

# Test Procedures for NTCIP-conformant Closed Circuit Television (CCTV) Camera Controllers

Version 01.02

December 2002

Prepared for:

E N T E R  P R I S E

Project Manager: Manny Agah (Arizona DOT)

In Cooperation with:



**FHWA Joint Program Office**



**I-95 Corridor  
Coalition**

**I-95 Corridor Coalition**

and

Prepared by:

**TREVILON™**



# Table of Contents

<b>INTRODUCTION</b> .....	<b>4</b>
ACKNOWLEDGMENTS .....	4
<b>RULES FOR FOLLOWING TEST PROCEDURES</b> .....	<b>5</b>
<b>EXECUTIVE SUMMARY</b> .....	<b>6</b>
<b>CCTV TEST CASES</b> .....	<b>12</b>
TEST CASE TC1: IDENTIFY DEVICE.....	12
TEST CASE TC2: IDENTIFY PRESET POSITION RANGE .....	13
TEST CASE TC3: IDENTIFY PAN LIMITS .....	14
TEST CASE TC4: TRUE NORTH OFFSET.....	16
TEST CASE TC5: IDENTIFY TILT LIMITS.....	17
TEST CASE TC6: IDENTIFY ZOOM LIMIT .....	18
TEST CASE TC7: IDENTIFY FOCUS LIMIT .....	19
TEST CASE TC8: IDENTIFY IRIS LIMIT .....	19
TEST CASE TC9: IDENTIFY PAN-TILT STEP ANGLE MINIMUM .....	20
TEST CASE TC10: GET AND SET LABEL.....	20
TEST CASE TC11: DISPLAY CAMERA LOCATION .....	24
TEST CASE TC12: DELTA PAN MOTION .....	29
TEST CASE TC13: ABSOLUTE PAN MOTION .....	30
TEST CASE TC14: CONTINUOUS PAN MOTION .....	31
TEST CASE TC15: CONTINUOUS PAN MOTION WITH STOP.....	32
TEST CASE TC16: DELTA TILT MOTION .....	34
TEST CASE TC17: ABSOLUTE TILT MOTION.....	35
TEST CASE TC18: CONTINUOUS TILT MOTION.....	37
TEST CASE TC19: CONTINUOUS TILT MOTION WITH STOP .....	38
TEST CASE TC20: DELTA ZOOM MOTION.....	40
TEST CASE TC21: ABSOLUTE ZOOM MOTION .....	41
TEST CASE TC22: CONTINUOUS ZOOM MOTION .....	42
TEST CASE TC23: CONTINUOUS ZOOM MOTION WITH STOP.....	44
TEST CASE TC24: DELTA FOCUS MOTION .....	46
TEST CASE TC25: ABSOLUTE FOCUS MOTION .....	47
TEST CASE TC26: CONTINUOUS FOCUS MOTION .....	48
TEST CASE TC27: CONTINUOUS FOCUS MOTION WITH STOP.....	50
TEST CASE TC28: DELTA IRIS MOTION .....	51
TEST CASE TC29: ABSOLUTE IRIS MOTION .....	53
TEST CASE TC30: CONTINUOUS IRIS MOTION .....	54
TEST CASE TC31: CONTINUOUS IRIS MOTION WITH STOP .....	55
TEST CASE TC32: PRESET POSITON .....	57
TEST CASE TC33: GET-SET ZONE .....	60
TEST CASE TC34: MOVE IN AND OUT OF ZONE.....	63
TEST CASE TC35: GET AVAILABILITY OF EQUIPMENT.....	68
TEST CASE TC36: CONTROL CAMERA POWER .....	69
TEST CASE TC37: CONTROL HEATER POWER.....	71
TEST CASE TC38: CONTROL WIPER.....	74
TEST CASE TC39: CONTROL WASHER .....	76

TEST CASE TC40: CONTROL BLOWER .....	78
TEST CASE TC41: GET AVAILABILITY OF LENS EQUIPMENT .....	80
TEST CASE TC42: CONTROL AUTO IRIS.....	81
TEST CASE TC43: CONTROL AUTO FOCUS .....	83
TEST CASE TC44: CABINET ALARM.....	85
TEST CASE TC45: ENCLOSURE ALARM .....	90
TEST CASE TC46: VIDEO LOSS ALARM .....	95
TEST CASE TC47: TEMPERATURE ALARM.....	99
TEST CASE TC48: PRESSURE ALARM.....	105
TEST CASE TC49: LOCAL-REMOTE ALARM.....	110
TEST CASE TC50: WASHER FLUID ALARM.....	115
TEST CASE TC51: MONITOR DISCRETE INPUT.....	120
TEST CASE TC52: MONITOR DISCRETE OUTPUT .....	125
TEST CASE TC53: PAGE DOWN.....	133
TEST CASE TC54: PAGE UP.....	133
TEST CASE TC55: CURSOR UP .....	134
TEST CASE TC56: CURSOR DOWN .....	134
TEST CASE TC57: CURSOR RIGHT .....	135
TEST CASE TC58: CURSOR LEFT.....	135
TEST CASE TC59: INCREMENT .....	136
TEST CASE TC60: DECREMENT VALUE .....	136
TEST CASE TC61: ENTER VALUE .....	137
TEST CASE TC62: ACTIVATE MENU .....	137
TEST CASE TC63: ACTIVATE MENU WITH TIMEOUT .....	138
TEST CASE TC64: DEACTIVATE MENU .....	138
<b>ANNEX – QUESTIONS ABOUT NTCIP 1205.....</b>	<b>140</b>

## Introduction

This document provides an initial set of test procedures for CCTV cameras that claim conformance to NTCIP 1205 per the ENTERPRISE/I-95 Procurement Specification Guide. These procedures have been written in a tool-neutral manner such that they can be performed by a variety of SNMP test tools, including the NTCIP Exerciser, the public domain tool developed by the Federal Highway Administration in 1996, and NTester, a new NTCIP conformance tester developed in cooperation with ENTERPRISE, the I-95 Corridor Coalition, and the FHWA.

The procedures listed in this document cover all of the various capabilities addressed by NTCIP 1205; however, many of these features are optional. Further, the procedures define several variables that should be customized to reflect project-specific limitations and/or to ensure robust testing. Thus, prior to performing a test according to these procedures, a tester should develop a formal test plan, that, among other things, defines the exact test cases to be implemented and the variables to use for each test case. For more information about test documentation see IEEE 829-1998.

This document is intended for manufacturers, system integrators, and other advanced users that are interested in testing a device for conformance. The nature of such testing is necessarily complex and requires extensive knowledge of the NTCIP standards. This document assumes that the reader possesses this knowledge. More information about the NTCIP is available from <http://www.ntcip.org>.

It should be noted that during the process of developing these test procedures, many ambiguities and issues were uncovered regarding NTCIP 1205. These issues were addressed as best as possible given the limitations of the project, but further clarification of these issues should be obtained from the Working Group prior to formally declaring a device passed or failed. A list of questions that have been submitted to the WG are contained in the Annex.

## ***Acknowledgments***

This document was originally prepared as a part of a cooperative effort among ENTERPRISE (<http://enterprise.prog.org/inch.html>), the I-95 Corridor Coalition, and the FHWA. The work was performed by Trevilon Corporation under a contract with the Arizona Department of Transportation.

## Rules for Following Test Procedures

The device under test shall 'pass' the test case if, and only if, it is awarded all possible points for the given test case. Otherwise it shall 'fail' the test case.

Each test case is divided up among several steps. The rules for each step type are defined as follows:

**DEFINE** – Most test cases start with one or more DEFINE steps. When implementing a DEFINE step the tester shall define a logical variable for later use in the test case. The value to be entered for this variable is defined by the Description clause and the Action Clause defines the precise syntax of the variable, including: the data type, the allowed range, and its default value. The Action clause also indicates whether the user should customize this variable for their specific test plan.

**GET** – When implementing a GET step, the tester shall send one or more SNMP Get commands containing the Object Identifiers for the Objects listed in the table to the device under test. The Object Identifiers may appear in any order and in any combination of GET commands (unless otherwise noted). The step also implicitly includes the decoding of the response and storing the requested data into the logical variable names shown in the table to the right of each object name so that they may be referenced later. The logical variables shall have the same syntax as the object retrieved (unless otherwise noted).

**SET** – When implementing a SET step, the tester shall send one or more SNMP Set commands containing the Object Identifiers and values for the Objects listed in the table to the device under test. The Object Identifiers may appear in any order (but shall be properly paired with the values) and in any combination of SET commands (unless otherwise noted). The values for block objects are given in '+' notation with each plus symbol indicating the linkage between the component field values.

**VERIFY/GOTO** – When implementing a VERIFY/GOTO step, the tester shall perform the comparison (typically between two variables or a variable and a constant) and then implement the indicated steps (a point assignment and/or goto command if passed and possibly a goto command if failed). A VERIFY/GOTO is intended to be a largely automated process and can be performed using information previously stored.

**USER-VERIFY** – When implementing a USER-VERIFY step, the tester shall perform the manual inspection indicated and record the result as pass or failed. In some instances, this step is merely used to allow the user to perform some manual action, in which case no points are assigned and the selection of pass or fail have no impact on the test. The failed condition may also have a goto command associated with it.

**PRECONDITION** – When implementing a PRECONDITION step, the tester shall ensure that the precondition has been met prior to performing the test.

**POSTCONDITION** – When implementing a POSTCONDITION step, the tester shall be aware that the device is left in an altered state from the starting condition.

**DELAY** – When implementing a DELAY step, the tester shall delay at least the time shown prior to implementing the next step.

## Executive Summary

This section contains a listing of test cases included within this document. The test case identifier, name, and description are shown in the following table.

Identifier	Name	Description
TC1	Identify Device	This test case ensures that the device under test contains valid information for the module make, model, and version number as well as other related information.
TC2	Identify Preset Position Range	This test case ensures that the device indicates that it supports the required number of preset positions.
TC3	Identify Pan Limits	This test case identifies and verifies the left and right panning limits and the home position of the device.
TC4	True North Offset	This test case ensures that the user can configure the true north setting in the camera.
TC5	Identify Tilt Limits	This test case ensures that the device indicates that it supports up and down tilting limits of the device.
TC6	Identify Zoom Limit	This test case ensures that the device indicates that it supports the required zoom limit of the device.
TC7	Identify Focus Limit	This test case ensures that the device indicates that it supports the required focus limit of the device
TC8	Identify Iris Limit	This test ensures that the device indicates that it supports the required iris limit of the device.
TC9	Identify Pan-Tilt Step Angle Minimum	This test case ensures that the device indicates that it supports the pan and tilt step angle minimum of the device
TC10	Get and Set Label	This test case verifies the number of labels the device can store. Test labels are stored in the device to verify storage capabilities
TC11	Display Camera Location	This test case tests the capability to

		display a text label on the video output.
TC12	Delta Pan Motion	This test case tests the delta panning motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC13	Absolute Pan Motion	This test case tests the absolute panning motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC14	Continuous Pan Motion	This test case tests the continuous panning motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.
TC15	Continuous Pan Motion with Stop	This test case tests the continuous panning motion of the camera by moving the camera and using the stop command to stop movement.
TC16	Delta Tilt Motion	This test case tests the delta tilt motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC17	Absolute Tilt Motion	This test case tests the absolute tilt motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC18	Continuous Tilt Motion	This test case tests the continuous tilt motion of the camera by moving the camera with with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.
TC19	Continuous Tilt Motion with Stop	This test case tests the continuous tilting motion of the camera by moving the camera and using the stop command to stop movement.

TC20	Delta Zoom Motion	This test case tests the delta zoom motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC21	Absolute Zoom Motion	This test case tests the absolute zoom motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC22	Continuous Zoom Motion	This test case tests the continuous zoom motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera
TC23	Continuous Zoom Motion with Stop	This test case tests the continuous zooming motion of the camera by moving the camera and using the stop command to stop movement.
TC24	Delta Focus Motion	This test case tests the delta focus motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them
TC25	Absolute Focus Motion	This test case tests the absolute focus motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC26	Continuous Focus Motion	This test case tests the continuous focus motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.
TC27	Continuous Focus Motion with Stop	This test case tests the continuous focus motion of the camera by moving the camera and using the stop command to stop movement.
TC28	Delta Iris Motion	This test case tests the delta iris motion of the camera by moving the camera with

		the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.
TC29	Absolute Iris Motion	This test case tests the absolute iris motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.
TC30	Continuous Iris Motion	This test case tests the continuous iris motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.
TC31	Continuous Iris Motion with Stop	This test case tests the continuous Iris motion of the camera by moving the camera and using the stop command to stop movement.
TC32	Preset Positon	This test case stores and moves the camera to preset camera positions
TC33	Get-Set Zone	This test case tests the storage of camera zones.
TC34	Move In and Out of Zone	This test case tests the labelling capability of zones by moving to areas within zones
TC35	Get Availability of Equipment	This test case identifies and verifies the availability of attached equipment to the camera.
TC36	Control Camera Power	This test case enables and disables this feature while the user verifies.
TC37	Control Heater Power	This test case enables and disables this feature while the user verifies.
TC38	Control Wiper	This test case enables and disables this feature while the user verifies.
TC39	Control Washer	This test case enables and disables this feature while the user verifies.
TC40	Control Blower	This test case enables and disables this feature while the user verifies.
TC41	Get Availability of Lens	This test case identifies and verifies the

	Equipment	availability of attached equipment to the camera.
TC42	Control Auto Iris	This test case enables and disables this feature while the user verifies.
TC43	Control Auto Focus	This test case enables and disables this feature while the user verifies
TC44	Cabinet Alarm	This test case tests the cabinet open alarm and the label associated with it.
TC45	Enclosure Alarm	This test case tests the enclosure alarm and the label associated with it.
TC46	Video Loss Alarm	This test case tests the video loss alarm and the label associated with it.
TC47	Temperature Alarm	This test case tests the temperature alarm and the label, thresholds, and current value associated with it.
TC48	Pressure Alarm	This test case tests the pressure alarm and the label, thresholds, and current value associated with it.
TC49	Local-Remote Alarm	This test case tests the local-remote alarm and the label associated with it.
TC50	Washer Fluid Alarm	This test case tests the washer fluid alarm and the label, thresholds, and current value associated with it.
TC51	Monitor Discrete Input	This test case verifies the state of discrete inputs and the label associated with it.
TC52	Monitor Discrete Output	This test case verifies the state of discrete outputs and the label associated with it.
TC53	Page Down	This test case sends a page down command to the CCTV.
TC54	Page Up	This test case sends a page up command to the CCTV
TC55	Cursor Up	This test case sends a cursor up command to the CCTV.
TC56	Cursor Down	This test case sends a cursor down command to the CCTV.
TC57	Cursor Right	This test case sends a cursor right command to the CCTV.

TC58	Cursor Left	This test case sends a cursor left command to the CCTV.
TC59	Increment	This test case sends an increment command to the CCTV.
TC60	Decrement Value	This test case sends and decrement command to the CCTV.
TC61	Enter Value	This test case sends an enter command to the CCTV
TC62	Activate Menu	This test case activates the menu of the CCTV.
TC63	Activate Menu with Timeout	This test case activates the menu with a timeout value. When the time has expired, the menu shall turn off.
TC64	Deactivate Menu	This test case turns off the internal camera menu.

# CCTV Test Cases

## Test Case *TC1: Identify Device*

This test case ensures that the device under test contains valid information for the module make, model, and version number as well as other related information.

### Step 1: DEFINE

Description: Please enter the number of rows required in the device's module table per the project requirements.

Action: Define the following variable(s) for later use

Variable Name: req\_globalMaxModules

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

### Step 2: DEFINE

Description: Please enter the module number for which you wish to retrieve make, model, and version information.

