Dialight IntelliLED EtherNet/IP Adapter

User Reference Manual

Lighting System

The IntelliLED Lighting System Server provides an automated interface, through which a user can control and monitor all the connected devices in a building-wide light control system.

Overview

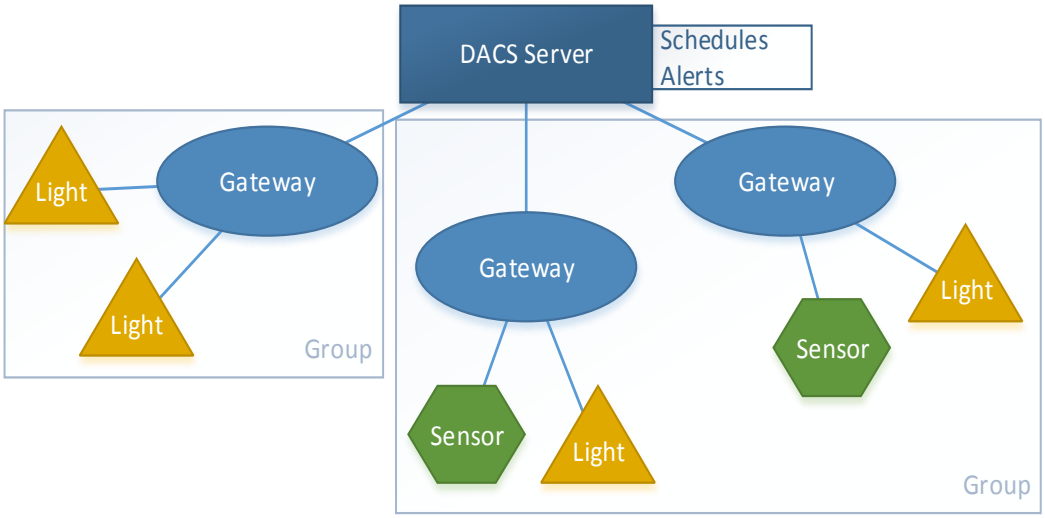
A IntelliLED Lighting system consists of multiple 'nodes' in a network:

- **IntelliLED Server**: The IntelliLED Server is the 'root' node of the system. It controls and monitors all other nodes on the network.
- **Gateway**: Gateways interact directly with the IntelliLED Server. They provide a network junction through which groups of lights and sensors can connect.
- **Devices**: "Devices" consist of any lights or sensors which are part of the lighting ecosystem. Devices participate in the network by connecting to a gateway.

In addition to these physical nodes, the IntelliLED Server provides a number of virtual 'objects' used for grouping, scheduling, and other diagnostic purposes:

- **Groups**: represent a logical 'grouping' of multiple devices.
- **Schedules**: contain a list of logical rules, used for controlling the behavior of the lighting system.
- **Alerts**: contain important diagnostic information intended for the system administrator.

An example of a typical lighting system tree is provided in the following chart:



Note: Every gateway, device, group, schedule and alert provides a unique "UID" string value, with which it can be identified or addressed.

EtherNet/IP Connectivity

The IntelliLED Server provides EtherNet/IP connectivity, via an Adapter interface running on the server itself. This EtherNet/IP Adapter acts as a protocol bridge between a PLC (or compatible EtherNet/IP device) and all participating nodes and objects in the Dialight lighting control system.
All devices, gateways, groups, schedules and alerts in the IntelliLED lighting system are modeled as a series of "IntelliLED Objects," which are uniquely identified by a string *UID* value. Each IntelliLED Object provides a set of "IntelliLED Object Properties" with read and/or write access. It is through the various IntelliLED Object Properties that an EtherNet/IP-enabled controller can interact with nodes and affect behavior on the IntelliLED lighting system.
On the EtherNet/IP side of the bridge; the application defines a set of vendor-specific EtherNet/IP application objects, through which incoming EtherNet/IP requests are translated and forwarded to the IntelliLED server. Once a response is received from the IntelliLED server, the adapter converts and delivers the response to the requesting device. These vendor-specific EtherNet/IP Application Objects are described in further detail in this document.

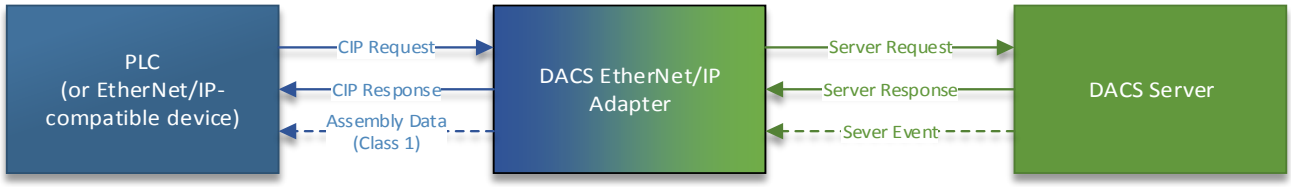


Figure 1- Communication between PLC and IntelliLED server

In addition to the explicit request/response paradigm, an EtherNet/IP-enabled device can *monitor* the lighting system for changes in state, using a Class 1 I/O connection and reacting to new data. These changes in state are triggered by "IntelliLED Server Events," originating from the lighting system.

