```
<cmd action = "schedule_summary"/>
```

Response

```
<resp action="schedule_summary" error="int">
  <sch_count> int </sch_count>
  <sch>
    <id> int </id>
    <desc>string</desc>
    <status>string</status>
  </sch>
  ....
  ....
  ....
  <sch>
    <id> int</id>
    <desc>string</desc>
    <status> string </status>
  </sch>
</resp>
```

Response Element	Data Type	Definition
sch_count	int	Number of schedules
sch		Schedule
id (within sch)	int	Identifies a schedule in this list. Given the id, the “schedule detail” command may be used to obtain more detailed information. Range 1 through 100.
desc (within sch)	string	Description of schedule
status	string	Schedule status. Possible values are: True False start time Disabled Not Sent start time has format “date time”, e.g. 09/18/07 06:01.

Example

```
<cmd action="schedule_summary"/>
```

```
<resp action="schedule_summary" error="0">
  <sch_count>2</sch_count>
  <sch>
    <id>1</id>
    <desc>MEAT</desc>
    <status>Not Sent</status>
  </sch>
  <sch>
```

```
<id>2</id>  
<desc>NIGHT</desc>  
<status>True</status>  
</sch>  
</resp>
```

1.17 schedule_detail

Reads details, given the ID of a schedule.

Command

```
<cmd action = "schedule_detail" id = "int"/>
```

Command Attribute	Data Type	Definition
id	int	Schedule ID from "schedule summary" command

Response

```
<resp action="schedule_detail" ...echo of command attributes... error="int">
  <id>int</id>
  <desc>string</desc>
  <usage> string</usage>
  <stagger>string</stagger>
  <enable> string </enable>
  <def_start_time>date time</def_start_time>
  <def_interval>int min</def_interval>
  <status> string|date time </status>
  <schedules>
    <sch_number>int</sch_number>
    <sch_detail>
      <type>string</type>
      <on_time>time</on_time>
      <off_time>time</off_time>
      <weekdays>string</weekdays>
      <holidays>string</holidays>
    </sch_detail>
    <sch_detail>
      <type> string </type>
      <on_time> time </on_time>
      <off_time> time </off_time>
      <weekdays>string</weekdays>
      <holidays>string</holidays>
    </sch_detail>
    .....
  </schedules>
</resp>
```

Response Element	Data Type	Definition
id	int	Schedule ID from schedule summary command
desc	string	Schedule description
usage	string	Misc Case Lighting Night Setback Shutdown Defrost Coord Defrost
stagger	string	yes no

enable	<i>string</i>	yes no
def_start_time	<i>date time</i>	Defrost start time - <i>date time</i> e.g. 09/18/07 16:01
def_interval	<i>int min</i>	Defrost interval - <i>int min</i> e.g. 30 min
status	<i>string date time</i>	Schedule status. Possible values are: True False <i>date time</i> Disabled Not Sent “ <i>date time</i> ” is the starting date and time, e.g. 09/18/07 06:01
schedules		Element that encloses a number of schedule details
sch_num (within schedules)	<i>int</i>	Number of schedule details
sch_detail(within schedules)		A schedule detail - includes an on time, off time and the weekdays and holidays during which it is active.
type (within sch_detail)	<i>string</i>	Schedule type - Standard Relative
on_time	<i>time</i>	Schedule On time - e.g. 12:00 PM
off_time	<i>time</i>	Schedule Off time - e.g. 2:00 PM
weekdays	<i>string</i>	Weekdays to which this schedule applies, e.g. SMTWRFA
holidays	<i>string</i>	Holidays to which this schedule applies, e.g. 12345678

Example 1

```
<cmd action="schedule_detail" id="2"/>
```

```
<resp id="2" action="schedule_detail" error="0">
```

```
<id>2</id>
```

```
<desc>NIGHT</desc>
```

```
<usage>Shutdown</usage>
```

```
<stagger>no</stagger>
```

```
<enable>yes</enable>
```

```
<def_start_time/>
```

```
<def_interval>30 min</def_interval>
```

```
<status>True</status>
```

```
<schedules>
```

```
<sch_number>2</sch_number>
```

```
<sch_detail>
```

```
<type>Standard</type>
```

```
<on_time>09:00 AM</on_time>
```

```
<off_time>09:00 PM</off_time>
```

```
<weekdays>SMTWRFA</weekdays>
```

```
<holidays>12345678</holidays>
```

```
</sch_detail>
```

```
<sch_detail>
```

```
<type>Standard</type>
```

```
<on_time>08:00 AM</on_time>
```

```
<off_time>10:00 PM</off_time>
```

```
<weekdays>SMTWRFA</weekdays>
```

```
<holidays>12345678</holidays>
```

```
</sch_detail>
```

```
</schedules>  
</resp>
```

Example 2

```
<cmd action="schedule_detail" id="3"/>  
<resp id="3" action="schedule_detail" error="0">  
  <id>3</id>  
  <desc>C</desc>  
  <usage>Case Lighting</usage>  
  <stagger>no</stagger>  
  <enable>yes</enable>  
  <def_start_time/>  
  <def_interval>30 min</def_interval>  
  <status>False</status>  
  <schedules>  
    <sch_number>2</sch_number>  
    <sch_detail>  
      <type>Standard</type>  
      <on_time>10:00 AM</on_time>  
      <off_time>04:00 PM</off_time>  
      <weekdays>W FA</weekdays>  
      <holidays>23 678</holidays>  
    </sch_detail>  
    <sch_detail>  
      <type>Standard</type>  
      <on_time>06:00 PM</on_time>  
      <off_time>08:00 PM</off_time>  
      <weekdays>W FA</weekdays>  
      <holidays>23 678</holidays>  
    </sch_detail>  
  </schedules>  
</resp>
```

1.18 read_store_schedule

Read the store operating hours.

Command

```
<cmd action = "read_store_schedule" time_format="int">
```

Command Attribute	Data Type	Definition
action	<i>string</i>	Command action which is always <i>read_store_schedule</i> .
time_format	<i>int</i>	format option which can be: 0 -> 12 hour format (hour = 10, minute = 54, units = "AM/PM") 1 -> 24 hour format 2 -> display format (ex. 10:00 AM, 02:00 PM) The default is 0, if this attribute is not used.

Response

```
<resp action = "read_store_schedule" time_format="2">
```

```
<currenttime time="date/time" timezone="int">
```

```
<daylightsavings>string</daylightsavings>
```

```
<spring>
```

```
<month>string</month>
```

```
<week>string</week>
```

```
<day>string</day>
```

```
<time>time</time>
```

```
</spring>
```

```
<fall>
```

```
<month>string</month>
```

```
<week>string</week>
```

```
<day>string</day>
```

```
<time>time</time>
```

```
</fall>
```

```
</currenttime>
```

```
<days count="int">
```

```
<day value="string" id="int">
```

```
<open>string</open>
```

```
<close>string</close>
```

```
</day>
```

```
...
```

```
...
```

```
</resp>
```

OR

```
<resp action = "read_store_schedule" time_format="1">
```

```
<currenttime time="date/time" timezone="int">
```

```
<daylightsavings>string</daylightsavings>
```

```
<spring>
```

```
<month>string</month>
```

```
<week>string</week>
```

```

    <day>string</day>
    <time>time</time>
  </spring>
  <fall>
    <month>string</month>
    <week>string</week>
    <day>string</day>
    <time>time</time>
  </fall>
</currenttime>
<days count="int">
<day value="string" id="int">
  <open>
    <hour>int</hour>
    <minute>int</minute>
  </open>
  <close>
    <hour>int</hour>
    <minute>int</minute>
  </close>
</day>
...
...
</resp>

```

OR

```

<resp action = "read_store_schedule" time_format="0">
  <currenttime time="date/time" timezone="int">
    <daylightsavings>string</daylightsavings>
    <spring>
      <month>string</month>
      <week>string</week>
      <day>string</day>
      <time>time</time>
    </spring>
    <fall>
      <month>string</month>
      <week>string</week>
      <day>string</day>
      <time>time</time>
    </fall>
  </currenttime>
  <days count="int">
  <day value="string" id="int">
    <open>
      <hour>int</hour>
      <minute>int</minute>
      <units>string</units>
    </open>
    <close>
      <hour>int</hour>

```

```

    <minute>int</minute>
    <units>string</units>
  </close>
</day>
...
...
</resp>

```

Response Element	Data Type	Definition
currenttime		contains daylight savings, time, timezone.
time	<i>date/time</i>	current date and time of unit.
timezone	<i>int</i>	timezone offset.
daylightsavings	<i>string</i>	Active or Inactive value
spring		Contains information on the spring ahead attributes.
fall		Contains information on the fall back attributes.
month	<i>string</i>	Jan, Feb, Mar, etc...
week	<i>string</i>	First, Second, Third, Fourth, Last.
time	<i>time</i>	Time of the daylight savings.
day		Element that contains either the display value or: hour, minute, units, or the value of the day.
value	<i>string</i>	Attribute of the day element. Will be the day of the week. (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)
id	<i>int</i>	Identifies a day of the week, may be 1 through 7
hour	<i>int</i>	Hour of day. 0-23, or 1-12.
minute	<i>int</i>	Minute of day, 0-23.
units	<i>string</i>	AM or PM value if the time format = 0.

Example 1:

```
<cmd action = "read_store_schedule">
```

```

<resp action = "read_store_schedule">
  <currenttime time="01:37PM 02/28/10" timezone="-500">
    <daylightsavings>Inactive</daylightsavings>
    <spring>
      <month>Jan</month>
      <week>Last</week>
      <day>Sunday</day>
      <time>02:00 AM</time>
    </spring>
    <fall>
      <month>Nov</month>
      <week>First</week>
      <day>Sunday</day>
      <time>02:00 AM</time>
    </fall>
  </currenttime>

```

```

<days count="7">
<day value="Monday" id="1">
  <open>07:00 AM</open>
  <close>08:00 PM</close>
</day>
<day value="Tuesday" id="2">
  <open>07:00 AM</open>
  <close>08:00 PM</close>
</day>
...
...
<day value="Sunday" id="7">
  <open>09:00 AM</open>
  <close>07:00 PM</close>
</day>
</days>
</resp>

```

Example 2:

```

<cmd action = "read_store_schedule" time_format="0">

<resp action = "read_store_schedule" time_format="1">
  <currenttime time="01:37PM 02/28/10" timezone="-500">
    <daylightsavings>Inactive</daylightsavings>
    <spring>
      <month>Jan</month>
      <week>Last</week>
      <day>Sunday</day>
      <time>02:00 AM</time>
    </spring>
    <fall>
      <month>Nov</month>
      <week>First</week>
      <day>Sunday</day>
      <time>02:00 AM</time>
    </fall>
  </currenttime>
  <days count="7">
<day value="Monday" id="1">
  <open>
    <hour>07</hour>
    <minute>00</minute>
    <units>AM</units>
  </open>
  <close>
    <hour>08</hour>
    <minute>00</minute>
    <units>PM</units>
  </close>

```

```

</day>
<day value="Tuesday" id="2">
  <open>
    <hour>07</hour>
    <minute>00</minute>
    <units>AM</units>
  </open>
  <close>
    <hour>08</hour>
    <minute>00</minute>
    <units>PM</units>
  </close>
</day>
...
...
<day value="Sunday" id="7">
  <open>
    <hour>09</hour>
    <minute>00</minute>
    <units>AM</units>
  </open>
  <close>
    <hour>07</hour>
    <minute>00</minute>
    <units>PM</units>
  </close>
</day>
</days>
</resp>

```

Example 2:

```

<cmd action = "read_store_schedule" time_format="l">

<resp action = "read_store_schedule" time_format="l">
  <currenttime time="01:37PM 02/28/10" timezone="-500">
    <daylightsavings>Inactive</daylightsavings>
    <spring>
      <month>Jan</month>
      <week>Last</week>
      <day>Sunday</day>
      <time>02:00</time>
    </spring>
    <fall>
      <month>Nov</month>
      <week>First</week>
      <day>Sunday</day>
      <time>02:00</time>
    </fall>
  </currenttime>

```

```
<days count="7">
<day value="Monday" id="1">
  <open>
    <hour>07</hour>
    <minute>00</minute>
  </open>
  <close>
    <hour>20</hour>
    <minute>00</minute>
  </close>
</day>
<day value="Tuesday" id="2">
  <open>
    <hour>07</hour>
    <minute>00</minute>
  </open>
  <close>
    <hour>20</hour>
    <minute>00</minute>
  </close>
</day>
...
...
<day value="Sunday" id="7">
  <open>
    <hour>09</hour>
    <minute>00</minute>
  </open>
  <close>
    <hour>19</hour>
    <minute>00</minute>
  </close>
</day>
</days>
</resp>
```

1.19 write_store_schedule

Write new schedule open/close time for a day of the week for the Operating Hours.

Command

```
<cmd action="write_store_schedule" auth="string" acct="string" user="string"
  password="string" id="int">
  <open>
    <hour>int</hour>
    <minute>int</minute>
  </open>
  <close>
    <hour>int</hour>
    <minute>int</minute>
  </close>
</cmd>
```

Command Attribute	Data Type	Definition
action	string	Command action which is always <i>write_store_schedule</i> .
auth	string	Authorization for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
acct	string	Account Code for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
user	string	User name for the C-Store version of SYSTEM MANAGER. Not used otherwise.
password	string	Password for the C-Store version of SYSTEM MANAGER. Not used otherwise.
id	int	Integer value of the day of week. (1 – 7 are the only valid id's) Monday = 1 Tuesday = 2 Wednesday = 3 Thursday = 4 Friday = 5 Saturday = 6 Sunday = 7
open		Complex tag containing the hour and minute tags. Must supply atleast one of <open> or <close> tags.
close		Complex tag containing the hour and minute tags. Must supply atleast one of <open> or <close> tags.
hour	int	Hour of the day (0 – 23).
minute	int	Minute of the day (0 – 59).

Response

```
<resp action="write_store_schedule" auth="string" acct="string" user="string"
  password="string" id="int" error="int">
```

```

<open>
  <hour>int</hour>
  <minute>int</minute>
</open>
<close>
  <hour>int</hour>
  <minute>int</minute>
</close>
</cmd>

```

Response Attribute	Data Type	Definition
action	string	Command action which is always <i>write store schedule</i> .
auth	string	Authorization for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
acct	string	Account Code for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
user	string	User name for the C-Store version of SYSTEM MANAGER. Not used otherwise.
password	string	Password for the C-Store version of SYSTEM MANAGER. Not used otherwise.
id	int	Integer value of the day of week. (1 – 7 are the only valid id's) Monday = 1 Tuesday = 2 Wednesday = 3 Thursday = 4 Friday = 5 Saturday = 6 Sunday = 7
error	int	Error code from attempt. If 0, then schedule was changed, else see Error Messages .
open		Complex tag containing the hour and minute tags.
close		Complex tag containing the hour and minute tags.
hour	int	Hour of the day (0 – 23).
minute	int	Minute of the day (0 – 59).

1.20set_store_time

Set the current date/time, time zone, and/or daylight savings fields for the store.

Command

```

<cmd action="set_store_time" auth="string" acct="string" user="string" password="string">
  <date month="int" day="int" year="int" hour="int" minute="int"/>
  <time_zone value="int"/>
  <spring month="int" week="int" day="int" hour="int" minute="int"/>
  <fall month="int" week="int" day="int" hour="int" minute="int"/>

```

```
<daylight_savings value="int"/>
</cmd>
```

Command Attribute	Data Type	Definition
action	<i>string</i>	Command action which is always <i>set store time</i> .
auth	<i>string</i>	Authorization for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
acct	<i>string</i>	Account Code for the EMA, RACK, SINGLE versions of the AK255. Not used if sending to C-Store version.
user	<i>string</i>	User name for the C-Store version of SYSTEM MANAGER. Not used otherwise.
password	<i>string</i>	Password for the C-Store version of SYSTEM MANAGER. Not used otherwise.
date		Contains month, day, year, hour, and minute attributes.
time_zone		Contains value attribute.
spring		Contains month, week, day, hour, and minute attributes.
fall		Contains month, week, day, hour, and minute attributes.
daylight_savings		Contains value attribute.
year	<i>int</i>	2 digit year.
month	<i>int</i>	Integer value of the month (1 – 12).
week	<i>int</i>	Integer value of the week. 1 – First Week 2 – Second Week 3 – Third Week 4 – Fourth Week 5 – Last Week
day	<i>int</i>	Integer value of the day of week. (1 – 7 are the only valid id's) Monday = 1 Tuesday = 2 Wednesday = 3 Thursday = 4 Friday = 5 Saturday = 6 Sunday = 7 Or day of month within the <date> element (1-31)
value	<i>int</i>	UTC <i>timezone</i> offset expressed as 100*hours. e.g. Eastern US time has an offset of -500 - or - For <i>daylight_savings</i> contains: 0 – Turn off daylight savings 1 – Turn on daylight savings

hour	<i>int</i>	Hour of the day (0 – 23).
minute	<i>int</i>	Minute of the day (0 – 59).

1.21 read_hvac_service

Read the information about the relay status.

Command

```
<cmd action="read_hvac_service" ahindex="int"/>
```

Command Attribute	Data Type	Definition
action	<i>string</i>	Command action which is always <i>read_hvac_service</i> .
ahindex	<i>int</i>	Number associated with the request hvac unit from reading the " <i>read_hvac</i> "

Response

```
<resp action="read_hvac_service" echo attributes error="int">
```

```
<relays total="int">
```

```
<relay name="string">
```

```
<value>int</value>
```

```
<status>string</status>
```

```
</relay>
```

```
...
```

```
...
```

```
</relays>
```

```
<inputs total="int">
```

```
<input name="string">
```

```
<value>int</value>
```

```
<status>string</value>
```

```
</input>
```

```
...
```

```
...
```

```
</inputs>
```

```
<sensors total="int">
```

```
<sensor name="string">
```

```
< status >string</ status >
```

```
<value>signed decimal</value>
```

```
<offset>signed decimal</offset>
```

```
<sensor name="string">
```

```
< status >string</ status >
```

```
<value>signed decimal</value>
```

```
</sensors>
```

```
</resp>
```

Response Attribute	Data Type	Definition
action	<i>string</i>	Command action which is always <i>write_store_schedule</i> .
relays		Complex element containing a <i>total</i> attribute and 1 or more <i>relay</i> elements.
relay		Complex element containing <i>name</i> , <i>value</i> , and <i>status</i> .

inputs		Complex element containing a <i>total</i> attribute and 1 or more <i>input</i> elements.
input		Complex element containing <i>name</i> , <i>value</i> , and <i>status</i> .
sensors		Complex element containing a <i>total</i> attribute and 1 or more <i>sensor</i> elements.
sensor		Complex element containing <i>name</i> , <i>value</i> , and <i>status</i> . Will contain an <i>offset</i> element if the <i>status</i> is set to <i>Auto</i> . If <i>status</i> is set to <i>Manual</i> , then the <i>value</i> is what the sensor is adjusted to.
total	<i>int</i>	Total number of; relays, inputs, or sensors.
name	<i>string</i>	Name of input.
id	<i>int</i>	Index for the relay/input/sensor.
value	<i>signed decimal</i>	Value of the sensor.
offset	<i>signed decimal</i>	Value of offset for sensor.
status	<i>int</i>	Status of input/relays. List can be: 1 - Manual On 2 - Manual Off 3 - Auto-Off 4 - Auto-On 5 - Auto_Off Locked 6 - Auto-On Locked 7 - Auto-Off Shutdown 8 - Auto-On Shutdown For sensors: 0 - Auto (for sensor) 2 - Manual (for sensor)

1.22 set_hvac_service

Set the inputs/relays/sensors information from the service screen. This command can use all or one of the relays/inputs/sensors.

Command

```
<cmd action="set_hvac_service" auth="string" acct="string" user="string"
password="string" ahindex = "int" units="u|U|s|S">
  <relays>
    <relay id="int" status="int">
      ...
    </relays>
  <inputs>
    <input id="int" status="int">
      ...
    </inputs>
  <sensors>
    <sensor id="int" status="int" value="signed decimal">
      ...
    </sensors>
</cmd>
```

Command Attribute	Data Type	Definition
action	<i>string</i>	Command action which is always <i>set hvac service</i> .
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	<i>string</i>	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	<i>string</i>	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
units	<i>char</i>	Optional value for "value" attribute if setting temperature. Either SI or US units -> "u U s S"
ahindex	<i>int</i>	Index of HVAC unit.
relays		Contains all "relay" elements.
inputs		Contains all "input" elements.
sensors		Contains all "sensor" elements.
relay		Contains the <i>id</i> , <i>status</i> associated with the relay in the list from reading the " <i>read_hvac_service</i> " command.
input		Contains the <i>id</i> , <i>status</i> associated with the input in the list from reading the " <i>read_hvac_service</i> " command.

sensor		Contains the <i>id</i> , <i>status</i> , <i>value</i> associated with the sensor in the list from reading the “ <i>read hvac service</i> ” command.
id	<i>int</i>	Index of the relay/input/sensor.
status	<i>int</i>	status can be (for inputs and relays): 0 -> Auto (where it will switch to either Auto-On or Auto-Off). 1 -> Manual On 2 -> Manual Off For sensors: 0 -> Auto (Meaning <i>sensor</i> will used <i>offset</i> if any given) 2 -> Manual (Meaning <i>sensor</i> will be set to <i>value</i> if given)
value	<i>float</i>	Used for sensor adjustment for: status=0 -> set sensor offset status=2 -> set sensor value

1.23read_hvacs

Read information on all HVAC's configured.

Command

```
<cmd action = "read_hvacs" />
```

Response

```
<resp action = "read_hvacs" error="int"/>
```

Response Element	Data Type	Definition
action	<i>string</i>	Command action which is always "read_hvacs"
hvac		Complex element to describe each HVAC unit.
ahindex	<i>int</i>	Index of HVAC unit.
name	<i>string</i>	Name of the SYSTEM MANAGER.
status	<i>string</i>	Current status of unit.
value	<i>double</i>	Current value of the unit
units	<i>string</i>	Degrees symbol for either SI or US.
error	<i>int</i>	Error code from command. If not 0, then see Error Messages .

Example:

```
<cmd action="read_hvac" />
```

```
<resp action="read_hvac" error="0">
  <hvac ahindex="1">
    <name>HVAC Main</name>
    <status>System Satisfied</status>
    <value>HVAC</type>
    <units>°F</units>
  </hvac>
  <total>1</total>
</resp>
```

1.24read_hvac_unit

Read details of specific HVAC unit.

Command

```
<cmd action = "read_hvac_unit" ahindex = "int" units = "u|U|s|S" error="int" />
```

Response Element	Data Type	Definition
action	<i>string</i>	Command action which is always "read_hvac_unit"
ahindex	<i>int</i>	Index of HVAC unit retrieved from "read_hvac"
units	<i>string</i>	Optional attribute to retrieve units for US or SI. Allowed values are "u, U, s, S".

Response

```
<resp action="read_hvac_unit" ahindex="int" units="string" error="int">
  <name>string</name>
  <status>string</status>
  <value>double</value>
  <units>string</units>
  <cooling_stages total="int">
    <cooling_stage stage="string" index="int" target="double" status="string">
      </cooling_stage>
    </cooling_stages>
  <aux_heats total=" int ">
    <aux_heat stage=" string " index=" int " target=" double " status=" string ">
      </aux_heat>
    </aux_heats>
  <heat_reclaims total=" int ">
    <heat_reclaim stage=" string " index=" int " target=" double " status=" string ">
      </heat_reclaim>
    </heat_reclaims>
  <dehumid_type type="string" type_id="int">
    <dehumid_control control_id="int">string</dehumid_control>
    <target units="string" unit_index="int" range="signed decimal">signed decimal</target>
    <stages range="signed decimal" units="string" units_index="int">
      <stage index="int" target="signed decimal" pre_delay="int" post_delay="int"/>
    ...
    ...
  </stages>
</resp>
```

Response Element	Data Type	Definition
resp		Return element.
name	<i>string</i>	Name of HVAC unit.
status	<i>string</i>	Status description of HVAC unit.
value	<i>double</i>	Value of HVAC unit.
Units	<i>string</i>	Units of the value (degf °F or degc °C)

cooling_stages		Complex element that contains all stages.
aux_heats		Complex element that contains all auxiliary heats.
heat_reclaims		Complex element that contains all heat reclaims.
cooling_stage		Contains information for one cooling stage.
stages		Contains information on humidity control stages.
aux_heat		Contains information for one auxiliary heat stage.
heat_reclaim		Contains information for one heat reclaim.
total	<i>int</i>	Total number of any of the stages.
stage	<i>string</i>	Stage name for cooling, aux heat, humidity control, or heat reclaim.
index	<i>int</i>	Index number of the stage.
target	<i>double</i>	Target setting for the stage.
status	<i>string</i>	Current status of the stage.
pre_delay	<i>int</i>	delay in minutes
post_delay	<i>int</i>	delay in minutes
range	<i>double</i>	range of target
error	<i>int</i>	Error code from command. If not 0, then see Error Messages .