Action: Define the following variable(s) for later use

Variable Name: req\_globalModuleEntry

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

### Step 3: GET

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	globalMaxModules.0	globalMaxModules

### Step 4: VERIFY/GOTO

If globalMaxModules >= req\_globalMaxModules

Award Points: 1

### Step 5: GET

Description: A user specified entry of the global module table will be retrieved in this step.

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	moduleDeviceNode.<req_globalModuleEntry>	moduleDeviceNode
	moduleMake.<req_globalModuleEntry>	moduleMake
	moduleModel.<req_globalModuleEntry>	moduleModel
	moduleVersion.<req_globalModuleEntry>	moduleVersion

moduleType.<req_globalModuleEntry>	moduleType
------------------------------------	------------

**Step 6: VERIFY/GOTO**

Description: This step ensures that the module references the CCTV node (i.e., 1.3.6.1.4.1.1206.4.2.7).

If moduleDeviceNode equals 1.3.6.1.4.1.1206.4.2.7

Award Points: 1

**Step 7: USER-VERIFY**

Description: Please verify that the device is reporting the manufacturer name for the requested module.

Points: 1

**Step 8: USER-VERIFY**

Description: Please verify that the device is reporting the model number of the module for the device under test.

Points: 1

**Step 9: USER-VERIFY**

Description: Please verify that the device is reporting the version of the module for the device under test.

Points: 1

**Step 10: USER-VERIFY**

Description: Please verify the module type being reported is correct for the module of the device under test.

Points: 1

***Test Case TC2: Identify Preset Position Range***

This test case ensures that the device indicates that it supports the required number of preset positions.

**Step 1: DEFINE**

Description: Please enter the number of preset positions that the CCTV procurement specifications required.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeMaxPreset

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 2: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	rangeMaxPreset.0	rangeMaxPreset

**Step 3: VERIFY/GOTO**

If rangeMaxPreset >= req\_rangeMaxPreset

Award Points: 1

**Test Case TC3: Identify Pan Limits**

This test case identifies and verifies the left and right panning limits and the home position of the device.

**Step 1: DEFINE**

Description: Please enter the speed at which you wish to pan the camera.

Action: Define the following variable(s) for later use

Variable Name: absolutePanSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 2: DEFINE**

Description: Please enter the pan left limit per the project requirements.

Action: Define the following variable(s) for later use

Variable Name: req\_rangePanLeftLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 27000

**Step 3: DEFINE**

Description: Please enter the pan right limit per the project requirements.

Action: Define the following variable(s) for later use

Variable Name: req\_rangePanRightLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 9000

**Step 4: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
--	--------	-----------

	rangePanRightLimit.0	rangePanRightLimit
	rangePanLeftLimit.0	rangePanLeftLimit

**Step 5: VERIFY/GOTO**

Description: Verify the left limit received is appropriate for the device. This object is measured in 1/100th degree units in a clockwise direction from the Home Position. NOTE: Since the limit is measured in the clockwise direction, a smaller number indicates a greater left-pan limit

If rangePanLeftLimit <= req\_rangePanLeftLimit  
Award Points: 1

**Step 6: VERIFY/GOTO**

Description: Verify the right limit received is appropriate for the device. This object is measured in 1/100th degree units in a clockwise direction from the Home Position.

If rangePanRightLimit >= req\_rangePanRightLimit  
Award Points: 1

**Step 7: SET**

Description: Move to Home Position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + absolutePanSpeed + '0'

**Step 8: USER-VERIFY**

Description: The camera was just panned to home position. Look at the direction the camera is pointed and remember this direction as home position. After pressing enter, note in which direction the camera is panning.

Points: 0

**Step 9: SET**

Description: Move to left limit

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + (-absolutePanSpeed) + rangePanLeftLimit

**Step 10: USER-VERIFY**

Description: The camera was just panned to the left limit. Verify that the camera panned in a leftward motion to a position <(rangePanLeftLimit/100)> clockwise degrees from home.

Points: 1

**Step 11: SET**

Description: Move to home position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + absolutePanSpeed + rangePanHomePosition

**Step 12: USER-VERIFY**

Description: Verify the camera moved back to the home position

Points: 1

**Step 13: SET**

Description: Move to right limit

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + absolutePanSpeed + rangePanRightLimit

**Step 14: USER-VERIFY**

Description: The camera was just panned to the right limit. Verify that the camera panned in a rightward motion to a position  $\langle \text{rangePanRightLimit}/100 \rangle$  clockwise degrees from home.

Points: 1

**Test Case TC4: True North Offset**

This test case ensures that the user can configure the true north setting in the camera.

**Step 1: DEFINE**

Description: Please enter a test value for the true North offset; this should be in 1/100ths of degrees clockwise from the home position. For example, a value of 500 would indicate 5.00 clockwise degrees from home.

Action: Define the following variable(s) for later use

Variable Name: alt\_rangeTrueNorthOffset

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 0

**Step 2: DEFINE**

Description: Please enter an errant value for true North offset between 36000 and 65534.

Action: Define the following variable(s) for later use

Variable Name: errant\_rangeTrueNorthOffset

Type: INTEGER

Lower Bound: 36000

Upper Bound: 65534

Get Value from User: Yes

Default value: 45000

**Step 3: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
rangeTrueNorthOffset.0	rangeTrueNorthOffset

**Step 4: SET**

Action: Set the following object(s) to the value indicated:

Object	Value
rangeTrueNorthOffset.0	alt_rangeTrueNorthOffset

**Step 5: VERIFY/GOTO**

If SnmpResponse.error = noError

Award Points: 1

**Step 5: SET**

Action: Set the following object(s) to the value indicated:

Object	Value
rangeTrueNorthOffset.0	errant_rangeTrueNorthOffset

**Step 6: VERIFY/GOTO**

If SnmpResponse.error = badValue

Award Points: 1

**Step 7: SET**

Description: Restore the true North back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
rangeTrueNorthOffset.0	rangeTrueNorthOffset

**Test Case TC5: Identify Tilt Limits**

This test case ensures that the device indicates that it supports up and down tilting limits of the device.

**Step 1: DEFINE**

Description: Please enter the project minimum requirement for the up tilt limit of the camera in 1/100<sup>th</sup> of degrees. For example, 500 would indicate 5.00 degrees up.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeTiltUpLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 4500

**Step 2: DEFINE**

Description: Please enter the project minimum requirement for the down tilt limit of the camera in

1/100<sup>th</sup> of degrees lower than 36000. For example, 35500 would indicate 355.00 degrees or 5.00 degrees downward.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeTiltDownLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 4500

**Step 3: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	rangeTiltUpLimit.0	rangeTiltUpLimit
	rangeTiltDownLimit.0	rangeTiltDownLimit

**Step 4: VERIFY/GOTO**

If rangeTiltUpLimit >= req\_rangeTiltUpLimit

Award Points: 1

**Step 5: VERIFY/GOTO**

If rangeTiltDownLimit <= req\_rangeTiltDownLimit

Award Points: 1

**Test Case TC6: Identify Zoom Limit**

This test case ensures that the device indicates that it supports the required zoom limit.

**Step 1: DEFINE**

Description: Please enter the project requirement for the number of zoom levels.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeZoomLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 10000

**Step 2: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	rangeZoomLimit.0	rangeZoomLimit

**Step 3: VERIFY/GOTO**

Description: Verify the zoom limitation received from device is larger than or equal to the project requirements.

If rangeZoomLimit >= req\_rangeZoomLimit

Award Points: 1

## Test Case TC7: Identify Focus Limit

This test case ensures that the device indicates that it supports the required focus limit.

### Step 1: DEFINE

Description: Please enter the project requirement for the number of focus levels.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeFocusLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 10000

### Step 2: GET

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	rangeFocusLimit.0	rangeFocusLimit

### Step 3: VERIFY/GOTO

Description: Verify the focus limitation received from device is larger than or equal to the project requirements.

If rangeFocusLimit >= req\_rangeFocusLimit

Award Points: 1

## Test Case TC8: Identify Iris Limit

This test ensures that the device indicates that it supports the required iris limit.

### Step 1: DEFINE

Description: Please enter the project requirement for the number of iris levels.

Action: Define the following variable(s) for later use

Variable Name: req\_rangeIrisLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 10000

### Step 2: GET

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	rangeIrisLimit.0	rangeIrisLimit

### Step 3: VERIFY/GOTO

Description: Specifies the iris range in arbitrary units. Used for absolute or offset control. Zero (0) identifies that iris limits are not supported. This number represents the scalar zoom positioning beginning with zero (0) for open and ending with 65535 for closed. Open is defined as the largest aperture setting. Closed is defined as the smallest aperture setting.

If rangeIrisLimit >= req\_rangeIrisLimit  
Award Points: 1

### **Test Case TC9: Identify Pan-Tilt Step Angle Minimum**

This test case ensures that the device indicates that it supports the pan and tilt step angle minimum of the device

#### **Step 1: DEFINE**

Description: Please enter the project required value of the minimum angle that will cause the camera to pan in 1/100ths of degrees. For example, a value of 50 will indicate that any command to pan 0.50 degrees or more will result in the camera moving.

Action: Define the following variable(s) for later use  
Variable Name: req\_rangeMinimumPanStepAngle  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 1

#### **Step 2: DEFINE**

Description: Please enter the project required value of the minimum angle that will cause the camera to tilt in 1/100ths of degrees. For example, a value of 50 will indicate that any command to tilt 0.50 degrees or more will result in the camera moving..

Action: Define the following variable(s) for later use  
Variable Name: req\_rangeMinimumTiltStepAngle  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 1

#### **Step 3: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
rangeMinimumPanStepAngle.0	rangeMinimumPanStepAngle
rangeMinimumTiltStepAngle.0	rangeMinimumTiltStepAngle

#### **Step 4: VERIFY/GOTO**

If rangeMinimumPanStepAngle <= req\_rangeMinimumPanStepAngle  
Award Points: 1

#### **Step 5: VERIFY/GOTO**

If rangeMinimumTiltStepAngle <= req\_rangeMinimumTiltStepAngle  
Award Points: 1

### **Test Case TC10: Get and Set Label**

This test case verifies the number of labels the device can store. Test labels are stored in the device to verify storage capabilities

**Step 1: DEFINE**

Description: Please enter the number of labels that the device is required to support.

Action: Define the following variable(s) for later use

Variable Name: req\_labelMaximum

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 2: DEFINE**

Description: Please enter the label row number to test.

Action: Define the following variable(s) for later use

Variable Name: index\_labelIndex

Type: INTEGER

Lower Bound: 1

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 3: DEFINE**

Description: Please enter the text to which the test label should be set.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelText

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'TEST'

**Step 4: DEFINE**

Description: Please enter the font for the test label.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelFontType

Type: INTEGER

Lower Bound: 1

Upper Bound: 2

Get Value from User: Yes

Default value: 1

**Step 5: DEFINE**

Description: Please enter the height for the test label.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelHeight

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 6: DEFINE**

Description: Please enter the color for the test label.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelColor

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes

Default value: 10

**Step 7: DEFINE**

Description: Please enter the row position for the start of the label text. A value of zero indicates the top of the screen and a value of 255 indicates the bottom of the screen.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelStartRow

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 8: DEFINE**

Description: Please enter the column in which the test label should start. A value of zero indicates the left side of the screen, a value of 255 indicates the right side of the screen.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelStartColumn

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 9: DEFINE**

Description: Please enter the value to set a label table entry status to.

Action: Define the following variable(s) for later use

Variable Name: alt\_labelStatus

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 10: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	labelMaximum.0	labelMaximum

**Step 11: VERIFY/GOTO**

If labelMaximum >= req\_labelMaximum

Award Points: 1

**Step 12: GET**

Description: Get and store the original values

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<index_labelIndex>	labelText
labelFontType.<index_labelIndex>	labelFontType
labelHeight.<index_labelIndex>	labelHeight
labelColor.<index_labelIndex>	labelColor
labelStartRow.<index_labelIndex>	labelStartRow
labelStartColumn.<index_labelIndex>	labelStartColumn
labelStatus.<index_labelIndex>	labelStatus

**Step 14: SET**

Description: Set the alternate user specified label information.

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<index_labelIndex>	alt_labelText
labelFontType.<index_labelIndex>	alt_labelFontType
labelHeight.<index_labelIndex>	alt_labelHeight
labelColor.<index_labelIndex>	alt_labelColor
labelStartRow.<index_labelIndex>	alt_labelStartRow
labelStartColumn.<index_labelIndex>	alt_labelStartColumn
labelStatus.<index_labelIndex>	alt_labelStatus

**Step 16: GET**

Description: Get the label information to make sure it was set correctly.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<index_labelIndex>	labelText2
labelFontType.<index_labelIndex>	labelFontType2
labelHeight.<index_labelIndex>	labelHeight2
labelColor.<index_labelIndex>	labelColor2
labelStartRow.<index_labelIndex>	labelStartRow2
labelStartColumn.<index_labelIndex>	labelStartColumn2
labelStatus.<index_labelIndex>	labelStatus2

**Step 17: VERIFY/GOTO**

If labelText2 equals alt\_labelText  
Award Points: 1

**Step 18: VERIFY/GOTO**

If labelFontType2 equals labelFontType  
Award Points: 1

**Step 19: VERIFY/GOTO**

If labelHeight2 equals labelHeight  
Award Points: 1

**Step 20: VERIFY/GOTO**

If labelColor2 equals labelColor  
Award Points: 1

**Step 21: VERIFY/GOTO**

If labelStartRow2 equals labelStartRow  
Award Points: 1

**Step 22: VERIFY/GOTO**

If labelStartColumn2 equals labelStartColumn  
Award Points: 1

**Step 23: VERIFY/GOTO**

If labelStatus2 equals labelStatus  
Award Points: 1

**Step 24: SET**

Description: Set the label back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<index_labelIndex>	labelText
labelFontType.<index_labelIndex>	labelFontType
labelHeight.<index_labelIndex>	labelHeight
labelColor.<index_labelIndex>	labelColor
labelStartRow.<index_labelIndex>	labelStartRow
labelStartColumn.<index_labelIndex>	alt_labelStartColumn
labelStatus.<index_labelIndex>	labelStatus

## ***Test Case TC11: Display Camera Location***

This test case tests the capability to display a text label on the video output.

**Step 1: DEFINE**

Description: Please enter the label row to use in the lable table for testing storing labels.

Action: Define the following variable(s) for later use

Variable Name: index\_LocationLabelIndex

Type: INTEGER

Lower Bound: 1  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 2

**Step 2: DEFINE**

Description: Please enter the value to set a label table entry text to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_LocLabelText  
Type: STRING  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 'LOCATION'

**Step 3: DEFINE**

Description: Please enter the value to set a label table entry font type to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_LocLabelFontType  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 1

**Step 4: DEFINE**

Description: Please enter the value to set a label table entry height to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_LocLabelHeight  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 5: DEFINE**

Description: Please enter the value to set a label table entry color to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_LocLabelColor  
Type: INTEGER  
Lower Bound: 1  
Upper Bound: 16  
Get Value from User: Yes  
Default value: 11

**Step 6: DEFINE**

Description: Please enter the value to set a label table entry start row to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_LocLabelStartRow  
Type: INTEGER  
Lower Bound: 0

Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 1

**Step 7: DEFINE**

Description: Please enter the value to set a label table entry start column to.

Action: Define the following variable(s) for later use  
 Variable Name: alt\_LocLabelStartColumn  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 1

**Step 8: DEFINE**

Description: Please enter the value to set a label table entry status to.