IntelliLED Object Properties

The IntelliLED EtherNet/IP Adapter provides a 'Get/Set' API, providing read and/or write access to a IntelliLED Object's individual properties. The properties available to the target IntelliLED Object depend on the underlying type; gateway, device, sensor, etc. Some IntelliLED Object types can inherit groups of properties from a common 'base' type.
The IntelliLED Server itself provides some 'root-level' properties only available by addressing the IntelliLED Server itself, using the reserved UID: "**Lighting-System**".
Each property is identified by a unique 16-bit ID. Each property is reported in its prescribed CIP Data Type.
The full list of properties is defined below, including the parent IntelliLED Object 'type,' allowable access, the unique ID, and a brief description. Some properties require additional 'argument' parameters. They are identified in the "Required Args" column.

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
Lighting System					
0x0001	Get	GATEWAY_LIST_SIZE	UDINT		
0x0002	Get	GATEWAY_LIST_GET_ELEMENT	STRING	Return the UID of the Gateway at the specified index	Index of the desired element (UDINT)
0x0003	Get	ALERT_LIST_SIZE	UDINT		
0x0004	Get	ALERT_LIST_GET_ELEMENT	STRING	Return the UID of the Alert at the specified index	Index (UDINT)
0x0005	Get	SCHEDULE_LIST_SIZE	UDINT		
0x0006	Get	SCHEDULE_LIST_GET_ELEMENT	STRING	Return the UID of the Schedule at the specified index	Index (UDINT)
0x0007	Get	GROUP_LIST_SIZE	UDINT		
0x0008	Get	GROUP_LIST_GET_ELEMENT	STRING	Return the UID of the Group at the specified index	Index (UDINT)
0x0009	Get	DEVICE_LIST_SIZE	UDINT		
0x000A	Get	DEVICE_LIST_GET_ELEMENT	STRING	Return the UID of the Device at the specified index	Index (UDINT)
0x000B	Get / Set	EMERGENCY_OVERRIDE	BOOL		
0x000C	Get	SYSTEM_POWER_CONSUMPTION	UDINT		
0x000D	Get	SYSTEM_SOFTWARE_VERSION	STRING		