Example:

```

<cmd action="read_hvac_unit" ahindex="1" units="u"/>

<resp action="read_hvac_unit" ahindex="1" units="u" error="0">
  <name>HVAC Main</name>
  <status>System Satisfied</status>
  <value>0.00</value>
  <units>°F</units>
  <cooling_stages total="2">
    <cooling_stage stage="Cooling 1" index="1" target="74.00" status="Maint. Capacity-Heat">
    </cooling_stage>
    <cooling_stage stage="Cooling 2" index="2" target="76.00" status="Maint. Capacity-Heat">
    </cooling_stage>
  </cooling_stages>
  <aux_heats total="2">
    <aux_heat stage="Gas Valve" index="1" target="69.00" status="System Satisfied">
    </aux_heat>
    <aux_heat stage="Aux Heat 2" index="2" target="67.00" status="System Satisfied">
    </aux_heat>
  </aux_heats>
  <heat_reclaims total="0">
  </heat_reclaims>
</resp>

```

1.25 write_hvac_unit

Sets a value for the specific HVAC unit.

Command

```
<cmd action = "write_hvac_unit" ahindex = "int" units="string"
      user="string" password = "string" auth = "string" acct = "string" >
  <val type = "string" stype="int" index = "int" set = "signed decimal" />
</cmd>
```

Command Element	Data Type	Definition
action	string	Command action which is always "write_hvac_unit"
ahindex	int	Index of HVAC unit retrieved from "read_hvac"
units	string	U or u for US, or S or s for SI. If not used, default is system setting.
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
val		Complex tag containing value to be changed.
type	string	The control type that will be changed. Accepted values are: cooling aux_heat heat_reclaim range (applies only to cooling, aux_heat, heat_reclaim) humidity
stype	int	stype is the type of set variable. Used if setting any of the cooling, aux_heat, or heat_reclaim stage information. If type = "range", then "stype" is not needed. If type = "humidity", then the 3 – range is available, 4 – type and 5 – control. 0 – target (index needed) except humidity (desiccant type) 1 – pre delay (index needed) 2 – post delay (index needed) 3 – range 4 – type

		5 – control
Index	<i>int</i>	Index retrieved from read_hvac_unit for the stage. Index is required as shown in the stype description.
offset_type	<i>int</i>	0 – Cooling Offset 1 – Heat Reclaim Offset 2 – Aux Heat Offset Only needed if “type” = “range”
set	<i>signed decimal / int</i>	Value which needs to be changed. For temps and percentages: float value. For humidity type: 0 - None 1 - Cooling 2 - Desiccant Whl For humidity control: 0 - Humidity 1 - Calc In Dew 2 - Inside Dew 3 - Calc OutDew 4 - Outside Dew For delay types: the <i>int</i> value of minutes for pre_delay and post_delay.

Command (examples of humidity control)

```
<cmd action="write_hvac_unit" ahindex = "1" units="U" auth = "12345" acct = "50">
  <val type="humidity" stype="4" set="0"/>
  <val type="humidity" stype="1" index="2" set="2"/>
  <val type="humidity" stype="0" index="2" set="32.1"/>
</cmd>
```

Response

```
<resp action="write_hvac_unit" repeat attributes ... error = "int" >
  <val repeat attributes ... error = "int" />
  ...
  ...
</resp>
```

1.26 write_hvac_setback

Modify the night setback for the cooling or heating on an HVAC unit.
Can modify all or one of the elements, but the ahindex must be supplied.

Command

```
<cmd action="write_hvac_setback" ahindex = "int" units="string" type = "string"
    user="string" password = "string" auth = "string" acct = "string" >
  <nightsetback value="int" type="string" offset="signed decimal" num_schds="int">
    <schedule index="int" sch_type="int">
      <start>int</start>
      <stop>int</stop>
      <weekdays>string</weekdays>
      <holidays>string</holidays>
    </schedule>
  </cmd>
```

Command Element	Data Type	Definition
action	string	Command action which is always "write_hvac_setback"
ahindex	int	Index of HVAC unit retrieved from "read_hvacs"
units	string	U or u for US, or S or s for SI. If not used, default is system setting.
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS.. May be sent to the SYSTEM MANAGER, but will be ignored
nightsetback		Element contains <i>value</i> , <i>offset</i> , <i>num_schds</i> , and <i>type</i> as attributes, and contains <i>schedules</i> element
value	int	0 – No (do not use nightsetback) 1 – Yes (use nightsetback)
type	string	The control type that will be changed. Accepted values are: cooling (cooling) heating (heat reclaim heating) aux_heat (auxiliary heating)
offset	signed decimal	Offset value in degrees of the night setback. Not used if removing night setback.
num_schds	int	number of schedules for the night setback. Not used if removing night setback. If turning on night setback, this will be used to set the number of total schedules for the setback.

schedule		Element contains <i>index</i> , <i>type</i> , <i>start</i> , <i>stop</i> , <i>weekdays</i> , <i>holidays</i> .
index	<i>int</i>	Index of the schedule to be changed. Range from 1 to <i>num_scheds</i> . NOTE: If the index of this schedule is not within the range of <i>num_scheds</i> , an error will occur.
sch_type	<i>int</i>	Required field specifying the type of schedule to be changed or added. 0 – Standard 1 – Relative
start	<i>int</i>	Time period range between -2359 to +2359
stop	<i>int</i>	Time period range between -2359 to +2359
weekdays	<i>string</i>	Weekdays to which this schedule applies, e.g. SMTWRFA
holidays	<i>string</i>	Holidays to which this schedule applies, e.g. 12345678

1.27 read_lighting

Read all lighting zones with relay information.

Command

```
<cmd action = "read_lighting" />
```

Response

```
<resp action = "read_lighting error="int">
```

Response Element	Data Type	Definition
unit_name	string	Name of SYSTEM MANAGER
device		Element that contains name, index, type.
value	string	Attribute containing current status of lighting point.
name	string	Name of lighting zone.
index	int	Index number of location of zone.
type	string	Type of lighting point.
stype	int	Reserved (internal use only). Not monitored
total	int	Total number of zones, relays, and dimmers.
All other attributes		Other attributes are reserved for internal use and not monitored.

Example:

```
<cmd action = "read_lighting" />
```

```
<resp action="read_lighting" error="0">
  <unit_name>Test CStore</unit_name>
  <device host="0" nodetype="255" addr="0" combo="7" comboindex="0" bpidx="0"
    node="0" mod="0" point="0" alarm="0" indent="0" arg1="1" value="Off">
    <name>Outside</name>
    <index>1</index>
    <type>ZONE</type>
    <stype>6</stype>
  </device>
  <device host="0" nodetype="3" alarm="0" online="1" addr="03-3.4" combo="7"
    comboindex="0" bpidx="30" node="3" mod="3" point="28" indent="2" arg1="1"
    value="0%">
    <name>Dimmer</name>
    <index>1</index>
    <type>ZONEDIMMER</type>
    <stype>9</stype>
  </device>
  <device host="0" nodetype="255" addr="0" combo="7" comboindex="1" bpidx="0"
    node="0" mod="0" point="0" alarm="0" indent="0" arg1="2" value="Off">
    <name>50% Sales</name>
    <index>2</index>
    <type>ZONE</type>
    <stype>6</stype>
```

```
</device>
<device host="0" nodetype="1" alarm="0" online="1" addr="03-3.2" combo="7"
  comboindex="1" bpidx="7" node="3" mod="3" point="10" indent="2" value="Off">
  <name>Relay1</name>
  <index>2</index>
  <type>LIGHTING</type>
  <stype>10</stype>
</device>
<total>4</total>
</resp>
```

1.28read_lighting_zone

Read specifics on a lighting zone.

NOTE:

Does not read panel information yet.

Command

```
<cmd action="read_lighting_zone" index="int"/>
```

Command Attribute or Element	Data Type	Definition
action	<i>string</i>	Command action description. Always “read_lighting_zone”
index	<i>int</i>	Index of lighting zone to read.

Response

```
<resp action="read_lighting_zone" index="int" error="int">
  <zone name="string" type="string" status="string" alarm="string">
    <schedules active="string" count="int">
      <schedule type="string" index="int">
        <on> time date </on>
        <off> time date </off>
        <days>string</days>
        <holidays>string</holidays>
      </schedule>
    </schedules>
  </zone>
  <relays count="int">
    <relay name="string" index="int" status="string">
    </relay>
  </relays>
  <run_times>
    <today>time</today>
    <yesterday>time</yesterday>
    <total_hrs>int</total_hrs>
    <total_maint>int</total_maint>
  </run_times>
</resp>
```

Response Element	Data Type	Definition
zone		Element containing <i>name</i> , <i>type</i> , <i>status</i> , <i>alarm</i> , and <i>schedules</i> associated with lighting zone.
schedules		Element containing <i>active</i> , <i>count</i> of schedules, and the <i>schedule</i> element for each schedule.
schedule		Each schedule contains <i>type</i> , <i>index</i> , the <i>on</i> and <i>off</i> time, <i>days</i> and <i>holidays</i> .
relays		Contains a list of <i>relay</i> types, and <i>count</i> of relays.

relay		Element that contains the <i>name</i> , <i>index</i> , and <i>status</i> for each relay for this lighting zone.
count	<i>int</i>	Number of <i>relays</i> .
index	<i>int</i>	Index is used for the <i>schedule</i> , and <i>relay</i> .
error	<i>int</i>	Error code if any problems arise from requesting data.
count	<i>int</i>	Count of <i>schedules</i> , <i>relays</i>
name	<i>string</i>	Name of <i>zone</i> , <i>relays</i>
status	<i>string</i>	Status of lighting <i>zone</i> , and <i>relays</i>
on	<i>time date</i>	On time for schedule
off	<i>time date</i>	Off time for schedule
days	<i>string</i>	String representation of the days the schedule is active. "SMTWRFA"
holidays	<i>string</i>	String representation of the holidays affecting this schedule. "12345678"
today	<i>time</i>	Run time today of zone in " <i>hrs:mins</i> "
yesterday	<i>time</i>	Run time yesterday of zone in " <i>hrs:mins</i> "
total_hrs	<i>int</i>	Total minutes this zone has operated. Right now is shown in minutes, so actual hours are total_hrs / 60.
total_maint	<i>int</i>	Total time since last maintenance. Right now is shown in minutes, so actual hours are total_maint / 60.

Example:

```
<cmd action="read_lighting_zone" index="1"/>
```

```
<resp action="read_lighting_zone" index="1" error="0">
  <zone name="Outside" type="AK2-SC255" status="Off" alarm="OK">
    <schedules active="Previous: 12:00AM To 12:00AM" count="2">
      <schedule type="STANDARD" index="1">
        <on>12:00AM</on>
        <off>12:00AM</off>
        <days>S-----A</days>
        <holidays>123-----</holidays>
      </schedule>
      <schedule type="STANDARD" index="2">
        <on>06:00AM</on>
        <off>10:00PM</off>
        <days>-MTWRF-</days>
        <holidays>12-----</holidays>
      </schedule>
    </schedules>
  </zone>
  <relays count="1">
    <relay name="Relay1" index="1" status="Off">
      </relay>
    </relays>
  <run_times>
```

```
<today>00:00</today>  
<yesterday>00:00</yesterday>  
<total_hrs>0</total_hrs>  
<total_maint>0</total_maint>  
</run_times>  
</resp>
```

1.29 set_zone_override

Set the lighting zone to a specific override status (Manual off, Manual on, Timed on, Auto).

Command

```
<cmd action = "set_zone_override" user="string" password="string" auth="string"
    acct="string" index = "int" relay="int" state="int" time="int" units="int" />
```

Response Element	Data Type	Definition
action	string	Command attribute "set_zone_override"
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
index	int	Index of the lighting zone to override.
relay	int	Index of the lighting relay associated with this lighting zone. If not used, then the main override for the zone will be set, otherwise the relay index will be used.
state	int	State to put the lighting zone: 0 -> Auto 1 -> Manual On 2 -> Manual Off 3 -> Timed On (cannot use if setting relay)
time	int	If using "Time On" then use this attribute to specify how long to override lights. (1 – 999 only)
units	int	units specifies what the time signifies: 0 -> sec 1 -> min 2 -> hour

Example 1)

```
<cmd action="set_zone_override" index="1" state="1" />
```

```
<resp action="set_zone_override" repeat attributes error="0">
```

Example 2)

```
<cmd action="set_zone_override" index="1" state="3" time="12" units="2" />
```

```
<resp action="set_zone_override" repeat attributes error="0">
```

Example 3)

```
<cmd action="set_zone_override" index="1" state="3" time="-1" units="2" />
```

```
<resp action="set_zone_override" repeat_attributes error="45">Invalid Hour  
</resp>
```

Example 4)

```
<cmd action="set_zone_override" index="1" relay="1" state="2"/>
```

```
<resp action="set_zone_override" repeat_attributes error="0"/>
```

1.30 write_lighting_zone

Modify information on the lighting zone, including schedules, and names.

Command

```
<cmd action="write_lighting_zone" user="string" password="string"
      auth="string" acct="string" index="int" name="string" num_scheds="int">
  <schedule index="int" sch_type="int">
    <start>int</start>
    <stop>int</stop>
    <weekdays>string</weekdays>
    <holidays>string</holidays>
  </schedule>
</cmd>
```

Command Element	Data Type	Definition
action	<i>string</i>	Tag used to identify type of xml command.
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	<i>string</i>	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	<i>string</i>	Password. Applies only to the AKCS.. May be sent to the SYSTEM MANAGER, but will be ignored
index	<i>int</i>	Lighting zone index retrieved from read_lighting. Index of schedule to be changed when within the <i>schedule</i> element. Required for both zone and schedule (if modifying an actual schedule).
name	<i>string</i>	If used, this will change the name of the zone.
num_scheds	<i>int</i>	Number of schedules needed for this lighting zone.
schedule		Element contains <i>index</i> , <i>sch_type</i> , <i>start</i> , <i>stop</i> , <i>weekdays</i> , and <i>holidays</i> associated with the requested change. Not needed if just modifying the number of schedules.
sch_type	<i>int</i>	Required field specifying the type of schedule to be changed or added. 0 – Standard 1 – Relative Not needed if not modifying the type of schedule.
start	<i>int</i>	Time period range between -2359 to +2359
stop	<i>int</i>	Time period range between -2359 to +2359

weekdays	<i>string</i>	Weekdays to which this schedule applies, e.g. SMTWRFA, can use any combination of these days: SA, or MTWRF, ...
holidays	<i>string</i>	Holidays to which this schedule applies, e.g. 12345678

Example 1:

This is used if adding a new schedule (2), and then changing all parameters of it.

```
<cmd action="write_lighting_zone" user="Supervisor" password="12345" auth="12345"
  acct="50"index="1" num_schds="2">
  <schedule index="2" sch_type="0">
    <start>1530</start>
    <stop>2330</stop>
    <weekdays>SA</weekdays>
    <holidays>12</holidays>
  </schedule>
</cmd>
```

Example 2:

If used, it will set the number of schedules to 1. If there were more schedules, they will be removed from this zone. Will also change the name of the zone to a user defined name supplied.

```
<cmd action="write_lighting_zone" user="Supervisor" password="12345" auth="12345"
  acct="50"index="1" name="Zone Outside" num_schds="1">
```

Example 3:

If used, it will only modify the holidays for schedule 2.

```
<cmd action="write_lighting_zone" user="Supervisor" password="12345" auth="12345"
  acct="50"index="1">
  <schedule index="2">
    <holidays>123</holidays>
  </schedule>
</cmd>
```

Example 4:

Setting up a relative schedule.

```
<cmd action="write_lighting_zone" user="Supervisor" password="12345" auth="12345"
  acct="50"index="1" num_schds="3">
  <schedule index="3" sch_type="1">
    <start>-100</start> - 1 hour before store schedule
    <stop>30</stop> - 30 minutes after store schedule
    <weekdays>MTWRF</weekdays>
    <holidays>345</holidays>
  </schedule>
</cmd>
```

1.31 read_holidays

Read holiday configuration

Command

```
<cmd action = "read_holidays" />
```

Response

```
<resp action = "read_holidays" error="int">
  <holiday_count>int</holiday_count>
  <holiday>
    <id>int</id>
    <desc>string</desc>
    <start>DDMM|MMDD</start>
    <end> DDMM|MMDD </end>
    <open>HOUR12|HOUR24</open>
    <close> HOUR12|HOUR24</close>
  </holiday>
  .
  .
  .
</resp>
```

Response Element	Data Type	Definition
holiday_count	<i>int</i>	Number of holidays in list, may be 1 through 8
holiday		Element that describes one holiday. Encloses elements id, desc, start, end, open, and close
id	<i>int</i>	Identifies a holiday, may be 1 through 8
desc	<i>string</i>	Description of holiday
start	<i>DDMM MMDD</i>	Starting month and day of this holiday period
end	<i>DDMM MMDD</i>	Ending month and day of this holiday period
open	<i>HOUR12 HOUR24</i>	Opening time during this holiday period
close	<i>HOUR12 HOUR24</i>	Closing time during this holiday period

Example

```
<cmd action="read_holidays"/>
<resp action="read_holidays" error="0">
  <holiday_count>2</holiday_count>
  <holiday>
    <id>1</id>
    <desc>XMASS</desc>
    <start>25/12</start>
    <end>31/12</end>
```

<open>10:00 AM</open>
<close>04:00 PM</close>
</holiday>
<holiday>
<id>2</id>
<desc>NEW YEAR</desc>
<start>01/01</start>
<end>02/01</end>
<open>11:00 AM</open>
<close>01:00 PM</close>
</holiday>
</resp>

1.32 write_holiday_sch

Write holiday schedule

Command

```
<cmd action = "write_holiday_sch" id="int" new="1" auth="string" acct="string">
  <desc>string</desc>
  <start>
    <day>int</day>
    <month>int</month>
  </start>
  <end>
    <day>int</day>
    <month>int</month>
  </end>
  <open>
    <hour>int</hour>
    <minute>int</minute>
  </open>
  <close>
    <hour>int</hour>
    <minute>int</minute>
  </close>
</cmd>
```

Command Attribute or Element	Data Type	Definition
id	int	Number of holiday whose schedule is to be changed. May be 1 through 8
new	int	If supplied, then using <id> is not necessary. If found, <new> must be 1, else error will show.
desc	string	Description of holiday. Only needed if changing description name, or using the <new> attribute.
start		Starting month and day of this holiday period
end		Ending month and day of this holiday period
open		Opening time during this holiday period
close		Closing time during this holiday period
day	int	Day of month in range 1 through 31
month	int	Month in range 1 through 12
hour	int	Hour in range 0 through 23
minute	int	Minute

Response

```
<resp action = "write_holiday" error="int">
```

```
<desc>string</desc>
<start>
  <day>int</day>
  <month>int</month>
</start>
<end>
  <day>int</day>
  <month>int</month>
</end>
<open>
  <hour>int</hour>
  <minute>int</minute>
</open>
<close>
  <hour>int</hour>
  <minute>int</minute>
</close>
</resp>
```

1.33 write_schedule_times

Command

```
<cmd action="write_schedule_times" id="int" auth="string" acct="string">
  <sch_number>int</sch_number>
  <sch_detail>
    <on_time>
      <hour>int</hour>
      <minute>int</minute>
    </on_time>
    <off_time>
      <hour>int</hour>
      <minute>int</minute>
    </off_time>
    <weekdays>string</weekdays>
    <holidays>string</holidays>
  </sch_detail>
  <sch_detail>
    <on_time>
      <hour>int</hour>
      <minute>int</minute>
    </on_time>
    <off_time>
      <hour>int</hour>
      <minute>int</minute>
    </off_time>
    <weekdays>string</weekdays>
    <holidays>string</holidays>
  </sch_detail>
  ---
  <sch_detail>
    <on_time>
      <hour>int</hour>
      <minute>int</minute>
    </on_time>
    <off_time>
      <hour>int</hour>
      <minute>int</minute>
    </off_time>
    <weekdays>string</weekdays>
    <holidays>string</holidays>
  </sch_detail>
</cmd>
```

command attribute	Data Type	Definition
Id	int	Identifies the schedule whose times are to be changed. Matches the value of an id element that is contained in an “sch” element that is listed in the “schedule_summary” command response.

		Range 1 through 100 but cannot exceed the number of schedules defined.
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters)
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters)

command element	Data Type	Definition
sch_number	<i>int</i>	The number of sch_detail elements Less than or equal to 12 for Dfrost and Defrost Coord timers Less than or equal to 8 for all other timers.
sch_detail		Contains an on time, off time, weekdays, and holidays for one schedule detail. One of these elements must be provided for each of the existing schedule details, even for details that have not changed.
on_time		on time for this schedule detail. Must be provided even if it is not a new value. Contains hour and minute elements
off_time		off time for this schedule detail. Must be provided for non-defrost schedules even if it is not a new value. Need not be provided for defrost and coord defrost schedules but if provided it is ignored. Contains hour and minute elements
Hour	<i>int</i>	Hour of the day. Range 0 through 23
minute	<i>int</i>	Minute of the hour. Range 0 through 59.
weekdays	<i>string</i>	Weekdays for this schedule detail. Must be provided even if it is not a new value. This is a string composed of one or more of the following characters: "SMTWRFA" for Sunday, Monday, Tuesday, Wednesday, Thursday, FRiday, Saturday respectively. The order is not important, not all characters need be present, but duplicates may not appear. For example, to indicate Monday and Saturday the string is: "MA". To indicate all days of the week the string is: "SMTWRFA"
holidays	<i>string</i>	Holidays for this schedule detail. Must be provided even if it is not a new value. This is a string composed of one or more of the following characters: "12345678" indicating the holiday number. The order is not important, not all need be present, but duplicates may not appear. For example, to indicate holidays 2 and 4 the string would be: "24"

Response

<resp action="write_schedule_times" echo of command attributes... error="int"/>

1.34set_offset

Set an offset for a particular SI point.

Command:

```
<cmd action="set_offset" user="string" pass="string" auth="string" acct="string"
nodetype="int" node="int" mod="int" point="int" value="signed decimal"/>
```

Response:

```
<resp action="set_offset" echo of command attributes... error="int"/>
```

Command Element	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS, but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
nodetype	int	Node type (Must be a sensor input)
node	int	Node address
mod	int	Module number within node.
point	int	Point number as displayed on the System Manager. For the relay output point, which is nodetype 1, the “point” element value is what would be displayed on the System Manager offset by 8. For example if the relay’s board point address is displayed as 1-3.2, the “point” XML element has a value of 8+2=10 For the relay enable input, which is nodetype 0, the offset is 0, that is, the value displayed on the System Manager matches the “point” XML element value.
value	signed decimal	New value of offset.

1.35read_suction_group

Read a specific suction group from information used from *read_devices* command. This command is only accessible if a rack system is enabled controller.

Command:

```
<cmd action="read_suction_group" rack_id="int" suction_id="int"/>
```

Response:

```
<resp action="read_suction_group" echo of command attributes... error="int">
  <name>string</name>
  <value units="string" units_index="int">signed decimal</value>
  <num_circuits>int</num_circuits>
  <suction_target units="string" units_index="int">signed decimal</suction_target>
  <suction_cutout units="string" units_index="int">signed decimal</suction_cutout>
  <suction_control type="string" type_id="int">
    <temp_target units="string" units_index="int">signed decimal</target>
    <temp_range units="string" units_index="int">signed decimal</range>
    <max_pressure_float units="string" units_index="int">signed
decimal</max_pressure_float>
    <post_defrost_delay units="string" units_index="int">int</post_defrost_delay>
  </suction_control>
  <auto_schedule>int</auto_schedule>
  <almhi_limit action="string" action_index="int" routingAction="int" units="string"
units_index="int">signed decimal</almhi_limit>
  <almhi_dur units="string" units_index="int">int</almhi_dur>
  <almlo_limit action="string" action_index="int" routingAction="int" units="sting"
units_index="int">signed decimal</almlo_limit>
  <almlo_dur units="string" units_index="int">int</almlo_dur>
</resp>
```

Response Element	Data Type	Definition
name	<i>string</i>	Describes name of the suction group
addr	<i>string</i>	Address of suction point
num_circuits	<i>int</i>	Number of circuits found in this suction group.
value	<i>signed decimal</i>	Value of suction pressure.
Units	<i>string</i>	Units of measurement (psi, bar, degf, degc, etc...)
units_index	<i>int</i>	Id of units
suction_target	<i>Signed decimal</i>	Value of target pressure. Holds units and units_index
suction_cutout	<i>Signed decimal</i>	Safety cutout value. Holds units and units index.
rack_id	<i>int</i>	1 based index of rack.
suction_id	<i>int</i>	1 based index of suction on this particular rack.
suction_control		Contains address, temperature target, range, max pressure float and post defrost delay. Contains two attributes: type and type_id

		type_id type 0 None 1 AK2-SC255 2 AKC 16x 3 Sensor 4 Dynamic
temp_target	<i>Signed decimal</i>	Value of temperature target. Holds units and units index.
range	<i>Signed decimal</i>	Range of temp target. Holds units and units index.
max_pressure_float	<i>Signed decimal</i>	Max floating pressure. Holds units and units index.
post_defrost_delay	<i>int</i>	Holds units and units index.
auto_schedule	<i>int</i>	Value of defrost scheduling. This value needs to be turned off if any defrost schedules will be changed for the circuits on this suction group. 0 – Off 1 – On
almhi_limit	<i>signed decimal</i>	Pressure trip value of high temperature alarm. Contains action, action_index, routingAction, units and units_index.
almhi_dur	<i>int</i>	Delay of high pressure alarm. Contains units="sec min hour" and units_index="1 2 3" 1 – sec 2 – min 3 – hour
almlo_limit	<i>signed decimal</i>	Pressure trip value of low temperature alarm. Contains action, action_index, routingAction, units and units_index.
almlo_dur	<i>int</i>	Delay of low pressure alarm. Contains units="sec min hour" and units_index="1 2 3"
routingAction	<i>int</i>	Routing action. Range is 1 through 15.
action	<i>string</i>	Disabled Log Only Normal Severe Critical
action_index	<i>Int</i>	1 – Disabled 2 – Log Only 3 – Normal 4 – Severe 5 – Critical

1.36 set_suction_group

Sets specific information for a particular suction group. See read_suction_group for details on tag definitions.