Action: Define the following variable(s) for later use  
 Variable Name: alt\_LocLabelStatus  
 Type: STRING  
 Lower Bound: 1  
 Upper Bound: 1  
 Get Value from User: Yes  
 Default value: 0xC0

**Step 9: GET**

Description: Get the current label setting and store for restoration later.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<index_LocationLabelIndex>	orig_labelText
labelFontType.<index_LocationLabelIndex>	orig_labelFontType
labelHeight.<index_LocationLabelIndex>	orig_labelHeight
labelColor.<index_LocationLabelIndex>	orig_labelColor
labelStartRow.<index_LocationLabelIndex>	orig_labelStartRow
labelStartColumn.<index_LocationLabelIndex>	orig_labelStartColumn
labelStatus.<index_LocationLabelIndex>	orig_labelStatus

**Step 10: SET**

Description: Set up the location label entry

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<index_LocationLabelIndex>	alt_LocLabelText
labelFontType.<index_LocationLabelIndex>	alt_LocLabelFontType
labelHeight.<index_LocationLabelIndex>	alt_LocLabelHeight

<input type="checkbox"/>	labelColor.<index_LocationLabelIndex>	alt_LocLabelColor
<input type="checkbox"/>	labelStartRow.<index_LocationLabelIndex>	alt_LocLabelStartRow
<input type="checkbox"/>	labelStartColumn.<index_LocationLabelIndex>	alt_LocLabelStartColumn
<input type="checkbox"/>	labelStatus.<index_LocationLabelIndex>	alt_LocLabelStatus

**Step 11: SET**

Description: Setup the location index pointing to the entry that was just set up.

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelLocationLabel.0	index_LocationLabelIndex

**Step 12: SET**

Description: Enable the text display of all labels by setting labelEnableTextDisplay.0 to 0x80

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelEnableTextDisplay..0	0x80

**Step 13: USER-VERIFY**

Description: Verify the location label is displayed

Points: 1

**Step 14: SET**

Description: Turn off the text display for all labels

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelEnableTextDisplay.0	0x00

**Step 15: USER-VERIFY**

Description: Verify there are no labels displayed

Points: 1

**Step 16: SET**

Description: Turn on the text display for all labels

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelEnableTextDisplay.0	0x80

**Step 17: USER-VERIFY**

Description: Verify the location label is displayed

Points: 1

**Step 18: SET**

Description: Turn off the status for the location label

Action: Set the following object(s) to the value indicated:

Object	Value
labelStatus.<index_LocationLabelIndex>	0x00

**Step 19: USER-VERIFY**

Description: Verify the location label is not displayed. Only the location label has been disabled. Other labels should be able to be displayed.

Points: 1

**Step 20: SET**

Description: Turn on the status for the location label

Action: Set the following object(s) to the value indicated:

Object	Value
labelStatus.<index_LocationLabelIndex>	0xC0

**Step 21: USER-VERIFY**

Description: Verify the location label is displayed

Points: 1

**Step 22: SET**

Description: Dereference the location label

Action: Set the following object(s) to the value indicated:

Object	Value
labelLocationLabel.0	0

**Step 23: USER-VERIFY**

Description: Verify the location label is no longer displayed

Points: 1

**Step 24: SET**

Description: Restore the original values for the label entry

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<index_LocationLabelIndex>	orig_LocLabelText
labelFontType.<index_LocationLabelIndex>	orig_LocLabelFontType
labelHeight.<index_LocationLabelIndex>	orig_LocLabelHeight
labelColor.<index_LocationLabelIndex>	orig_LocLabelColor
labelStartRow.<index_LocationLabelIndex>	orig_LocLabelStartRow

<input type="checkbox"/>	labelStartColumn.<index_LocationLabelIndex>	orig_LocLabelStartColumn
<input type="checkbox"/>	labelStatus.<index_LocationLabelIndex>	orig_LocLabelStatus

## Test Case TC12: Delta Pan Motion

This test case tests the delta panning motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

### Step 1: DEFINE

Description: Please enter the amount of movement to utilize within the delta pan motion test. Specifies measurement in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: deltaPanMovement

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 200

### Step 2: DEFINE

Description: Please enter the speed of the movement to utilize within the delta pan motion test.

Action: Define the following variable(s) for later use

Variable Name: deltaPanMoveSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

### Step 3: SET

Description: Move the camera in a clockwise direction

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	positionPan.0	'1' + deltaPanMoveSpeed + deltaPanMovement

### Step 4: USER-VERIFY

Description: Verify the camera moved in a clockwise direction at the movement and speed specified by the test variables deltaPanMovement and deltaPanMoveSpeed.

Points: 1

### Step 5: SET

Description: Move the camera in a counterclockwise direction

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	positionPan.0	'1' + -

		(deltaPanMoveSpeed) + deltaPanMovement
--	--	---

**Step 6: USER-VERIFY**

Description: Verify the camera moved in a counterclockwise direction at the movement and speed specified by the test variables deltaPanMovement and deltaPanMoveSpeed.

Points: 1

**Test Case TC13: Absolute Pan Motion**

This test case tests the absolute panning motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the first position to move the camera for the absolute pan motion test. Specifies a position in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: absolutePanPosition

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 0

**Step 2: DEFINE**

Description: Please enter the second position to move the camera for the absolute pan motion test. Specifies a position in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: absolutePanPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 9000

**Step 3: DEFINE**

Description: Please enter the speed of the movement to utilize within the absolute pan motion test.

Action: Define the following variable(s) for later use

Variable Name: absolutePanSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 4: SET**

Description: Move the camera to the first position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + absolutePanSpeed + absolutePanPosition

**Step 5: USER-VERIFY**

Description: Verify that the camera moved to position indicated by absolutePanPosition

Points: 1

**Step 6: SET**

Description: Move the camera to the second position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + absolutePanSpeed + absolutePanPosition2

**Step 7: USER-VERIFY**

Description: Verify that the camera moved to position indicated by absolutePanPosition2

Points: 1

***Test Case TC14: Continuous Pan Motion***

This test case tests the continuous panning motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous pan motion test.

Action: Define the following variable(s) for later use

Variable Name: contPanSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the alternate timeout value for continuous movement.

Action: Define the following variable(s) for later use

Variable Name: alt\_contPanTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the pan motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutPan.0	orig_timeoutPan

**Step 4: SET**

Description: Set the pan motion timeout

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutPan.0	alt_contPanTimeout

**Step 5: SET**

Description: Start the camera moving in a clockwise direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'3' + conPanSpeed

**Step 6: USER-VERIFY**

Description: Verify the camera stops moving after a time of alt\_contPanTimeout

Points: 1

**Step 7: SET**

Description: Set the camera moving in a counterclockwise direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'3' + (-contPanSpeed)

**Step 8: USER-VERIFY**

Description: Verify the camera moves in a counterclockwise direction and stops after alt\_contPanTimeout.

Points: 1

**Step 9: SET**

Description: Set the pan motion timeout back to original.

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutPan.0	orig_timeoutPan

**Test Case TC15: Continuous Pan Motion with Stop**

This test case tests the continuous panning motion of the camera by moving the camera and using the stop command to stop movement.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous pan motion test.

Action: Define the following variable(s) for later use

Variable Name: contPanSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the delay before sending a stop command for the continuous movement. Measured in milliseconds.

Action: Define the following variable(s) for later use

Variable Name: contPanTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the pan motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutPan.0	orig_timeoutPan

**Step 4: SET**

Description: Set the pan motion timeout to unused (0)

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutPan.0	0

**Step 5: SET**

Description: Start the camera moving in a clockwise direction

Action: Set the following object(s) to the value indicated:

Object	Value
postionPan.0	'3' + contPanSpeed

**Step 6: DELAY**

Delay: <contPanTimeout> milliseconds:

**Step 7: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'0' + '0' + '0' + '0'

**Step 8: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 9: SET**

Description: Set the camera moving in a counterclockwise direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'3' + (-conPanSpeed)

**Step 10: DELAY**

Delay: <contPanTimeout> milliseconds:

**Step 11: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'0' + '0' + '0' + '0'

**Step 12: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 13: SET**

Description: Set the pan motion timeout back to original.

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutPan.0	orig_timeoutPan

**Test Case TC16: Delta Tilt Motion**

This test case tests the delta tilt motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the amount of movement to utilize within the delta tilt motion test.

Specifies measurement in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: deltaTiltMovement

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535  
Get Value from User: Yes  
Default value: 200

**Step 2: DEFINE**

Description: Please enter the speed of the movement to utilize within the delta tilt motion test.

Action: Define the following variable(s) for later use  
Variable Name: deltaTiltMoveSpeed  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 127  
Get Value from User: Yes  
Default value: 64

**Step 3: SET**

Description: Move the camera in a up direction

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionTilt.0	'1' + deltaTiltMoveSpeed + deltaTiltMovement

**Step 4: USER-VERIFY**

Description: Verify the camera moved in a up direction at the movement and speed specified by the test variables deltaTiltMovement and deltaTiltMoveSpeed.

Points: 1

**Step 5: SET**

Description: Move the camera in a down direction

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionTilt.0	'1' + - (deltaTiltMoveSpeed) + deltaTiltMovement

**Step 6: USER-VERIFY**

Description: Verify the camera moved in a down direction at the movement and speed specified by the test variables deltaTiltMovement and deltaTiltMoveSpeed.

Points: 1

**Test Case TC17: Absolute Tilt Motion**

This test case tests the absolute tilt motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the first position to move the camera for the absolute tilt motion test. Specifies a position in 1/100th degrees.

Action: Define the following variable(s) for later use  
 Variable Name: absoluteTiltPosition  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 65535  
 Get Value from User: Yes  
 Default value: 0

**Step 2: DEFINE**

Description: Please enter the second position to move the camera for the absolute tilt motion test. Specifies a position in 1/100th degrees.

Action: Define the following variable(s) for later use  
 Variable Name: absoluteTiltPosition2  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 65535  
 Get Value from User: Yes  
 Default value: 4500

**Step 3: DEFINE**

Description: Please enter the speed of the movement to utilize within the absolute tilt motion test.

Action: Define the following variable(s) for later use  
 Variable Name: absoluteTiltSpeed  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 127  
 Get Value from User: Yes  
 Default value: 64

**Step 4: SET**

Description: Move the camera to the first position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionTilt.0	'2' + absoluteTiltSpeed + absoluteTiltPosition

**Step 5: USER-VERIFY**

Description: Verify that the camera moved to position indicated by absoluteTiltPosition

Points: 1

**Step 6: SET**

Description: Move the camera to the second position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionTilt.0	'2' + absoluteTiltSpeed + absoluteTiltPosition2

**Step 7: USER-VERIFY**

Description: Verify that the camera moved to position indicated by absoluteTiltPosition2

Points: 1

### **Test Case TC18: Continuous Tilt Motion**

This test case tests the continuous tilt motion of the camera by moving the camera with with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.

#### **Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous tilt motion test.

Action: Define the following variable(s) for later use

Variable Name: conTiltSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

#### **Step 2: DEFINE**

Description: Please enter the alternate timeout value for continuous movement.

Action: Define the following variable(s) for later use

Variable Name: alt\_contTiltTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

#### **Step 3: GET**

Description: Get the tilt motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutTilt.0	orig_timeoutTilt

#### **Step 4: SET**

Description: Set the pan motion timeout

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutTilt.0	alt_contTiltTimeout

#### **Step 5: SET**

Description: Start the camera moving in a upwards direction

Action: Set the following object(s) to the value indicated:

Object	Value
--------	-------

positionTilt.0	'3' + conTiltSpeed
----------------	--------------------

**Step 6: USER-VERIFY**

Description: Verify the camera stops moving after a time of alt\_contTiltTimeout

Points: 1

**Step 7: SET**

Description: Set the camera moving in a downward direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionTilt.0	'3' + (-conTiltSpeed)

**Step 8: USER-VERIFY**

Description: Verify the camera moves in a downward direction and stops after alt\_contTiltTimeout.

Points: 1

**Step 9: SET**

Description: Set the tilt motion timeout back to original.

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutTilt.0	orig_timeoutTilt

**Test Case TC19: Continuous Tilt Motion with Stop**

This test case tests the continuous tilting motion of the camera by moving the camera and using the stop command to stop movement.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous tilt motion test.

Action: Define the following variable(s) for later use

Variable Name: conTiltSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the delay before sending a stop command for the continuous movement. Measured in milliseconds.

Action: Define the following variable(s) for later use

Variable Name: contTiltTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535  
Get Value from User: Yes  
Default value: 100

**Step 3: GET**

Description: Get the tilt motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutTilt.0	orig_timeoutTilt

**Step 4: SET**

Description: Set the tilt motion timeout to unused (0)

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutTilt.0	0

**Step 5: SET**

Description: Start the camera moving in a upward direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionTilt.0	'3' + conTiltSpeed

**Step 6: DELAY**

Delay: <contTiltTimeout> milliseconds:

**Step 7: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionTilt.0	'0' + '0' + '0' + '0'

**Step 8: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 9: SET**

Description: Set the camera moving in a downward direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionTilt.0	'3' + (-conTiltSpeed)

**Step 10: DELAY**

Delay: <contTiltTimeout> milliseconds:

**Step 11: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionTilt.0	'0' + '0' + '0' + '0'

**Step 12: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 13: SET**

Description: Set the Tilt motion timeout back to original.

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutTilt.0	orig_timeoutTilt

**Test Case TC20: Delta Zoom Motion**

This test case tests the delta zoom motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the amount of movement to utilize within the delta zoom motion test. Specifies measurement in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: deltaZoomMovement

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 200

**Step 2: DEFINE**

Description: Please enter the speed of the movement to utilize within the delta zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: deltaZoomMoveSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 3: SET**

Description: Move the camera lens to telephoto zoom

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionZoomLens.0	'1' + deltaZoomMoveSpeed + deltaZoomMovement

**Step 4: USER-VERIFY**

Description: Verify the camera lens moved towards a telephoto position at the movement and speed specified by the test variables deltaZoomMovement and deltaZoomMoveSpeed.

Points: 1

**Step 5: SET**

Description: Move the camera lens to wide zoom

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionZoomLens.0	'1' + - (deltaZoomMoveSpeed) + deltaZoomMovement

**Step 6: USER-VERIFY**

Description: Verify the camera lens moved towards a wide angle position at the movement and speed specified by the test variables deltaZoomMovement and deltaZoomMoveSpeed.

Points: 1

**Test Case TC21: Absolute Zoom Motion**

This test case tests the absolute zoom motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the first position to move the camera for the absolute zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteZoomPosition

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 0

**Step 2: DEFINE**

Description: Please enter the second position to move the camera for the absolute zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteZoomPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 450

**Step 3: DEFINE**

Description: Please enter the speed of the movement to utilize within the absolute zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteZoomSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 4: SET**

Description: Move the camera lens to the first zoom position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionZoomLens.0	'2' + absoluteZoomSpeed + absoluteZoomPosition

**Step 5: USER-VERIFY**

Description: Verify that the camera lens moved to position indicated by absoluteZoomPosition

Points: 1

**Step 6: SET**

Description: Move the camera lens to the second zoom position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionZoomLens.0	'2' + absoluteZoomSpeed + absoluteZoomPosition2

**Step 7: USER-VERIFY**

Description: Verify that the camera lens moved to position indicated by absoluteZoomPosition2

Points: 1

**Test Case TC22: Continuous Zoom Motion**

This test case tests the continuous zoom motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: conZoomSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127  
Get Value from User: Yes  
Default value: 64

**Step 2: DEFINE**

Description: Please enter the alternate timeout value for continuous movement.