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
0x000E	Set	GROUP_LIST_CREATE_ELEMENT	None*	Create a new Group object. Creation is performed asynchronously, and completion is signaled by the "GROUP_LIST_SIZE Changed" event. Once the signal has been triggered, a client can retrieve the UID of the newly-created object by getting the UID of the last element in the Group list (GROUP_LIST_GET_ELEMENT).	
0x000F	Set	GROUP_LIST_DELETE_ELEMENT	UINT32	Delete the Group object located at the given index.	The index of the Group object to delete, as positioned in the GROUP_LIST array.
0x0010	Set	SCHEDULE_LIST_CREATE_ELEMENT	None*	Create a new Schedule object. *Creation is performed asynchronously, and completion is signaled by the "SCHEDULE_LIST_	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
				SIZE Changed" event. Once the signal has been triggered, a client can retrieve the UID of the newly-created object by getting the UID of the last element in the Schedule list (SCHEDULE_LIST_GET_ELEMENT).	
0x0011	Set	SCHEDULE_LIST_DELETE_ELEMENT	UINT32	Delete the Schedule object located at the given index.	The index of the Schedule object to delete, as positioned in the SCHEDULE_LIST array.
Device					
0x0100	Get / Set	NAME	STRING	Friendly name for the device.	
0x0101	Get	UID	STRING	Unique identifier	
0x0102	Get	MODEL_NAME	STRING	This is a human readable model description	
0x0103	Get	MODEL_NUMBER	STRING	This is manufacturers part number for the device	
0x0104	Get	MANUFACTURER	STRING	String representation of the product manufacturer	
0x0105	Get	DEVICE_TYPE	USINT	Device object subtype (see	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
				Appendix A for enumerated list of device sub-types)	
0x0106	Get	CONNECTED	BOOL	Is the device currently online	
0x0107	Get	FIRMWARE_VERSION	STRING	Software version of the device	
0x0108	Get / Set	IDENTIFY	BOOL	Is identify mode currently enabled?	
Basic Light (Inherits: Device)					
0x0200	Get / Set	LIGHT_LEVEL	USINT	Dimmer level of light	
0x0201	Get / Set	OVERRIDE	BOOL	Flag for if the level is currently overridden	
HB Light (Inherits: Device, Basic Light)					
0x0300	Get	POWER_CONSUMED	UDINT	Watts consumed x 10	
0x0301	Get	LUMENS_OUTPUT	UDINT	light output in lumens	
0x0302	Get	TEMPERATURE	INT	Temp in F (signed INT)	
0x0303	Get	LAMP_HOURS	UDINT	Total hours of lamp operation	
0x0304	Get	RF_STRENGTH	UINT	Number representing signal strength	
0x0305	Get	RF_PARENT	STRING	UID of network parent	
Basic Sensor (Inherits: Device)					
0x0400	Get	SENSOR_SUB_TYPE	USINT	Sensor sub-type (see Appendix A for enumerated list of sensor sub-types)	
WOS Sensor (Inherits: Device, Basic Sensor)					
0x0450	Get	MOTION_SENSED	BOOL	True == Motion Sensed	
0x0451	Get	BATT_VOLTAGE	UINT	Voltage x10	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
WDLH Sensor (Inherits: Device, Basic Sensor)					
0x0500	Get	LUX	UINT	Light level reading in LUX	
Gateway (Inherits: Device)					
0x0600	Get	JN_FIRMWARE_VERSION	STRING	Firmware version of JN co-processor	
0x0601	Get	SERVER_IP_ADDR	STRING	IP address as a string	
0x0602	Get	SERVER_PORT	UINT	port addr	
0x0603	Get	ETHERNET_TYPE	STRING	DHCP or Static	
0x0604	Get	GATEWAY_IP_ADDR	STRING	Ethernet IP address of the gateway itself	
0x0605	Get	GATEWAY_NETMASK	STRING	Network mask of the gateway	
0x0606	Get	GATEWAY_ROUTER	STRING	Ethernet IP gateway address setting for the gateway	
0x0607	Get	WIFI_SSID	STRING	SSID for the onboard Wi-Fi host	
0x0608	Get	WIFI_CHANNEL	UINT	Wi-Fi channel to use for hostapd	
0x0609	Get / Set	DISCOVER	BOOL	Is discovery mode enabled	
0x060A	Get	GWY_DEVICE_LIST_SIZE	UDINT	Number of elements in gateway device list	
0x060B	Get	GWY_DEVICE_LIST_GET_ELEMENT	STRING	Get device UID at index	Index (UDINT)
Group					
0x0700	Get / Set	GRP_NAME	STRING	Friendly name for the group	
0x0701	Get / Set	GRP_DESCRIPTION	STRING	User defined description for the group	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
0x0702	Get	GRP_POWER_CONSUMED	UDINT	Total power for all devices in the group	
0x0703	Get / Set	GRP_LIGHT_LEVEL	USINT	Dimmer level for the group	
0x0704	Get / Set	GRP_OVERRIDE	BOOL	Flag to command group manual override	
0x0705	Get	GRP_DEVICE_LIST_SIZE	UDINT	Get number of elements in group device list	
0x0706	Get	GRP_DEVICE_LIST_GET_ELEMENT	STRING	Get device UID at index	Index (UDINT)
0x0707	Set	GRP_DEVICE_LIST_ADD_ELEMENT	STRING	Add device to group	The UID of the device to add. (STRING)
0x0708	Set	GRP_DEVICE_LIST_REMOVE_ELEMENT	STRING	Remove device from group	The UID of the device to remove. (STRING)
0x0709	Get	GRP_SCHED_LIST_SIZE	UDINT	Get number of elements in group schedule list	
0x070A	Get	GRP_SCHED_LIST_GET_ELEMENT	STRING	Get schedule UID at index	Index (UDINT)
0x070B	Set	GRP_SCHED_LIST_ADD_ELEMENT	STRING	Add Schedule to group	The UID of the device to add. (STRING)
0x070C	Set	GRP_SCHED_LIST_REMOVE_ELEMENT	STRING	Remove Schedule from group	The UID of the device to remove. (STRING)
0x070D	Get / Set	GRP_WOS_ENABLED	BOOL	Flag for if WOS is currently enabled	
0x070E	Get / Set	GRP_WDLH_ENABLED	BOOL	Flag for if WDLH is currently enabled	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
0x070F	Get / Set	GRP_TARGET_LUX	UINT	Target lux value for DLH to adjust to	
0x0710	Get	GRP_MEASURED_LUX	UINT	Current measured lux for the group	
0x0711	Get / Set	GRP_WDLH_MODE	USINT	Current WDLH mode (See Appendix A for enumerated list of modes.)	
0x0712	Get	GRP_STEPDOWN_LIST_SIZE	UINT32		
0x0713	Get	GRP_STEPDOWN_LIST_GET_ELEMENT	STRUCT of STEPDOWN_ELEMENT	Get stepdown schedule item at index.	Index (UINT32)
		DURATION	UINT8		
		LIGHT_LEVEL	UINT8		
0x0714	Set	GRP_STEPDOWN_LIST_ADD_ELEMENT	STRUCT of STEPDOWN_ELEMENT	Add stepdown schedule item to end of current list.	
		DURATION	UINT8		
		LIGHT_LEVEL	UINT8		
0x0715	Set	GRP_STEPDOWN_LIST_REMOVE_ELEMENT		Remove stepdown schedule item from index.	Index (UINT32)
Schedule					
0x0800	Get	SCH_NAME	STRING	Friendly name for the schedule	
0x0801	Get	SCH_DESCRIPTION	STRING	User defined description for the schedule	
0x0802	Get	SCH_DOWEEK	USINT	Days of the week the schedule should run (Note: follows the Unix cron format:	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
				both bit 0 and 7 denote Sunday, bit 1-6 denote Mon-Sat).	
0x0803	Get	SCH_EVENT_LIST_SIZE	UDINT	Number of events in the schedule	
0x0804	Get	SCH_EVENT_LIST_GET_ELEMENT	STRUCT of EVENT_DATA	Event configuration structure	Index (UDINT)
		EVENT_ID	UDINT		
		EVENT_TIME_SEC	USINT		
		EVENT_TIME_MIN	USINT		
		EVENT_TIME_HOUR	USINT		
		EVENT_PARAM	UINT		
		EVENT_VALUE	USINT		
0x0805	Set	SCH_EVENT_LIST_ADD_ELEMENT		Add event to the current schedule. * The EVENT_ID value is ignored. The server will populate this value.	
		EVENT_ID	UINT32		
		EVENT_TIME_SEC	UINT8		
		EVENT_TIME_MIN	UINT8		
		EVENT_TIME_HOUR	UINT8		
		EVENT_PARAM	UINT16		
		EVENT_VALUE	UINT8		
0x0806	Set	SCH_EVENT_LIST_REMOVE_ELEMENT		Remove event from index.	Index (UINT32)
Alerts					
0x1000	Get	ALERT_EVENT_UID	STRING	Event UID that triggered the event	
0x1001	Get	ALERT_UID	STRING	Unique identifier for the alert itself	
0x1002	Get	ALERT_TYPE	USINT	The severity of the Alert (see Appendix A for enumerated list of alert types)	
0x1003	Get	ALERT_STRING	STRING	Information string about the alert	