One or all of the elements may be used when setting parameters in the unit.

Command:

```
<cmd action="set_suction_group" rack_id="int" suction_id="int" user="string" password =  
"string" auth = "string" acct = "string" units="token">  
  <suction_target>signed decimal</suction_target>  
  <suction_cutout>signed decimal</suction_cutout>  
  <suction_control>  
    <value>int</value>  
    <temp_target>signed decimal</target>  
    <temp_range>signed decimal</range>  
    <max_pressure_float>signed decimal</max_pressure_float>  
    <post_defrost_delay>int</post_defrost_delay>  
  </suction_control>  
  <auto_schedule>int</auto_schedule>  
  <almhi_limit>  
    <value>signed decimal</value>  
    <routingAction>int</routingAction>  
    <action_index >int</action_index >  
  </almhi_limit>  
  <almhi_dur>  
    <value>int</value>  
    <units_index>int</units_index >  
  </almhi_dur>  
  <almlo_limit>  
    <value>signed decimal</value>  
    <routingAction>int</routingAction>  
    <action_index >int</action_index >  
  </almlo_limit>  
  <almlo_dur>  
    <value>int</value>  
    <units_index>int</units_index >  
  </almlo_dur>  
</cmd>
```

Response:

```
<resp action="set_suction_group" echo of command attributes... error="int">  
  <name error="int">string</name>  
  ...  
  ...  
</resp>
```

1.37 read_circuit

Reads specific information regarding a circuit from a particular suction group.

Command:

```
<cmd action="read_circuit" rack_id="int" suction_id="int" circuit_id="int"/>
```

Response:

```
<resp action="read_circuit" echo of command attributes... error="int">
  <name fix_type="string" fix_type_index="int" name_index="int">string</name>
  <status status_id="int">string</status>
  <values count="int">
    <value name="string" units="string" units_index="int">signed decimal</value>
    <value name="string" units="string" units_index="int">signed decimal</value>
    ...
  </values>
  <temp_target units="string" units_index="int">signed decimal</temp_target>
  <temp_range units="string" units_index="int">signed decimal</temp_range>
  <temp_control index="int">string</temp_control >
  <defrosts type="string" type_index="int">
    <term type="string" type_index="int">
      <value units="string" units_index="int">signed decimal</value>
    </term>
    <drip_delay>int</drip_delay>
    <min_defrost use_min="int">int</min_defrost>
    <defrost_dur>int</defrost_dur>
    <defrost_loop num_defrosts="int">
      <defrost num="int">
        <hour>int</hour>
        <minute>int</minute>
      </defrost>
      <defrost num="int">
        ...
      </defrost>
    </defrost_loop>
  </defrosts>
  <alarms>
    <alarm sensor_num="int">
      <almhi_limit action="string" action_index="int" routingAction="int" units="string"
units_index="int">signed decimal</almhi_limit>
      <almhi_dur units="string" units_index="int">int</almhi_dur>
      <almlo_limit action="string" action_index="int" routingAction="int" units="sting"
units_index="int">signed decimal</almlo_limit>
      <almlo_dur units="string" units_index="int">int</almlo_dur>
      <almhi_limit_dual action="string" action_index="int" routingAction="int" units="string"
units_index="int">signed decimal</almhi_limit_dual >
      <almhi_dur_dual units="string" units_index="int">int</almhi_dur_dual >
      <almlo_limit_dual action="string" action_index="int" routingAction="int" units="sting"
units_index="int">signed decimal</almlo_limit_dual >
      <almlo_dur_dual units="string" units_index="int">int</almlo_dur_dual >
```

```

</alarm>
<alarm sensor_num="int">
...
...
</alarm>
</alarms>
</resp>

```

Response Element	Data Type	Definition																																													
name	<i>string</i>	Describes name of the circuit. Contains: fix_type fix_type_index Box 0 Multi deck 1 Single deck 2 Reach-In 3 Service 4 Miscellaneous 5 Contains name_index (see description)																																													
man_defrost		Complex type describing if circuit to be placed in manual/auto defrost. Contains 3 attributes: defrost, max_time, allow_term																																													
defrost	<i>int</i>	This is a mandatory attribute for the man_defrost 0 – Auto 1 – Man On 2 – Man Off																																													
max_time	<i>int</i>	Optional Attribute for man_defrost Time in defrost (0 – 255)																																													
allow_term	<i>int</i>	Optional Attribute for man_defrost 0 – No 1 – Yes																																													
name_index	<i>int</i>	If the case is user defined, or a list of the following based on the fix_type: <table> <tr> <td>Box Type</td><td>Multi</td><td>Single</td></tr> <tr> <td>0 – Meat Box</td><td>14 – MD Cheese</td><td>26 – SD Meat</td></tr> <tr> <td>1 – FzMeat Bx</td><td>15 – MD Juice</td><td>27 – SDFZ Meat</td></tr> <tr> <td>2 – Hold Box</td><td>16 – MD Meat</td><td>28 – SD Fish</td></tr> <tr> <td>3 – Poul Box</td><td>17 – MDFZ Meat</td><td>29 – SDFZ Fish</td></tr> <tr> <td>4 – Fish Box</td><td>18 – MDFZ Food</td><td>30 – SD Icecream</td></tr> <tr> <td>5 – FzFish Bx</td><td>19 – MDFZ Veg</td><td>31 – SD Juice</td></tr> <tr> <td>6 – Dairy Box</td><td>20 – MDFZ Bak</td><td>32 – SDFZ Food</td></tr> <tr> <td>7 – Prod Box</td><td>21 – MDFZ Fish</td><td>33 – SDFZ Veg</td></tr> <tr> <td>8 – Icecream Bx</td><td>22 – Pkg Deli</td><td>34 – SDFZ Bak</td></tr> <tr> <td>9 – FzFood Bx</td><td>23 – Produce</td><td>35 – Is Cheese</td></tr> <tr> <td>10 – Bakery Bx</td><td>24 – MD Dairy</td><td>36 – Is Prod</td></tr> <tr> <td>11 – Meat Prep</td><td>25 – User Def.</td><td>37 – SD Dairy</td></tr> <tr> <td>12 – Prod Prep</td><td></td><td>38 – User Def.</td></tr> <tr> <td>13 – User Def.</td><td></td><td></td></tr> </table>	Box Type	Multi	Single	0 – Meat Box	14 – MD Cheese	26 – SD Meat	1 – FzMeat Bx	15 – MD Juice	27 – SDFZ Meat	2 – Hold Box	16 – MD Meat	28 – SD Fish	3 – Poul Box	17 – MDFZ Meat	29 – SDFZ Fish	4 – Fish Box	18 – MDFZ Food	30 – SD Icecream	5 – FzFish Bx	19 – MDFZ Veg	31 – SD Juice	6 – Dairy Box	20 – MDFZ Bak	32 – SDFZ Food	7 – Prod Box	21 – MDFZ Fish	33 – SDFZ Veg	8 – Icecream Bx	22 – Pkg Deli	34 – SDFZ Bak	9 – FzFood Bx	23 – Produce	35 – Is Cheese	10 – Bakery Bx	24 – MD Dairy	36 – Is Prod	11 – Meat Prep	25 – User Def.	37 – SD Dairy	12 – Prod Prep		38 – User Def.	13 – User Def.		
Box Type	Multi	Single																																													
0 – Meat Box	14 – MD Cheese	26 – SD Meat																																													
1 – FzMeat Bx	15 – MD Juice	27 – SDFZ Meat																																													
2 – Hold Box	16 – MD Meat	28 – SD Fish																																													
3 – Poul Box	17 – MDFZ Meat	29 – SDFZ Fish																																													
4 – Fish Box	18 – MDFZ Food	30 – SD Icecream																																													
5 – FzFish Bx	19 – MDFZ Veg	31 – SD Juice																																													
6 – Dairy Box	20 – MDFZ Bak	32 – SDFZ Food																																													
7 – Prod Box	21 – MDFZ Fish	33 – SDFZ Veg																																													
8 – Icecream Bx	22 – Pkg Deli	34 – SDFZ Bak																																													
9 – FzFood Bx	23 – Produce	35 – Is Cheese																																													
10 – Bakery Bx	24 – MD Dairy	36 – Is Prod																																													
11 – Meat Prep	25 – User Def.	37 – SD Dairy																																													
12 – Prod Prep		38 – User Def.																																													
13 – User Def.																																															

		Reach-In 39 – RI Icecream 40 – RI Juice 41 – RIFZ Food 42 – RIFZ Veg 43 – RIFZ Bak 44 – User Def.	Service 45 – Srv Meat 46 – Srv Fish 47 – Srv Deli 48 – User Def.	Miscellaneous 49 – Prep Food 50 – Salad Bar 51 – Floral 52 – Cakes 53 – Beverage 54 – Produce 55 – User Def.
fix_type	string	Fixture type		
fix_type_index	int	Fixture type index		
status	string	Status of circuit ID Description 0 Thermostat Cutin 1 Thermostat Cutout 2 Drip Down 3 Defrosting 4 Manual Defrost 5 Temp Control, Man On 6 Temp Control, Man Off 7 Shutdown 8 Pump-Out 9 Equalize		
status_id	int	See status for id association		
values		Contains all temperature values for each sensor on the circuit. Contains the count of sensors for this circuit.		
count	int	Count of sensors per circuit.		
value	signed decimal	Value of temperature. Contains name, units and units_index.		
units	string	Units of measurement (psi, bar, degf, degc, etc...)		
units_index	int	Id of units		
temp_target	Signed decimal	Value of target temperature. Holds units and units_index		
temp_range	Signed decimal	+/- Range of target.		
temp_control	string	Temperature control type. Contains index attribute: 0 – None 1 – Solenoid 2 – EEPR		
defrosts		Complex element containing: term, drip_delay, min_defrost, defrost_dur, and defrost_loop. Contains two attributes: type_index type 0 – None 1 – Hot Gas 2 – Time off 3 – Air 4 – Electric		

term		Termination type containing a value element. contains attributes: type_index type 0 – Defrost sensor 1 – Disch air snsr 2 – On/Off Input 3 – None 4 – Hot gas return
type	<i>string</i>	type values (see above descriptions)
type_index	<i>int</i>	See type for details.
drip_delay	<i>int</i>	Drip down delay in seconds.
min_defrost	<i>int</i>	Minimum defrost time in minutes. Contains use_min attribute
use_min	<i>int</i>	Use minimum defrost time: 0 – No 1 – Yes
defrost_dur	<i>int</i>	Defrost duration in minutes.
defrost_loop		Contains all defrosts for this circuit. Has defrost_num attribute and defrost elements
defrost_num	<i>int</i>	Number of defrosts for this circuit.
defrost		A defrost schedule containing num, hour, minute, and units (0 – AM; 1 – PM)
num	<i>int</i>	defrost schedule number (1 – max defrosts)
hour	<i>int</i>	Hour value of defrost (0 – 23)
minute	<i>int</i>	Minute value of defrost (0 – 59)
alarms		Stores all alarms configured for each sensor on the circuit. Contains alarm element.
alarm		An individual alarm for a sensor. Contains all alarm parameters with an attribute sensor_num.
sensor_num	<i>int</i>	The id of the sensor for the alarm information.
almhi_limit	<i>signed decimal</i>	Temperature trip value of high temperature alarm. Contains action, action_index, routingAction, units and units_index.
almhi_dur	<i>int</i>	Delay of high temperature alarm. Contains units="sec min hour" and units_index="1 2 3" 1 – sec 2 – min 3 – hour
almlo_limit	<i>signed decimal</i>	Temperature trip value of low temperature alarm. Contains action, action_index, routingAction, units and units_index.
almlo_dur	<i>int</i>	Delay of low temperature alarm. Contains units="sec min hour" and units_index="1 2 3"
almhi_limit_dual	<i>signed decimal</i>	Used for the dual temp alarms. Temperature trip value of high temperature alarm. Contains action, action_index, routingAction, units and units_index.
almhi_dur_dual	<i>int</i>	Used for the dual temp alarms.

		Delay of high temperature alarm. Contains units="sec min hour" and units_index="1 2 3" 1 – sec 2 – min 3 – hour
almlo_limit_dual	<i>signed decimal</i>	Used for the dual temp alarms. Temperature trip value of low temperature alarm. Contains action, action_index, routingAction, units and units_index.
almlo_dur_dual	<i>int</i>	Used for the dual temp alarms. Delay of low temperature alarm. Contains units="sec min hour" and units_index="1 2 3"
routingAction	<i>int</i>	Routing action. Range is 1 through 15.
action	<i>string</i>	Disabled Log Only Normal Severe Critical
action_index	<i>int</i>	1 – Disabled 2 – Log Only 3 – Normal 4 – Severe 5 – Critical

1.38 set_circuit

Set variables within the particular circuit. See read_circuit for details on elements.

When setting parameters, one or all elements may be changed at one time.

Note:

There are no checks to determine if some settings cannot be changed if certain flags are not set. However, an error flag will be associated with the setting notifying that the attempt was unsuccessful.

Command:

```
<cmd action="set_circuit" rack_id="int" suction_id="int" circuit_id="int" user="string"
password = "string" auth = "string" acct = "string">
  <name name_index="int">
    <value>string</value>
  </name>
  <man_defrost defrost="int" max_time="int" allow_term="int"/>
  <temp_control>int</temp_control>
  <temp_target>signed decimal</temp_target>
  <temp_range>signed decimal</temp_range>
  <defrosts>
    <type_index>int</type_index>
    <term>
      <type_index>int</type_index>
      <value>signed decimal</value>
    </term>
    <drip_delay>int</drip_delay>
    <use_min>int</use_min>
    <min_defrost>int</min_defrost>
    <defrost_dur>int</defrost_dur>
    <num_defrosts>int</num_defrosts>
    <defrost_loop>
      <defrost num="int">
        <hour>int</hour>
        <minute>int</ minute >
      </defrost>
      <defrost num="int">
        ...
      </defrost>
    </defrost_loop>
  </defrosts>
  <alarms>
    <alarm sensor_num="int">
      <almhi_limit>
        <value>signed decimal</value>
        <routingAction>int</routingAction>
        <action_index >int</action_index >
      </almhi_limit>
      <almhi_dur>
        <value>int</value>
      </almhi_dur>
    </alarm>
  </alarms>
</cmd>
```

```
<units_index>int</units_index >
</almhi_dur>
<almlo_limit>
  <value>signed decimal</value>
  <routingAction>int</routingAction>
  <action_index >int</action_index >
</ almlo_limit>
< almlo_dur>
  <value>int</value>
  <units_index>int</units_index >
</ almlo_dur>
</alarm>
<alarm sensor_num="int">
  ...
</alarm>
</alarms>
</cmd>
Response:
<resp action="set_circuit" echo of command attributes... error="int">
  <name error="int">string</name>
  ...
  ...
</resp>
```

1.39read_condenser

Reads specific information regarding a condenser for a particular rack.

Command:

```
<cmd action="read_condenser" rack_id="int"/>
```

Response:

```
<resp action="read_condenser" echo of command attributes... error="int">
  <name>string</name>
  <value units="string" units_index="int">signed decimal</value>
  <cond_type index="int">string</cond_type>
  <fans type="string" type_index="int" num_fans="int"/>
  <control_sensor index="int">string</control_sensor>
  <control_type index="int">string</ control_type >
  <control_method type="string" type_index="int">
    <stage_cutin>
      <stage num="int">signed decimal</stage>
      ...
      <stage num="int">signed decimal</stage>
    </stage_cutin>
    <stage_cutout>
      <stage num="int">signed decimal</stage>
      ...
      <stage num="int">signed decimal</stage>
    </stage_cutout>
  </control_method>
  <target units="string" units_index="int">signed decimal</target>
  <min_cond units="string" units_index="int">signed decimal</min_cond>
  <max_cond units="string" units_index="int">signed decimal</max_cond>
  <cond_delta units="string" units_index="int">signed decimal</cond_delta>
  <almhi_limit action="string" action_index="int" routingAction="int" units="string"
units_index="int">signed decimal</almhi_limit>
  <almhi_dur units="string" units_index="int">int</almhi_dur>
  <almlo_limit action="string" action_index="int" routingAction="int" units="sting"
units_index="int">signed decimal</almlo_limit>
  <almlo_dur units="string" units_index="int">int</almlo_dur>
</resp>
```

Response Element	Data Type	Definition
name	string	name of condenser
cond_type	string	condenser type with index: 0 – None 1 – Air cooled 2 – Evaporative 3 – Water Cooled
fans		Descriptions of fans. Contains type, type_index, and num_fans. type_index – type 0 – Single Fan

		1 – Multi. Fan 2 – 2-Speed
num_fans	<i>int</i>	Number of fans. Will only show if using Multi. Fan
control_sensor	<i>string</i>	Controlling sensor. Contains index attribute: 0 – Dropleg Temp 1 – Disch Press 2 – Sat Cond Temp 3 – Avg Cond Temp 4 – Dropleg Press
control_method		Complex element containing type and type_index: 1 – Cut In/Out 2 – Target Also contains either: (stage_cutin, and stage_cutout)
control_type	<i>string</i>	If the control_method is based on target, then this is the target control type which index: 1 – Neutral Zone 3 – Rate of change
stage_cutin		Staging for the fans when using the Cut In/Out method for each fan. Contains stage and num.
stage_cutout		Staging for the fans when using the Cut In/Out method for each fan. Contains stage and num.
stage	<i>signed decimal</i>	Stage value for cutin/cutout.
num	<i>int</i>	Stage number.
target	<i>signed decimal</i>	If not using cutin/cutout, nor Condensing temp then this is the target control temperature/pressure. Contains units and units_index.
units	<i>string</i>	Units of measurement (psi, bar, degf, degc, etc...)
units_index	<i>int</i>	Id of units
min_cond	<i>signed decimal</i>	Minimum condensing temperature. Contains units and units_index. Only used if using the Condenser temp via the control_sensor tag.
max_cond	<i>signed decimal</i>	Maximum condensing temperature. Contains units and units_index. Only used if using the Condenser temp via the control_sensor tag.
cond_delta	<i>signed decimal</i>	Condenser delta temperature. Contains units and units_index. Only used if using the Condenser temp via the control_sensor tag.
alarms		Contains all alarm settings for each sensor.
alarm		Contains the sensor_num if needed. If no sensor_num is supplied, function will assume that the alarm parameters are for ALL sensors associated with this circuit. If sensor_num is supplied, the function will set parameters for just that sensor num. sensor_num -> (1 - # sensors).

almhi_limit	<i>signed decimal</i>	Pressure trip value of high pressure alarm. Contains action, action_index, routingAction, units and units_index.
almhi_dur	<i>int</i>	Delay of high pressure alarm. Contains units="sec min hour" and units_index="1 2 3" 1 – sec 2 – min 3 – hour
almlo_limit	<i>signed decimal</i>	Pressure trip value of low pressure alarm. Contains action, action_index, routingAction, units and units_index.
almlo_dur	<i>int</i>	Delay of low pressure alarm. Contains units="sec min hour" and units_index="1 2 3"
routingAction	<i>int</i>	Routing action. Range is 1 through 15.
action	<i>string</i>	Disabled Log Only Normal Severe Critical
action_index	<i>int</i>	1 – Disabled 2 – Log Only 3 – Normal 4 – Severe 5 – Critical

1.40 set_condenser

Sets information regarding a particular condenser for a specific rack. See read_condenser for definitions of attributes and elements.

Note:

There are no checks to determine if some settings cannot be changed if certain flags are not set. However, an error flag will be associated with the setting notifying that the attempt was unsuccessful.

Command:

```
<cmd action="set_condenser" rack_id="int" user="string" password = "string" auth =
"string" acct = "string">
  <num_fans>int</num_fans>
  <control_sensor>int</control_sensor>
  <control_method>int</control_method>
  <stage_cutin>
    <stage num="int">signed decimal</stage>
    ...
    <stage num="int">signed decimal</stage>
  </stage_cutin>
  <stage_cutout>
    <stage num="int">signed decimal</stage>
    ...
    <stage num="int">signed decimal</stage>
  </stage_cutout>
  <target>signed decimal</target>
  <min_cond>signed decimal</min_cond>
  <max_cond>signed decimal</max_cond>
  <cond_delta>signed decimal</cond_delta>
  <almhi_limit>
    <value>signed decimal</value>
    <routingAction>int</routingAction>
    <action_index >int</action_index >
  </almhi_limit>
  <almhi_dur>
    <value>int</value>
    <units_index>int</units_index >
  </almhi_dur>
  <almlo_limit>
    <value>signed decimal</value>
    <routingAction>int</routingAction>
    <action_index >int</action_index >
  </almlo_limit>
  <almlo_dur>
    <value>int</value>
    <units_index>int</units_index >
  </almlo_dur>
</cmd>
```

Response:

```
<resp action="set_condenser" echo of command attributes... error="int">  
  <num_fans error="int">int</num_fans>  
  ...  
  ...  
</resp>
```

1.41 read_inputs

Read the summary of the digital ON/OFF inputs (node type = 0))

Command

```
<cmd action= "read_inputs"/>
```

Response

```
<resp action="read_inputs" error="int">
  <input>
    <host>int</host>
    <legacy>int</legacy>
    <name>string</name>
    <addr>string</addr>
    <node>int</node>
    <mod>int</mod>
    <point>int</point>
    <units>string</units>
  </input>
  ....
  ....
  ....
  <total_count>int</total_count>
</resp>
```

Response Element	Data Type	Definition
input		Describes an on/off input. Contains elements: host, legacy, name, node, mod, point
host	int	SYSTEM MANAGER unit number 0 through 9. The I/O is attached to this unit.
legacy	int	1 = legacy I/O, 0 = not legacy I/O
name	string	Name of point
addr	string	Address as it appears on the System Manager's display e.g. 01-1.3
node	int	Node address
mod	int	Module number within node.
point	int	Point number within module (if module exists) or node.
units	string	Units of measurement.
total_count	int	Count of input descriptions in the list

Example

```
<cmd action="read_inputs"/>
```

```
<resp action="read_inputs" error="0">
  <input>
    <host>0</host>
```

```
<legacy>0</legacy>  
<name>A1 Defrost</name>  
<addr>03-2.4</addr>  
<node>3</node>  
<mod>2</mod>  
<point>4</point>  
<units>On/Off</units>  
</input>  
<total_count>1</total_count>  
</resp>
```

1.42 read_relays

Read the summary of the digital Relay outputs (node type 1).

Command

```
<cmd action="read_relays"/>
```

Response

```
<resp action="read_relays" error="int">
```

```
<relay>
```

```
<host>int</host>
```

```
<legacy>int</legacy>
```

```
<name>string</name>
```

```
<addr>string</addr>
```

```
<node>int</node>
```

```
<mod>int</mod>
```

```
<point>int</point>
```

```
<units>string</units>
```

```
</relay>
```

```
<total_count>int</total_count>
```

```
</resp>
```

Response Element	Data Type	Definition
Relay		Describes a relay output. Contains elements: host, legacy, name, node, mod, point
Host	<i>int</i>	SYSTEM MANAGER unit number 0 through 9. The I/O is attached to this unit.
Legacy	<i>int</i>	1 = legacy I/O, 0 = not legacy I/O
Name	<i>string</i>	Name of point
Addr	<i>string</i>	node, mod, point as displayed on the System Manager
Node	<i>int</i>	Node address
Mod	<i>int</i>	Module number within node.
Point	<i>int</i>	Point number as displayed on the System Manager offset by 8. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 8+2=10
Units	<i>string</i>	Units of measurement.
total_count	<i>int</i>	Count of relay descriptions in the list

Example

```
<cmd action=" read_relays "/>
```

```
<resp action=" read_relays " error="0">
```

```
<relay>
```

```
<host>0</host>
```

```
<legacy>0</legacy>
<name>Rail Heat 1</name>
<addr>01-3.1</addr>
<node>1</node>
<mod>3</mod>
<point>9</point>
<units>On/Off</units>
</relay>
<relay>
  <host>0</host>
  <legacy>0</legacy>
  <name>Rail Heat 2</name>
  <addr>01-3.2</addr>
  <node>1</node>
  <mod>3</mod>
  <point>10</point>
  <units>On/Off</units>
</relay>
<relay>
  <host>0</host>
  <legacy>0</legacy>
  <name>Misc Relay 1</name>
  <addr>01-1.2</addr>
  <node>1</node>
  <mod>1</mod>
  <point>10</point>
  <units>On/Off</units>
</relay>
<total_count>3</total_count>
</resp>
```

1.43 read_alarm_relays

Read the summary of alarm related relays and their enable inputs.