Action: Define the following variable(s) for later use  
Variable Name: alt\_contZoomTimeout  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 65535  
Get Value from User: Yes  
Default value: 100

**Step 3: GET**

Description: Get the zoom motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutZoom.0	orig_timeoutZoom

**Step 4: SET**

Description: Set the zoom motion timeout

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutZoom.0	alt_conZoomTimeout

**Step 5: SET**

Description: Start the camera lens moving in a telephoto angle direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoomLens.0	'3' + conZoomSpeed

**Step 6: USER-VERIFY**

Description: Verify the camera lens stops moving after a time of alt\_contZoomTimeout

Points: 1

**Step 7: SET**

Description: Set the camera lens moving in a wide angle direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoomLens.0	'3' + (-conZoomSpeed)

**Step 8: USER-VERIFY**

Description: Verify the camera moves in a wide angle direction and stops after alt\_contZoomTimeout.

Points: 1

**Step 9: SET**

Description: Set the zoom motion timeout back to original.

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutZoom.0	orig_timeoutZoom

**Test Case TC23: Continuous Zoom Motion with Stop**

This test case tests the continuous zooming motion of the camera by moving the camera and using the stop command to stop movement.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous zoom motion test.

Action: Define the following variable(s) for later use

Variable Name: conZoomSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the delay before sending a stop command for the continuous movement. Measured in milliseconds.

Action: Define the following variable(s) for later use

Variable Name: contZoomTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the zoom motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	timeoutZoom.0	orig_timeoutZoom

**Step 4: SET**

Description: Set the zoom motion timeout to unused (0)

Action: Set the following object(s) to the value indicated:

	Object	Value
--	--------	-------

timeoutZoom.0	0
---------------	---

**Step 5: SET**

Description: Start the camera lens moving towards the telephoto direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoomLens.0	'3' + conZoomSpeed

**Step 6: DELAY**

Delay: <contZoomTimeout> milliseconds:

**Step 7: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoomLens.0	'0' + '0' + '0' + '0'

**Step 8: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 9: SET**

Description: Set the camera moving towards the wide direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoomLens.0	'3' + (-conZoomSpeed)

**Step 10: DELAY**

Delay: <contZoomTimeout> milliseconds:

**Step 11: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionZoom.0	'0' + '0' + '0' + '0'

**Step 12: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 13: SET**

Description: Set the zoom motion timeout back to original.

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutZoom.0	orig_timeoutZoom

## Test Case TC24: Delta Focus Motion

This test case tests the delta focus motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them

### Step 1: DEFINE

Description: Please enter the amount of movement to utilize within the delta focus motion test. Specifies measurement in 1/100th degrees.

Action: Define the following variable(s) for later use

Variable Name: deltaFocusMovement

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 200

### Step 2: DEFINE

Description: Please enter the speed of the movement to utilize within the delta focus motion test.

Action: Define the following variable(s) for later use

Variable Name: deltaFocusMoveSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

### Step 3: SET

Description: Move the camera lens to far focus

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'1' + deltaFocusMoveSpeed + deltaFocusMovement

### Step 4: USER-VERIFY

Description: Verify the camera moved the lens towards telephoto position at the movement and speed specified by the test variables deltaFocusMovement and deltaFocusMoveSpeed.

Points: 1

### Step 5: SET

Description: Move the camera lens to near focus

Action: Set the following object(s) to the value indicated:

	Object	Value
--	--------	-------

positionFocusLens.0	'1' + - (deltaFocusMoveSpeed) + deltaFocusMovement
---------------------	--

**Step 6: USER-VERIFY**

Description: Verify the camera lens move towards near focus at the movement and speed specified by the test variables deltaFocusMovement and deltaFocusMoveSpeed.

Points: 1

**Test Case TC25: Absolute Focus Motion**

This test case tests the absolute focus motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

**Step 1: DEFINE**

Description: Please enter the first position to move the camera for the absolute focus motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteFocusPosition

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 0

**Step 2: DEFINE**

Description: Please enter the second position to move the camera for the absolute focus motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteFocusPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 4500

**Step 3: DEFINE**

Description: Please enter the speed of the movement to utilize within the absolute focus motion test.

Action: Define the following variable(s) for later use

Variable Name: absoluteFocusSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 4: SET**

Description: Move the camera lens to the first focus position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'2' + absoluteFocusSpeed + absoluteFocusPosition

**Step 5: USER-VERIFY**

Description: Verify that the camera lens moved to position indicated by absoluteFocusPosition

Points: 1

**Step 6: SET**

Description: Move the camera lens to the second focus position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'2' + absoluteFocusSpeed + absoluteFocusPosition2

**Step 7: USER-VERIFY**

Description: Verify that the camera lens moved to position indicated by absoluteFocusPosition2

Points: 1

**Test Case TC26: Continuous Focus Motion**

This test case tests the continuous focus motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous focus motion test.

Action: Define the following variable(s) for later use

Variable Name: conFocusSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the alternate timeout value for continuous movement.

Action: Define the following variable(s) for later use

Variable Name: alt\_contFocusTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the focus motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutFocus.0	orig_timeoutFocus

**Step 4: SET**

Description: Set the focus motion timeout

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutFocus.0	alt_contFocusTimeout

**Step 5: SET**

Description: Start the camera lens moving in a far focus direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionFocusLens.0	'3' + conFocusSpeed

**Step 6: USER-VERIFY**

Description: Verify the camera lens stops moving after a time of alt\_contFocusTimeout

Points: 1

**Step 7: SET**

Description: Set the camera lens moving in a near focus direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionFocusLens.0	'3' + (-conFocusSpeed)

**Step 8: USER-VERIFY**

Description: Verify the camera moves in a near focus direction and stops after alt\_contFocusTimeout.

Points: 1

**Step 9: SET**

Description: Set the zoom motion timeout back to original.

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutFocus.0	orig_timeoutFocus

## Test Case TC27: Continuous Focus Motion with Stop

This test case tests the continuous focus motion of the camera by moving the camera and using the stop command to stop movement.

### Step 1: DEFINE

Description: Please enter the speed of the movement to utilize within the continuous focus motion test.

Action: Define the following variable(s) for later use

Variable Name: conFocusSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

### Step 2: DEFINE

Description: Please enter the delay before sending a stop command for the continuous movement. Measured in milliseconds.

Action: Define the following variable(s) for later use

Variable Name: contFocusTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

### Step 3: GET

Description: Get the focus motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutFocus.0	orig_timeoutFocus

### Step 4: SET

Description: Set the focus motion timeout to unused (0)

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutFocus.0	0

### Step 5: SET

Description: Start the camera lens moving towards the far focus direction

Action: Set the following object(s) to the value indicated:

Object	Value
positionFocusLens.0	'3' + conFocusSpeed

### Step 6: DELAY

Delay: <contFocusTimeout> milliseconds:

**Step 7: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'0' + '0' + '0' + '0'

**Step 8: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 9: SET**

Description: Set the camera moving towards the near focus direction

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'3' + (-conFocusSpeed)

**Step 10: DELAY**

Delay: &lt;contFocusTimeout&gt; milliseconds:

**Step 11: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionFocusLens.0	'0' + '0' + '0' + '0'

**Step 12: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 13: SET**

Description: Set the focus motion timeout back to original.

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutFocus.0	orig_timeoutFocus

**Test Case TC28: Delta Iris Motion**

This test case tests the delta iris motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.

**Step 1: DEFINE**

Description: Please enter the amount of movement to utilize within the delta focus motion test.

Specifies measurement in scaler units.

Action: Define the following variable(s) for later use

Variable Name: deltaIrisMovement

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 200

**Step 2: DEFINE**

Description: Please enter the speed of the movement to utilize within the delta focus motion test.

Action: Define the following variable(s) for later use

Variable Name: deltaIrisMoveSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 3: SET**

Description: Move the camera lens iris towards a closed position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'1' + deltaIrisMoveSpeed + deltaIrisMovement

**Step 4: USER-VERIFY**

Description: Verify the camera moved the lens towards a closed position at the movement and speed specified by the test variables deltaIrisMovement and deltaIrisMoveSpeed.

Points: 1

**Step 5: SET**

Description: Move the camera lens iris towards an open position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'1' + - (deltaIrisMoveSpeed) + deltaIrisMovement

**Step 6: USER-VERIFY**

Description: Verify the camera lens move towards an open position at the movement and speed specified by the test variables deltaIrisMovement and deltaIrisMoveSpeed.

Points: 1

## Test Case TC29: Absolute Iris Motion

This test case tests the absolute iris motion of the camera by moving the camera with several different speed and direction parameters and allowing the user to verify them.

### Step 1: DEFINE

Description: Please enter the first position to move the camera lens for the absolute Iris motion test.

Action: Define the following variable(s) for later use

Variable Name: absolutelrisPosition

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 0

### Step 2: DEFINE

Description: Please enter the second position to move the camera lens for the absolute Iris motion test.

Action: Define the following variable(s) for later use

Variable Name: absolutelrisPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 4500

### Step 3: DEFINE

Description: Please enter the speed of the movement to utilize within the absolute Iris motion test.

Action: Define the following variable(s) for later use

Variable Name: absolutelrisSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

### Step 4: SET

Description: Move the camera lens to the first iris position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'2' + absolutelrisSpeed + absolutelrisPosition

### Step 5: USER-VERIFY

Description: Verify that the camera lens moved to position indicated by absolutelrisPosition

Points: 1

### Step 6: SET

Description: Move the camera lens to the second iris position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'2' + absolutelIrisSpeed + absolutelIrisPosition2

**Step 7: USER-VERIFY**

Description: Verify that the camera lens moved to position indicated by absolutelIrisPosition2

Points: 1

**Test Case TC30: Continuous Iris Motion**

This test case tests the continuous iris motion of the camera by moving the camera with the continuous command using the timeout parameter to stop the camera. The user verifies the movement and stopping of the camera.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous iris motion test.

Action: Define the following variable(s) for later use

Variable Name: conIrisSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the alternate timeout value for continuous movement.

Action: Define the following variable(s) for later use

Variable Name: alt\_contIrisTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the iris motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	timeoutIris.0	orig_timeoutIris

**Step 4: SET**

Description: Set the iris motion timeout

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutIris.0	alt_conIrisTimeout

**Step 5: SET**

Description: Start the camera lens moving towards a closed position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'3' + conIrisSpeed

**Step 6: USER-VERIFY**

Description: Verify the camera lens stops moving after a time of alt\_conIrisTimeout

Points: 1

**Step 7: SET**

Description: Set the camera lens moving towards an open position

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionIrisLens.0	'3' + (-conIrisSpeed)

**Step 8: USER-VERIFY**

Description: Verify the camera moves towards an open and stops after alt\_conIrisTimeout.

Points: 1

**Step 9: SET**

Description: Set the iris motion timeout back to original.

Action: Set the following object(s) to the value indicated:

	Object	Value
	timeoutIris.0	orig_timeoutIris

**Test Case TC31: Continuous Iris Motion with Stop**

This test case tests the continuous Iris motion of the camera by moving the camera and using the stop command to stop movement.

**Step 1: DEFINE**

Description: Please enter the speed of the movement to utilize within the continuous iris motion test.

Action: Define the following variable(s) for later use

Variable Name: conIrisSpeed

Type: INTEGER

Lower Bound: 0

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 2: DEFINE**

Description: Please enter the delay before sending a stop command for the continuous movement. Measured in milliseconds.

Action: Define the following variable(s) for later use

Variable Name: conIrisTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 65535

Get Value from User: Yes

Default value: 100

**Step 3: GET**

Description: Get the iris motion timeout.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
timeoutIris.0	orig_timeoutIris

**Step 4: SET**

Description: Set the iris motion timeout to unused (0)

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutIris.0	0

**Step 5: SET**

Description: Start the camera lens moving towards the closed position

Action: Set the following object(s) to the value indicated:

Object	Value
positionIrisLens.0	'3' + conIrisSpeed

**Step 6: DELAY**

Delay: <conIrisTimeout> milliseconds:

**Step 7: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionIrisLens.0	'0' + '0' + '0' + '0'

**Step 8: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 9: SET**

Description: Set the camera moving towards the open position

Action: Set the following object(s) to the value indicated:

Object	Value
positionIrisLens.0	'3' + (-conIrisSpeed)

**Step 10: DELAY**

Delay: <conIrisTimeout> milliseconds:

**Step 11: SET**

Description: Send the stop command

Action: Set the following object(s) to the value indicated:

Object	Value
positionIrisLens.0	'0' + '0' + '0' + '0'

**Step 12: USER-VERIFY**

Description: Verify the camera stops moving.

Points: 1

**Step 13: SET**

Description: Set the iris motion timeout back to original.

Action: Set the following object(s) to the value indicated:

Object	Value
timeoutIris.0	orig_timeoutIris

## ***Test Case TC32: Preset Positon***

This test case stores and moves the camera to preset camera positions

**Step 1: DEFINE**

Description: Please enter a pan position for the first preset position in the preset position test.

Action: Define the following variable(s) for later use

Variable Name: presetPanPosition1

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 0

**Step 2: DEFINE**

Description: Please enter a tilt position for the first preset position in the preset position test.

Action: Define the following variable(s) for later use

Variable Name: presetTiltPosition1

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 0

**Step 3: DEFINE**

Description: Please enter a pan position for the second preset position in the preset position test.

Action: Define the following variable(s) for later use

Variable Name: presetPanPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 450

**Step 4: DEFINE**

Description: Please enter a tilt position for the second preset position in the preset position test.

Action: Define the following variable(s) for later use

Variable Name: presetTiltPosition2

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 100

**Step 5: DEFINE**

Description: Please enter the movement speed to utilize within the preset positions test.

Action: Define the following variable(s) for later use

Variable Name: presetMovementSpeed

Type: INTEGER

Lower Bound: -127

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 6: DEFINE**

Description: Please enter a index to use for the first preset position

Action: Define the following variable(s) for later use

Variable Name: presetStore1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 7: DEFINE**

Description: Please enter a index to use for the second preset position

Action: Define the following variable(s) for later use

Variable Name: presetStore2

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 2

**Step 8: SET**

Description: Goto position 1

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + presetMovementSpeed + presetPanPosition1
	positionTilt.0	'2' + presetMovementSpeed + presetTiltPosition1

**Step 9: USER-VERIFY**

Description: Verify the camera is in position 1

Points: 1

**Step 10: SET**

Description: Set position 1 as a preset

Action: Set the following object(s) to the value indicated:

	Object	Value
	presetStorePosition.0	presetStore1

**Step 11: SET**

Description: Goto position 2

Action: Set the following object(s) to the value indicated:

	Object	Value
	positionPan.0	'2' + presetMovementSpeed + presetPanPosition2
	positionTilt.0	'2' + presetMovementSpeed + presetTiltPosition2

**Step 12: USER-VERIFY**

Description: Verify the camera is in position 2

Points: 1

**Step 13: SET**

Description: Set position 2 as a preset

Action: Set the following object(s) to the value indicated:

	Object	Value
	presetStorePosition.0	presetStore2

**Step 14: SET**

Description: Goto preset position 1

Action: Set the following object(s) to the value indicated:

	Object	Value
	presetGotoPosition.0	presetStore1

**Step 15: USER-VERIFY**

Description: Verify the camera moved to position 1

Points: 1

**Step 15: USER-VERIFY**

Description: Verify the camera moved to position 2

Points: 1

**Step 16: SET**

Description: Goto preset position 2

Action: Set the following object(s) to the value indicated:

	Object	Value
	presetGotoPosition.0	presetStore2

**Test Case TC33: Get-Set Zone**

This test case tests the storage of camera zones.

**Step 1: DEFINE**

Description: Please enter the number of zones required by the project.

Action: Define the following variable(s) for later use

Variable Name: req\_ZoneMaximum

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 255

**Step 2: DEFINE**

Description: Please enter the zone row to use in the zone table for testing storing zones.