Property ID	Access	Property Name	CIP Data Type	Description	Required Args
0x1004	Get / Set	ALERT_STATE	USINT	The state of the alert (see Appendix A for enumerated list of alert types)	
0x1005	Get	ALERT_TIMESTAMP	UDINT	Unix timestamp of when the alert occurred	
0x1006	Get	ALERT_DEVICE_UID	STRING	Device UID that caused alert	

IntelliLED Server Events

In some cases it may not be efficient to continually 'poll' a desired property, to detect changes in state. For these cases, the IntelliLED Server provides a simple "Event" system, to alert controllers of changes in the state of the system.
Each event is identified by a unique, 8-bit ID number. The complete list of possible IntelliLED Server "Events" is defined below, accompanied with a brief description.

Event ID	Event Name	Description
1	GATEWAY_LIST_SIZE Changed	Indicates a change in the value of the Lighting System's GATEWAY_LIST_SIZE attribute
2	ALERT_LIST_SIZE Changed	Indicates a change in the value of the Lighting System's ALERT _LIST_SIZE attribute
3	SCHEDULE_LIST_SIZE Changed	Indicates a change in the value of the Lighting System's SCHEDULE _LIST_SIZE attribute
4	GROUP_LIST_SIZE Changed	Indicates a change in the value of the Lighting System's GROUP _LIST_SIZE attribute
5	DEVICE_LIST_SIZE Changed	Indicates a change in the value of the Lighting System's DEVICE _LIST_SIZE attribute
6	EMERGENCY_OVERRIDE Changed	Indicates a change in the value of the Lighting System's EMERGENCY_OVERRIDE attribute
7	Motion Sensed	Indicates a sensor device has detected motion.
8	GWY_DEVICE_LIST_SIZE Changed	Indicates a change in the value of a gateway's GWY_DEVICE _LIST_SIZE attribute.
9	GRP_LIGHT_LEVEL Changed	Indicates a change in the value of a group's GRP_LIGHT_LEVEL attribute.
10	GRP_OVERRIDE Changed	Indicates a change in the value of a group's GRP_ OVERRIDE attribute.
11	GRP_DEVICE_LIST_SIZE Changed	Indicates a change in the value of a gateway's GRP_DEVICE_LIST_SIZE attribute.
12	GRP_SCHEDULE_LIST_SIZE Changed	Indicates a change in the value of a gateway's GRP_ SCHEDULE _LIST_SIZE attribute.
13	GRP_WOS_ENABLED Changed	Indicates a change in the value of a gateway's GRP_ WOS_ENABLED attribute.
14	GRP_WDLH_ENABLED Changed	Indicates a change in the value of a gateway's GRP_ WDLH_ENABLED attribute.
15	SCH_EVENT_LIST_SIZE Changed	Indicates a change in the value of a schedule's SCH_EVENT_LIST_SIZE attribute.