Command

```
<cmd action= "read_alarm_relays"/>
```

Response

```
<resp action="read_alarm_relays" error="int">
```

```
<count>int</count>
```

```
<alarm_relay>
```

```
<host>int</host>
```

```
<name>string</name>
```

```
<number>int</number>
```

```
<relay_output>
```

```
<addr>string</addr>
```

```
<nodetype>int</nodetype>
```

```
<node>int</node>
```

```
<mod>int</mod>
```

```
<point>int</point>
```

```
<value>int</value>
```

```
</relay_output>
```

```
<enable_input>
```

```
<addr>string</addr>
```

```
<node>int</node>
```

```
<mod>int</mod>
```

```
<point>int</point>
```

```
<value>int</value>
```

```
</enable_input>
```

```
</alarm_relay>
```

```
<alarm_relay>
```

```
<host>int</host>
```

```
<name>string</name>
```

```
<number>int</number>
```

```
<relay_output>
```

```
<addr>string</addr>
```

```
<nodetype>int</nodetype>
```

```
<node>int</node>
```

```
<mod>int</mod>
```

```
<point>int</point>
```

```
<value>int</value>
```

```
</relay_output>
```

```
<enable_input>
```

```
<addr>string</addr>
```

```
<node>int</node>
```

```
<mod>int</mod>
```

```
<point>int</point>
```

```
<value>int</value>
```

```

    </enable_input>
  </alarm_relay>
</resp>

```

Response Element	Data Type	Definition
count	<i>int</i>	The number of alarm relays in the list
alarm_relay		Element that contains all information for one alarm relay
host	<i>int</i>	SYSTEM MANAGER unit number 0 through 9. The relay and enable input if any are attached to this unit.
name	<i>string</i>	Name of the alarm relay
number	<i>int</i>	Identifying number of the alarm relay. Range is 1 through 10
relay_output		Element that contains the relay output's address and value. The elements that it contains are: addr node mod point value
enable_input		Element that contains the address and value of the associated enable input if any. The elements that it contains are: addr node mod point value If there is no enable input this element appears, but is empty.
addr	<i>string</i>	Node,mod,point as displayed on the System Manager
nodetype	<i>int</i>	Node type
node	<i>int</i>	Node address
mod	<i>int</i>	Module number within node.
point	<i>int</i>	Point number as displayed on the System Manager. For the relay output point, which is nodetype 1, the "point" element value is what would be displayed on the System Manager offset by 8. For example if the relay's board point address is displayed as 1-3.2, the "point" XML element has a value of 8+2=10 For the relay enable input, which is nodetype 0, the offset is 0, that is, the value displayed on the System Manager matches the "point" XML element value.
value	<i>string</i>	Value. 0 or 1

1.44read_sensors

Read the summary of the analog sensor inputs (node type = 2))

Command

```
<cmd action= "read_sensors"/>
```

Response

```
<resp action="read_sensors" error="int">
  <sensor>
    <host>int</host>
    <legacy>int</legacy>
    <name>string</name>
    <addr>string</addr>
    <node>int</node>
    <mod>int</mod>
    <point>int</point>
    <units>string</units>
  </sensor>
  ....
  ....
  ....
  <total_count>int</total_count>
</resp>
```

Response Element	Data Type	Definition
sensor		Describes a sensor. Contains elements: host,legacy,name,node,mod,point
host	int	SYSTEM MANAGER unit number 0 through 9. The I/O is attached to this unit.
legacy	int	1 = legacy I/O, 0 = not legacy I/O
name	string	Name of point
addr	string	node,mod,point as displayed on the System Manager
node	int	Node address
mod	int	Module number within node.
point	int	Point number as displayed on the System Manager offset by 16. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 16+2=18.
units	string	Engineering units in displayable form. For example: °F
total count	int	Count of input descriptions in the list

Example

```
<cmd action=" read_sensors "/>
```

```
<resp action=" read_sensors " error="0">
  <sensor>
    <host>0</host>
    <legacy>0</legacy>
    <name>Monitor Tmp1</name>
    <addr>01-1.1</addr>
    <node>1</node>
    <mod>1</mod>
    <point>17</point>
    <units>°F</units>
  </sensor>
  <sensor>
    <host>0</host>
    <legacy>0</legacy>
    <name>Monitor Tmp2</name>
    <addr>03-1.1</addr>
    <node>3</node>
    <mod>1</mod>
    <point>17</point>
    <units>°F</units>
  </sensor>
  ---
  ---
  ---
  <sensor>
    <host>0</host>
    <legacy>0</legacy>
    <name>Misc Sensor 11</name>
    <addr>01-2.1</addr>
    <node>1</node>
    <mod>2</mod>
    <point>17</point>
    <units>°F</units>
  </sensor>
  <sensor>
    <host>0</host>
    <legacy>0</legacy>
    <name>Monitor Tmp3</name>
    <addr>03-1.2</addr>
    <node>3</node>
    <mod>1</mod>
    <point>18</point>
    <units>°F</units>
  </sensor>
  <total_count>16</total_count>
</resp>
```

1.45read_var_outs

Read the summary of the analog variable outputs (node type = 3)

Command

```
<cmd action= "read_var_outs"/>
```

Response

```
<resp action="read_var_outs" error="int">
  <var_output>
    <host>int</host>
    <legacy>int</legacy>
    <name>string</name>
    <addr>string</addr>
    <node>int</node>
    <mod>int</mod>
    <point>int</point>
    <units>string</units>
  </var_output>
  ....
  ....
  <total_count>int</total_count>
</resp>
```

Response Element	Data Type	Definition
var_output		Describes a variable output. Contains elements: host,legacy,name,node,mod,point
host	int	SYSTEM MANAGER unit number 0 through 9. The I/O is attached to this unit.
legacy	int	1 = legacy I/O, 0 = not legacy I/O
name	string	Name of point
addr	string	node,mod,point as displayed on the System Manager
node	int	Node address
mod	int	Module number within node.
point	int	Point number as displayed on the System Manager offset by 24. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 24+2=26
units	string	Engineering units in displayable form. For example: °F
total_count	int	Count of input descriptions in the list

Example

```
<cmd action=" read_var_outs "/>
<resp action=" read_var_outs " error="0">
  <total_count>0</total_count>
</resp>
```

1.46 read_input

Read the value (On or Off) of the digital ON/OFF inputs (node type = 0))

Command

```
<cmd action= "read_input" num_only="int" valid_only="int">
<input node="int" mod=" int" point=" int"/>
<input node=" int" mod=" int" point=" int"/>
...
...
...
<input node=" int" mod=" int" point=" int"/>
</cmd>
```

Command Attribute	Data Type	Definition
num_only	int	If num_only defined and = "1" then display numeric value for enumeration, i.e. On is displayed as 1 and Off is displayed as 0.
valid_only	int	If this attribute is set equal to "1" then A value that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.

Command Element	Data Type	Definition
input		Describes digital input address.

attribute of cmd's input element	Data Type	Definition
node	int	Node address
mod	int	Module within node.
point	int	Point number. Range 1 - 8
name	string	Name of Point (Returned in response ONLY).

Response

```
<resp action=" read_input" error="0">
<input node="int" mod=" int" point=" int" name="string">Off|On|0|1</input>
<input node=" int" mod=" int" point=" int" name="string">Off|On|0|1</input>
...
...
...
<input node=" int" mod=" int" point=" int" name="string">Off|On|0|1</input>
</resp>
```

Example 1

```
<cmd action= "read_input">  
  <input node="63" mod="2" point="5"/>  
  <input node="12" mod="8" point="3"/>  
</cmd>
```

```
<resp action=" read_input" error="0">  
  <input node="63" mod="2" point="5" name="A1 Defrost">Off</input>  
  <input node="12" mod="8" point="3" name=" A2 Defrost">On</input>  
</resp>
```

Example 2

```
<cmd action= "read_input" num_only="1">  
  <input node="63" mod="2" point="5"/>  
  <input node="12" mod="8" point="3"/>  
</cmd>
```

```
<resp action=" read_input" error="0">  
  <input node="63" mod="2" point="5" name="Misc 1 Input">0</input>  
  <input node="12" mod="8" point="3" name=" Misc 2 Input">1</input>  
</resp>
```

1.47 read_relay

Command

```
<cmd action="read_relay" num_only="int" valid_only="1">
<relay node="int" mod="int" point="int"/>
<relay node="int" mod="int" point="int"/>
...
...
...
<relay node="int" mod="int" point="int"/>
</cmd>
```

Command Attribute	Data Type	Definition
num_only	int	If num_only defined and = "1" then display numeric value for enumeration, i.e. On is displayed as 1 and Off is displayed as 0.
valid_only	int	If this attribute is set equal to "1" then A value that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.

Command Element	Data Type	Definition
relay		Describes relay output address.

Attribute of cmd's relay element	Data Type	Definition
node	int	Node address
mod	int	Module within node.
point	int	Point number as displayed on the System Manager offset by 8. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 8+2=10
name	string	Name of Point (Returned in response ONLY).

Response

```
<resp action="read_relay" error="0">
<relay node="int" mod="int" point="int" name="string">Off|On|0|1</relay>
<relay node="int" mod="int" point="int" name="string">Off|On|0|1</relay>
...
...
```

...
<relay node=" int" mod=" int" point=" int" name="string">Off|On|0|1</relay>
</resp>

Example 1

```
<cmd action= "read_relay">  
  <relay node="1" mod="2" point="9"/>  
  <relay node="2" mod="8" point="11"/>  
</cmd>
```

```
<resp action=" read_relay" error="0">  
  <relay node="1" mod="2" point="9" name="Misc 1 Relay">Off</relay>  
  <relay node="2" mod="8" point="11" name="Misc 2 Relay">On</relay>  
</resp>
```

Example 2

```
<cmd action= "read_relay" num_only="1">  
  <relay node="1" mod="2" point="9"/>  
  <relay node="2" mod="8" point="11"/>  
</cmd>
```

```
<resp action=" read_relay" num_only="1" error="0">  
  <relay node="1" mod="2" point="9" name="Misc 1 Relay">0</relay>  
  <relay node="2" mod="8" point="11" name="Misc 2 Relay">1</relay>  
</resp>
```

1.48read_sensor

Read the value of the analog sensor inputs (node type = 2)

Command

```
<cmd action= "read_sensor" units="string" valid_only="1">
< sensor node="int" mod=" int" point=" int"/>
< sensor node=" int" mod=" int" point=" int"/>
...
...
...
< sensor node=" int" mod=" int" point=" int"/>
</cmd>
```

Command attribute	Data Type	Definition
units	string	U or u for US, or S or s for SI format. Default for EMA version is SI, for Rack version is US
valid_only	int	If this attribute is set equal to "1" then: 1. A temperature value that is outside of the valid range appears as the string NaN instead of the most recent good value read. 2. A value that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.

Command element	Data Type	Definition
sensor		Describes address of sensor

Attribute of cmd's sensor element	Data Type	Definition
node	int	Node address
mod	int	Module within node.
point	int	Point number as displayed on the System Manager offset by 16. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 16+2=18
name	string	Name of Point (Returned in response ONLY).

Response

```
<resp units="string" action=" read_sensor" error="0">
<sensor offset="int" node="int" mod="int" point="int" name="string">signed decimal</
sensor >
<sensor offset="int" node="int" mod="int" point="int" name="string">signed decimal</
sensor >
```

...
...
...

```
< sensor offset="int" node="int" mod="int" point="int" name="string">signed
decimal</sensor>
</resp>
```

Resp element	Data Type	Definition
sensor	<i>string</i>	Describes address of sensor, which contains formatted value. Contains following attributes: offset, parval, units, units_index, and attributes sent with cmd element.
offset	<i>signed decimal</i>	Value of offset for this sensor.
parval	<i>signed decimal</i>	Float value of the sensor value.
units	<i>string</i>	Type of units associated with this sensor.
units_index	<i>int</i>	ID of units. (See description in Section 6)

Example 1

```
<cmd action= "read_sensor" units="S">
  < sensor node="1" mod="2" point="17"/>
  < sensor node="2" mod="1" point="18"/>
</cmd>
```

```
<resp units="S" action=" read_sensor" error="0">
  < sensor node="1" offset="1.3" mod="2" point="17" name="Comp 1">0.0 Bar</ sensor >
  < sensor node="2" offset="0.0" mod="1" point="18" name="Case 1">-17.8 °C </ sensor >
</resp>
```

Example 2

```
<cmd action= "read_sensor" units="U">
  < sensor node="1" mod="2" point="17"/>
  < sensor node="2" mod="1" point="18"/>
</cmd>
```

```
<resp units="U" action=" read_sensor" error="0">
  < sensor node="1" offset="1.3" mod="2" point="17" name="Comp 1">0.0 psi</ sensor >
  < sensor node="2" offset="2.1" mod="1" point="18" name="Case 1"> 0.0 °F </ sensor >
</resp>
```

1.49read_var_out

Read the value of the analog variable outputs (node type = 3)

Command

```
<cmd action= "read_var_out" valid_only="1">
< var_output node="int" mod=" int" point=" int"/>
< var_output node=" int" mod=" int" point=" int"/>
...
...
...
< var_output node=" int" mod=" int" point=" int"/>
</cmd>
```

Command attribute	Data Type	Definition
valid_only	int	If this attribute is set equal to "1" then A value that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.

Command element	Data Type	Definition
var_output		Describes address of variable output

Attribute of cmd's var_output element	Data Type	Definition
node	int	Node address
mod	int	Module within node.
point	int	Point number as displayed on the System Manager offset by 24. For example if the board point address is displayed as 1-3.2, the xml "point" has a value of 24+2=26

"units" is not used in command because all the variable outputs in the system are in voltage and return in percentage.

Response

```
<resp action=" read_var_out" error="0">
< var_output node="int" mod=" int" point=" int">signed decimal</ var_output >
< var_output node=" int" mod=" int" point=" int"> signed decimal </ var_output >
...
...
...
< var_output node=" int" mod=" int" point=" int"> signed decimal </ var_output >
</resp>
```

Example

```
<cmd action= "read_var_out" units="S">  
  < var_output node="2" mod="2" point="24"/>  
  < var_output node="3" mod="3" point="25"/>  
</cmd>
```

```
<resp units="S" action=" read_var_out" error="0">  
  < var_output node="2" mod="2" point="24"> 0.0 %</ var_output >  
  < var_output node="3" mod="3" point="25"> 0.0 %</ var_output >  
</resp>
```

1.50read_monitor_summary

Lists names, types, and addresses of all monitor points.

Command

```
<cmd action="read_monitor_summary"/>
```

Response

```
<resp action="read_monitor_summary" ...repeat command attributes...error="int ">
  <unit_name>string</unit_name>
  <software>string</software>
  <monitor>
    <input>
      <name>string</name>
      <type>string</type>      temp, defrost, clean, digital
      <node_type>int</node_type>
      <node>int</node>
      <mod>int</mod>
      <point>int</point>
    </input>
    ...
    <input>
      <name>string</name>
      <type>string</type>      temp, defrost, clean, digital
      <node_type>int</node_type>
      <node>int</node>
      <mod>int</mod>
      <point>int</point>
    </input>
  </monitor>
  ...
  <monitor>
    <input>
      <name>string</name>
      <type>string</type>      temp, defrost, clean, digital
      <node_type>int</node_type>
      <node>int</node>
      <mod>int</mod>
      <point>int</point>
    </input>
    ...
    <input>
      <name>string</name>
      <type>string</type>      temp, defrost, clean, digital
      <node_type>int</node_type>
      <node>int</node>
      <mod>int</mod>
      <point>int</point>
    </input>
  </monitor>
</resp>
```

Response	Data Type	Definition
unit_name	<i>string</i>	Name of unit.
software	<i>string</i>	version and type of software in controller.
monitor		Element tha contains associated analog and digital monitor inputs. Each input is described by an input element
input		Element that contains name, nodetype, node, mod, and point for one monitoring I/O point
name	<i>string</i>	Name
type	<i>string</i>	Type of monitoring input point. Possible values are: Monitoring MoniDefrost MoniClean MoniDigi
nodetype	<i>int</i>	Node type. Values are: 0 = digital input 2 = analog input
node	<i>int</i>	Node address
mod	<i>int</i>	Module within node.
point	<i>int</i>	Point number. Range depends upon nodetype; Nodetype Range 0 (digital input) 1 through 8 2 (analog input) 17 through 23 The System Manager displays analog inputs in the range 1 through 8. The “point” parameter for analog inputs is the point number as would be displayed + an offset of 16

Example

```
<cmd action="read_monitor_summary"/>
```

```
<resp action="read_monitor_summary" error="0">
```

```
  <unit_name/>
```

```
  <software>E02.090</software>
```

```
  <monitor>
```

```
    <input>
```

```
      <name>Temp 2-01</name>
```

```
      <type>Monitoring</type>
```

```
      <nodetype>2</nodetype>
```

```
      <node>10</node>
```

```
      <mod>1</mod>
```

```
      <point>17</point>
```

```
    </input>
```

```
    <input>
```

```
      <name>Defrost 2-01</name>
```

```
      <type>MoniDefrost</type>
```

```
<nodetype>0</nodetype>
<node>11</node>
<mod>1</mod>
<point>1</point>
</input>
<input>
  <name>Clean 2-01</name>
  <type>MoniClean</type>
  <nodetype>0</nodetype>
  <node>11</node>
  <mod>1</mod>
  <point>2</point>
</input>
<input>
  <name>Digital 2-01</name>
  <type>MoniDigi</type>
  <nodetype>0</nodetype>
  <node>11</node>
  <mod>1</mod>
  <point>3</point>
</input>
</monitor>
<monitor>
  <input>
    <name>Temp 4-01</name>
    <type>Monitoring</type>
    <nodetype>2</nodetype>
    <node>12</node>
    <mod>1</mod>
    <point>17</point>
  </input>
</monitor>
</resp>
```

1.51 read_monitor_detail

Read detailed monitoring point information. The input parameters may identify the analog point used as a temperature input, the digital input used to indicate defrost, the digital input used to indicate cleaning, or the general purpose digital input.

Command

```
<cmd action="read_monitor_detail" nodetype="int" node="int" mod="int" point="int"
      valid_only="int"/>
```

OR

```
<cmd action="read_monitor_detail">
  <monitor nodetype="int" node="int" mod="int" point="int" valid_only="int"/>
  <monitor nodetype="int" node="int" mod="int" point="int" valid_only="int"/>
  <monitor nodetype="int" node="int" mod="int" point="int" valid_only="int"/>
  ...
  <monitor nodetype="int" node="int" mod="int" point="int" valid_only="int"/>
</cmd>
```

Command Attribute	Data Type	Definition						
monitor		Complex type if wanting more than one detail to be returned. Is not required, unless more than one detail is to be requested. All other tags must be placed in <monitor> if more than one, else only within the <cmd> tag.						
nodetype	int	Node type. Values are: 0 = digital input 2 = analog input						
node	int	Node address						
mod	int	Module within node.						
point	int	Point number. Range depends upon nodetype; <table><tr><td>Nodetype</td><td>Range</td></tr><tr><td>0 (digital input)</td><td>1 through 8</td></tr><tr><td>2 (analog input)</td><td>17 through 23</td></tr></table> The System Manager displays analog inputs in the range 1 through 8. The “point” parameter for analog inputs is the point number as would be displayed + an offset of 16	Nodetype	Range	0 (digital input)	1 through 8	2 (analog input)	17 through 23
Nodetype	Range							
0 (digital input)	1 through 8							
2 (analog input)	17 through 23							
valid_only	int	If this attribute is set equal to “1” then: 1. A temperature value (temp_value) that is outside of the valid range appears as the string NaN instead of the most recent good value read. 2. A value (temp_value, dig_value, def_value, or clean_value) that is from a controller that is not online is displayed as the character * instead of the last value read when the controller was online.						

Response

```
<resp action="read_monitor_detail" ...repeat command attributes...error="int ">
  <temp_name>string</temp_name>
  <temp_value units="string" statcode="int">temp</temp_value>
  <temp_alm_state>int</temp_alm_state>
  <almhi_limit units="string"> temp</almhi_limit>
  <almhi_dur units="string">int</almhi_dur>
  <almlo_limit units="sting">temp</almlo_limit>
  <almlo_dur units="string">int</almlo_dur>
  <dig_name>string</dig_name>
  <dig_value statcode="int"> 0|1</dig_value>
  <dig_alm_state>int</dig_alm_state>
  <dig_alm_cond>int</dig_alm_cond>
  <dig_alm_dur units="string">int</dig_alm_dur>
  <def_name>string</def_name>
  <def_value statcode="int">0|1</def_value>
  <def_alm_state>int</def_alm_state>
  <def_alm_cond>int</def_alm_cond>
  <def_alm_dur units="string">int</def_alm_dur>
  <def_delay units="string">int</def_delay>
  <clean_name>string</clean_name>
  <clean_value statcode="int"> 0|1</clean_value>
  <clean_delay units="string">int</clean_delay>
</resp>
```

OR

```
<resp action="read_monitor_detail" error="int ">
  <monitor ...repeat command attributes...>
    <temp_name>string</temp_name>
    <temp_value units="string" statcode="int">temp</temp_value>
    <temp_alm_state>int</temp_alm_state>
    <almhi_limit units="string"> temp</almhi_limit>
    <almhi_dur units="string">int</almhi_dur>
    <almlo_limit units="sting">temp</almlo_limit>
    <almlo_dur units="string">int</almlo_dur>
    <dig_name>string</dig_name>
    <dig_value statcode="int"> 0|1</dig_value>
    <dig_alm_state>int</dig_alm_state>
    <dig_alm_cond>int</dig_alm_cond>
    <dig_alm_dur units="string">int</dig_alm_dur>
    <def_name>string</def_name>
    <def_value statcode="int">0|1</def_value>
    <def_alm_state>int</def_alm_state>
    <def_alm_cond>int</def_alm_cond>
    <def_alm_dur units="string">int</def_alm_dur>
    <def_delay units="string">int</def_delay>
    <clean_name>string</clean_name>
    <clean_value statcode="int"> 0|1</clean_value>
    <clean_delay units="string">int</clean_delay>
  </monitor>
```

```

<monitor ...repeat command attributes...>
...
...
</monitor>
...
...
</resp>

```

Response Element	Data Type	Definition
temp_name	<i>string</i>	The name of the temperature analog input.
temp_value	<i>temp</i>	The analog monitor point value This element has two attributes: “units”, and “statcode” The “units” attribute indicates engineering units as follows: “degf” = degrees Fahrenheit “degc” = degrees Celsius The “statcode” attribute provides the points status: 1 = offline 2 = online 3 = startup
temp_alm_state	<i>int</i>	1 = temperature input point is in alarm 0 = temperature input point is not in alarm
almhi_limit	<i>temp</i>	High temperature alarm limit. This element has a “units” attribute that indicates engineering units as follows: “degf” = degrees Fahrenheit “degc” = degrees Celsius
almhi_dur	<i>int</i>	High temperature alarm duration. A high temperature alarm condition exists if the temperature is above almhi for the duration almhi_dur . This elements has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes “hr” = hours”
almlo_limit	<i>temp</i>	Low temperature alarm limit. This element has a “units” attribute that indicates engineering units as follows: “degf” = degrees Fahrenheit “degc” = degrees Celsius
almlo_dur	<i>int</i>	Low temperature alarm duration. A low temperature alarm condition exists if the temperature is below almlo for the duration almlo_dur .

		This element has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes “hr” = hours”
dig_value		Digital input. Values are: 0 = digital input is off 1 = digital input is on This element has a “statcode” attribute which provides the points status: 1 = offline 2 = online 3 = startup
dig_alm_state	<i>int</i>	1 = digital input is in alarm 0 = digital input is not in alarm
dig_alm_cond	<i>int</i>	Digital input alarm condition 0 = alarm if digital input is off 1 = alarm if digital input is on
dig_alm_dur	<i>int</i>	Digital input alarm duration. A digital input alarm condition exists if the digital input is equal to almdig for the duration almdig_dur. This element has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes “hr” = hours”
def_name	<i>string</i>	The name of the digital input assigned to indicate defrost.
def_value	<i>int</i>	Defrost indication. Values are: 0 = defrost is off 1 = defrost is on This element has a “statcode” attribute which provides the points status: 1 = offline 2 = online 3 = startup
def_alm_state	<i>int</i>	1 = defrost input is in alarm 0 = defrost input is not in alarm
def_alm_cond	<i>int</i>	Alarm defrost condition 0 = alarm if defrost input is off 1 = alarm if defrost input is on
def_alm_dur	<i>int</i>	Defrost input alarm duration. A defrost input alarm condition exists if the defrost input is equal to almdef for the duration almdef_dur. This element has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes

		“hr” = hours”
def_delay	<i>int</i>	Defrost delay in seconds, minutes, or hours This element has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes “hr” = hours”
clean_name	<i>string</i>	Name if the digital input assigned as “clean” input.
clean_value		Clean indication. Values are: 0 = clean input is off 1 = clean input is on This element has a “statcode” attribute which provides the points status: 1 = offline 2 = online 3 = startup
clean_delay	<i>int</i>	Post clean delay This element has a “units” attribute that may have the following values: “sec” = seconds “min” = minutes “hr” = hours”

1.52set_monitor_point

Set the monitoring point value or alarm parameters. (NOTE: The digital alarm condition and defrost alarm condition cannot be changed due to the nature of the alarm. These alarm conditions are part of the alarm, and cannot be changed within the alarm.)