Action: Define the following variable(s) for later use

Variable Name: index\_ZoneIndex

Type: INTEGER

Lower Bound: 1

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 3: DEFINE**

Description: Please enter an alternate value to set a zone table entry label to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_zoneLabel  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 4: DEFINE**

Description: Please enter an alternate value to set a zone table entry left limit to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_zoneLeftLimit  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 0

**Step 5: DEFINE**

Description: Please enter an alternate value to set a zone table entry right limit to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_zoneRightLimit  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 400

**Step 6: DEFINE**

Description: Please enter an alternate value to set a zone table entry up limit to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_zoneUpLimit  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 450

**Step 7: DEFINE**

Description: Please enter an alternate value to set a zone table entry down limit to.

Action: Define the following variable(s) for later use  
Variable Name: alt\_zoneDownLimit  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 35999  
Get Value from User: Yes  
Default value: 0

**Step 8: GET**

Description: Get the maximum storage limit for zones

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
zoneMaximum.0	zoneMaximum

**Step 9: VERIFY/GOTO**

If zoneMaximum >= req\_ZoneMaximum

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 10: GET**

Description: Get the zone about to be changed and store

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
zoneLabel.<index_ZoneIndex>	orig_zoneLabel
zonePanRightLimit.<index_ZoneIndex>	orig_zonePanRightLimit
zoneTiltUpLimit.<index_ZoneIndex>	orig_zoneTiltUpLimit
zoneTiltDownLimit.<index_ZoneIndex>	orig_zoneTiltDownLimit
zonePanLeftLimit.<index_ZoneIndex>	orig_zonePanLeftLimit

**Step 11: SET**

Description: Set the zone entry

Action: Set the following object(s) to the value indicated:

Object	Value
zoneLabel.<index_ZoneIndex>	alt_zoneLabel
zonePanRightLimit.<index_ZoneIndex>	alt_zonePanRightLimit
zoneTiltUpLimit.<index_ZoneIndex>	alt_zoneTiltUpLimit
zoneTiltDownLimit.<index_ZoneIndex>	alt_zoneTiltDownLimit
zonePanLeftLimit.<index_ZoneIndex>	alt_zonePanLeftLimit

**Step 12: VERIFY/GOTO**

Description: Verify no error was received

If

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 13: SET**

Description: Set the zone back to original value

Action: Set the following object(s) to the value indicated:

Object	Value
--------	-------

zoneLabel.<index_ZoneIndex>	orig_zoneLabel
zonePanRightLimit.<index_ZoneIndex>	orig_zonePanRightLimit
zoneTiltUpLimit.<index_ZoneIndex>	orig_zoneTiltUpLimit
zoneTiltDownLimit.<index_ZoneIndex>	orig_zoneTiltDownLimit
zonePanLeftLimit.<index_ZoneIndex>	orig_zonePanLeftLimit

## **Test Case TC34: Move In and Out of Zone**

This test case tests the labelling capability of zones by moving to areas within zones

### **Step 1: DEFINE**

Description: Please enter an alternate value to set a zone table entry label to.

Action: Define the following variable(s) for later use

Variable Name: alt\_zoneLabel

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

### **Step 2: DEFINE**

Description: Please enter an alternate value to set a zone table entry left limit to.

Action: Define the following variable(s) for later use

Variable Name: alt\_zoneLeftLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 0

### **Step 3: DEFINE**

Description: Please enter an alternate value to set a zone table entry right limit to.

Action: Define the following variable(s) for later use

Variable Name: alt\_zoneRightLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 400

### **Step 4: DEFINE**

Description: Please enter an alternate value to set a zone table entry up limit to.

Action: Define the following variable(s) for later use

Variable Name: alt\_zoneUpLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 450

**Step 5: DEFINE**

Description: Please enter an alternate value to set a zone table entry down limit to.

Action: Define the following variable(s) for later use

Variable Name: alt\_zoneDownLimit

Type: INTEGER

Lower Bound: 0

Upper Bound: 35999

Get Value from User: Yes

Default value: 0

**Step 6: DEFINE**

Description: Please enter the index of zone entry to utilize in the move in and out of zone test.

Action: Define the following variable(s) for later use

Variable Name: indexZone

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 7: DEFINE**

Description: Please enter a value to set a label table entry text to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelText

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'Zone1'

**Step 8: DEFINE**

Description: Please enter a value to set a label table entry font type to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelFontType

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 9: DEFINE**

Description: Please enter a value to set a label table entry height to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelHeight

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 10: DEFINE**

Description: Please enter a value to set a label table entry color to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelColor

Type: INTEGER

Lower Bound: 0

Upper Bound: 16

Get Value from User: Yes

Default value: 11

**Step 11: DEFINE**

Description: Please enter a value to set a label table entry start row to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelStartRow

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 12: DEFINE**

Description: Please enter a value to set a label table entry start column to.

Action: Define the following variable(s) for later use

Variable Name: alt\_LocLabelStartColumn

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 13: DEFINE**

Description: Please enter a value to set a label table entry status to.

Action: Define the following variable(s) for later use

Variable Name: alt\_ZLabelStatus

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 14: DEFINE**

Description: Please enter the movement speed to utilize within the preset positions test.

Action: Define the following variable(s) for later use

Variable Name: zoneMoveSpeed

Type: INTEGER

Lower Bound: -127

Upper Bound: 127

Get Value from User: Yes

Default value: 64

**Step 15: GET**

Description: Get the zone about to be changed and store

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
zoneLabel.<indexZone>	orig_zoneLabel
zonePanRightLimit.<indexZone>	orig_zonePanRightLimit
zoneTiltUpLimit.<indexZone>	orig_zoneTiltUpLimit
zoneTiltDownLimit.<indexZone>	orig_zoneTiltDownLimit
zonePanLeftLimit.<indexZone>	orig_zonePanLeftLimit

**Step 16: SET**

Description: Configure the zone entry

Action: Set the following object(s) to the value indicated:

Object	Value
zoneLabel.<indexZone>	alt_zoneLabel
zonePanRightLimit.<indexZone>	alt_zonePanRightLimit
zoneTiltUpLimit.<indexZone>	alt_zoneTiltUpLimit
zoneTiltDownLimit.<indexZone>	alt_zoneTiltDownLimit
zonePanLeftLimit.<indexZone>	alt_zonePanLeftLimit

**Step 17: GET**

Description: Get the label information and store

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<alt_zoneLabel>	orig_labelText
labelFontType.<alt_zoneLabel>	orig_labelFontType
labelHeight.<alt_zoneLabel>	orig_labelHeight
labelColor.<alt_zoneLabel>	orig_labelColor
labelStartRow.<alt_zoneLabel>	orig_labelStartRow
labelStartColumn.<alt_zoneLabel>	orig_labelStartColumn
labelStatus.<alt_zoneLabel>	orig_labelStatus

**Step 18: SET**

Description: Configure the label to be used

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alt_zoneLabel>	alt_ZlabelText

labelFontType.<alt_zoneLabel>	alt_ZlabelFontType
labelHeight.<alt_zoneLabel>	alt_ZlabelHeight
labelColor.<alt_zoneLabel>	alt_ZlabelColor
labelStartRow.<alt_zoneLabel>	alt_ZlabelStartRow
labelStartColumn.<alt_zoneLabel>	alt_ZlabelStartColumn
labelStatus.<alt_zoneLabel>	alt_ZlabelStatus

**Step 19: SET**

Description: Move the camera into the set zone.

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'2' + zoneMoveSpeed + (alt_zoneRightLimit - 1)
positionTilt.0	'2' + zoneMoveSpeed + (alt_zoneUpLimit - 1)

**Step 20: USER-VERIFY**

Description: Verify the label associated with the zone is displayed

Points: 1

**Step 21: SET**

Description: Move the camera out of the zone.

Action: Set the following object(s) to the value indicated:

Object	Value
positionPan.0	'2' + zoneMoveSpeed + (alt_zoneRightLimit + 1)
positionTilt.0	'2' + zoneMoveSpeed + (alt_zoneUpLimit + 1)

**Step 22: USER-VERIFY**

Description: Verify the label associated with the zone is not displayed

Points: 1

**Step 23: SET**

Description: Restore the original zone entry

Action: Set the following object(s) to the value indicated:

Object	Value
zoneLabel.<indexZone>	orig_zoneLabel
zonePanRightLimit.<indexZone>	orig_zonePanRightLimit
zoneTiltUpLimit.<indexZone>	orig_zoneTiltUpLimit

zoneTiltDownLimit.<indexZone>	orig_zoneTiltDownLimit
zonePanLeftLimit.<indexZone>	orig_zonePanLeftLimit

**Step 24: SET**

Description: Restore the original label entry

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alt_zoneLabel>	orig_labelText
labelFontType.<alt_zoneLabel>	orig_labelFontType
labelHeight.<alt_zoneLabel>	orig_labelHeight
labelColor.<alt_zoneLabel>	orig_labelColor
labelStartRow.<alt_zoneLabel>	orig_labelStartRow
labelStartColumn.<alt_zoneLabel>	orig_labelStartColumn
labelStatus.<alt_zoneLabel>	orig_labelStatus

**Test Case TC35: Get Availability of Equipment**

This test case identifies and verifies the availability of attached equipment to the camera.

**Step 1: PRE-CONDITIONS**

Description: The object 'systemCameraEquipped' must be configured to project requirements prior to this test.

**Step 2: DEFINE**

Description: Please enter the required equipment that is attached to the device.

Action: Define the following variable(s) for later use

Variable Name: req\_systemCameraEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

**Step 3: DEFINE**

Description: Please enter a value for the systemCameraEquipped. This value should be different than req\_systemCameraEquipped

Action: Define the following variable(s) for later use

Variable Name: alt\_systemCameraEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

**Step 4: GET**

Description: Get the equipped object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	orig_systemCameraEquipped

**Step 5: VERIFY/GOTO**

If orig\_systemCameraEquipped equals req\_systemCameraEquipped

Award Points: 1

**Step 6: SET**

Description: Set equipped to an alternate value

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	alt_systemCameraEquipped

**Step 7: GET**

Description: Get the changed value

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	systemCameraEquipped

**Step 8: VERIFY/GOTO**

If systemCameraEquipped equals alt\_systemCameraEquipped

Award Points: 1

**Step 9: SET**

Description: Set back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	orig_systemCameraEquipped

**Test Case TC36: Control Camera Power**

This test case enables and disables this feature while the user verifies.

**Step 1: DEFINE**

Description: Storage for the original state of the systemCameraFeatureControl object.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraFeatureControl

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: No

**Step 2: DEFINE**

Description: Storage for the original value of the object systemCameraEquipped.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: No

**Step 3: GET**

Description: Get the camera equipped object

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	orig_systemCameraEquipped

**Step 4: VERIFY/GOTO**

If systemCameraEquipped bitwise AND 0x80

Award Points: 1

**Step 5: SET**

Description: Turn on bit 7

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	systemCameraEquipped   0x80

**Step 6: GET**

Description: Get the feature control object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureControl.0	systemCameraFeatureControl

**Step 7: SET**

Description: Turn on bit 7 camera power.

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl   0x80

**Step 8: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
--------	-----------

systemCameraFeatureStatus.0	systemCameraFeatureStatus
-----------------------------	---------------------------

**Step 9: VERIFY/GOTO**

Description: Ensure that bit 7 is set

If systemCameraFeatureStatus bitwise AND 0x80

Award Points: 1

**Step 10: USER-VERIFY**

Description: Camera Power is on

Points: 1

**Step 11: SET**

Description: Turn off bit 7 - camera power

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureStatus.0	systemCameraFeatureControl & 0x7F

**Step 12: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 13: VERIFY/GOTO**

Description: Ensure that bit 7 is cleared

If systemCameraFeatureStatus bitwise AND 0x80

Award Points: 1

**Step 14: USER-VERIFY**

Description: Camera Power is off

Points: 1

**Step 15: SET**

Description: Set the feature control back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	orig_systemCameraEquipped

**Test Case TC37: Control Heater Power**

This test case enables and disables this feature while the user verifies.

**Step 1: DEFINE**

Description: Storage for the original state of the systemCameraFeatureControl object.

Action: Define the following variable(s) for later use  
 Variable Name: store\_systemCameraFeatureControl  
 Type: STRING  
 Lower Bound: 2  
 Upper Bound: 2  
 Get Value from User: No

**Step 2: DEFINE**

Description: Storage for the original value of the object systemCameraEquipped.

Action: Define the following variable(s) for later use  
 Variable Name: store\_systemCameraEquipped  
 Type: STRING  
 Lower Bound: 1  
 Upper Bound: 1  
 Get Value from User: No

**Step 3: GET**

Description: Get the camera equipped object

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	orig_systemCameraEquipped

**Step 4: VERIFY/GOTO**

If systemCameraEquipped bitwise AND 0x40  
 Award Points: 1

**Step 5: SET**

Description: Turn on bit 6

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	systemCameraEquipped   0x40

**Step 6: GET**

Description: Get the feature control object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	systemCameraEquipped

**Step 7: SET**

Description: Turn on bit 6 heater power.

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl

		0x40
--	--	------

**Step 8: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 9: VERIFY/GOTO**

Description: Ensure that bit 6 is set

If systemCameraFeatureStatus bitwise AND 0x40  
Award Points: 1

**Step 10: USER-VERIFY**

Description: Heater Power is on

Points: 1

**Step 11: SET**

Description: Turn off bit 6 - heater power

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureStatus.0	systemCameraFeatureControl & 0xBF

**Step 12: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 13: VERIFY/GOTO**

Description: Ensure that bit 6 is cleared

If systemCameraFeatureStatus bitwise AND 0x40  
Award Points: 1

**Step 14: USER-VERIFY**

Description: Heater Power is off

Points: 1

**Step 15: SET**

Description: Set the feature control back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
--------	-------

systemCameraFeatureControl.0	orig_systemCameraEquipped
------------------------------	---------------------------

### **Test Case TC38: Control Wiper**

This test case enables and disables this feature while the user verifies.

#### **Step 1: DEFINE**

Description: Storage for the original state of the systemCameraFeatureControl object.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraFeatureControl

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: No

#### **Step 2: DEFINE**

Description: Storage for the original value of the object systemCameraEquipped.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: No

#### **Step 3: GET**

Description: Get the camera equipped object

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	orig_systemCameraEquipped

#### **Step 4: VERIFY/GOTO**

If systemCameraEquipped bitwise AND 0x20

Award Points: 1

#### **Step 5: SET**

Description: Turn on bit 5

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	systemCameraEquipped   0x20

#### **Step 6: GET**

Description: Get the feature control object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	systemCameraEquipped

**Step 7: SET**

Description: Turn on bit 5 for wiper.

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl   0x20

**Step 8: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 9: VERIFY/GOTO**

Description: Ensure that bit 5 is set

If systemCameraFeatureStatus bitwise AND 0x20

Award Points: 1

**Step 10: USER-VERIFY**

Description: Wiper is on

Points: 1

**Step 11: SET**

Description: Turn off bit 5 - wiper control

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl & 0xDF

**Step 12: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 13: VERIFY/GOTO**

Description: Ensure that bit 5 is cleared

If systemCameraFeatureStatus bitwise AND 0x20

Award Points: 1

**Step 14: USER-VERIFY**

Description: Wiper is off

Points: 1

**Step 15: SET**

Description: Set the feature control back to the original value

Action: Set the following object(s) to the value indicated:

	Object	Value
	systemCameraFeatureControl.0	orig_systemCameraEquipped

**Test Case TC39: Control Washer**

This test case enables and disables this feature while the user verifies.