EtherNet/IP Application Objects

The IntelliLED EtherNet/IP Adapter provides a set of vendor-specific CIP Application Objects, through which a controller can interact with the IntelliLED Object Properties and IntelliLED Server Events, outlined above.

"IntelliLED Control" Object

The "IntelliLED Control" object is the CIP object through which all "Get Property" and "Set Property" requests are directed.

Class Code

The Class ID of the IntelliLED Control object is **100 (hex 0x64)**.

Class Attributes

The Adapter provides the following attributes at the "class" level (**instance 0**):

Attribute	Access	Name	CIP Data Type	Description
0x01	Get	Revision	UINT	Get the current revision of the class definition. The current revision of the class is **2.**

Instances

The Adapter provides only a single instance of the IntelliLED Control object: **instance 1 (hex: 0x01)**.

Instance Services

The "IntelliLED Control" object provides object-specific CIP services, providing a means through which to Get & Set IntelliLED Server properties. These services are defined below with a brief description. Service-specific request data is outlined in the following sections.

Service Code	Service Name	Description
0x4B	Get IntelliLED Property	Read the value of the specified IntelliLED Property, from the object with the specified IntelliLED Object UID.
0x4C	Set IntelliLED Property	Write a value to the specified IntelliLED Property, to the object with the specified IntelliLED Object UID.

"Get Property" Request Data

The "Get Property" service accepts a Property ID value and IntelliLED Object UID value, and returns the value of the specified property. Additionally, some IntelliLED Server Properties require an additional parameter, such as the desired Index of an array. The request data for a "Get Property" service request is defined below:

Required?	Name	EIP Data type	Description
Required	Property ID	UINT	The 16-bit property identifier of the desired property. For the full list of properties, refer to the table of properties, defined above in "IntelliLED Object Properties."
Required	IntelliLED Object UID	STRING	The UID specifying the desired IntelliLED device, gateway, group, schedule or alert being addressed. **Note:** The Lighting System has a reserved UID string value of "**Lighting-System**"
Conditional	Parameter	<varies>	For some properties, an additional parameter may be required. These properties are identified in the "IntelliLED Object Properties" table, above, in the "Additional Request Args" column.

Example

An example "Get Property" service request is outlined below. This example retrieves the current "System Software Version," from the IntelliLED Server Lighting System.

Service Code	75 (hex: 0x4B)		
Class ID	100 (hex: 0x64)		
Instance Number	1 (hex: 0x01)		
Request Data (Length: 19 bytes)			
Name	Property ID	Target IntelliLED Object UID	Parameter
Raw Data	0D 00	0F 00 4C 69 67 68 74 69 6E 67 2D 53 79 73 74 65 6D	[N/A]
Value	[UINT] 13 (0x000D) ('SYSTEM_ SOFTWARE_ VERSION')	[STRING] 'Lighting-System'	[N/A]

Note: the length of the request data will fluctuate, with variable-length String values.

Set Property

The “Set Property” service accepts a Property ID value and IntelliLED Object UID value, *as well* as the new value to assign to the property. The request data for a “Set Property” service request is defined below:

Required?	Name	EIP Data type	Description
Required	Property ID	UINT	The 16-bit property identifier of the desired property. For the full list of properties, refer to the table of properties, defined above in “IntelliLED Object Properties.”
Required	IntelliLED Object UID	STRING	The UID specifying the desired IntelliLED device, gateway, group, schedule or alert being addressed. **Note:** The Lighting System has a reserved UID string value of “**Lighting-System**"
Required	Value	<varies>	The new value to assign to the target property. Value must match the data type of the property, as defined in the above table “IntelliLED Object Properties.”

Example

An example “Set Property” service request is outlined below. This example assigns the Light Level of a light named “MyExampleLight” to 100% (hex 0xFF).

Service Code	76 (hex: 0x4C)		
Class ID	100 (hex: 0x64)		
Instance Number	1 (hex: 0x01)		
Request Data (Length: 19 bytes)			
Name	Property ID	Target IntelliLED Object UID	Value
Raw Data	00 02	0E 00 4D 79 45 78 61 6D 70 6C 65 4C 69 67 68 74	FF
Value	[UINT] 512 (0x0200) (‘LIGHT_LEVEL’)	[STRING] ‘MyExampleLight’	[USINT] 255 (0xFF) (100%)

Note: the length of the request data will fluctuate, with variable-length String values.