Command

```
<cmd action="set_monitor_point" user=" string" password=" string" auth=" string"
  acct=" string" node=" int" mod="int" point="int" units="string">
  <delay_units>int</delay_units>
  <temp_name>string</temp_name>
  <almhi_limit > temp</almhi_limit>
  <almhi_dur>int</almhi_dur>
  <almlo_limit>temp</almlo_limit>
  <almlo_dur>int</almlo_dur>
  <dig_alm_cond>int</dig_alm_cond>
  <dig_alm_dur>int</dig_alm_dur>
  <def_alm_cond>int</def_alm_cond>
  <def_alm_dur>int</def_alm_dur>
  <def_delay>int</def_delay>
  <clean_delay>int</clean_delay>
</cmd>
```

Cmd Element	Data Type	Definition				
nodetype (For Reference Only)	<i>int</i>	Nodetype must always be 2 for the temperature sensor input. No need to supply.				
node	<i>int</i>	Node address				
mod	<i>int</i>	Module within node.				
point	<i>int</i>	Point number. Range depends upon nodetype; <table><tr><th>Nodetype</th><th>Range</th></tr><tr><td>2 (analog input)</td><td>17 through 23</td></tr></table> The System Manager displays analog inputs in the range 1 through 8. The “point” parameter for analog inputs is the point number as would be displayed + an offset of 16	Nodetype	Range	2 (analog input)	17 through 23
Nodetype	Range					
2 (analog input)	17 through 23					
units	<i>string</i>	u U for US or s S for SI (default is set by controller if not used)				
delay_units	<i>int</i>	Element that describes the unit of time associated with alarm duration delays. 0 = seconds 1 = minutes 2 = hours Default for this 1				
temp_name	<i>string</i>	The name of this monitoring point.				
almhi_limit	<i>temp</i>	High temperature alarm limit.				
almhi_dur	<i>int</i>	High temperature alarm delay.				
almlo_limit	<i>temp</i>	Low temperature alarm limit.				
almlo_dur	<i>int</i>	Low temperature alarm delay.				

dig_name	<i>string</i>	The name of the digital input for this monitoring point.
dig_alm_cond	<i>int</i>	Digital input alarm condition 0 = alarm if digital input is off 1 = alarm if digital input is on
dig_alm_dur	<i>int</i>	Digital input alarm delay.
def_name	<i>string</i>	The name of the defrost input for this monitoring point.
def_alm_cond	<i>int</i>	Alarm defrost condition 0 = alarm if defrost input is off 1 = alarm if defrost input is on
def_alm_dur	<i>int</i>	Defrost input alarm delay.
def_delay	<i>int</i>	Defrost delay in minutes. Range is 0 – 240.
clean_name	<i>string</i>	Name of the clean input for this monitoring point.
clean_delay	<i>int</i>	Post clean delay in minutes. Range is 5 – 240.

1.53 read_history

Command

```
<cmd action="read_history" nodetype="int" node="int" cid="int" vid="int" tag="string"
  pnum="int" hist_index="int" mod="int" point="int" sample_rate="int" units="string">
  <starttime>
    <year>int</year>
    <month>int</month>
    <day>int</day>
    <hour>int</hour>
    <minute>int</minute>
    <second>int</second>
  </starttime>
</cmd>
```

command attribute	Data Type	Definition
nodetype	int	Node type of device that contains requested values
node	int	Node address of device that contains requested values
tag	string	XML tag that identifies the requested value. Tags are defined in the edf file associated with device in XMLNAME_SECTION Not used if cid and vid are used
cid	int	Component ID - not used if tag is used
vid	int	Variable ID - not used if tag is used
pnum	int	System Manager database parameter number.
hist_index	int	History local datapoint number. One based, that is, range starts with 1.
mod	int	module within node
point	int	Point within module Ranges are: 1 – 8 for digital inputs (nodetype 0) 9 – 16 for digital outputs (nodetype 1) 17 – 24 for analog inputs (nodetype 2) 25 – 28 for analog outputs (nodetype 3)
sample_rate	int	Requested sample rate in seconds. Even though any int value may be entered, it will be converted to only a few values as follows: If integer is: then value is: > 0 and <= 5 5 seconds >5 and <= 30 30 seconds >30 and <= 60 1 minute >60 and <= 120 2 minutes >120 and <= 600 10 minutes >600 and <= 1800 30 minutes <= 0 or >1800 1 hour

		When this attribute is omitted, the sample_rate defaults to 1 hour.
units	<i>string</i>	Units – “U”, “u”, “S”, or “s”

command element	Data Type	Definition
starttime		Describes starting time for returned history data. Contains elements year,month,day,hour,minute,second
year	<i>int</i>	Two digit year
month	<i>int</i>	Month
day	<i>int</i>	Day of month
hour	<i>int</i>	Hour of day 0 through 23
minute	<i>int</i>	Minute
second	<i>int</i>	Second

Response

<resp action="read_history" ...echo the command attributes... actual_sample_rate="int"
error=" int ">

<starttime>

<year> *int* </year>

<month> *int* </month>

<day> *int* </day>

<hour> *int* </hour>

<minute> *int* </minute>

<second>*int*</second>

<epoch> *uint* </epoch>

</starttime>

<display>*string*</display> (appears only when using “tag” command attribute)

<name>*string*</name>

<data>

<y>*signed decimal* | *string*</y>

<y> *signed decimal* | *string* </y>

<y> *signed decimal* | *string*</y>

.....

.....

.....

<y> *signed decimal* | *string*</y>

</data>

<unit>*string*</unit>

<digital> *int* </digital>

<timezone> *int* </timezone>

<daylightsavings> *int* </daylightsavings>

<total_number> *int* </total_number>

<stoptime>

<year> *int* </year>

<month> *int* </month>

<day> *int* </day>

```

<hour> int </hour>
<minute> int </minute>
<second> int </second>
<epoch> uint </epoch>
</stoptime>
</resp>

```

response attribute	Data Type	Definition
actual_sample_rate	int	Interval in seconds used for actual recording in the 255

response element	Data Type	Definition
starttime		Describes starting time for returned history data. Contains elements year,month,day,hour,minute,second,epoch
year	int	Two digit year
month	int	Month
day	int	Day of month
hour	int	Hour of day 0 through 23
minute	int	Minute
second	int	Second
epoch	uint	Date and time expresses as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.
display	string	XML name defined in XMLNAME_SECTION of edf file Appears only when “tag” command attribute is used.
name	string	Parameter name defined in PARAMETER_SECTION of edf file
data		Data values. Contains “y” elements, one for each data value.
y	signed decimal	A single data value or “----” if value undefined
unit	string	Unit that applies to all data values. e.g. °C or °F
digital	int	1 = data values are digital, 0 = data values are either analog or are invalid
timezone	int	UTC time zone offset expressed as 100*hours. e.g. Eastern US time has an offset of -500
daylightsavings	int	0 = daylight savings not in effect 1 = daylight savings is in effect
total_number	int	Count of data values returned
stoptime		Describes ending time for returned history data. Contains elements year,month,day,hour,minute,second,epoch

Example 1

```
<cmd action="read_history" nodetype="16" node="5" cid="184" vid="3"
sample_rate="30"><starttime><year>07</year><month>9</month><day>17</day><hour>1
7</hour><minute>3</minute><second/></starttime></cmd>
<resp cid="184" sample_rate="30" nodetype="16" vid="3" action="read_history" node="5"
real_sample_rate="30" error="0">
  <starttime>
    <year>07</year>
    <month>9</month>
    <day>17</day>
    <hour>17</hour>
    <minute>3</minute>
    <second/>
    <epoch>1190048580</epoch>
  </starttime>
  <name>Control temp A</name>
  <data>
    <y>-----</y>
    <y>-200.0</y>
    <y>-200.0</y>
    <y>*****</y>
    <y>*****</y>      ("*****" indicated undefined at this time, powered down etc)
    <y>*****</y>
    <y>*****</y>
    <y>-200.0</y>
    <y>-200.0</y>
    .
    .
    .
    <y>-200.0</y>
    <y>-200.0</y>
  </data>
  <unit>°C </unit>
  <digital>0</digital>
  <timezone>-500</timezone>
  <daylightsavings>1</daylightsavings>
  <total_number>73</total_number>
  <stoptime>
    <year>7</year>
    <month>9</month>
    <day>17</day>
    <hour>17</hour>
    <minute>39</minute>
    <second>0</second>
    <epoch>1190050740</epoch>
  </stoptime>
</resp>
```

Example 2

```
<cmd action="read_history" nodetype="16" node="5" cid="184" vid="2"
sample_rate="30"><starttime><year>07</year><month>9</month><day>17</day><hour>1
7</hour><minute>3</minute><second/></starttime></cmd>
<resp cid="184" sample_rate="30" nodetype="16" vid="2" action="read_history" node="5"
real_sample_rate="30" error="0">
  <starttime>
    <year>07</year>
    <month>9</month>
    <day>17</day>
    <hour>17</hour>
    <minute>3</minute>
    <second/>
    <epoch>1190048580</epoch>
  </starttime>
  <name>Case Status A</name>
  <data>
    <y>-----</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
    <y>*****</y>
    <y>*****</y>
    .
    .
    .
    <y>*****</y>
    <y>*****</y>
    <y>*****</y>
    <y>*****</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
    <y>0</y>
  </data>
  <unit/>
  <digital>0</digital>
  <timezone>-500</timezone>
  <daylightsavings>1</daylightsavings>
  <total_number>91</total_number>
  <stoptime>
    <year>7</year>
    <month>9</month>
    <day>17</day>
    <hour>17</hour>
    <minute>48</minute>
    <second>0</second>
```

```
<epoch>1190051280</epoch>
</stoptime>
</resp>
```

Example 3

```
<cmd action="read_history" nodetype="16" node="5" tag="evapintemp1"
sample_rate="30"><starttime><year>07</year><month>9</month><day>19</day><hour>1
1</hour><minute>3</minute></starttime></cmd>
<resp node="5" tag="evapintemp1" action="read_history" sample_rate="30" nodetype="16"
real_sample_rate="30" error="0">
  <starttime>
    <year>07</year>
    <month>9</month>
    <day>19</day>
    <hour>11</hour>
    <minute>3</minute>
    <epoch>1190203165</epoch>
  </starttime>
  <display>Air In</display>
  <name>Control temp A</name>
  <data>
    <y>-200.0</y>
    <y>-200.0</y>
    <y>-200.0</y>
    .
    .
    .
    <y>-200.0</y>
    <y>-200.0</y>
    <y>-200.0</y>
    <y>-200.0</y>
  </data>
  <unit>°C</unit>
  <digital>0</digital>
  <timezone>-500</timezone>
  <daylightsavings>1</daylightsavings>
  <total_number>101</total_number>
  <stoptime>
    <year>7</year>
    <month>9</month>
    <day>19</day>
    <hour>12</hour>
    <minute>49</minute>
    <second>25</second>
    <epoch>1190206165</epoch>
  </stoptime>
</resp>
```

Example 4

```
<cmd action="read_history" nodetype="2" node="1" mod="1" point="17" sample_rate="5"
units="S"><starttime><year>08</year><month>05</month><day>01</day><hour>0</hour>
<minute>0</minute><second>0</second></starttime></cmd>
```

```
sample_rate="5" action="read_history" real_sample_rate="5" error="0">
<starttime>
  <year>08</year>
  <month>05</month>
  <day>01</day>
  <hour>0</hour>
  <minute>0</minute>
  <second>0</second>
  <epoch>1209600000</epoch>
</starttime>
<name>Monitor Tmp1</name>
<data>
  <y>25.5</y>
  <y>25.5</y>
  <y>25.5</y>
  <y>25.2</y>
  <y>25.2</y>
  <y>25.2</y>
  ---
  ---
  ---
  <y>25.2</y>
  <y>25.2</y>
  <y>25.2</y>
  <y>25.5</y>
</data>
<unit>°C</unit>
<digital>0</digital>
<timezone>-500</timezone>
<daylightsavings>1</daylightsavings>
<total_number>169</total_number>
<stoptime>
  <year>8</year>
  <month>5</month>
  <day>1</day>
  <hour>0</hour>
  <minute>14</minute>
  <second>0</second>
  <epoch>1209600840</epoch>
</stoptime>
</resp>
```

1.54read_history_cfg

This command returns history configurations of the history data point. Can use the start_point, or can use the nodetype, node with a combination of (cid, vid), (mod, point), (tag), (pnun), or (hist_index). This will only return the sample_rate for the requested point.

Command

```
<cmd action="read_history_cfg" start_point="0"/>
```

command attribute	Data Type	Definition										
start_point	int	The starting point in the 600 history data points. The maximum returned number of points is 100 Range 0 through 599. If not specified then start_point defaults to zero.										
nodetype	int	Node type of device that contains requested values										
node	int	Node address of device that contains requested values										
tag	string	XML tag that identifies the requested value. Tags are defined in the device's edf file in XMLNAME_SECTION Not used if cid and vid or mod and point or hist_index are used										
cid	int	Component ID - not used if tag or mod/point or hist_index is used										
vid	int	Variable ID - not used if tag or mod/point or hist_index is used										
mod	int	Module within node - not used if tag or cid/vid or hist_index is used.										
point	int	Point within module node - not used if tag or cid/vid or hist_index is used. Range depends upon nodetype; <table><tr><td>Nodetype</td><td>Range</td></tr><tr><td>0 (digital input)</td><td>1 through 8</td></tr><tr><td>1 (digital output)</td><td>9 through 16 (offset of 8 + point as displayed)</td></tr><tr><td>2 (analog input)</td><td>17 through 24 (offset of 16 + point as displayed)</td></tr><tr><td>3 (analog output)</td><td>25 through 28 (offset of 24 + point as displayed)</td></tr></table>	Nodetype	Range	0 (digital input)	1 through 8	1 (digital output)	9 through 16 (offset of 8 + point as displayed)	2 (analog input)	17 through 24 (offset of 16 + point as displayed)	3 (analog output)	25 through 28 (offset of 24 + point as displayed)
Nodetype	Range											
0 (digital input)	1 through 8											
1 (digital output)	9 through 16 (offset of 8 + point as displayed)											
2 (analog input)	17 through 24 (offset of 16 + point as displayed)											
3 (analog output)	25 through 28 (offset of 24 + point as displayed)											
pnum	int	System Manager database parameter number										
hist_index	int	History local datapoint number. One based, that is, range starts with 1. Not used if tag or cid/vid or mod/point is used.										

Response

```
<resp action="read_history_cfg" ...echo command attributes ...error="int">
```

```
<history_cfg>
```

```
<devicename>string</devicename> (appears only if device is generic)
```

```

<name>string</name>
<pnum>int</pnum>
<nodetype> int </nodetype>
<node> int </node>
<mod> int </mod>
<point> int </point>
<cid> int </cid>          (appears only if device is generic)
<vid> int </vid>          (appears only if device is generic)
<hist_index> int </hist_index>
</history_cfg>
....
....
....
<history_cfg>
  <devicename>string</devicename>      (appears only if device is generic)
  <name>string</name>
  <pnum>int</pnum>
  <nodetype> int </nodetype>
  <node> int </node>
  <mod> int </mod>
  <point> int </point>
  <cid> int </cid>          (appears only if device is generic)
  <vid> int </vid>          (appears only if device is generic)
  <hist_index> int </hist_index>
</history_cfg>
<total_list> int </total_list>
<total_points> int </total_points>
<unit_addr>int</unit_addr>
<unit_name>string</unit_name>
<timezone>int</timezone>
<daylightsavings>int</daylightsavings>
<current_secs>uint</current_secs>
</resp>

```

response element	Data Type	Definition
history_cfg		Describes history configuration for a single point Contains elements: name, devicename, nodetype, node, mod, point, cid, vid, hist_index
devicename	<i>string</i>	Name of device
name	<i>string</i>	Name of the point
pnum	<i>int</i>	System Manager database parameter number
nodetype	<i>int</i>	Node type of the device that contains the point
node	<i>int</i>	Node address of the device that contains the point
mod	<i>int</i>	Module number
point	<i>int</i>	Point number within module or node Range depends upon nodetype; Nodetype Range

		0 (digital input) 1 through 8 1 (digital output) 9 through 16 (offset of 8 + point as displayed) 2 (analog input) 17 through 24 (offset of 16 + point as displayed) 3 (analog output) 25 through 28 (offset of 24 + point as displayed)
cid	int	Component ID appears only if device is generic
vid	int	Variable ID appears only if device is generic
hist_index	int	Index within 600 possible history configured points. Range 1 through 600
units	string	(degf, degc, psi, bar, etc...) Section 6
unittype	int	See defined value in Section 6
actual_sample_rate	int	The rate at which the samples are actually recorded in the System Manager's history files.
total_list	int	Number of point configurations displayed in this result
total_points	int	Total number of history configured points
unit_addr	int	System Manager unit number, range 0 through 9
unit_name	string	System Manager unit name
timezone	int	UTC time zone offset expressed as 100*hours. e.g. Eastern US time has an offset of -500
daylightsavings	int	0 = daylight savings not active, 1 = daylight savings active
current_secs	uint	Current date and time expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.

Example

```

<cmd action="read_history_cfg" start_point="0"></cmd>
<resp start_point="0" action="read_history_cfg" error="0">
  <history_cfg>
    <devicename>Device P1-01a</devicename>
    <name>5: Main switch</name>
    <nodetype>16</nodetype>
    <node>5</node>
    <mod>0</mod>
    <point>0</point>
    <cid>78</cid>
    <vid>8</vid>
    <hist_index>1</hist_index>
  </history_cfg>
  <history_cfg>
    <devicename>Device P1-01a</devicename>
    <name>5: AK2 error</name>
    <nodetype>16</nodetype>
    <node>5</node>
    <mod>0</mod>
    <point>0</point>
  </history_cfg>

```

```
<cid>2</cid>
<vid>8</vid>
<hist_index>2</hist_index>
</history_cfg>
.....
.....
.....
<history_cfg>
  <name>Misc Sensor 1</name>
  <nodetype>2</nodetype>
  <node>1</node>
  <mod>1</mod>
  <point>17</point>
  <hist_index>17</hist_index>
</history_cfg>
<total_list>17</total_list>
<total_points>17</total_points>
<unit_addr>0</unit_addr>
<unit_name/>
<timezone>-500</timezone>
<daylightsavings>1</daylightsavings>
<current_secs>1190283347</current_secs>
</resp>
```

1.55read_device_history_cfg

This command will return all of the history configurations for a specific generic device.

Command

```
<cmd action="read_device_history_cfg" nodetype="16" node="int"/>
```

command attribute	Data Type	Definition
nodetype	int	Node type of device. Always set to 16
node	int	Node address of device

Response

```
<resp action="read_device_history_cfg" ...echo command attributes... error="0">
  <history_cfg>
    <name>string</name>
    <cid> int </cid>
    <vid> int </vid>
    <hist_index> int </hist_index>
  </history_cfg>
  <history_cfg>
    <name> string </name>
    <cid> int </cid>
    <vid> int </vid>
    <hist_index> int </hist_index>
  </history_cfg>
  <timezone> int </timezone>
  <daylightsavings> int </daylightsavings>
  <current_secs> uint </current_secs>
  <total_points> int </total_points>
</resp>
```

response element	Data Type	Definition
history_cfg		Describes history configuration for a single point Contains elements: name, cid, vid, hist_index
name	string	Name of the point
cid	int	Component ID
vid	int	Variable ID
hist_index	int	Index within 600 possible history configured points. Range 1 through 600
timezone	int	UTC time zone offset expressed as 100*hours. e.g. Eastern US time has an offset of -500
daylightsavings	int	0 = daylight savings not active, 1 = daylight savings active
current_secs	uint	Current date and time expresses as epoch time.

		Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.
total_points	int	Total number of history configured points for this device (also number that appears in the list)

Example

```

<cmd action="read_device_history_cfg" nodetype="16" node="5"/>
<resp nodetype="16" action="read_device_history_cfg" node="5" error="0">
  <history_cfg>
    <name>5: Main switch</name>
    <cid>78</cid>
    <vid>8</vid>
    <hist_index>1</hist_index>
  </history_cfg>
  <history_cfg>
    <name>5: AK2 error</name>
    <cid>2</cid>
    <vid>8</vid>
    <hist_index>2</hist_index>
  </history_cfg>
  ....
  ....
  ....
  <history_cfg>
    <name>5: Case Status A</name>
    <cid>184</cid>
    <vid>2</vid>
    <hist_index>3</hist_index>
  </history_cfg>
  <timezone>-500</timezone>
  <daylightsavings>1</daylightsavings>
  <current_secs>1190049162</current_secs>
  <total_points>16</total_points>
</resp>

```

1.56 write_digi_op

This command writes the operation mode to on/off inputs and relay outputs.
Must be authorized as supervisor.

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="write_digi_op" auth="string" acct="string" user="string" password="string">
  <relays>
    <relay node="int" mod="int" point="int" op_mode="int"/>
    <relay node="int" mod="int" point="int" op_mode="int"/>
    ....
    ....
    <relay node="int" mod="int" point="int" op_mode="int"/>
  </relays>
  <inputs>
    <input node="int" mod="int" point="int" op_mode="int"/>
    <input node="int" mod="int" point="int" op_mode="int"/>
    ....
    ....
    <input node="int" mod="int" point="int" op_mode="int"/>
  </inputs>
</cmd>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS, but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored

command element or attribute	Data Type	Definition
relays		Contains a list of the relay elements.

relay		Addresses and describes how to change the operation status of a single relay output. Has the attributes <code>node</code> , <code>mod</code> , <code>point</code> , and <code>op_mode</code>										
node (relay attr)	<i>int</i>	node address of device that contains relay output										
mod (relay attr)	<i>int</i>	module number of the relay output										
point (relay attr)	<i>int</i>	point number of the relay output										
op_mode (relay attr)	<i>int</i>	Value to which to set the relay output's operation mode. May have the following values: <table><tr><td>Value</td><td>Definition</td></tr><tr><td>0</td><td>Auto On</td></tr><tr><td>1</td><td>Auto Off</td></tr><tr><td>2</td><td>Manual On</td></tr><tr><td>3</td><td>Manual Off</td></tr></table>	Value	Definition	0	Auto On	1	Auto Off	2	Manual On	3	Manual Off
Value	Definition											
0	Auto On											
1	Auto Off											
2	Manual On											
3	Manual Off											
inputs		Contains a list of the on/off input elements.										
input		Addresses and describes how to change the operation status of a single on/off input. Has the attributes <code>node</code> , <code>mod</code> , <code>point</code> , and <code>op_mode</code>										
node (input attr)	<i>int</i>	node address of device that contains on/off input										
mod (input attr)	<i>int</i>	module number of the on/off input										
point (input attr)	<i>int</i>	point number of the on/off input This is actually 8+ the point number as displayed on the 255. For example, the point value to be sent for the relay out whose address is displayed as 1.1.2 is 10.										
op_mode (input attr)	<i>int</i>	Value to which to set the on/off input's operation mode. May have the following values: <table><tr><td>Value</td><td>Definition</td></tr><tr><td>0</td><td>Auto On</td></tr><tr><td>1</td><td>Auto Off</td></tr><tr><td>2</td><td>Manual On</td></tr><tr><td>3</td><td>Manual Off</td></tr></table>	Value	Definition	0	Auto On	1	Auto Off	2	Manual On	3	Manual Off
Value	Definition											
0	Auto On											
1	Auto Off											
2	Manual On											
3	Manual Off											

Response

```

<resp action="write_digi_op" ...echo the command attributes... error="int">
  <relays>
    <relay op_mode=" int" mod=" int" point=" int" node=" int" error="int"/> (error case)
    <relay op_mode=" int" mod=" int" point=" int" node=" int">string </relay>
  ....
  ....
    <relay op_mode=" int" mod=" int" point=" int" node=" int">string </relay>
  </relays>
  <inputs>
    <input op_mode=" int" mod=" int" point=" int" node=" int" error="int"/> (error case)
    <input op_mode=" int" mod=" int" point=" int" node=" int">string</input>
  ....
  ....
    <input op_mode=" int" mod=" int" point=" int" node=" int">string</input>
  </inputs>
</resp>

```

response element or attribute	Data Type	Definition
relays		Contains a list of the relay elements.
relay	<i>string</i>	Shows the address and describes how the operation status of a single relay output has been changed, or has not been changed due to an error. Its text value is the operation status of the relay and may be: Auto-On Auto_Off Manual On Manual Off This element has the attributes: node, mod, point, and op_mode.
node (relay attr)	<i>int</i>	node address of device that contains relay output
mod (relay attr)	<i>int</i>	module number of the relay output
point (relay attr)	<i>int</i>	Point number of the relay output Range is 9 through 16 (offset of 8 + point as displayed on System Manager)
inputs		Contains a list of the on/off input elements.
input	<i>string</i>	Shows the address and describes how the operation status of a single on/off input has been changed, or has not been changed due to an error. Its text value is the operation status of the on/off input and may be: Auto-On Auto_Off Manual On Manual Off This element has the attributes: node, mod, point, and op_mode.
node (input attr)	<i>int</i>	node address of device that contains on/off input
mod (input attr)	<i>int</i>	module number of the on/off input
point (input attr)	<i>int</i>	point number of the on/off input

Example 1

```
<cmd action="write_digi_op" auth="12345" acct="50"><relays><relay node="1" mod="1" point="10" op_mode="2"/></relays></cmd>
```

```
<resp auth="12345" action="write_digi_op" acct="50" error="0">
  <relays>
    <relay point="10" mod="1" node="1" op_mode="2">Manual On</relay>
  </relays>
</resp>
```

Example 2

```
<cmd action="write_digi_op" auth="12345" acct="50"><relays><relay node="1" mod="1" point="10" op_mode="3"/></relays></cmd>
```

```
<resp auth="12345" action="write_digi_op" acct="50" error="0">
  <relays>
    <relay point="10" mod="1" node="1" op_mode="3">Manual Off</relay>
  </relays>
</resp>
```

1.57 write_val

Write to generic device parameters. Must be authorized as supervisor.
The variable may be specified using either XML tag or cid, vid.