**Step 1: DEFINE**

Description: Storage for the original state of the systemCameraFeatureControl object.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraFeatureControl

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: No

**Step 2: DEFINE**

Description: Storage for the original value of the object systemCameraEquipped.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: No

**Step 3: GET**

Description: Get the camera equipped object

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	systemCameraEquipped.0	orig_systemCameraEquipped

**Step 4: VERIFY/GOTO**

If systemCameraEquipped bitwise AND 0x10

Award Points: 1

**Step 5: SET**

Description: Turn on bit 4

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	systemCameraEquipped   0x10

**Step 6: GET**

Description: Get the feature control object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	systemCameraEquipped

**Step 7: SET**

Description: Turn on bit 4 for washer.

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl   0x10

**Step 8: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 9: VERIFY/GOTO**

Description: Ensure that bit 4 is set

If systemCameraFeatureStatus bitwise AND 0x10

Award Points: 1

**Step 10: USER-VERIFY**

Description: Wiper is on

Points: 1

**Step 11: SET**

Description: Turn off bit 4 - washer control

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl & 0xEF

**Step 12: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 13: VERIFY/GOTO**

Description: Ensure that bit 4 is cleared

If systemCameraFeatureStatus bitwise AND 0x10

Award Points: 1

**Step 14: USER-VERIFY**

Description: Washer is off

Points: 1

**Step 15: SET**

Description: Set the feature control back to the original value

Action: Set the following object(s) to the value indicated:

	Object	Value
	systemCameraEquipped.0	orig_systemCameraEquipped

**Test Case TC40: Control Blower**

This test case enables and disables this feature while the user verifies.

**Step 1: DEFINE**

Description: Storage for the original state of the systemCameraFeatureControl object.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraFeatureControl

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: Yes

**Step 2: DEFINE**

Description: Storage for the original value of the object systemCameraEquipped.

Action: Define the following variable(s) for later use

Variable Name: store\_systemCameraEquipped

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: Yes

**Step 3: GET**

Description: Get the camera equipped object

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	orig_systemCameraEquipped

**Step 4: VERIFY/GOTO**

If systemCameraEquipped bitwise AND 0x08

Award Points: 1

**Step 5: SET**

Description: Turn on bit 3

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	systemCameraEquipped   0x08

**Step 6: GET**

Description: Get the feature control object.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraEquipped.0	systemCameraEquipped

**Step 7: SET**

Description: Turn on bit 3 for blower.

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl   0x08

**Step 8: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 9: VERIFY/GOTO**

Description: Ensure that bit 3 is set

If systemCameraFeatureStatus bitwise AND 0x08

Award Points: 1

**Step 10: USER-VERIFY**

Description: Blower is on

Points: 1

**Step 11: SET**

Description: Turn off bit 3 - blower control

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraFeatureControl.0	systemCameraFeatureControl & 0xF7

**Step 12: GET**

Description: Get the status of the equipped.

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemCameraFeatureStatus.0	systemCameraFeatureStatus

**Step 13: VERIFY/GOTO**

Description: Ensure that bit 3 is cleared

If systemCameraFeatureStatus bitwise AND 0x08

Award Points: 1

**Step 14: USER-VERIFY**

Description: Blower is off

Points: 1

**Step 15: SET**

Description: Set the feature control back to the original value

Action: Set the following object(s) to the value indicated:

Object	Value
systemCameraEquipped.0	orig_systemCameraEquipped

***Test Case TC41: Get Availability of Lens Equipment***

This test case identifies and verifies the availability of attached equipment to the camera.

**Step 1: PRE-CONDITIONS**

Description: The object 'systemLensEquipped' must be configured to project requirements prior to this test.

**Step 2: DEFINE**

Description: Please enter the required equipment for attachment to the device.

Action: Define the following variable(s) for later use

Variable Name: req\_systemLensEquipped

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: No

**Step 3: DEFINE**

Description: Please enter an alternate value for the systemLensEquipped. This value should be different than req\_systemCameraEquipped

Action: Define the following variable(s) for later use

Variable Name: alt\_systemLensEquipped

Type: INTEGER

Lower Bound: 1

Upper Bound: 1

Get Value from User: No

**Step 4: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	systemLensEquipped.0	orig_systemLensEquipped

**Step 5: VERIFY**

**Step 6: SET**

Description: Set the equipped to a user specified value.

Action: Set the following object(s) to the value indicated:

	Object	Value
	systemLensEquipped.0	alt_systemLensEquipped

**Step 7: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	systemLensEquipped.0	systemLensEquipped

**Step 8: VERIFY**

Description: Ensure the equipped was set properly.

**Step 9: SET**

Description: Set back to original

Action: Set the following object(s) to the value indicated:

	Object	Value
	systemLensEquipped.0	

**Test Case TC42: Control Auto Iris**

This test case enables and disables this feature while the user verifies.

**Step 1: GET**

Description: Get the systemLensEquipped

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensEquipped.0	systemLensEquipped

**Step 2: VERIFY/GOTO**

If systemLensEquipped bitwise AND 0x80

Award Points: 1

**Step 3: SET**

Description: Turn on bit 7 - Auto Iris

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensEquipped.0	systemLensEquipped   0x80

**Step 4: GET**

Description: Get the feature control object

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureControl.0	orig_systemLensFeatureControl

**Step 5: SET**

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensFeatureControl.0	orig_systemLensFeatureControl   0x80

**Step 6: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureStatus.0	systemLensFeatureStatus

**Step 7: VERIFY/GOTO**

If systemLensFeatureStatus bitwise AND 0x80

Award Points: 1

**Step 8: USER-VERIFY**

Description: Auto Iris is on

Points: 1

**Step 9: SET**

Description: Turn off the auto iris

Action: Set the following object(s) to the value indicated:

Object	Value
--------	-------

systemLensFeatureControl.0	orig_systemLensFeatureControl & 0x7F
----------------------------	--------------------------------------

**Step 10: GET**

Description: Get the status

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureStatus.0	systemLensFeatureStatus

**Step 11: VERIFY/GOTO**

If systemLensFeatureStatus bitwise AND 0x80

Award Points: 1

**Step 12: USER-VERIFY**

Description: Auto Iris is off

Points: 1

**Step 13: SET**

Description: Set the feature control back to original

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensFeatureControl.0	orig_systemLensFeatureControl

**Test Case TC43: Control Auto Focus**

This test case enables and disables this feature while the user verifies

**Step 1: GET**

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensEquipped.0	systemLensEquipped

**Step 2: VERIFY**

Description: Verify that bit 6 - Auto Focus is set

**Step 3: SET**

Description: Equip the device with auto focus

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensEquipped..0	

**Step 4: GET**

Description: Get the feature control

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureControl.0	orig_systemLensFeatureControl

**Step 5: SET**

Description: Turn on the auto focus

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensFeatureStatus.0	orig_systemLensFeatureControl   0x40

**Step 6: GET**

Description: Get the status of auto focus

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureStatus.0	systemLensFeatureStatus

**Step 7: VERIFY**

**Step 8: USER-VERIFY**

Description: Auto Focus is on

Points: 1

**Step 9: SET**

Description: Turn off auto focus

Action: Set the following object(s) to the value indicated:

Object	Value
systemLensFeatureStatus.0	orig_systemLensFeatureControl & 0xBF

**Step 10: GET**

Description: Get the status

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
systemLensFeatureStatus.0	systemLensFeatureStatus

**Step 11: VERIFY**

**Step 12: USER-VERIFY**

Description: Auto Focus is off

Points: 1

**Step 13: SET**

Description: Set feature control back to original

Action: Set the following object(s) to the value indicated:

	Object	Value
	systemLensFeatureControl.0	orig_systemLensFeatureControl

**Test Case TC44: Cabinet Alarm**

This test case tests the cabinet open alarm and the label associated with it.

**Step 1: DEFINE**

Description: Please enter the label entry to use for the alarm

Action: Define the following variable(s) for later use

Variable Name: alarmCLabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 12

**Step 2: DEFINE**

Description: Please enter the text for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'C'

**Step 3: DEFINE**

Description: Please enter the font type for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

**Step 4: DEFINE**

Description: Please enter the text height for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 5: DEFINE**

Description: Please enter the text color for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabColor1

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes

Default value: 4

**Step 6: DEFINE**

Description: Please enter the start row for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabStartRow1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 13

**Step 7: DEFINE**

Description: Please enter the start column for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabStartColumn1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 8: DEFINE**

Description: Please enter the status for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmCLabStatus1

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 9: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

	Object	Value
	labelText.<alarmCLabIndex>	alarmCLabText1
	labelFontType.<alarmCLabIndex>	alarmCLabFontType1
	labelHeight.<alarmCLabIndex>	alarmCLabHeight1

<input type="checkbox"/>	labelColor.<alarmCLabIndex>	alarmCLabColor1
<input type="checkbox"/>	labelStartRow.<alarmCLabIndex>	alarmCLabStartRow1
<input type="checkbox"/>	labelStartColumn.<alarmCLabIndex>	alarmCLabStartColumn1
<input type="checkbox"/>	labelStatus.<alarmCLabIndex>	alarmCLabStatus1

**Step 10: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLabelIndex.0	'alarmCLabIndex' 00 00 00 00 00 00

**Step 11: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 12: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLatchClear.0	0x00

**Step 13: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 14: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

<input type="checkbox"/>	Object	Variables
<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 15: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x80

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 16: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x80  
 Award Points: 1  
 and Goto Step: 0  
 Otherwise Goto Step: 0

**Step 17: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 18: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 19: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x80  
 Award Points: 1  
 and Goto Step: 0  
 Otherwise Goto Step: 0

**Step 20: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x80  
 Award Points: 1  
 and Goto Step: 0  
 Otherwise Goto Step: 0

**Step 21: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 22: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 23: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
--	--------	-----------

<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 24: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 25: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x80

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 26: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 27: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLatchClear.0	0x00

**Step 28: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

<input type="checkbox"/>	Object	Variables
<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 29: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 30: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

## **Test Case TC45: Enclosure Alarm**

This test case tests the enclosure alarm and the label associated with it.

### **Step 1: DEFINE**

Description: Please enter the label entry to use for the alarm

Action: Define the following variable(s) for later use

Variable Name: alarmELabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 12

### **Step 2: DEFINE**

Description: Please enter the text for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmELabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'C'

### **Step 3: DEFINE**

Description: Please enter the font type for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmELabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

### **Step 4: DEFINE**

Description: Please enter the text height for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmELabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

### **Step 5: DEFINE**

Description: Please enter the text color for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmELabColor1  
 Type: INTEGER  
 Lower Bound: 1  
 Upper Bound: 16  
 Get Value from User: Yes  
 Default value: 4

**Step 6: DEFINE**

Description: Please enter the start row for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmELabStartRow1  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 13

**Step 7: DEFINE**

Description: Please enter the start column for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmELabStartColumn1  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 1

**Step 8: DEFINE**

Description: Please enter the status for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmELabStatus1  
 Type: STRING  
 Lower Bound: 1  
 Upper Bound: 1  
 Get Value from User: Yes  
 Default value: 0xC0

**Step 9: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

	Object	Value
	labelText.<alarmELabIndex>	alarmELabText1
	labelFontType.<alarmELabIndex>	alarmELabFontType1
	labelHeight.<alarmELabIndex>	alarmELabHeight1
	labelColor.<alarmELabIndex>	alarmELabColor1
	labelStartRow.<alarmELabIndex>	alarmELabStartRow1

<input type="checkbox"/>	labelStartColumn.<alarmELabIndex>	alarmELabStartColumn1
<input type="checkbox"/>	labelStatus.<alarmELabIndex>	alarmELabStatus1

**Step 10: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLabelIndex.0	00 'alarmELabIndex' 00 00 00 00 00

**Step 11: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 12: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLatchClear.0	0x00

**Step 13: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 14: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

<input type="checkbox"/>	Object	Variables
<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 15: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x40

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 16: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x40

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 17: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 18: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 19: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x40

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 20: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x40

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 21: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 22: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 23: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 24: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 25: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x40

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 26: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 27: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 28: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 29: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 30: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

## **Test Case TC46: Video Loss Alarm**

This test case tests the video loss alarm and the label associated with it.

### **Step 1: DEFINE**

Description: Please enter the label entry to use for the alarm

Action: Define the following variable(s) for later use

Variable Name: alarmVLabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 12

### **Step 2: DEFINE**

Description: Please enter the text for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmVLabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'v'

### **Step 3: DEFINE**

Description: Please enter the font type for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmVLabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

### **Step 4: DEFINE**

Description: Please enter the text height for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmVLabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

### **Step 5: DEFINE**

Description: Please enter the text color for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmVLabColor1

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes  
Default value: 4

**Step 6: DEFINE**

Description: Please enter the start row for the alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmVLabStartRow1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 14

**Step 7: DEFINE**

Description: Please enter the start column for the alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmVLabStartColumn1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 1

**Step 8: DEFINE**

Description: Please enter the status for the alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmVLabStatus1  
Type: STRING  
Lower Bound: 1  
Upper Bound: 1  
Get Value from User: Yes  
Default value: 0xC0

**Step 9: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alarmVLabIndex>	alarmVLabText1
labelFontType.<alarmVLabIndex>	alarmVLabFontType1
labelHeight.<alarmVLabIndex>	alarmVLabHeight1
labelColor.<alarmVLabIndex>	alarmVLabColor1
labelStartRow.<alarmVLabIndex>	alarmVLabStartRow1
labelStartColumn.<alarmVLabIndex>	alarmVLabStartColumn1
labelStatus.<alarmVLabIndex>	alarmVLabStatus1

**Step 10: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLabelIndex.0	00 00 'alarmVLabIndex' 00 00 00 00

**Step 11: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 12: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 13: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 14: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 15: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x20

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 16: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x20

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 17: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 18: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 19: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x20

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 20: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x20

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 21: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 22: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 23: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 24: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 25: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x20

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 26: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 27: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 28: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 29: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 30: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Test Case TC47: Temperature Alarm**

This test case tests the temperature alarm and the label, thresholds, and current value associated with it.

**Step 1: DEFINE**

Description: Please enter the threshold to utilize when testing the temperature alarm.

Action: Define the following variable(s) for later use

Variable Name: temperatureThreshold

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: Yes

Default value: '0x20 0x40'

### **Step 2: DEFINE**

Description: Please enter the label entry to use for the alarm

Action: Define the following variable(s) for later use

Variable Name: alarmTlabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 12

### **Step 3: DEFINE**

Description: Please enter the text for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmTlabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'P'

### **Step 4: DEFINE**

Description: Please enter the font type for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmTlabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

### **Step 5: DEFINE**

Description: Please enter the text height for the alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmTlabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

### **Step 6: DEFINE**

Description: Please enter the text color for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmTLabColor1  
 Type: INTEGER  
 Lower Bound: 1  
 Upper Bound: 16  
 Get Value from User: Yes  
 Default value: 4

**Step 7: DEFINE**

Description: Please enter the start row for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmTLabStartRow1  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 12

**Step 8: DEFINE**

Description: Please enter the start column for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmTLabStartColumn1  
 Type: INTEGER  
 Lower Bound: 0  
 Upper Bound: 255  
 Get Value from User: Yes  
 Default value: 1

**Step 9: DEFINE**

Description: Please enter the status for the alarm label.