"IntelliLED Event History" Object

The "IntelliLED Event History" object stores a history of every IntelliLED Event raised by the IntelliLED server. For each individual type of event (defined above, in the "IntelliLED Server Events" table), the "IntelliLED Event History" stores a list of IntelliLED Server Objects which have recently raised the specific event.
Each event's history is stored in the form of a circular buffer. Each history buffer can store up to 255 entries. For any given event history buffer, an EtherNet/IP-compatible device can retrieve:

- The current index of the most recently-occurring instance of the event.
- Given a desired index; the UID of the IntelliLED Server Object which raised the event

Class Code

The Class ID of the IntelliLED Event Queue object is **101** (**hex 0x65).**

Class Attributes

The "IntelliLED Event History" object provides the following attributes at the "class" level (**instance 0**):

Attribute	Access	Name	CIP Data Type	Description
0x01	Get	Revision	UINT	Get the current revision of the class definition. The initial revision of the class is **1.**

Instances

The Adapter provides only a single instance of the IntelliLED Event History object: **instance 1 (hex 0x01).**

Instance Services

The "IntelliLED Event History" object provides object-specific CIP services, providing a means through which to retrieve the index of the 'most recent' instance of a given event; as well as a means through which to retrieve specific instances of a given event. These services are defined below, with a brief description.
Service-specific request data is outlined in the following sections.

Service Code	Service Name	Description
0x4B	Get IntelliLED Event Entry	Retrieve the UID of the IntelliLED object which raised the event with the specified index, in the Event Monitor Assembly.
0x4C	Get Current Index	Retrieve the 'current index' of the most recent event entry, in the specified event's history.

"Get IntelliLED Event Entry" Request Data

The "Get IntelliLED Event Entry" service accepts an 8-bit "Event ID" value, and a desired 8-bit "Index" value, representing the index within the circular buffer from which to retrieve the event data. The service returns the UID of the IntelliLED Server Object which raised the desired event. The request data for a "Get IntelliLED Event Entry" service request is defined below:

Required or Optional	Name	CIP Data type	Description
Required	Event ID	USINT	The 8-bit identifier of the desired event. For the full list of supported events, refer to the full table: IntelliLED Server Events.
Required	Buffer Index	USINT	The 8-bit index – within the event's circular buffer – of the desired event history entry.

Example

An example "Get IntelliLED Event Entry" service request is outlined below. This example retrieves UID of the IntelliLED 'gateway' object which raised the "GWY_DEVICE_LIST_SIZE Changed" event. It retrieves the UID from index '120' of the event's circular buffer.

Service Code	75 (hex: 0x4B)	
Class ID	101 (hex: 0x65)	
Instance Number	1 (hex: 0x01)	
Request Data (Length: 2 bytes)		
Name	Event ID	Buffer Index
Raw Data	08	78
Value	[USINT] 8 (0x08) ('GWY_DEVICE_LIST_SIZE Changed')	[USINT] 120 (0x78) ('Index 120')

"Get Current Index" Request Data

The "Get Current Index" service accepts an Event ID value, and returns the 'current index' of the most recent instance within that event's circular buffer. The request data for a "Get Current Index" service request is defined below:

Required or Optional	Name	CIP Data type	Description
Required	Event ID	USINT	The 8-bit Event identifier of the desired event history buffer. For the full list of supported events, refer to the table IntelliLED Server Events.

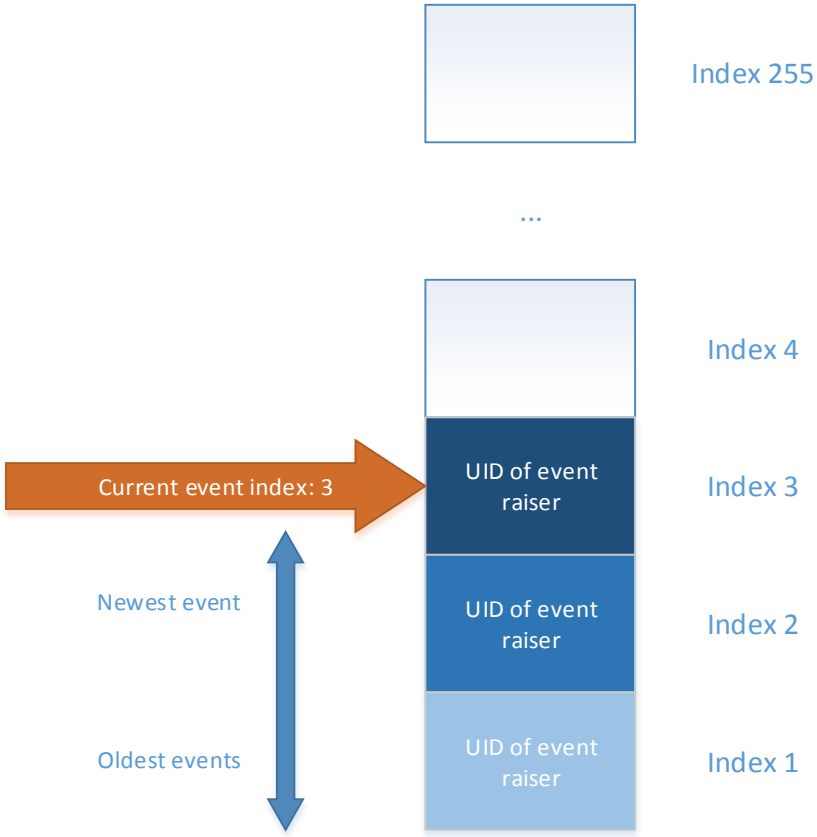
Example

An example "Get Current Index" service request is outlined below. This example retrieves the most recent index of the IntelliLED "EMERGENCY_OVERRIDE Changed" event.