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="write_val" units="string" auth="string" acct="string"
      user="string" password="string">
  <val nodetype="int" node="int" tag="string" set="signed decimal|int"/>
  <val nodetype="int" node="int" tag="string" set="signed decimal|int"/>
  <val nodetype="int" node="int" cid="int" vid="int" set="signed decimal|int"/>
  <val nodetype="int" node="int" cid="int" vid="int" set="signed decimal|int"/>
</cmd>
```

command attribute	Data Type	Definition
units	<i>string</i>	U or u for US, or S or s for SI.
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	<i>string</i>	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	<i>string</i>	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored

command element or attribute	Data Type	Definition
val		This element's attributes provide the address of a variable and the value to be written. Its attributes are: nodetype, node, tag, cid, vid, set.
nodetype (val attr)	<i>int</i>	Node type of device that contains the variable
node (val attr)	<i>int</i>	Node address of device that contains variable
tag (val attr)	<i>string</i>	XML tag as listed in the XMLNAME_SECTION of device's edf file. These also appear in the response to the read_parm_info command.

		The variable is identified either with its tag or with its cid,vid combination.
cid (val attr)	<i>int</i>	Component ID of the variable as listed in the PARAMETER_SECTION of the device's edf file. These also appear in the response to the read_parm_info command. The variable is identified either with its tag or with its cid,vid combination
vid (val attr)	<i>int</i>	Variable ID of the variable as listed in the PARAMETER_SECTION of the device's edf file. These also appear in the response to the read_parm_info command. The variable is identified either with its tag or with its cid,vid combination
set (val attr)	<i>signed decimal or int</i>	Value to which variable is to be set.

Response

```

<resp action="write_val" ...echo command attributes... error="0">
  <val nodetype="int" set=" int" tag=" string" node=" int" display="string"
    name=" string">value</val>
  <val nodetype=" int" set=" int" tag=" string" node=" int" display=" string"
    name=" string"> int </val>
  <val nodetype=" int" cid=" int" set=" int" vid=" int" node=" int" display=" string"
    name=" string.">On</val>
  ....
  ....
  ....
  <val nodetype=" int" cid=" int" set=" int" vid=" int" node=" int" display=" string"
    name=" string.">On</val>
</resp>

```

response element or attribute	Data Type	Definition
error (resp attr)	<i>int</i>	If the data field is read-only or the set data is beyond the range of the parameter min-max settings, the error attribute of resp element will be set as "2".
val		This element's attributes echo the attributes of the corresponding "val" element in the command. Its text value is a reading of the changed variable. This elements attributes are: nodetype, node, tag, cid, vid, set, display, name.
nodetype (val attr)	<i>int</i>	Node type of device that contains the variable
node (val attr)	<i>int</i>	Node address of device thet contains variable
tag (val attr)	<i>string</i>	XML tag as listed in the XMLNAME_SECTION of device's edf file. These also appear in the response to the read_parm_info command.

		The variable is identified either with its tag or with its cid,vid combination.
cid (val attr)	<i>int</i>	Component ID of the variable as listed in the PARAMETER_SECTION of the device's edf file. These also appear in the response to the read_parm_info command. The variable is identified either with its tag or with its cid,vid combination
vid (val attr)	<i>int</i>	Variable ID of the variable as listed in the PARAMETER_SECTION of the device's edf file. These also appear in the response to the read_parm_info command. The variable is identified either with its tag or with its cid,vid combination
set (val attr)	<i>signed decimal or int</i>	Value to which variable was to be set by the command.
display	<i>string</i>	Descriptive XML name of the variable as listed in the XMLNAME_SECTION of device's edf file
name	<i>string</i>	Descriptive parameter name of the variable as listed in the PARAMETER_SECTION of the device's edf file
online	<i>string</i>	States whether the device is online with a value of TRUE or FALSE. If this state returns FALSE, then the point is not written.

Example

```
<cmd units="S" action="write_val" auth="12345" acct="50">
  <val nodetype="16" node="95" tag="wtest1" set="1"/>
  <val nodetype="16" node="95" tag="wtest2" set="2"/>
  <val nodetype="16" node="95" cid="37" vid="17" set="1"/>
  <val nodetype="16" node="95" cid="2" vid="8" set="5"/>
</cmd>
```

```
<resp acct="50" auth="12345" action="write_val" units="S" error="0">
  <val nodetype="16" set="1" tag="wtest1" node="95" display="Main Switch" name="Main
    switch" online="TRUE">On</val>
  <val nodetype="16" set="2" tag="wtest2" node="95" display="Therm Type"
    name="Thermostat type" online="FALSE"></val>
  <val nodetype="16" cid="37" set="1" vid="17" node="95" display="AKV Injection Ctrl."
    name="AKV Injection Ctrl." online="TRUE">On</val>
  <val nodetype="16" cid="2" set="5" vid="8" node="95" online="TRUE" error="2"/>
</resp>
```

1.58write_alarm_ack

Acknowledge a single alarm or all alarms.

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="write_alarm_ack" auth="string" acct="string"
      user="string" password="string" ref="int | all"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
user	string	User name. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the SYSTEM MANAGER, but will be ignored
ref	int or "all"	If a single alarm is to be acknowledged then ref equals the alarm's reference number. If all alarms are to be acknowledged then ref equals "all".

Response

```
<resp action="write_alarm_ack" ...echo of commands attributes... error="0">
<alarm ref="int">"string"</alarm>
<alarm ref="int">"string"</alarm>
....
....
....
<alarm ref="int">"string"</alarm>
</resp>
```

response element or attribute	Data Type	Definition
alarm	<i>string</i>	Element whose text value is “acknowledged” if the alarm has already been acknowledged, or “success” if this command has acknowledged it. It has one attribute, “ref”, which is the alarm reference number. In the case where the command’s “ref” attribute’s value is a single alarm’s ID, there is only one “alarm” element within the “resp” element. In the case where the command’s “ref” attribute’s value is “all” there is one “alarm” element for each already acknowledged or just successfully acknowledged alarm within the “resp” element.
ref (alarm attr)	<i>int</i>	Alarm reference number.

1.59 write_alarm_clear

Clear a single alarm or all alarms.

Command

```
<cmd action="write_alarm_clear" auth="string" acct="string" user="string"
password="string" ref="int | all"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters)
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters)
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
ref	int or "all"	If a single alarm is to be cleared then ref equals the alarm's reference number. If all alarms are to be cleared then ref equals "all".

Response

```
<resp action="write_alarm_clear" ...echo of commands attributes... error="0">
<alarm ref="int">"string"</alarm>
<alarm ref="int">" string"</alarm>
....
....
....
<alarm ref="int">" string"</alarm>
</resp>
```

response element or attribute	Data Type	Definition
alarm	string	Text value is "success" if this command has cleared the alarm. It has one attribute, "ref", which is the alarm reference number. In the case where the command's "ref" attribute's value is a single alarm's ID, there is only one "alarm" element within the "resp" element. In the case where the command's "ref" attribute's value is "all" there is one "alarm" element within the "resp" element for each successfully cleared alarm.
ref (alarm attr)	int	Alarm reference number.

1.60 set_defrost

Sets a generic device's defrost on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_defrost" auth="string" acct="string" user="string" password="string"
nodetype="16" node="int" operation="1|0"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_defrost" ...echo command attributes... error="0">
<operation>string</operation>
</resp>
```

response element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_defrost" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_defrost" node="95" acct="50" auth="12345" nodetype="16"
operation="1" error="0">
<operation>Success</operation>
</resp>
```

1.61 set_light

Sets a generic device's lighting on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_light" auth="string" acct="string" user="string" password="string"
nodetype="16" node="int" operation="int"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS, but will be ignored
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_light" ...echo command attributes... error="0">
<operation>string</operation>
</resp>
```

response element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_light" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_light" node="95" acct="50" auth="12345" nodetype="16" operation="1"
error="0">
<operation>Success</operation>
</resp>
```

1.62 set_main_switch

Sets a generic device's main switch on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_main_switch" auth="string" acct="string" user="string"
password="string" nodetype="16" node="int" operation="int"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_main_switch" ...echo command attributes... error="0">
<operation>string</operation>
</resp>
```

response element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_main_switch" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_main_switch" node="95" acct="50" auth="12345" nodetype="16"
operation="1" error="0">
<operation>Success</operation>
</resp>
```

1.63 set_cleaning

Sets a generic device's cleaning mode on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_cleaning" auth="string" acct="string" user="string" password="string"
nodetype="16" node="int" operation="int"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_cleaning" ...echo command attributes... error="0">
<operation> string </operation>
</resp>
```

response element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_cleaning" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_cleaning" node="95" acct="50" auth="12345" nodetype="16"
operation="1" error="0">
<operation>Success</operation>
</resp>
```

1.64 set_night_setback

Sets a generic device's night setback mode on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_night_setback" auth="string" acct="string" user="string"
password="string" nodetype="16" node="int" operation="int"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255. May be sent to the AKCS but will be ignored.
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_night_setback" ...echo command attributes... error="0">
<operation> string </operation>
</resp>
```

response element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_night_setback" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_night_setback" node="95" acct="50" auth="12345" nodetype="16"
operation="1" error="0">
<operation>Success</operation>
</resp>
```

1.65set_shutdown

Sets a generic device's shutdown mode on or off.

Must be authorized as supervisor

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="set_shutdown" auth="string" acct="string" user="string" password="string"
nodetype="16" node="int" operation=" int"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255 May be sent to the AKCS but will be ignored
acct	string	Authorization code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255 May be sent to the AKCS but will be ignored
user	string	User name. Applies only to the AKCS. May be sent to the AK255, but will be ignored
password	string	Password. Applies only to the AKCS. May be sent to the AK255, but will be ignored
nodetype	int	Always set to "16" (type generic device)
node	int	Node address
operation	int	"0" = off, "1" = on

Response

```
<resp action="set_shutdown" ...echo command attributes... error="0">
<operation> string </operation>
</resp>
```

command element	Data Type	Definition
operation	string	Element whose text value is "Success" if command is successful.

Example

```
<cmd action="set_shutdown" auth="12345" acct="50" nodetype="16" node="95"
operation="1"/>
```

```
<resp action="set_shutdown" node="95" acct="50" auth="12345" nodetype="16"
operation="1" error="0">
<operation>Success</operation>
</resp>
```

1.66 read_system_status

Reads the system status and returns the system flags.

Command

```
<cmd action="read_system_status"/>
```

command attribute	Data Type	Definition
action	<i>string</i>	Fixed string of command name.

Response

```
<cmd response="read_system_status" error="0">
  <status>0</status>
  <override>0</override>
  <critical_alarm>0</critical_alarm>
  <beta_code>0</beta_code>
  <nodes_offline>0</nodes_offline>
  <remote_not_able>0</remote_not_able>
  <active_users>0</active_users>
  <history_active>0</history_active>
  <host_network_fail>0</host_network_fail>
  <factory_app>0</factory_app>
  <berg_pin_missing>0</berg_pin_missing>
  <battery_low>0</battery_low>
  <load_shed_pending>0</load_shed_pending>
  <db_cleared>0</db_cleared>
</cmd>
```

command element	Data Type	Definition		
status	<i>int</i>	System Status	INT	Definition
		Running	0	Unit Nodes -All tasks initialized -Database not cleared
		Load Shed	1	Once or more nodes offline
		IO Scan	2	Unit is running Beta code
		Upgrade	3	Unit currently processing a software update command
		Load DB	4	Unit currently processing a Load DB command
		Save DB	5	Unit currently processing a Save DB command
		Forced Reset	6	Unit has been reset by user.
		Bootup	7	When the unit comes out of reset
		Startup	8	One or more critical Alarms
override	<i>int</i>	One or more asset in override		

critical_alarm	int	One or more critical Alarms
beta_code	int	Unit is running Beta code
nodes_offline	int	Once or more nodes offline
remote_not_able	int	HW & SW not okay for Remote upgrade Factory app not present.
active_users	int	User logged on the local unit
history_active	int	Set if history pullback is active
host_network_fail	int	Units on the host network offline
factory_app	int	Running in factory app
berg_pin_missing	int	Set if berg pin not present
battery_low	int	Battery is low
load_shed_pending	int	Load shed scheduled for future
db_cleared	int	If the DB cleared

Example:

```

<cmd action="read_system_status"/>

<cmd response="read_system_status" error="0">
  <status>0</status>
  <override>0</override>
  <critical_alarm>0</critical_alarm>
  <beta_code>0</beta_code>
  <nodes_offline>0</nodes_offline>
  <remote_not_able>0</remote_not_able>
  <active_users>0</active_users>
  <history_active>0</history_active>
  <host_network_fail>0</host_network_fail>
  <factory_app>0</factory_app>
  <berg_pin_missing>0</berg_pin_missing>
  <battery_low>0</battery_low>
  <load_shed_pending>0</load_shed_pending>
  <db_cleared>0</db_cleared>
</cmd>

```

1.67read_license_data

Reads the license data of the unit.

Command

```
<cmd action="read_license_data"/>
```

command attribute	Data Type	Definition
Action	<i>string</i>	Fixed string of command name.

Response

```
<resp action="read_license_data" error="0">
  <licenses total="int">
    <license desc="String" type="int" value="String">int</license>
  </licenses>
</resp>
```

command element	Data Type	Definition
error	<i>int</i>	error flag
licenses	<i>complex</i>	Holds all licenses that are within controller
total	<i>int</i>	total licenses on this controller
license	<i>complex</i>	holds description, type, and value of license
desc	<i>string</i>	Description of license type
type	<i>int</i>	LICENSED_REFRIG 1 LICENSED_HVAC 2 LICENSED_MULTI 3 LICENSED_CARRIER 11 LICENSED_PFOCUS 12 LICENSED_LITES 18 LICENSED_SINGLES 20 LICENSED_HVACS 21 LICENSED_C_STORE 23 LICENSED_X_HIST 25 // still gets set & shows, but History ignores it LICENSED_ETHER 28 LICENSED_MUNTER 13 LICENSED_LENNOX 14 LICENSED_CUTLR_MB 15 LICENSED_SIEMS_MB 16 LICENSED_CAR_UPC 17 LICENSED_COMTROL 19 LICENSED_LOGGER 22 LICENSED_SQD_MB 26 LICENSED_AKCS 50 // convenience store LICENSED_AKCS2 51 // convenience store for Epta LICENSED_ALL 254
value	<i>string</i>	The actual license key

Example:

```
<resp action="read_license_data" error="0">  
  <licenses total="1">  
    <license desc="Refrigeration & HVAC" type="3" value="2d8-c27-e53-6f7-ce7-71e-  
      b716">1</license>  
  </licenses>  
</resp>
```

1.68 alarm_summary

Provides a summary of alarms.

Command

```
<cmd action="alarm_summary"
  day="int"
  date_format="int"
  time_format="int"/>
```

Command Attribute	Data Type	Attribute Value Definition
day	int	Start of time period whose alarms are to be summarized. "1" to indicate today. "2" to indicate yesterday. "3" to indicate two_days_ago "n" to indicate n-1 days ago where n is a positive integer "0" to indicate all alarms regardless of time that they occurred. This is the default if "day" is not specified.
date_format	int	"0" - month, day, year order with a two digit year. "2" - day, month year order with a two digit year.
time_format	int	"0" - 12 hour clock (e.g. 10:30PM) "1" - 24 hour clock (e.g. 22:30) If this attribute is not specified it is defaulted to 12 hour clock

Response

```
<resp action="alarm_summary" ...repeat command attributes...error="int">
  <total>int</total>
  <active>
    <ref> int </ref>
    <ref> int </ref>
    ...
    <ref> int </ref>
    <total_active> int </ total_active >
  </active>
  <acked>
    <ref> int </ref>
    <ref> int </ref>
    ...
    <ref> int </ref>
    < total_acked> int </ total_acked>
  </acked>
  <cleared>
```

```

    <ref> int </ref>
    <ref> int </ref>
    ...
    <ref> int </ref>
    < total_cleared> int </ total_cleared>
</cleared>
<oldest>
    <time> time date</time>
    <ref> int </ref>
</oldest>
<newest>
    <time> time date</time>
    <ref> int </ref>
</newest>
</resp>

```

Response Element	Data Type	Definition
total	<i>int</i>	Contains the total of both active and cleared alarms that occurred within the requested time period.
active		Contains one or more “ref” elements, one for each active, unacknowledged, alarm that occurred within the requested time period. Contains a single “total_active” element that provides a count of active alarms. Contains only unacknowledged alarms.
acked		Contains a “ref” element for every active, acked alarm. Contains the “total_acked” element which is the count of the “ref” elements.
cleared		Contains one or more “ref” elements, one for each cleared alarm that occurred within the requested time period. Contains a single “total_cleared” element that provides a count of cleared alarms.
oldest		Contains elements “time” and “ref”. This provides the time, date, and reference number of the oldest alarm. If any alarms are returned for the requested time period, this is the oldest of those returned. If none are returned, but the System Manager’s alarm list is not empty, this is the oldest in the list.
newest		Contains elements “time” and “ref”.

		These provide the time, date, and reference number of the newest alarm.
ref	<i>int</i>	Contains a reference number. When the System Manager creates its internal representation of an alarm it assigns to it the next reference number in sequence. The sequence spans both active and cleared alarms. Range 1 through 2147483647
time	<i>time date</i>	Contains the time in 12 or 24 hour format and date in MMDDYY or DDMMYY format.
total_active	<i>int</i>	The number of active, unacknowledged alarms
total_acked	<i>int</i>	The number of active, acknowledged alarms
total_cleared		The number of cleared alarms
error	<i>int</i>	Error code , 0 = ok, see “Error Codes” in “Reference Information” section for other error codes.

Example

```

<cmd action="alarm_summary"/>
<resp action="alarm_summary" error="0">
  <total>53</total>
  <active>
    <ref>7</ref>
    <unacknowledged>1</unacknowledged>
    <total_active>1</total_active>
  </active>
  <cleared>
    <ref>53</ref>
    <ref>52</ref>
    ---
    ---
    ---
    <ref>4</ref>
    <ref>3</ref>
    <ref>2</ref>
    <ref>1</ref>
    <total_cleared>52</total_cleared>
  </cleared>
  <oldest>
    <time>03:42PM 01/05/08</time>
    <ref>1</ref>
  </oldest>
  <newest>
    <time>02:56PM 07/05/08</time>
    <ref>53</ref>
  </newest>
</resp>

```

1.69 alarm_detail

Returns details for a single alarm record.

Command

```
<cmd action="alarm_detail"
  only="string"
  before="int"
  after="int"
  current="int"
  newest="string"
  oldest="string"
  expanded="int"
  lang="string"
  date_format="int"
  time_format="int"
  units="string"/>
```

Command Attribute	Data Type	Attribute Value Definition
only	string	“active” return only an active alarm “cleared” return only a cleared alarm “acked” return only an alarm that has been acked “unacked” return only an alarm that has not been acked “any” return any alarm
before	int	Return the alarm whose reference number is less than and closest to this attribute’s value, subject to the “only” criteria. If there is no such alarm, error 12: Error in finding alarm ref. number, is returned.
after	int	Return the alarm whose reference number is greater than and closest to this attribute’s value, subject to the “only” criteria If there is no such alarm, error 12: Error in finding alarm ref. number, is returned
current	int	Return the alarm whose reference number is equal to this attribute’s value, subject to the “only” criteria. If there is no such alarm, error 12: Error in finding alarm ref. number, is returned
newest	string	If this attribute is specified, return the most recent alarm, subject to the “only” criteria. Always specify its value as “true” If there is no such alarm, error 12: Error in finding alarm ref. number, is returned
oldest	string	If this attribute is specified, return the oldest alarm, subject to the “only” criteria. Always specify its value as “true” If there is no such alarm, error 12: Error in finding alarm ref. number, is returned
expanded	int	The value of this attribute selects a short or long response. If not specified the response is short. See the Response section for a description of short and long responses.

		Values are “1” for short and “2” for long.
lang	<i>string</i>	Language: “c” or “C” Chinese “s” or “S” Spanish “p” or “P” Portugese “g” or “G” German “f” or “F” French “d” or “D” Dutch “j” or “J” Japanese “e” or “E” English
date_format	<i>int</i>	“0” for MMDDYY or “2” for DDMMYY
time_format	<i>int</i>	“0” 12 hour format (e.g. 10:30 PM) “1” 24 hour format (e.g. 22:30)
units	<i>string</i>	“u” or “U” U.S. units “s” or “S” SI units

Response

<resp action=”alarm_detail”...repeat command attributes... error=”int”/>

```

<ref>int</ref>
<storeName> string </storeName>
<unitNumber> int </unitNumber>
<unitName>string</unitName>
<deviceName>string</deviceName>
<deviceModel>string</deviceModel>
<nodetype> int </nodetype>
<addr> int </addr>
<class>string</class>
<name>string</name>
<status>string</status>
<action>string</action>
<clearable>string</clearable>
<value> string </value>
<setting> string </setting>
<ack>int</ack>
<acknowledgement> string </acknowledgement>
<occurDate>date</occurDate>
<occurTime>time</occurTime>
<clearDate> date</clearDate>
<clearTime> time</clearTime>
<ackDate>date</ackDate>
<ackTime>time</ackTime>
<epoch>uint</epoch>
<epoch_acked>uint</epoc_acked>
<epoch_cleared> uint </epoch_cleared>
<ref_tag_id>string</ref_tag_id>
</resp>

```

In the following table, elements that have a “yes” in the “long only” column only appear in a response when the command’s “expanded” attribute equals “2” for long.

Response Element	Long only	Data Type	Definition
ref		<i>int</i>	Alarm reference number that is unique within an System Manager unit Range 1 to 2147483647
storeName	yes	<i>string</i>	Configured name of store. Max size 16 characters.
unitNumber		<i>int</i>	System Manager unit number, range 0 through 9, 0 is master
unitName		<i>string</i>	SYSTEM MANAGER unit name. Max size 16 characters
deviceName		<i>string</i>	Name of device, e.g. AKC55, EKC204A
deviceModel	yes	<i>string</i>	e.g. 084B8520
nodetype	yes	<i>int</i>	Number associated with type of device that is the source of the alarm. See “Node Types” in “Reference Information” for a list of node types.
addr		<i>string</i>	The address of the device that has sent the alarm in board/point display format. This is the board and point number formatted as would be seen on the System Manager’s local display.
class	yes	<i>string</i>	Alarm class name - used for alarm routing.
name		<i>string</i>	Name of alarm
name_id		<i>int</i>	Id of the name of generic alarm. Only exists if name is from an edf file.
device_id		<i>string</i>	Device id, which is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x
status		<i>string</i>	“active” or “cleared”
action		<i>string</i>	“Critical Severe Normal Log only Disabled”
clearable		<i>int</i>	1 = may be cleared, 0 = may not be cleared
value	yes	<i>string or signed decimal</i>	Value when tripped. For a simple alarm this is “Trip OK” Otherwise it is a numeric value.
current_value		<i>string or signed decimal</i>	Current value
parval		<i>int or signed decimal</i>	The current value without any formatting.
units		<i>string</i>	units (see description from READ COMMANDS Section)
unittype		<i>int</i>	unit type (see description from READ COMMANDS Section)
setting	yes	<i>string</i>	Describes alarm

ack		<i>int</i>	0 = alarm not acknowledged When non-zero, the alarm has been acked and the value indicates the authorization number and the account number. The account number is the remainder resulting from value/256. It has range 1-66 The authorization number is value/256 ignoring the remainder. It has range 1-7. Does not apply to AKCS.
acknowledgement		<i>string</i>	"No" its not acknowledged "Yes" if acknowledged.
occurDate		<i>date</i>	Date when alarm tripped
occurTime		<i>time</i>	Time when alarm tripped
clearDate		<i>date</i>	Date when alarm cleared
clearTime		<i>time</i>	Time when alarm cleared
ackDate		<i>date</i>	Date when alarm was acked
ackTime		<i>time</i>	Time when alarm was acked
epoch		<i>int</i>	Date and time when alarm tripped expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (Local time) of January 1, 1970.
epoch_acked		<i>int</i>	Date and time when alarm was acked expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (Local time) of January 1, 1970.
epoch_cleared		<i>int</i>	Date and time when alarm cleared expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (Local time) of January 1, 1970.
ref_tag_id		<i>string</i>	Reference to tag_id associated with the data point.