Action: Define the following variable(s) for later use  
 Variable Name: alarmTLabStatus1  
 Type: STRING  
 Lower Bound: 1  
 Upper Bound: 1  
 Get Value from User: Yes  
 Default value: 0xC0

**Step 10: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

	Object	Value
	labelText.<alarmTLabIndex>	alarmTLabText1
	labelFontType.<alarmTLabIndex>	alarmTLabFontType1
	labelHeight.<alarmTLabIndex>	alarmTLabHeight1
	labelColor.<alarmTLabIndex>	alarmTLabColor1
	labelStartRow.<alarmTLabIndex>	alarmTLabStartRow1

<input type="checkbox"/>	labelStartColumn.<alarmTlabIndex>	alarmTlabStartColumn1
<input type="checkbox"/>	labelStatus.<alarmTlabIndex>	alarmTlabStatus1

**Step 11: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLabelIndex.0	00 00 00 'alarmTlabIndex' 00 00 00

**Step 12: SET**

Description: Set up the Threshold

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmPressureHighLowThreshold.0	temperatureThreshold

**Step 13: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 14: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLatchClear.0	0x00

**Step 15: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 16: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

<input type="checkbox"/>	Object	Variables
<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 17: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x10

Award Points: 1

and Goto Step: 0  
Otherwise Goto Step: 0

**Step 18: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x10

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 19: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 20: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 21: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x10

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 22: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x10

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 23: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 24: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 25: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 26: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 27: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x10

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 28: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 29: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 30: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 31: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 32: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Test Case TC48: Pressure Alarm**

This test case tests the pressure alarm and the label, thresholds, and current value associated with it.

**Step 1: DEFINE**

Description: Please enter the threshold to utilize when testing the pressure alarm.

Action: Define the following variable(s) for later use

Variable Name: pressureThreshold

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: Yes

Default value: '0x20 0x40'

**Step 2: DEFINE**

Description: Please enter the label entry to use for the pressure alarm

Action: Define the following variable(s) for later use

Variable Name: alarmPLabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 11

**Step 3: DEFINE**

Description: Please enter the text for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'P'

**Step 4: DEFINE**

Description: Please enter the font type for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

**Step 5: DEFINE**

Description: Please enter the text height for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 6: DEFINE**

Description: Please enter the text color for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabColor1

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes

Default value: 4

**Step 7: DEFINE**

Description: Please enter the start row for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabStartRow1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 11

**Step 8: DEFINE**

Description: Please enter the start column for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabStartColumn1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 9: DEFINE**

Description: Please enter the status for the pressure alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmPLabStatus1

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 10: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alarmPLabIndex>	alarmWFLabText1
labelFontType.<alarmPLabIndex>	alarmWFLabFontType1
labelHeight.<alarmPLabIndex>	alarmWFLabHeight1
labelColor.<alarmPLabIndex>	alarmWFLabColor1
labelStartRow.<alarmPLabIndex>	alarmWFLabStartRow1
labelStartColumn.<alarmPLabIndex>	alarmWFLabStartColumn1
labelStatus.<alarmPLabIndex>	alarmWFLabStatus1

**Step 11: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLabelIndex.0	00 00 00 00 'alarmPLabIndex' 00 00

**Step 12: SET**

Description: Set up the Threshold

Action: Set the following object(s) to the value indicated:

Object	Value
alarmPressureHighLowThreshold.0	washerThreshold

**Step 13: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 14: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 15: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 16: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 17: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x08

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 18: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x08

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 19: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 20: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 21: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x08

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 22: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x08

Award Points: 1

and Goto Step: 0  
Otherwise Goto Step: 0

**Step 23: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 24: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 25: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 26: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 27: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x08

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 28: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 29: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 30: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 31: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 32: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

***Test Case TC49: Local-Remote Alarm***

This test case tests the local-remote alarm and the label associated with it.

**Step 1: DEFINE**

Description: Please enter the label entry to use for the local-remote alarm

Action: Define the following variable(s) for later use

Variable Name: alarmLLabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 12

**Step 2: DEFINE**

Description: Please enter the text for the local-remotealarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmLLabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'L'

**Step 3: DEFINE**

Description: Please enter the font type for the local-remote alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmLLabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2  
Get Value from User: Yes  
Default value: 1

**Step 4: DEFINE**

Description: Please enter the text height for the local-remote alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmLLabHeight1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 5: DEFINE**

Description: Please enter the text color for the local-remote alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmLLabColor1  
Type: INTEGER  
Lower Bound: 1  
Upper Bound: 16  
Get Value from User: Yes  
Default value: 4

**Step 6: DEFINE**

Description: Please enter the start row for the local-remote alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmLLabStartRow1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 15

**Step 7: DEFINE**

Description: Please enter the start column for the local-remote alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmLLabStartColumn1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 1

**Step 8: DEFINE**

Description: Please enter the status for the local-remote alarm label.

Action: Define the following variable(s) for later use  
Variable Name: alarmLLabStatus1  
Type: STRING  
Lower Bound: 1  
Upper Bound: 1

Get Value from User: Yes  
Default value: 0xC0

**Step 9: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alarmLLabIndex>	alarmLLabText1
labelFontType.<alarmLLabIndex>	alarmLLabFontType1
labelHeight.<alarmLLabIndex>	alarmLLabHeight1
labelColor.<alarmLLabIndex>	alarmLLabColor1
labelStartRow.<alarmLLabIndex>	alarmLLabStartRow1
labelStartColumn.<alarmLLabIndex>	alarmLLabStartColumn1
labelStatus.<alarmLLabIndex>	alarmLLabStatus1

**Step 10: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLabelIndex.0	00 00 00 00 00 'alarmLLabIndex' 00 00

**Step 11: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 12: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 13: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 14: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus

<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus
--------------------------	--------------------	------------------

**Step 15: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x04

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 16: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x04

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 17: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	alarmLatchClear.0	0x00

**Step 18: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

<input type="checkbox"/>	Object	Variables
<input type="checkbox"/>	alarmStatus.0	alarmStatus
<input type="checkbox"/>	alarmLatchStatus.0	alarmLatchStatus

**Step 19: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x04

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 20: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x04

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 21: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 22: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 23: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 24: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 25: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x04

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 26: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 27: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 28: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 29: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 30: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Test Case TC50: Washer Fluid Alarm**

This test case tests the washer fluid alarm and the label, thresholds, and current value associated with it.

**Step 1: DEFINE**

Description: Please enter the threshold to utilize when testing the washer fluid alarm.

Action: Define the following variable(s) for later use

Variable Name: washerThreshold

Type: STRING

Lower Bound: 2

Upper Bound: 2

Get Value from User: Yes

Default value: '0x80 0x80'

**Step 2: DEFINE**

Description: Please enter the label entry to use for the washer fluid alarm

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabIndex

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 11

**Step 3: DEFINE**

Description: Please enter the text for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabText1

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'WF'

**Step 4: DEFINE**

Description: Please enter the font type for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabFontType1

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

**Step 5: DEFINE**

Description: Please enter the text height for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabHeight1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 6: DEFINE**

Description: Please enter the text color for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabColor1

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes

Default value: 4

**Step 7: DEFINE**

Description: Please enter the start row for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabStartRow1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 8: DEFINE**

Description: Please enter the start column for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabStartColumn1

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 1

**Step 9: DEFINE**

Description: Please enter the status for the washer fluid alarm label.

Action: Define the following variable(s) for later use

Variable Name: alarmWFLabStatus1

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 10: SET**

Description: Set up alarm label

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<alarmWFLabIndex>	alarmWFLabText1
labelFontType.<alarmWFLabIndex>	alarmWFLabFontType1
labelHeight.<alarmWFLabIndex>	alarmWFLabHeight1
labelColor.<alarmWFLabIndex>	alarmWFLabColor1
labelStartRow.<alarmWFLabIndex>	alarmWFLabStartRow1
labelStartColumn.<alarmWFLabIndex>	alarmWFLabStartColumn1
labelStatus.<alarmWFLabIndex>	alarmWFLabStatus1

**Step 11: SET**

Description: Set up the label references

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLabelIndex.0	00 00 00 00 00 00 'alarmWFLabIndex'

**Step 12: SET**

Description: Set up the Threshold

Action: Set the following object(s) to the value indicated:

Object	Value
alarmWasherFluidHighLowThreshold.0	washerThreshold

**Step 13: USER-VERIFY**

Description: Ensure no labels for this alarm are shown

Points: 1

**Step 14: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 15: USER-VERIFY**

Description: Turn on the alarm and verify the label for that alarm is displayed

Points: 1

**Step 16: GET**

Description: Get the status and latch status of alarm

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 17: VERIFY/GOTO**

Description: Verify the status for alarm is on

If alarmStatus equals 0x02

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 18: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x02

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 19: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

Object	Value
alarmLatchClear.0	0x00

**Step 20: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
alarmStatus.0	alarmStatus
alarmLatchStatus.0	alarmLatchStatus

**Step 21: VERIFY/GOTO**

Description: Verify the status for the alarm is on

If alarmStatus equals 0x02  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Step 22: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x02  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Step 23: USER-VERIFY**

Description: Verify the label for the corresponding alarm is on and deactivate the alarm

Points: 1

**Step 24: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 25: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 26: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Step 27: VERIFY/GOTO**

Description: Verify the latch status is on for the alarm

If alarmLatchStatus equals 0x02  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Step 28: USER-VERIFY**

Description: Verify the label for the corresponding alarm is off

Points: 1

**Step 29: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	alarmLatchClear.0	0x00

**Step 30: GET**

Description: Get the status and latch status of alarms

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	alarmStatus.0	alarmStatus
	alarmLatchStatus.0	alarmLatchStatus

**Step 31: VERIFY/GOTO**

Description: Verify the status is off for the alarm

If alarmStatus equals 0x00  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Step 32: VERIFY/GOTO**

Description: Verify the latch status is off for the alarm

If inputLatchStatus equals 0x00  
Award Points: 1  
and Goto Step: 0  
Otherwise Goto Step: 0

**Test Case TC51: Monitor Discrete Input**

This test case verifies the state of discrete inputs and the label associated with it.

**Step 1: DEFINE**

Description: Please enter the input (1-8) to use in the monitor discrete input test.

Action: Define the following variable(s) for later use  
Variable Name: input1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 8  
Get Value from User: Yes  
Default value: 1

**Step 2: DEFINE**

Description: Please enter the label entry to use for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabelIndex1  
Type: INTEGER  
Lower Bound: 0

Upper Bound: 255  
Get Value from User: Yes  
Default value: 5

**Step 3: DEFINE**

Description: Please enter the text for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabText1  
Type: STRING  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 'In1'

**Step 4: DEFINE**

Description: Please enter the font type for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabFontType1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 2  
Get Value from User: Yes  
Default value: 1

**Step 5: DEFINE**

Description: Please enter the text height for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabHeight1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 6: DEFINE**

Description: Please enter the text color for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabColor1  
Type: INTEGER  
Lower Bound: 1  
Upper Bound: 16  
Get Value from User: Yes  
Default value: 4

**Step 7: DEFINE**

Description: Please enter the start row for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabStartRow1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255

Get Value from User: Yes  
Default value: 10

**Step 8: DEFINE**

Description: Please enter the start column for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabStartColumn1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 9: DEFINE**

Description: Please enter the status for the discrete input.

Action: Define the following variable(s) for later use  
Variable Name: inputLabStatus1  
Type: STRING  
Lower Bound: 1  
Upper Bound: 1  
Get Value from User: Yes  
Default value: 0xC0

**Step 10: SET**

Description: Set up input label 1

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<inputLabelIndex1>	inputLabText1
labelFontType.<inputLabelIndex1>	inputLabFontType1
labelHeight.<inputLabelIndex1>	inputLabHeight1
labelColor.<inputLabelIndex1>	inputLabColor1
labelStartRow.<inputLabelIndex1>	inputLabStartRow1
labelStartColumn.<inputLabelIndex1>	inputLabStartColumn1
labelStatus.<inputLabelIndex1>	inputLabStatus1

**Step 11: SET**

Description: Set up the label references [input1]=inputLabelIndex1

Action: Set the following object(s) to the value indicated:

Object	Value
inputLabelIndex.0	[input1]=inputLabelIndex1

**Step 12: USER-VERIFY**

Description: Turn off inputs and verify that no labels corresponding to the inputs are displayed

Points: 1

**Step 13: SET**

Description: Clear latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	inputLatchClear.0	0x00

**Step 14: USER-VERIFY**

Description: Turn on the input and verify the label for that input is displayed

Points: 1

**Step 15: GET**

Description: Get the status and latch status of inputs

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	inputStatus.0	inputStatus
	inputLatchStatus.0	inputLatchStatus

**Step 16: VERIFY/GOTO**

Description: Verify the Status is on for the input

If inputStatus equals correct bit is on

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 17: VERIFY/GOTO**

Description: Verify the latch status is on for the input

If inputLatchStatus equals correct bit is on

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 18: USER-VERIFY**

Description: Verify the label for the corresponding input is on

Points: 1

**Step 19: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	inputLatchClear.0	0x00

**Step 20: GET**

Description: Get the status and latch status of inputs

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	inputStatus.0	inputStatus
	inputLatchStatus.0	inputLatchStatus

**Step 21: VERIFY/GOTO**

Description: Verify the Status is on for the input

If inputStatus equals correct bit is on

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 22: VERIFY/GOTO**

Description: Verify the latch status is on for the input

If inputLatchStatus equals correct bit is on

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 23: USER-VERIFY**

Description: Verify the label for the corresponding input is on, turn off the input after it has been checked

Points: 1

**Step 24: GET**

Description: Get the status and latch status of inputs

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	inputStatus.0	inputStatus
	inputLatchStatus.0	inputLatchStatus

**Step 25: VERIFY/GOTO**

Description: Verify the Status is off for the input

If inputStatus equals correct bit is off

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 26: VERIFY/GOTO**

Description: Verify the latch status is on for the input

If inputLatchStatus equals correct bit is on

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 27: USER-VERIFY**

Description: Verify the label for the corresponding input is off

Points: 1

**Step 28: SET**

Description: Clear the latch

Action: Set the following object(s) to the value indicated:

	Object	Value
	inputLatchClear.0	0x00

**Step 29: GET**

Description: Get the status and latch status of inputs

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
	inputStatus.0	inputStatus
	inputLatchStatus.0	inputLatchStatus

**Step 30: VERIFY/GOTO**

Description: Verify the Status is off for the input

If inputStatus equals correct bit is off

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 31: VERIFY/GOTO**

Description: Verify the latch status is off for the input

If inputLatchStatus equals correct bit is off

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 32: USER-VERIFY**

Description: Verify the label for the corresponding input is off

Points: 1

**Test Case TC52: Monitor Discrete Output**

This test case verifies the state of discrete outputs and the label associated with it.

**Step 1: DEFINE**

Description: Please enter the first output (1-8) to use in the monitor discrete input test.

Action: Define the following variable(s) for later use

Variable Name: output1

Type: INTEGER

Lower Bound: 0

Upper Bound: 8  
Get Value from User: Yes  
Default value: 1

**Step 2: DEFINE**

Description: Please enter the label entry to use for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabelIndex1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 5

**Step 3: DEFINE**

Description: Please enter the text for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabText1  
Type: STRING  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 'Out1'

**Step 4: DEFINE**

Description: Please enter the font type for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabFontType1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 2  
Get Value from User: Yes  
Default value: 1

**Step 5: DEFINE**

Description: Please enter the text height for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabHeight1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 6: DEFINE**

Description: Please enter the text color for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabColor1  
Type: INTEGER  
Lower Bound: 1  
Upper Bound: 16

Get Value from User: Yes  
Default value: 4

**Step 7: DEFINE**

Description: Please enter the start row for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabStartRow1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 8: DEFINE**

Description: Please enter the start column for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabStartColumn1  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes  
Default value: 10

**Step 9: DEFINE**

Description: Please enter the status for the discrete output number 1.

Action: Define the following variable(s) for later use  
Variable Name: outputLabStatus1  
Type: STRING  
Lower Bound: 1  
Upper Bound: 1  
Get Value from User: Yes  
Default value: 0xC0

**Step 10: DEFINE**

Description: Please enter the second output (1-8) to use in the monitor discrete input test.