Service Code	75 (hex: 0x4B)
Class ID	101 (hex: 0x65)
Instance Number	1 (hex: 0x01)
Request Data (Length: 1 byte)	
Name	Event ID
Raw Data	06
Value	[USINT] 6 (0x06) ('EMERGENCY_OVERRIDE Changed')

Event History: Circular Buffer Behavior

Each event type maintains its own circular buffer. Index 0 (zero) of any buffer is never used; if the reported 'current index' of any buffer returns zero, it indicates an empty buffer with no events. Any reported 'current index' with a non-zero value indicates one or more events, stored in the buffer. Once more than 255 events have been received, the most recent event is placed in index '1,' with previous events stored at index 255, 254, 253, etc.



Assembly Object: “Event Monitor”

To avoid the inefficient task of constantly ‘polling’ the IntelliLED Event History object for new events, a single assembly is provided through which a PLC may establish a real-time, Class 1 I/O connection. Each byte within the assembly represents the “Current Index” value of the IntelliLED Server Event, whose ID matches the index of the byte.
Byte index 1 in the assembly contains the current index for the IntelliLED Server Event with ID of “1” (‘GATEWAY_LIST_SIZE Changed’). Byte index 2 contains the event count for the IntelliLED Server Event with ID of “2” (‘ALERT_LIST_SIZE Changed’), and so on. Refer to the “IntelliLED Server Events” table, for the complete list of supported events.
The byte at index 0 is not used. Bytes beyond index 15 are reserved for future use.

Usage

A PLC or EtherNet/IP-enabled device will monitor the values within this assembly. When a relevant event count is changed, the PLC will retrieve the relevant event parameters from the “IntelliLED Event History” object, as described in the previous section.

Class 1 I/O Connection Parameters

A Class 1 “Input-Only” connection can be established with the Event Monitor assembly, using the following connection points:

Originator to Target Connection Point (“Heartbeat”):

- Instance 198 (hex: 0xC6)
- Size: 0
- Run/Idle header: no

Target to Originator Connection Point (“Event Monitor”):

- Instance 100 (hex: 0x64)
- Size: 128
- Run/Idle header: no

PLC Example Logic

Included with the IntelliLED EtherNet/IP Adapter, a number of PLC example programs have been provided. These examples illustrate simple use-cases which end-users can build upon or use for reference.

Common Subroutines

In all example programs, two subroutines are referenced frequently; “EncodeRequest” and “DecodeResponse.” These subroutines allow a user to quickly ‘pack’ and ‘unpack’ their desired request parameters (and corresponding response values) to and from an simple array of bytes, to be used in the PLC’s “Message” instruction when interacting with the IntelliLED EtherNet/IP Adapter.

“EncodeRequest” Subroutine

The “EncodeRequest” subroutine takes 4 parameters:

1. **RequestBlock** – This parameter should be the same as the “return” value. It represents an array of bytes, to be used in a MESSAGE instruction, when requesting data from the IntelliLED Server.
2. **PropertyID** – The ID of the property being retrieved (or assigned).
3. **TargetUID** – The UID string identifying the IntelliLED Object being interacted with.
4. **Parameter** – An optional parameter to be appended to the end of the request block. For “Get Property” requests, this can be the desired index of an element in an array. For “Set” requests, this should be the value being assigned to the propery.

It returns the resulting packed block of request data. The resulting request data should be used in a subsequent MESSAGE instruction.

“DecodeResponse” Subroutine

The “DecodeResponse” subroutine takes 3 parameters:

1. **ResponseBlock** – This parameter should be the same as the “return” value. It represents an array of bytes, to be used in a MESSAGE instruction, when requesting data from the IntelliLED Server.
2. **ResponseType** – A string, identifying the data type into which the response block should be parsed. The subroutine supports the following data type strings:
    a. “INT”
    b. “DINT”
    c. “STRING”
3. **ResponseBlockReturn** – This parameter should be the same as the “return” value. It contains the resulting parsed data value from the response block, placed into the member associated with the described “ResponseType.”

It returns a structure containing the various supported data types. The parsed value will be located in the member with the prescribed data type (“ResponseType”).