Example 1

```

<cmd action = "alarm_detail" current = "1" expanded="1"/>
<resp expanded="1" current="1" action="alarm_detail" error="0">
  <ref>1</ref>
  <unitNumber>0</unitNumber>
  <unitName/>
  <deviceName>AK-SC255</deviceName>
  <addr>0</addr>
  <name>Database Cleared</name>
  <status>active</status>
  <action>Normal</action>
  <acknowledgement>No</acknowledgement>
  <occurDate>01/31/95</occurDate>
  <occurTime>12:01AM</occurTime>
  <clearDate/>
  <clearTime/>

```

```
<epoch>791510502</epoch>  
<epoch_cleared>0</epoch_cleared>  
<ref_tag_id>No Tag</ref_tag_id>  
</resp>
```

Example 2

```
<cmd action = "alarm_detail" current = "1" expanded="2"/>  
<resp expanded="2" current="1" action="alarm_detail" error="0">  
  <ref>1</ref>  
  <storeName/>  
  <unitNumber>0</unitNumber>  
  <unitName/>  
  <deviceName>AK-SC255</deviceName>  
  <deviceModel/>  
  <nodetype>255</nodetype>  
  <addr>0</addr>  
  <class>System Alarms</class>  
  <name>Database Cleared</name>  
  <status>active</status>  
  <action>Normal</action>  
  <value>Trip</value>  
  <setting>Alarm if error</setting>  
  <acknowledgement>No</acknowledgement>  
  <occurDate>01/31/95</occurDate>  
  <occurTime>12:01AM</occurTime>  
  <clearDate/>  
  <clearTime/>  
  <epoch>791510502</epoch>  
  <epoch_cleared>0</epoch_cleared>  
  <ref_tag_id>No Tag</ref_tag_id>  
</resp>
```

1.70 write_alarm_ack

Acknowledge a single alarm or all alarms.

Note that the cmd attributes auth and acct are used when communicating to an AK255 and the attributes user and password are used when communicating to an AKCS. Only one of these pairs needs to be provided.

Command

```
<cmd action="write_alarm_ack" auth="string" acct="string"
      user="string" password="string" ref="int | all"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255.
user	string	User name. Applies only to the AK355/AKCS.
password	string	Password. Applies only to the AK355/AKCS.
ref	int or "all"	If a single alarm is to be acknowledged then ref equals the alarm's reference number. If all alarms are to be acknowledged then ref equals "all".

Response

```
<resp action="write_alarm_ack" ...echo of commands attributes... error="0">
<alarm ref="int">"string"</alarm>
<alarm ref="int">"string"</alarm>
....
....
....
<alarm ref="int">"string"</alarm>
</resp>
```

response element or attribute	Data Type	Definition
alarm	string	Element whose text value is "acknowledged" if the alarm has already been acknowledged, or "success" if this command has acknowledged it. It has one attribute, "ref", which is the alarm reference number. In the case where the command's "ref" attribute's value is a single alarm's ID, there is only one "alarm" element within the "resp" element.

		In the case where the command's "ref" attribute's value is "all" there is one "alarm" element for each already acknowledged or just successfully acknowledged alarm within the "resp" element.
ref (alarm attr)	<i>int</i>	Alarm reference number.

1.71 write_alarm_clear

Clear a single alarm or all alarms.

Command

```
<cmd action="write_alarm_clear" auth="string" acct="string" user="string"
password="string" ref="int | all"/>
```

command attribute	Data Type	Definition
auth	string	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255.
acct	string	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255.
user	string	User name. Applies only to the AK355/AKCS.
password	string	Password. Applies only to the AK355/AKCS.
ref	int or "all"	If a single alarm is to be cleared then ref equals the alarm's reference number. If all alarms are to be cleared then ref equals "all".

Response

```
<resp action="write_alarm_clear" ...echo of commands attributes... error="0">
<alarm ref="int">"string"</alarm>
<alarm ref="int">"string"</alarm>
....
....
....
<alarm ref="int">"string"</alarm>
</resp>
```

response element or attribute	Data Type	Definition
alarm	string	Text value is "success" if this command has cleared the alarm. It has one attribute, "ref", which is the alarm reference number. In the case where the command's "ref" attribute's value is a single alarm's ID, there is only one "alarm" element within the "resp" element. In the case where the command's "ref" attribute's value is "all" there is one "alarm" element within the "resp" element for each successfully cleared alarm.
ref (alarm attr)	int	Alarm reference number.

1.72 start_history_query

Command

```
<cmd action="start_history_query" hist_index="int" sample_rate="int" units="string"
    averaged_over="int"
    compress="int" start_epoch="int" stop_epoch="int">
</cmd>
```

command attribute	Data Type	Definition
action (required)	<i>token</i>	Fixed value of the command. Must be exact (start_history_query).
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255.
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255.
user	<i>string</i>	User name. Applies only to the AK355/AKCS.
password	<i>string</i>	Password. Applies only to the AK355/AKCS.
hist_index (required)	<i>int</i>	History local datapoint number. One based, that is, range starts with 1.
sample_rate (required)	<i>int</i>	Requested sample rate in seconds. If this attribute is not specified then the sample rate is equal to the rate at which samples are recorded in the history files. The sample rate is specified in seconds and must be one of the following: 5 30 60 (1 minute) 120 (2 minutes) 600 (10 minutes) 900 (15 minutes) 1800 (30 minutes) 3600 (1 hour)
units (NOT required)	<i>string</i>	units = "U", "u", "S", or "s" Default is system units.
averaged_over (NOT required)	<i>int</i>	Averaging interval in seconds. Each returned value is the average of all samples that appear within each time interval of this size. If this attribute is not specified then sample values are returned, not averages. averaged_over must be specified in seconds; must be greater than the sample_rate if provided or the recorded rate if not; and must be one of the following: 30 60 (1 minute)

		120 (2 minutes) 600 (10 minutes) 900 (15 minutes) 1800 (30 minutes) 3600 (1 hour) 10800 (3 hours) 21600 (6 hours) 43200 (12 hours) 86400 (1 day)
compress (NOT required)	<i>int</i>	1 = force data to be compressed. 0 = force data to be uncompressed. If this attribute is unspecified then returned data is uncompressed if its size is less than or equal to 1024 and compressed if its size is greater than 1024..
start_epoch (required)	<i>uint</i>	Date and time of the beginning of the history interval expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM of January 1, 1970 Local time of the controller. NOTE: If authorized, the epoch time is UTC. This is needed to ensure the existing functionality did not change.
stop_epoch (required)	<i>uint</i>	Date and time of the end of the history interval expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM of January 1, 1970 Local time of the controller. NOTE: If authorized, the epoch time is UTC. This is needed to ensure the existing functionality did not change.

Response

```
<resp action=" start_history_query " ...echo the command attributes...error=" int ">
<query_id> int </query_id>
</resp>
```

response element	Data Type	Definition
resp	<i>token</i>	Fixed value of the command. Must be exact (start history query).
query_id	<i>int</i>	Identifies the query started by the start_history_query command.

1.73read_query_status

Command

```
<cmd action="read_query_status" query_id= "int"/>
```

This command is sent periodically to determine when the history data collecting activity started by `start_history_query` has stopped.

The final `read_query_status` response provides information about the collected data such as the number and size of the data fields. This is used to interpret the results of the subsequent `read_history_data` command.

If there is a possibility that more than one client may send the `start_history_query` then the `read_query_status` command must be sent within two minutes of the `start_history_query` command, and in the case where more than one `read_query_status` is necessary, no more than two minutes must be allowed to expire between them.

If command sends proper authorization, the returned start and stop epoch times are returned as UTC epoch time, instead of the local epoch time.

command attribute	Data Type	Definition
action (required)	<i>token</i>	Fixed value of the command. Must be exact (<code>read_query_status</code>).
auth	<i>string</i>	Authorization code. Must be five characters and all must be numeric (no alpha characters) Applies only to the AK255.
acct	<i>string</i>	Account code. Must be two characters and both must be numeric (no alpha characters) Applies only to the AK255.
user	<i>string</i>	User name. Applies only to the AK355/AKCS.
password	<i>string</i>	Password. Applies only to the AK355/AKCS.
query_id (required)	<i>int</i>	Identifies a query that was previously started by the <code>start_history_query</code> command

Response

```
<resp action="read_query_status " ...echo the command attributes...error=" int ">
  <status>active|complete|more|idle</status>
  <field_count>int</field_count>
  <field_size>int</field_size>
  <exp>int</exp>
  <unit>string</unit>
  <actual_sample_rate>int</actual_sample_rate>
  <offset>-800</offset>
  <starttime>
    <year>int</year>
    <month>int</month>
    <day>int</day>
    <hour>int</hour>
    <minute>int</minute>
    <second>int</second>
    <epoch>uint</epoch>
  </starttime>
  <stoptime>
    <year>int</year>
```

```

<month>int</month>
<day>int</day>
<hour>int</hour>
<minute>int</minute>
<second>int</second>
<epoch>uint</epoch>
</stopttime>
</resp>

```

response element	Data Type	Definition
status	string	Query status. active = query is being executed complete = query is no longer being executed and results are available for the entire requested time interval more = query is no longer being executed and results are available for part of the requested time interval. These results must be read with a read_query_data command and then another start_history_query must be issued for the remaining portion of the interval. <i>This is the only element returned if status="active"</i>
field_count	int	The number of fields of size field_size that will appear in the binary data block. These fields start after the query_id which occupies the first four bytes of the block.
field_size	int	Size of the field in bytes.
exp	int	Exponent.. The position of the decimal point is determined for all values in a binary response by exp. This is a power of ten that is applied to the integer value to create a decimal value. For example, if the variable's value is 24536 and exp is "-1" the variable's value is to be interpreted as 2453.6. Another example: if the variable's value is 24536 and exp is "-3", the variable's value is to be interpreted as 24.536..
min	int	Minimum value of field (without decimal).
max	int	Maximum value of field (without decimal)
unit	string	Unit that applies to all data values. Possible values are: - psi - bar - degf - degfd - degc - degcd - percent - ppm

		v amp kw kwh hz gpm fps ph min hr sec fc lpm lps
actual_sample_rate	int	The rate at which the samples are actually recorded in the System Manager's history files. This may differ from the sample_rate requested in the start_history_query command
offset	int	The offset of the controller (i.e. -800)
starttime		Describes starting time for returned history data. Contains elements year,month,day,hour,minute,second
year	int	Two digit year
month	int	Month
day	int	Day of month
hour	int	Hour of day 0 through 23
minute	int	Minute
second	int	Second
epoch	uint	Date and time of the beginning of the history interval expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM of January 1, 1970 Local time. NOTE: If authorized, the epoch time is UTC. This is needed to ensure the existing functionality did not change.
stoptime		Describes ending time for returned history data. Contains elements year,month,day,hour,minute,second,epoch

Only the status element is returned if status is "active".

1.74 read_query_data (binary data is returned)

Command

<cmd action="read_query_data" query_id= "int"/>

This command is to be sent only after the read_query_status command has returned a status not equal to "active"

If there is a possibility that more than one client may send the start_history_query then the read_query_data command must be sent within two minutes of the last read_query_status command.

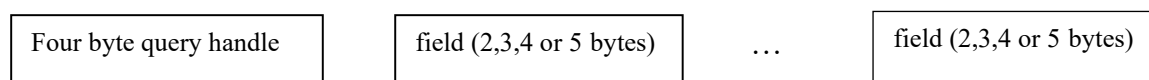
attribute	Data Type	Definition
action (required)	token	Fixed value of the command. Must be exact (read_query_data).
query_id (required)	int	Identifies a query that was previously started by the start_history_query command

Response

The response is a block of binary data.

The first 4 bytes contains the query_id number in high to low order.

The remainder consists of the data fields where data bytes within a field are ordered high (first received) to low.



The first byte of a field determines how the remaining bytes of the field are to be interpreted and may have the following values:

- 0x00 The remainder of the field is a data sample or an average value in high to low byte order.
- 0x01 No sample read yet.
The remainder of the field is 0.
- 0x02 No sample because time is outside of collection range.
The remainder of the field is 0.
- 0x03 No sample because of gap in collection activity.
The remainder of the field is 0.
- 0x04 No sample because of node offline.
The remainder of the field is 0.
- 0x05 No sample because history collection suspended.
The remainder of the field is 0.
- 0x06 Some samples were unavailable and so could not be used to compute an average for this interval. The value that appears in the remainder of the field is the average of those that were available.
- 0x07 No valid samples were available during this interval and so no average could be computed. The remainder of the field is 0.

0xFF Invalid sample for unknown reason.
The remainder of the field is 0.

The data portion of the field is to be interpreted as a decimal value where the position of the decimal point is indicated by the “exp” status response element. For example, if the field has an integer value of 15234 and exp is “-1” then the field is to be interpreted as the decimal number 1523.4.

If exp is zero, then there is no implied decimal.

1.75abort_query

Command

<cmd action="abort_query" query_id= "*int*" />

This command aborts the history data collection activity started by a start_history_query command.

command attribute	Data Type	Definition
action (required)	<i>token</i>	Fixed value of the command. Must be exact (abort_query).
query_id (required)	<i>int</i>	Identifies a query that was previously started by the start_history_query command

Response

<resp action=" abort_query " ...*echo the command attributes*...error=" *int* " />

Alarm Messages

The System Manager may be configured to send XML formatted alarm messages to a specific ip address and port using TCP/IP. These messages are unsolicited and do not use the HTTP protocol. The System Manager will send an alarm message every 30 seconds until it receives an acknowledgement message from the alarm receiver through the TCP/IP channel. The alarm receiver must send an acknowledgement message in the format described below to indicate to the System Manager that it has received and saved the alarm message successfully.

Alarm Message Format

```
<alarm>
  <ref>int</ref>
  <storeName> string </storeName>
  <unitNumber> int </unitNumber>
  <unitName>string</unitName>
  <deviceName>string</deviceName>
  <deviceModel>string</deviceModel>
  <nodetype> int </nodetype>
  <node>int</node>
  <mod>int</mod>
  <point>int</mod>
  <cid>int</cid>
  <vid>int</vid>
  <addr> int </addr>
  <class>string</class>
  <name>string</name>
  <name_id>int</name_id>
  <device_id>string</device_id>
  <status>string</status>
  <action>string</action>
  <acked_clearable>string</acked_clearable>
  <value parval="signed decimal|int" units="string" units_index="int"> string </value>
  <current_value parval="signed decimal|int" units="string"
units_index="int">string</current_value>
  <setting> string </setting>
  <acknowledgement> string </acknowledgement>
  <occurDate>date</occurDate>
  <occurTime>time</occurTime>
  <clearDate> date</clearDate>
  <clearTime> time</clearTime>
  <epoch>uint</epoch>
  <epoch_cleared> uint </epoch_cleared>
  <severity>string</severity>
  <ipAddr>string</ipAddr>
  <macAddr>string</macAddr>
  <storeId1>string</storeId1>
  <storeId2>string</storeId2>
  <xmitNum>int</xmitNum>
  <routingAction>int</routingAction>
  <active_state>int</active_state>
```

```

<acked_state>int</acked_state>
<cleared_state>int</cleared_state>
<ackDate>string</ackDate>
<ackTime>string</ackTime>
<epoch_acked>int</epoch_acked>
<ackUserAcct>string</ackUserAcct>
<inactiveDate>string</inactiveDate>
<inactiveTime>string</inactiveTime>
<epoch_inactive>int</epoch_inactive>
<clearUserAcct>string</clearUserAcct>
<listed_as>int</listed_as>
</alarm>

```

Element	Data Type	Definition
alarm		Identifies this message as an alarm
ref	int	Alarm reference number that is unique within an System Manager unit Range 1 to 2147483647 Starts at 1 after alarm log is cleared
storeName	string	Configured name of store. Max size 16 characters.
unitNumber	int	System Manager unit number, range 0 through 9, 0 is master
unitName	string	SYSTEM MANAGER unit name. Max size 16 characters
deviceName	string	Name of device, e.g. AKC55, EKC204A
deviceModel	string	e.g. 084B8520
nodetype	int	Number associated with type of device that is the source of the alarm. See “Node Types” in “Reference Information” for a list of node types.
addr	string	The address of the device that has sent the alarm in board/point display format. This is the board and point number formatted as would be seen on the System Manager’s local display.
class	string	Alarm class name - used for alarm routing.
class_id	int	ID for class name
level_id	int	ID for the alarm type (i.e. Hi_Temp, etc...)
name	string	Name of alarm
name_id	int	Id of the name of generic alarm. Only exists if name is from an edf file. NOTE: this is a 0-based index If alarm_id in edf states 14, the name_id returned is 13....
device_id	string	Device id, which is composed of the <MODEL> from the EDF file concatenated with underscore and the <VERSION> from the EDF file e.g, 080Z0124_012x
status	string	“active” or “cleared”

action	<i>string</i>	“Critical Normal Log only Disabled”
acked_clearable	<i>int</i>	1 = may be cleared, 0 = may not be cleared
value	<i>string or signed decimal</i>	Value when tripped. For a simple alarm this is “Trip OK” Otherwise it is a numeric value.
current_value	<i>string</i>	Current value
parval	<i>int or signed decimal</i>	The current value without any formatting. Can be null if no value exists.
units	<i>string</i>	units (see description from READ COMMANDS Section)
units_index	<i>int</i>	unit index (see description from READ COMMANDS Section)
setting	<i>string</i>	Describes alarm
acknowledgement	<i>string</i>	“No” its not acknowledged “Yes” if acknowledged.
occurDate	<i>date</i>	Date when alarm tripped
occurTime	<i>time</i>	Time when alarm tripped
clearDate	<i>date</i>	Date when alarm cleared
clearTime	<i>time</i>	Time when alarm cleared
epoch	<i>int</i>	Date and time when alarm tripped expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.
epoch_cleared	<i>int</i>	Date and time when alarm cleared expressed as epoch time. Epoch time is defined as the number of seconds elapsed since 12:00 AM (local SYSTEM MANAGER time) of January 1, 1970.
severity	<i>string</i>	“Critical Normal”
ipAddr	<i>string</i>	Ip address of the transmitting System Manager. This is 15 characters formatted as follows: 999.999.999.999
macAddr	<i>string</i>	Mac (Ethernet) address of the System Manager. This uniquely identifies the System Manager. It is 12 characters formatted as 6 Hex ASCII encoded bytes. For example: 3A290E214F60
storeId1	<i>string</i>	An additional store identifier. Eight characters. If fewer than eight characters were entered in System Manager, it is left justified with trailing spaces.
storeId2	<i>string</i>	An additional store identifier. Eight characters. If fewer than eight characters were entered in System Manager, it is left justified with trailing spaces.
xmitNum	<i>int</i>	Starts equal to 1 and increments with each retransmission of an alarm message. Allows for processing of duplicate transmissions by receiver
routingAction	<i>int</i>	Routing action. Range is 1 through 15.

active_state	<i>int</i>	The active_state is one when an alarm condition occurs and changes to 0 when the alarm condition returns to normal
acked_state	<i>int</i>	The acked_state becomes 1 when an alarm is acknowledged. This may be performed manually at the System Manager's front panel or through the XML interface via the write_alarm_ack command. The ak255 R version does not require that an alarm is acked for it to be considered cleared when the alarm condition returns to normal. Therefore it forces the acked_state to a value of 1. The same is true for the E version when the action code is 12 through 15.
cleared_state	<i>int</i>	The cleared_state becomes 1 when an alarm is manually cleared. This may be performed at the System Manager's front panel or through the XML interface via the write_alarm_clear command. The ak255 R version does not require that an alarm is manually cleared for it to be considered cleared when the alarm condition returns to normal. Therefore the ak255 forces the cleared_state to a value of 1. The same is true for the E version when the action code is 9 through 15.
ackDate	<i>string</i>	The date that the alarm was acknowledged.
ackTime	<i>string</i>	The time that the alarm was acknowledged.
epoch_acked	<i>int</i>	The epoch time that the alarm was acknowledged
ackUserAcct	<i>string</i>	Authorization level - Account in format 99-99. Identifies the user that acknowledged the alarm.
inactiveDate	<i>string</i>	Date on which the alarm became inactive.
inactiveTime	<i>string</i>	Time at which the alarm became inactive
epoch_inactive	<i>int</i>	Epoch time at which the alarm became inactive.
clearUserAcct	<i>string</i>	Authorization level - Account in format 99-99. Identifies the user that cleared the alarm
listed_as	<i>int</i>	Indicates in which list the SYSTEM MANAGER displays the alarm. Its values may be 1, 2, or 3 indicating the active, acked, and cleared lists respectively.

Acknowledgement Message Format

<ack ref="int"/>

Element	Data Type	Definition
ack		Identifies this message as an acknowledgement

Attribute	Data Type	Definition
ref	<i>int</i>	Alarm reference number that is unique within an System Manager unit Range 1 to 2147483647 Starts at 1 after a new database installation.

2 Establishing a Trusted Browser Connection

Figure 1 describes the process of establishing a secure connection between the web browser and the SM800A. While enabling HTTPS in the settings ensures the encryption of the data exchanges it does not prevent man-in-the-middle attacks. To certify the authorship of the messages by a trusted third party (certificate authority) and obtain a secure browser session the following steps must be completed:

- 3 Select a certificate authority and obtain the root or intermediate certificate file.
- 4 Import the root or intermediate certificate in the browser.
Note: These steps can be skipped if no custom CA is used.
- 5 Create a Certificate Signing Request (CSR) using the gen_csr XML command. Decode the returned b64 string and save it to a text file with the file name indicated in the XML response.
- 6 Send the .csr file to the certificate authority (CA) to have it signed.
- 7 Upload the signed certificate using the load_cert XML command. Make sure it has the same file name as the .csr file and a .pem file extension. It must match the last CSR generated by the System Manager, i.e. no other gen_csr requests should be made.

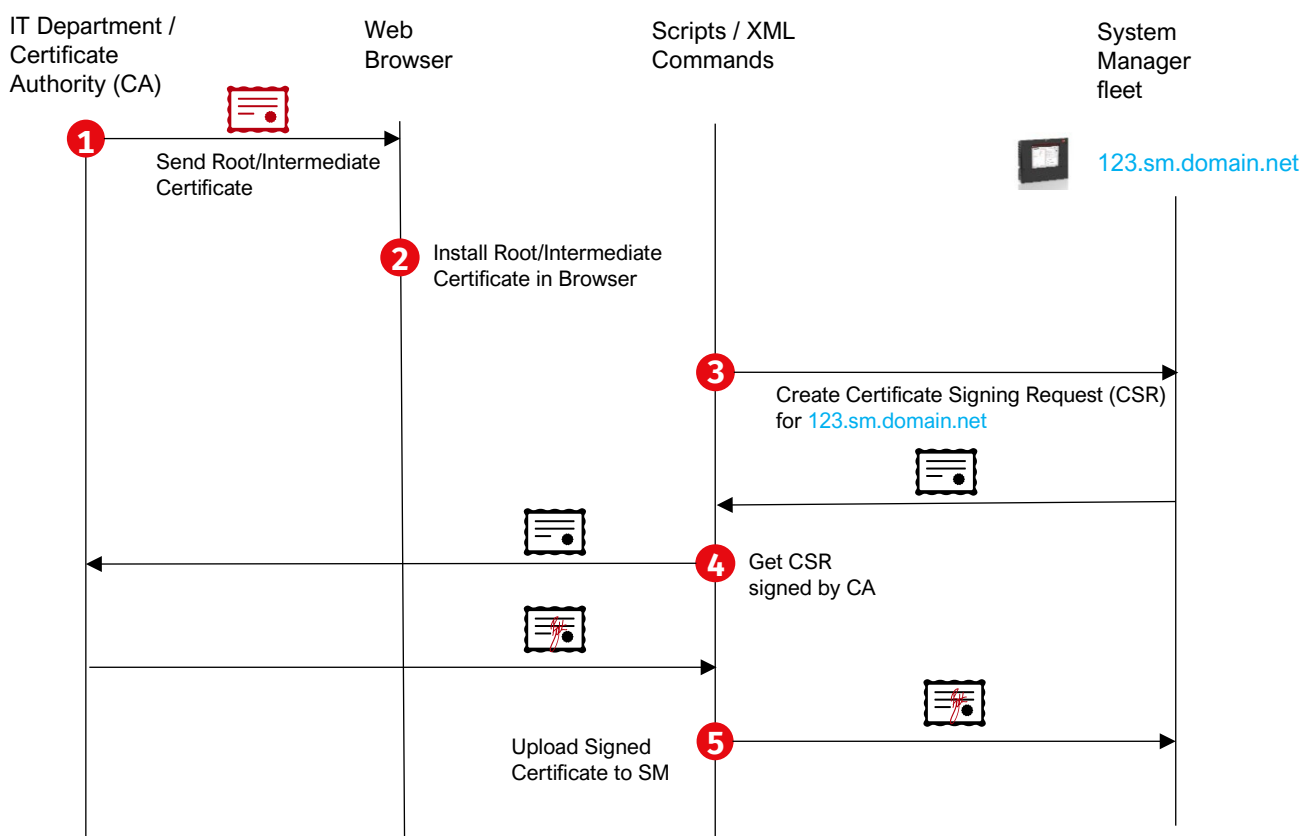


Figure 1: Certificate Generation for the System Manager.

It is also possible to create the signed key pair with the OpenSSL library and upload it to the System Manager. If the domain of the certificate contains a wildcard the key pair can be shared among many System Managers that run in that domain.

- 8 Select a certificate authority and obtain the root or intermediate certificate file.
- 9 Import the root or intermediate certificate in the browser.
Note: These steps can be skipped if no custom CA is used.
- 10 Create a Certificate Signing Request (CSR) along with its private key using the OpenSSL library.
- 11 Send the .csr file to the certificate authority (CA) to have it signed.
- 12 Upload the private key and the signed certificate as a B64 encoded string using the load_key_pair XML command.

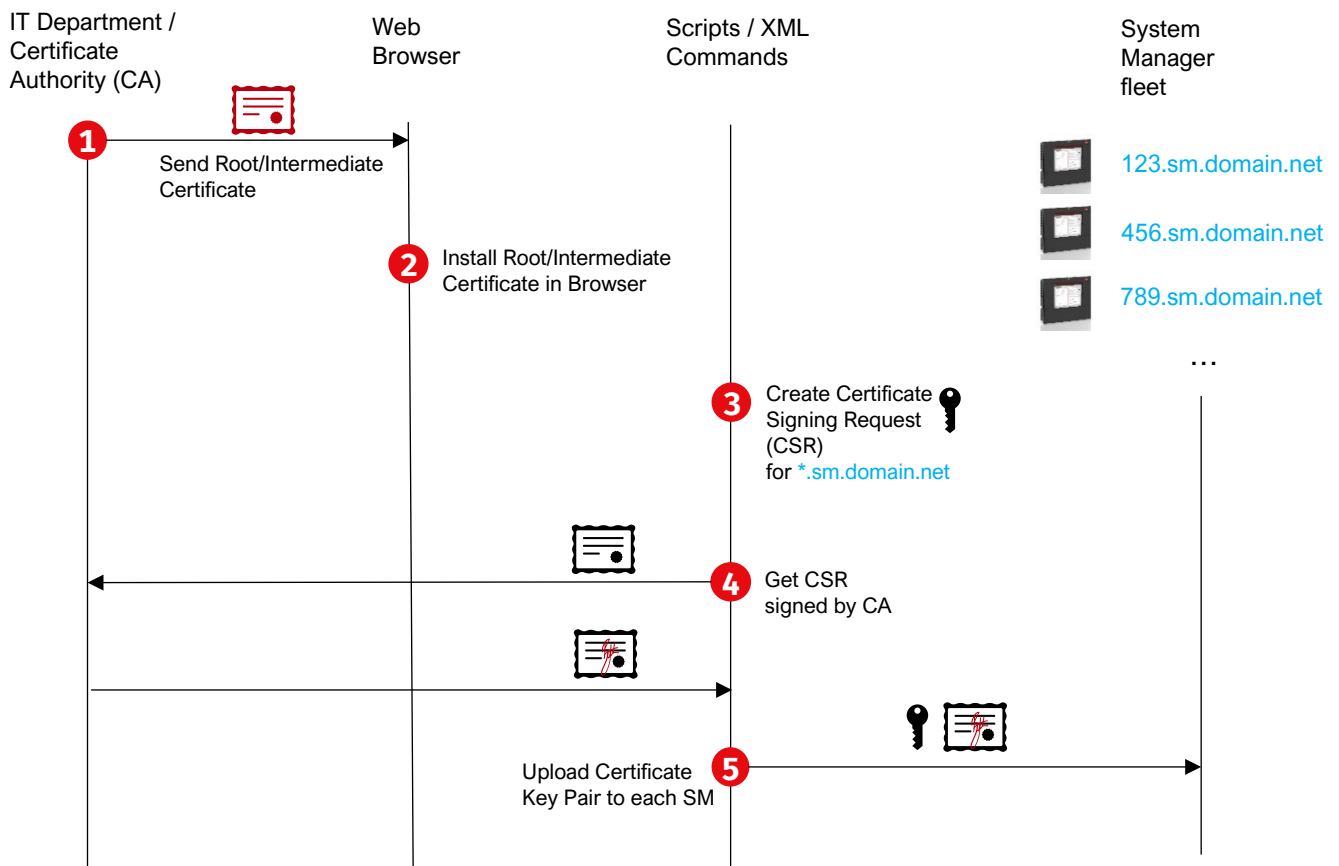


Figure 2: Certificate Generation for multiple System Managers.

Creating a Custom Certificate Authority

Instead of using a trusted third-party to sign the CSR, a custom certificate authority can be created for testing purposes.

- 1 Install OpenSSL on your computer. Make sure the PATH variable is set such that it can be accessed from the command line.
- 2 Generate the private key called *ca.key* by running the following command from the command line:

```
openssl genrsa -des3 -out ca.key 2048
```

It will ask for the pass phrase, please keep the pass phrase in a safe place because it will be required for signing certificate requests.

- 3 Generate the root certificate *ca.crt* with an expiration time of 1 year based on the private key:

```
openssl req -x509 -new -nodes -key ca.key -sha256 -days 365 -out ca.crt
```

Enter your information:

Country Name (2 letter code) [AU]:*DE*

State or Province Name (full name) [Some-State]:*HH*

Locality Name (eg, city) []:*Hamburg*

Organization Name (eg, company) [Internet Widgits Pty Ltd]:*Danfoss*

Organizational Unit Name (eg, section) []:*ClimateSolutions*

Common Name (e.g. server FQDN or YOUR name) []:*John*

Email Address []:*john.doe@danfoss.com*

- 4 Create an extension file *ca.ext* as described below:

authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

subjectAltName = @alt_names

[alt_names]

DNS.1 =sm800a.danfoss.com

IP.1 =192.168.0.17

where DNS and IP should be set to the address of the web interface of the System Manager. At least one entry is required (DNS or IP).

- 5 To sign a CSR named *sm800a-danfoss-com.csr* using the CA files created above the following command should be run from the same folder containing the .crt, .csr and .key files.

```
openssl x509 -req -days 365 -CA ca.crt -CAkey ca.key -CAcreateserial -CAserial  
ca.serial -in sm800a-danfoss-com.csr -out sm800a-danfoss-com.pem -extfile ca.ext
```

It generates a signed certificate called *sm800a-danfoss-com.pem* that can be uploaded to the System Manager. The name of .csr and .pem should match but can be chosen arbitrarily. Here, the name *sm800a-danfoss-com* describes the domain the certificate is issued for.

Encoding of Files

The System Manager XML interface uses the Base64 (B64) encoding to transfer files. The encoding can be done by an online tool or using the following scripts.

Below are given two example scripts in Powershell that can be used to turn a file into a B64 string and vice versa.

- 1 Create a file called *encode_b64.ps1* with the following content:

```
param(
    [parameter(mandatory=$true)][string]$file
)

$Content1 = get-content $file -Encoding UTF8 -Raw
$Bytes = [System.Text.Encoding]::UTF8.GetBytes($Content1)
Write-Host "Bytes: " $Bytes.Length
$Encoded = [System.Convert]::ToBase64String($Bytes)
$Encoded | set-content ($file + ".b64")
Write-Host "ENCODED: " $Encoded
```

It can then be called from the command line with `./encode_b64` with the absolute file path as parameter.

- 2 Create a file called `decode_b64.ps1` with the following content:

```
param(
    [parameter(mandatory=$true)][string]$file
)

# Decode
$Content1 = get-content $file
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($Content1)) | Out-File -Encoding "ASCII" ($file + ".dec")
$Content2 = get-content ($file + ".dec")
Write-Host "DECODED: " $Content2
```

It can then be called from the command line with `./decode_b64` with the B64 string as only parameter.

Commands and Responses

2.1 `gen_ssc`

The command generates a self-signed certificate which is not recognized as secure by the browser. It puts the system manager's certificates in a state that is equivalent to the factory settings and should be considered as a reset. It returns a file that contains the summary of the certificate as b64 encoded string. All fields of the request are optional.

Command

```
<cmd action="gen_ssc">
  <!-- Optional: -->
  <organization>string</organization>
  <domain>string</domain>
  <domain>string</domain>
  <ip>string</ip>
  <locality>string</locality>
  <country>string</country>
```

</cmd>

Command Attribute	Data Type	Attribute value definition
organization	<i>string</i>	Name of the organization this certificate is issued for.
domain	<i>string</i>	Domain name covered by the certificate. This can either be a full domain name or a wildcard DNS record marked by an *. Multiple entries are possible.
ip	<i>string</i>	IP covered by the certificate.
locality	<i>string</i>	Denotes the city in which the organization is located.
country	<i>string</i>	Contains the 2-character ISO format country code of the organization.

Response

<resp action="gen_ssc" time="int" filename="string"

access_area="user" offset="int" error="int">

<organization>string</organization>

<domain>string</domain>

<domain>string</domain>

<ip>string</ip>

<locality>string</locality>

<country>string</country>

<encodedfile>

<b64>string</b64>

</encodedfile>

<num_bytes>int</num_bytes>

<tot_bytes>int</tot_bytes>

<enc_bytes>int</enc_bytes>

<offset>int</offset>

<done>int</done>

</resp>

Response Element or Attribute	Data Type	Definition
time	<i>int</i>	Timestamp of certificate in epoch time.
filename	<i>string</i>	File name of the certificate summary.
organization	<i>string</i>	Name of the organization this certificate is issued for.
domain	<i>string</i>	Domain name covered by the certificate. This can either be a full domain name or a wildcard DNS record marked by an *. Multiple entries are possible.

ip	<i>string</i>	IP covered by the certificate.
locality	<i>string</i>	Denotes the city in which the organization is located.
country	<i>string</i>	Contains the 2-character ISO format country code of the organization.
b64	<i>string</i>	File with certificate summary as B64 encoded string.
num_bytes	<i>int</i>	Number of bytes of the encoded file.
enc_bytes	<i>int</i>	Number of bytes of the encoded file.
tot_bytes	<i>int</i>	Number of bytes of the decoded file.
offset	<i>int</i>	Offset for files split in multiple commands (not used).
done	<i>int</i>	Flag marking the file as complete (not used).

EXAMPLE:

Request:

```
<cmd action="gen_ssc">
  <!-- Optional: -->
  <organization>Danfoss</organization>
  <domain>sm800a.danfoss.com</domain>
  <domain>*.danfoss.com</domain>
  <ip>192.178.0.17</ip>
  <locality>Nordborg</locality>
  <country>DK</country>
</cmd>
```

Response:

```
<resp action="gen_ssc" time="int" filename="cert.info"
  access_area="user" offset="0" error="0">
  <organization>Danfoss</organization>
  <domain>sm70.danfoss.com</domain>
  <domain>*.danfoss.net</domain>
  <ip>192.168.0.17</ip>
  <locality>Nordborg</locality>
  <country>DK</country>
  <encodedfile>
    <b64>
      ...
    </b64>
  </encodedfile>
```

```

    <num_bytes>3604</num_bytes>
    <tot_bytes>2703</tot_bytes>
    <enc_bytes>3604</enc_bytes>
    <offset>3604</offset>
    <done>1</done>
</resp>

```

2.2 gen_csr

The command generates a certificate signing request. The organization and at least one domain or ip must be submitted, otherwise an error will be returned (error code: 68). Other fields are optional. Currently, up to three domains and one IP are supported.

Command

```

<cmd action="gen_csr">
    <!-- Mandatory: -->
    <organization>string</organization>
    <domain>string</domain>
    <!-- Optional: -->
    <domain>string</domain>
    <ip>string</ip>
    <locality>string</locality>
    <country>string</country>
</cmd>

```

Command Attribute	Data Type	Attribute value definition
organization	string	Name of the organization this certificate is issued for.
domain	string	Domain name covered by the certificate. This can either be a full domain name or a wildcard DNS record marked by an *. Multiple entries are possible.
ip	string	IP covered by the certificate.
locality	string	Denotes the city in which the organization is located.
country	string	Contains the 2-character ISO format country code of the organization.

Response

```

<resp action="gen_csr" time="int" filename="string"
    access_area="string" offset="int" error="int">
    <organization>string</organization>

```

```

<domain>string</domain>
<domain>string</domain>
<ip>string</ip>
<locality>string</locality>
<country>string</country>
<encodedfile>
    <b64>string</b64>
</encodedfile>
<num_bytes>int</num_bytes>
<tot_bytes>int</tot_bytes>
<enc_bytes>int</enc_bytes>
<offset>int</offset>
<done>int</done>
</resp>

```

Response Element or Attribute	Data Type	Definition
Time	<i>int</i>	Timestamp of certificate in epoch time.
Filename	<i>string</i>	File name of the certificate signing request.
Organization	<i>string</i>	Name of the organization this certificate is issued for.
Domain	<i>string</i>	Domain name covered by the certificate. This can either be a full domain name or a wildcard DNS record marked by an *. Multiple entries are possible.
Ip	<i>string</i>	IP covered by the certificate.
Locality	<i>string</i>	Denotes the city in which the organization is located.
Country	<i>string</i>	Contains the 2-character ISO format country code of the organization.
b64	<i>string</i>	File with the certificate signing request as B64 encoded string.
num_bytes	<i>int</i>	Number of bytes of the encoded file.
enc_bytes	<i>int</i>	Number of bytes of the encoded file.
tot_bytes	<i>int</i>	Number of bytes of the decoded file.
offset	<i>int</i>	Offset for files split in multiple commands (not used).
done	<i>int</i>	Flag marking the file as complete (not used).

EXAMPLE:

Request:

```
<cmd action="gen_csr">
```

```
<!-- Optional: -->
<organization>Danfoss</organization>
<domain>sm800a.danfoss.com</domain>
<domain>*.danfoss.com</domain>
<ip>192.178.0.17</ip>
<locality>Nordborg</locality>
<country>DK</country>
</cmd>
```

Response:

```
<resp action="gen_csr" time="int" filename="sm800-danfoss-com.csr"
  access_area="user" offset="0" error="0">
  <organization>Danfoss</organization>
  <domain>sm800a.danfoss.com</domain>
  <domain>*.danfoss.com</domain>
  <ip>192.168.0.17</ip>
  <locality>Nordborg</locality>
  <country>DK</country>
  <encodedfile>
    <b64>
      ...
    </b64>
  </encodedfile>
  <num_bytes>3604</num_bytes>
  <tot_bytes>2703</tot_bytes>
  <enc_bytes>3604</enc_bytes>
  <offset>3604</offset>
  <done>1</done>
</resp>
```

2.3 load_cert

The command loads a signed certificate to the system manager. The certificate file is passed as a b64 encoded string. The ending of the file must have a .pem extension to indicate that a certificate is uploaded. The system manager returns a string representation of the current certificate or an error if the uploaded certificate is not valid (error code: 132).

The name of the uploaded certificate should match the name of the CSR, i.e. the system manager expects a file named 'sm800-danfoss-com.pem' when a CSR named 'sm800-danfoss-com.csr' has been generated before. Otherwise, the user will be informed that no matching private key was found (error code: 133). If the upload was successful, the system manager responds with a b64 encoded file containing the summary of the updated certificate.

Command

```
<cmd done="1" action="load_cert" offset="int" index="int" filename="string" >
<b64>string<b64>
</cmd>
```

Command Attribute	Data Type	Attribute value definition
offset	<i>int</i>	The number of bytes in the b64 encoded string.
done	<i>int</i>	Indicates that the file is uploaded in one piece. Must be set to 1.
index	<i>int</i>	File index (not used).
filename	<i>string</i>	Name of the certificate file (.pem).
b64	<i>string</i>	B64 encoded signed certificate file (.pem).

Response

```
<resp action="load_cert" time="int" filename="string"
  access_area="string" offset="int" error="int">
  <encodedfile>
    <b64>string</b64>
  </encodedfile>
  <num_bytes>int</num_bytes>
  <tot_bytes>int</tot_bytes>
  <enc_bytes>int</enc_bytes>
  <offset>int</offset>
  <done>int</done>
</resp>
```

Response Element or Attribute	Data Type	Definition
Time	<i>int</i>	Timestamp of certificate in epoch time.
filename	<i>string</i>	File name of the certificate info file.
b64	<i>string</i>	File with the certificate info encoded as b64 string.
num_bytes	<i>int</i>	Number of bytes of the encoded info file.
enc_bytes	<i>int</i>	Number of bytes of the encoded info file.
tot_bytes	<i>int</i>	Number of bytes of the decoded info file.

offset	<i>int</i>	Offset for files split in multiple commands (not used).
done	<i>int</i>	Flag marking the file as complete (not used).

EXAMPLE:

Request:

```
<cmd done="1" action="load_cert" offset="1232" index="1" filename="sm800-danfoss-com.csr.pem" >
<b64>...</b64>
</cmd>
```

Response:

```
<resp action="load_cert" time="int" filename="sm800-danfoss-com.csr.pem "
  access_area="user" offset="0" error="0">
  <encodedfile>
    <b64>
      ...
    </b64>
  </encodedfile>
  <num_bytes>3604</num_bytes>
  <tot_bytes>2703</tot_bytes>
  <enc_bytes>3604</enc_bytes>
  <offset>3604</offset>
  <done>1</done>
</resp>
```

2.4 load_key_pair

The command loads a presigned certificate key pair to the system manager. The private key and the certificate file are passed as b64 encoded strings. The endings of the files must be '.key' and '.pem' to indicate that a key pair is uploaded. The system manager returns a string representation of the current certificate or an error if the uploaded certificate is not valid (error code: 132). If the upload was successful, the system manager responds with a b64 encoded file containing the summary of the updated certificate.

Command

```
<cmd action="load_key_pair">
  <key done="1" offset="int" index="1" filename="string">
```

```
<b64>string</b64>
</key>
<cert done="1" offset=" int " index="1" filename="string">
<b64>string</b64>
</cert>
</cmd>
```

Command Attribute	Data Type	Attribute value definition
offset	<i>int</i>	The number of bytes in the b64 encoded string.
done	<i>int</i>	Indicates that the file is uploaded in one piece. Must be set to 1.
index	<i>int</i>	File index (not used).
filename	<i>string</i>	Name of key or certificate file.
b64	<i>string</i>	B64 encoded string of key (.key) or certificate file (.pem)

Response

```
<resp action="load_key_pair" time="int" error="0">
  <key done="1" filename="string" index="1" offset="int" file_id="int">
    </key>
  <cert done="1" filename="string" index="1" offset="int" file_id="int">
    </cert>
  <encodedfile>
    <b64>string</b64>
  </encodedfile>
  <num_bytes>int</num_bytes>
  <tot_bytes>int</tot_bytes>
  <enc_bytes>int</enc_bytes>
  <offset>int</offset>
</done>1</done>
</resp>
```

Response Element or Attribute	Data Type	Definition
Time	<i>int</i>	Timestamp of certificate in epoch time.
Filename	<i>string</i>	File name of the certificate signing request.
b64	<i>string</i>	File with the certificate summary as B64 encoded string.
num_bytes	<i>int</i>	Number of bytes of the certificate summary as encoded file.

enc_bytes	int	Number of bytes of the encoded file.
tot_bytes	int	Number of bytes of the certificate summary as decoded file.
offset	int	Offset for files split in multiple commands (not used).
done	int	Flag marking the file as complete (not used).

EXAMPLE:

Request:

```
<cmd action="load_key_pair">
  <key done="1" offset="1751" index="1" filename="danfoss-net.key">
    <b64>...</b64>
  </key>
  <cert done="1" offset="1350" index="1" filename=" danfoss-net.pem">
    <b64>...</b64>
  </cert>
</cmd>
```

Response:

```
<resp action="load_key_pair" time="1677038051" error="0">
  <key done="1" filename="danfoss-net.key" index="1" offset="2272" file_id="10">
    </key>
  <cert done="1" filename="danfoss-net.pem" index="1" offset="1748" file_id="11">
    </cert>
  <encodedfile>
    <b64>...</b64>
  </encodedfile>
  <num_bytes>4084</num_bytes>
  <tot_bytes>3063</tot_bytes>
  <enc_bytes>4084</enc_bytes>
  <offset>4084</offset>
<done>1</done>
</resp>
```

3 Reference Information

Node Types

Node Type	Definition
0	On/Off Input
1	Relay Output
2	Sensor Input
3	Variable Output
16	Generic Device
255	Empty Node

3.1 Error Codes

Error Code	Definition
0	No error - command was successful
1	Response is too large for the XML transmission buffer which is 32K bytes
2	No attributes associated with cmd element or Command has no “cmd” element or Could not write to a variable Could not read a variable Could not read a point Could not write on/off input or relay output operation mode
3	Bad command input
4	Error in CGI post parameters
5	Error converting the XML response from its internal representation to the response XML string
6	Error decoding URL string
7	Memory Overflow
8	Response data compression failed
9	Error parsing request
10	Data access error - a number of causes that are not further resolved
11	Undefined command action attribute value Attribute value is invalid
12	Error in Alarm Reference Number or Alarm not configured
13	The input cgi parameter string is too long.
14	Device address is not found
15	Failed in authorization to write
16	No history configured
17	No history data returned, can be time error
18	Create file errors
19	Read file errors
20	Exceeded maximum size of device list
21	not used
22	Invalid History Index
23	Could not find board point index from nodetype,node,mod, and point.
24	Could not find History Index
25	History query has not been initialized before attempting to start a history query.
26	An hq_start_query has been attempted while a history query is active
27	No command attributes found when some are expected
28	A beginning of epoch has been provided without an end or vise-versa
29	Could not convert from time and date format to epoch seconds
30	Command requires a query_id but none provided
31	Error building XML response
32	The provided query_id does not match an active query
33	An hq_get_data command has been attempted but data is not yet available
34	A sample rate has been provided that is not one of those allowed
35	The averaging interval is less than or equal to the sample rate at which data has been recorded

36	The stop epoch time is less than the start epoch time or the stop epoch time minus the start epoch time is less than the averaging interval
37	The device_id provided does not match any currently known to the System Manager. Note that only devices that have been configured are known to the System Manager
38	No Id attribute
39	Invalid Id attribute
40	No schedule number
41	Invalid schedule number
42	No schedule details
43	Missing on_time
44	Missing hour
45	Invalid hour
46	Missing minute
47	Invalid minute
48	Missing off_time
49	Missing weekdays
50	Invalid weekdays
51	Missing holidays
52	Invalid holidays
53	Missing or invalid holiday_start
54	Missing or invalid holiday_end
55	Missing or invalid holiday_open
56	Missing or invalid holiday_close
57	Holiday not configured
58	Missing description
59	Cannot clear alarm
60	Missing or invalid store_open
61	Missing or invalid store_closed
62	Alarm Not Configured
63	Node Offline
64	Missing Index
65	Invalid Index
66	Relay Cannot be Timed On
67	The name being changed has too many characters. If changing the store_name, or unit_name, must be <= 16 characters, if changing the storeId1, or storeId2, must be <= 8 characters.
68	Missing elements associated with the command element.
69	Tag_ID not Defined
70	Missing Leaf Node for Data
71	No Change in Data
72	Failed to update database
73	Not writable
77	Invalid type
78	System busy
79	Asset not Configured for Load Shed
80	Invalid Power Rating for Asset
81	Invalid Start-up Delay for Asset

82	Config Busy
83	I/O Scanning in Progress
84	Too Many Defrosts in Progress
85	Trying to load a bad database version into unit
86	Database load failed
87	No available web memory available
88	Demand response not configured (Only the AK355 system)

Danfoss A/S

Climate Solutions • danfoss.com • +45 7488 2222

Any information, including, but not limited to information on selection of product, its application or use, product design, weight, dimensions, capacity or any other technical data in product manuals, catalogues descriptions, advertisements, etc. and whether made available in writing, orally, electronically, online or via download, shall be considered informative, and is only binding if and to the extent, explicit reference is made in a quotation or order confirmation. Danfoss cannot accept any responsibility for possible errors in catalogues, brochures, videos and other material. Danfoss reserves the right to alter its products without notice. This also applies to products ordered but not delivered provided that such alterations can be made without changes to form, fit or function of the product.

All trademarks in this material are property of Danfoss A/S or Danfoss group companies. Danfoss and the Danfoss logo are trademarks of Danfoss A/S. All rights reserved.