Action: Define the following variable(s) for later use  
Variable Name: output2  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 8  
Get Value from User: Yes  
Default value: 2

**Step 11: DEFINE**

Description: Please enter the label entry to use for the discrete output number 2.

Action: Define the following variable(s) for later use  
Variable Name: outputLabelIndex2  
Type: INTEGER  
Lower Bound: 0  
Upper Bound: 255  
Get Value from User: Yes

Default value: 6

**Step 12: DEFINE**

Description: Please enter the text for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabText2

Type: STRING

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 'Out2'

**Step 13: DEFINE**

Description: Please enter the font type for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabFontType2

Type: INTEGER

Lower Bound: 0

Upper Bound: 2

Get Value from User: Yes

Default value: 1

**Step 14: DEFINE**

Description: Please enter the text height for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabHeight2

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 10

**Step 15: DEFINE**

Description: Please enter the text color for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabColor2

Type: INTEGER

Lower Bound: 1

Upper Bound: 16

Get Value from User: Yes

Default value: 4

**Step 16: DEFINE**

Description: Please enter the start row for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabStartRow2

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 11

**Step 17: DEFINE**

Description: Please enter the start column for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabStartColumn2

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

Default value: 11

**Step 18: DEFINE**

Description: Please enter the status for the discrete output number 2.

Action: Define the following variable(s) for later use

Variable Name: outputLabStatus2

Type: STRING

Lower Bound: 1

Upper Bound: 1

Get Value from User: Yes

Default value: 0xC0

**Step 19: GET**

Description: Get original label 1

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<outputLabelIndex1>	orig_labelText1
labelFontType.<outputLabelIndex1>	orig_labelFontType1
labelHeight.<outputLabelIndex1>	orig_labelHeight1
labelColor.<outputLabelIndex1>	orig_labelColor1
labelStartRow.<outputLabelIndex1>	orig_labelStartRow1
labelStartColumn.<outputLabelIndex1>	orig_labelStartColumn1
labelStatus.<outputLabelIndex1>	orig_labelStatus1

**Step 20: GET**

Description: Get original label 2

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
labelText.<outputLabelIndex2>	orig_labelText2
labelFontType.<outputLabelIndex2>	orig_labelFontType2
labelHeight.<outputLabelIndex2>	orig_labelHeight2
labelColor.<outputLabelIndex2>	orig_labelColor2
labelStartRow.<outputLabelIndex2>	orig_labelStartRow2

<input type="checkbox"/>	labelStartColumn.<outputLabelIndex2>	orig_labelStartColumn2
<input type="checkbox"/>	labelStatus.<outputLabelIndex2>	orig_labelStatus2

**Step 21: SET**

Description: Set up output label 1

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelText.<outputLabelIndex1>	outputLabText1
<input type="checkbox"/>	labelFontType.<outputLabelIndex1>	outputLabFontType1
<input type="checkbox"/>	labelHeight.<outputLabelIndex1>	outputLabHeight1
<input type="checkbox"/>	labelColor.<outputLabelIndex1>	outputLabColor1
<input type="checkbox"/>	labelStartRow.<outputLabelIndex1>	outputLabStartRow1
<input type="checkbox"/>	labelStartColumn.<outputLabelIndex1>	outputLabStartColumn1
<input type="checkbox"/>	labelStatus.<outputLabelIndex1>	outputLabStatus1

**Step 22: SET**

Description: Set up output label 2

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	labelText.<outputLabelIndex2>	outputLabText2
<input type="checkbox"/>	labelFontType.<outputLabelIndex2>	outputLabFontType2
<input type="checkbox"/>	labelHeight.<outputLabelIndex2>	outputLabHeight2
<input type="checkbox"/>	labelColor.<outputLabelIndex2>	outputLabColor2
<input type="checkbox"/>	labelStartRow.<outputLabelIndex2>	outputLabStartRow2
<input type="checkbox"/>	labelStartColumn.<outputLabelIndex2>	outputLabStartColumn2
<input type="checkbox"/>	labelStatus.<outputLabelIndex2>	outputLabStatus2

**Step 23: SET**

Description: Turn off all outputs

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
<input type="checkbox"/>	outputControl.0	0x00 0x00

**Step 24: SET**

Description: Set up the label references for the outputs. [output1]=outputLabelIndex1  
[output2]=outputLabelIndex2

Action: Set the following object(s) to the value indicated:

<input type="checkbox"/>	Object	Value
--------------------------	--------	-------

outputLabelIndex.0	
--------------------	--

**Step 25: SET**

Description: Turn on output 1. If output 1 = 1 then set outputControl to 0x01 0x10

Action: Set the following object(s) to the value indicated:

Object	Value
outputControl.0	

**Step 26: USER-VERIFY**

Description: Verify that output 1 is on

Points: 1

**Step 27: GET**

Description: Get the status of outputs

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
outputStatus.0	outputStatus

**Step 28: VERIFY/GOTO**

Description: Verify the output is on

If outputStatus equals or 0x01

Award Points: 1

and Goto Step: 0

Otherwise Goto Step: 0

**Step 29: SET**

Description: Turn on output 2. If output 2 = 2 then set outputControl to 0x02 0x10

Action: Set the following object(s) to the value indicated:

Object	Value
outputControl.0	

**Step 30: USER-VERIFY**

Description: Verify that output 2 is on

Points: 1

**Step 31: GET**

Description: Get the status of outputs

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
outputStatus.0	outputStatus

**Step 32: VERIFY/GOTO**

Description: Verify the output 1 and 2 is on

If outputStatus equals or 0x03  
 Award Points: 1  
 and Goto Step: 0  
 Otherwise Goto Step: 0

**Step 33: SET**

Description: Turn off all outputs

Action: Set the following object(s) to the value indicated:

Object	Value
outputControl.0	0x00 0x00

**Step 34: USER-VERIFY**

Description: Verify that all outputs are off

Points: 1

**Step 35: GET**

Description: Get the status of outputs

Action: Get the following object(s) and store the returned value in the indicated variables:

Object	Variables
outputStatus.0	outputStatus

**Step 36: VERIFY/GOTO**

Description: Verify the output 1 and 2 are off

If outputStatus equals or 0x00  
 Award Points: 1  
 and Goto Step: 0  
 Otherwise Goto Step: 0

**Step 37: SET**

Description: Restore label 1

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<outputLabelIndex1>	orig_labelText1
labelFontType.<outputLabelIndex1>	orig_labelFontType1
labelHeight.<outputLabelIndex1>	orig_labelHeight1
labelColor.<outputLabelIndex1>	orig_labelColor1
labelStartRow.<outputLabelIndex1>	orig_labelStartRow1
labelStartColumn.<outputLabelIndex1>	orig_labelStartColumn1
labelStatus.<outputLabelIndex1>	orig_labelStatus1

**Step 38: SET**

Description: Restore label 2

Action: Set the following object(s) to the value indicated:

Object	Value
labelText.<outputLabelIndex2>	orig_labelText2
labelFontType.<outputLabelIndex2>	orig_labelFontType2
labelHeight.<outputLabelIndex2>	orig_labelHeight2
labelColor.<outputLabelIndex2>	orig_labelColor2
labelStartRow.<outputLabelIndex2>	orig_labelStartRow2
labelStartColumn.<outputLabelIndex2>	orig_labelStartColumn2
labelStatus.<outputLabelIndex2>	orig_labelStatus2

### ***Test Case TC53: Page Down***

This test case sends a page down command to the CCTV.

#### **Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

#### **Step 2: SET**

Description: Send the page down command

Action: Set the following object(s) to the value indicated:

Object	Value
menuControl.0	pageDown (1)

#### **Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the page down command on the screen.

Points: 1

#### **Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

### ***Test Case TC54: Page Up***

This test case sends a page up command to the CCTV

#### **Step 1: GET**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

Action: Get the following object(s) and store the returned value in the indicated variables:

	Object	Variables
--	--------	-----------

**Step 2: SET**

Description: Send the page up command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	pageUp (2)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the page down command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

### ***Test Case TC55: Cursor Up***

This test case sends a cursor up command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

**Step 2: SET**

Description: Send the cursor up command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	cursorUp (3)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the cursor up command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

### ***Test Case TC56: Cursor Down***

This test case sends a cursor down command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view

the execution.

**Step 2: SET**

Description: Send the cursor down command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	cursorDown (4)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the cursor down command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC57: Cursor Right**

This test case sends a cursor right command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

**Step 2: SET**

Description: Send the cursor right command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	cursorRight (5)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the cursor right command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC58: Cursor Left**

This test case sends a cursor left command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view

the execution.

**Step 2: SET**

Description: Send the cursor left command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	cursorLeft (6)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the cursor left command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC59: Increment**

This test case sends an increment command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

**Step 2: SET**

Description: Send the increment value command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	incrementValue (7)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the increment value command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC60: Decrement Value**

This test case sends and decrement command to the CCTV.

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view

the execution.

**Step 2: SET**

Description: Send the decrement value command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	decrementValue (8)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the decrement command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC61: Enter Value**

This test case sends an enter command to the CCTV

**Step 1: PRE-CONDITIONS**

Description: Run the test titled "Activate Menu" or "Activate Menu with Timeout" in order to display menu. Ensure menu is in a position so that the user can instruct this command and view the execution.

**Step 2: SET**

Description: Send the enter value command.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuControl.0	enterValue (9)

**Step 3: USER-VERIFY**

Description: Ensure the device correctly executed the enter value command on the screen.

Points: 1

**Step 4: POST-CONDITIONS**

Description: Run the test titled "Deactivate Menu" if the user wishes to turn off the menu.

**Test Case TC62: Activate Menu**

This test case activates the menu of the CCTV.

**Step 1: SET**

Description: Activate the menu.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuActivate.0	255

**Step 2: USER-VERIFY**

Description: Verify that the menu is displayed on the video output and stays displayed until commanded to be turned off.

Points: 1

**Test Case TC63: Activate Menu with Timeout**

This test case activates the menu with a timeout value. When the time has expired, the menu shall turn off.

**Step 1: DEFINE**

Description: Used to test menu activation for a number of seconds.

Action: Define the following variable(s) for later use

Variable Name: activateMenuTimeout

Type: INTEGER

Lower Bound: 0

Upper Bound: 255

Get Value from User: Yes

**Step 2: SET**

Description: Activate menu with a timeout.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuActivate.0	activateMenuTimeout

**Step 3: USER-VERIFY**

Description: Verify that the menu is displayed on the video output for activateMenuTimeout seconds.

Points: 1

**Test Case TC64: Deactivate Menu**

This test case turns off the internal camera menu.

**Step 1: SET**

Description: Deactivate the menu.

Action: Set the following object(s) to the value indicated:

	Object	Value
	menuActivate.0	0

**Step 2: USER-VERIFY**

Description: Verify that the menu has disappeared from the video output.

Points: 1

### Annex – Questions about NTCIP 1205

1. Why is the Database Management conformance group listed as an option for 1205? I might not be seeing the big picture but I couldn't really think of a reason that CCTV should have this conformance group included at all. Database management is for controlling the acceptance of large amounts of object SET's. Basically the central controller would put the database into a mode that would queue up or buffer the SNMP SET's and not change the actual value of all of the objects until commanded to do so.
2. Why is the Time Management conformance group listed as an option for 1205? At first I thought the time and date might be utilized in labels as an option but I could not find it there. Then I thought the schedule or the report conformance group but those aren't even options. I just couldn't find a usage in 1205.
3. Why isn't the Security conformance group from 1201 an option for 1205? It seems to me that this group should be at least an option.
4. The description of the position objects is hard to understand.
  - 4.1. How do the four modes of operation act? (definition) It is not obvious at all what the exact value of each byte is supposed to be.
  - 4.2. Byte 1 is apparently a choice of stop movement, delta, absolute, or continuous movement, what are the enumerations? I read the PositionReference syntax but I am still confused by it. How do I know if byte 3 and 4 are position or offset.
  - 4.3. I assume that a (+) value means a range from 0..127 and (-) means a range from 128..255 in literal byte value. Is this a correct assumption?
5. labelFontType object - what is the definition of these two options for this object? How do I know what these look like? Should I care?
6. labelHeight object - What is this measurement in? Is it pixels, characters? (same for labelStartRow)
7. zoneLabel object - Is zero (0) a valid label? or does it mean no label
8. The 1205 standard requires the CCTV conformance group which includes the label table, however, I only see one minor usage of this label table. The major usage of this label table is in the Extended conformance group (zones). Is this intended or should the label table be in the Extended conformance group.
9. Discrete input and output
  - 9.1. I noticed that inputs and outputs are associated with label indexes. Is the label supposed to be displayed on the video output when the discrete input is ON? same for discrete outputs? So this means the video could have many labels scattered across the screen?

## 9.2. Same question for alarms

10. I noticed that the timeout parameters can be a value of zero which means that the particular timeout is not supported. What happens if this object is set to a different value? `genErr`?
11. I noticed that the objects `systemCameraFeatureStatus` and `systemLensFeatureStatus` have similar functionality but `systemCameraFeatureStatus` is read-only and `systemLensFeatureStatus` is read-write. Are these both supposed to be read-write?
12. Why is `alarmTemperatureCurrentValue`, `alarmPressureCurrentValue`, `alarmWasherFluidCurrentValue` read-write? It seems that this would never be SET through the SNMP interface.
13. Why is `alarmLabelIndex` read-only? It seems that one would want to be able to change the label reference for these alarms. `inputLabelIndex` and `outputLabelIndex` are too, why are these fixed?
14. If `alarmLatchClear` is set with `latch = ON`, does the `alarmLatchClear = 0` when read or would the `alarmLatchClear` need to be set with `latch = OFF` before it is latchable.
15. Is the alarm truly latched or is it this type of alarm just triggered and the value is held until cleared?
16. If an alarm is to display a label on the video when it occurs, does it happen upon a latch status or does it occur during an active status? I am assuming these are two distinct types of alarms, active and latched.
  - 16.1. What about `inputLatch`? Does it have the same behavior.
17. What happens when there are overlapping zones? Should the first zone in the table that fits the position of the camera be the one that displays the label? The standard should indicate how the correct zone is determined.
18. True North Offset: The definition of the true North offset is incomplete. This is the offset between true North and the Home position in the clockwise or counterclockwise direction.
19. What is the purpose of value for the Home Position? It appears to be a measurement from one arbitrary point to another arbitrary point.
20. Is it a requirement that the camera be able to pan, tilt, and zoom to the home position; or could a manufacturer define Home to be a position that the camera can not reach?
21. Is there anyway to send a command to the camera to force it to the Home position (assuming that the Home position is reachable)? It would appear that if the camera is pointing one degree clockwise from Home and you told it to go to the absolute position of Home with a speed of +1, it could not get there if the Pan Right Limit was less than 360 degrees.
22. If, in the above case, you told the camera to go home with a speed of +1, should it report back an error, or should it pan to the Pan Right Limit (or go home)? Same issues with tilt and zoom.
23. What does a positive speed mean in a move command; what about a negative speed?

24. rangeMinimumPanStepAngle and rangeMinimumTiltStepAngle: The title for these are confusing (Maximum Pan Step Angle Parameter). If this is intended to be minimum, it seems that the step angle can be equal to or larger than this value. It seems you would want the limit to be a maximum so that the device does not step larger than this limit. The title in the text and the object name should have the same verbage. The descriptive text should indicate that any step made by the device cannot be larger than this value ( or smaller whichever is intended ).
25. labelStatus – This description of this usage is very ambiguous. Is bit 7 for determining whether this particular label entry is currently being displayed? Is bit 6 for turning off a particular entry?