Example 01: Set a light level to 100%, when a motion sensor is triggered

In this simple example, suppose a PLC wants to turn a specific light on, whenever an accompanying motion sensor detects motion.
Assume:

- The light has a UID of "MyExampleLight"
- The sensor has a UID of "MyExampleSensor"

This example is implemented with the following logic:

1. Read the input data from the IntelliLED EtherNet/IP Adapter, over the network.
2. Check the value of the input byte at **index 7 ("Motion Sensed").**
3. When the value changes:
    a. Query the "IntelliLED Event History" object: **Service 0x4B, Class 0x65, Instance 1**. Using a MESSAGE instruction, we request the IntelliLED **"Motion Sensed" (value: 0x07)** event history, using the value we retrieved from the input data, in the previous step. This retrieves the UID of the device which raised the most recent instance of the "Motion Sensed" event.
        i. Since requests for event data consist of only 2 bytes, use the MOV and COP instructions to copy the desired "Event ID" and "Event Index" to a request structure.
    b. Use the "DecodeResponse" subroutine to parse the response data. Examine the UID of the resulting event-raiser, in the response data. If the UID matches our desired sensor ("MyExampleSensor"):
    c. Send a request to the "IntelliLED Control" object: **Service 0x4C, Class 0x64, Instance 1**. The request data should contain the desired property we wish to affect UID of the desired light ("MyExampleLight"), followed by an 8-bit unsigned integer, setting the light level to max (255). Use the "EncodeRequest" subroutine to pack the parameters into the request structure.

Using This Example

Using Studio 5000 designer, Import the provided file: "Example01_MotionSensed.L5K."
Before downloading the example logic to the PLC, ensure the device IP Addresses are updated, to point at the correct target devices. Also ensure that the 'target' sensor and light UIDs are updated, to reflect valid UIDs of actual devices on your test setup. These UIDs can be assigned via the PLC Tags: "**SensorUID**" and "**LightUID**."
With the logic running on the PLC, you will need to manually set the light level of the target light to 0%. Once the light has been manually turned off, perform some motion in front of the target motion sensor. The PLC should automatically turn the target light on, at 100% strength.

Example 02: Determine Which 'Group' Consumes the Most Power

In this example, an EtherNet/IP-compatible device wants to enumerate through the list of all "Groups" defined by the IntelliLED Server, and identify the group consuming the most power.
This example is implemented with the following logic:

1. Determine the number of groups defined by the Lighting System:
    a. Using the "EncodeRequest" subroutine, encode a request for **Property 0x0007 ("Group list size")**, from the reserved Server UID **"Lighting-System."**
    b. Query the "IntelliLED Control" object**: Service 0x4B, Class 0x64, Instance 1**, using the request data encoded in the previous step.
    c. Decode the response. The returned value represents the total number of Groups contained in the Lighting System
2. Find the Group with the highest power consumption.
    a. For each index [0 – "Group list size"], determined in the previous step:
        i. Retrieve the UID of the 'current group':
            1. Encode a request for **property 0x0008 ("Group list: get element"),** from the reserved server UID **"Lighting-System."** In the 'parameter' field, include the desired, current **index**.
            2. Query the "IntelliLED Control" object: **Service 0x4B, Class 0x64, Instance 1**, using the request block created in the previous step.
            3. Decode the response. The returned value represents the UID of the Group located at the current index.
        ii. Using the UID of the current group, retrieve the current power consumption:
            1. Encode a request for property **0x0702 ("Group power consumed"),** from **the UID retrieved in the previous step**.
            2. Query the "IntelliLED Control" object: **Service 0x4B, Class 0x64, Instance 1**, using the request data from the previous step.
            3. The returned value represents the current power consumption of the current group.
        iii. If the power consumed is greater than the currently-found "Max" value, then save the UID and Power Consumption as the new "Max" value.

Using This Example

Using Studio 5000 designer, Import the provided file:
"Example02_DetectGroupPowerConsumption.L5K."
Before downloading the example logic to the PLC, ensure the device IP Addresses are updated, to point at the correct target devices.
With the logic running on the PLC, you will need to manually 'toggle' the PLC's control bit, labeled "**StartRequest**." Each time this bit is turned 'on,' the PLC will run through the list of groups, determining which group consumes the most power and storing the result in the tag: "**MAX_GROUP_UID**."
Using the IntelliLED WebUI, fiddle with the various light levels of your groups. Each time you change a group's light level, trigger the "StartRequest" bit on the PLC to update the determined 'highest consumer.'

Appendix A: Enumerations

Device Sub-Types

Value	Name
0x01	Light
0x02	Basic Light
0x03	Sensor
0x04	Gateway

Sensor Sub-Types

Value	Name
0x01	WOCC Sensor
0x02	WDLH Sensor

Group WDLH Modes

Value	Name
0x01	Avg. Mode
0x02	Min Mode
0x03	Max Mode

Alert Types

Value	Name
0x01	Advisory
0x02	Warning
0x03	Error

Alert States

Value	Name
0x01	New
0x02	Acknowledged
0x03	Delete