uCIFI Schedule Framework

Approved Version: 1.0 - 2025-05-30

Open Mobile Alliance

OMA-TS-uCIFI-Schedule-Framework-V1_0-20250530-A

main: 19 Nov 2025 09:01:00 *rev: 73db574*

Table of Contents

1. Scope
2. References
    2.1. Normative References
3. Terminology and Conventions
    3.1. Conventions
    3.2. Definitions
    3.3. Abbreviations
4. Introduction
    4.1. Version 1.0
5. Objective of the uCIFI Schedule Framework
6. Target vertical smart city businesses
7. General requirements
8. Use cases to be supported by the uCIFI Schedule Framework
    8.1. Device to perform action on a local actuator at a specific time
    8.2. Device to perform action on a local actuator upon local sensor change
    8.3. Device(s) to send notification(s) and perform action(s) on remote actuators
    8.4. Device to control dimming and color temperature on local actuators based on local and remote sensors
9. Key Concepts
    9.1. Reference Time
    9.2. Day and Night Calendars
    9.3. Compound Operations
    9.4. Multiple Sensor Inputs
    9.5. Off when inactive
10. Syntax Rules
11. Data Model
    11.1. Program Function
    11.2. Calendar
    11.3. Program Scheduler
    11.4. General View
    11.5. Detailed Description
        11.5.1. 3457: Program Manager
        11.5.2. 3452: Program Scheduler
        11.5.3. 3453: Calendar
        11.5.4. 3454: Program Function
    11.6. 3455: Input Filter
    11.7. 3456: Output Filter
    11.8. 3458:Program Monitor
12. Examples
    12.1. Complex Lighting Use Case – Compact Form
13. Simulation exercise
    13.1. Complex Lighting Use Case – Constrained Device Implementation
    13.2. Simulation exercise
14. Join the uCIFI Alliance
Appendix A. Change History (Informative)
    A.1 Approved Version History

Table of Figures

Figure: 9.1.-1 timestamps must be referred to UTC
Figure: 9.1.-2 LwM2M provides UTC offset within Device object 3
Figure: 9.1.-3 DST change mechanism in Device Extension object 3410
Figure: 9.2.-1 adjusting the graphical representation
Figure: 9.2.-2 Day representation (Midnight-to-Midnight)
Figure: 9.4.-1 multiple expressions
Figure: 11.1.-1 relationship between the argument value and the resource
Figure: 11.2.-1 Define specific calendar rules
Figure: 11.3.-1 Calendar rule and a specific program function
Figure: 11.4.-1 Visual representation of the main architecture
Figure: 12.1.-1 Associated to a Smart Lighting application
Figure: 12.1.-2 The corresponding dimming values
Figure: 12.1.-3 The sensor input
Figure: 12.1.-4 The remaining Program Functions
Figure: 12.1.-5 Attach an Input Filter
Figure: 12.1.-6 Input Sources of the Program Scheduler
Figure: 13.-1 Program Scheduler instances
Figure: 13.1.-1 Constrained Device Implementation
Figure: 13.1.-2 Constrained Device Implementation - Associated with the time-related actions
Figure: 13.1.-3 Constrained Device Implementation - 2 Associated to the dynamic sensor actions
Figure: 13.1.-4 Constrained Device Implementation - 1. three Program Functions
Figure: 13.1.-5 Constrained Device Implementation - 2. three Program Functions
Figure: 13.1.-6 Constrained Device Implementation - Input Filters
Figure: 13.1.-7 Compound Operation
Figure: 13.1.-8 1-Pedestrian Program Functions
Figure: 13.1.-9 2-Pedestrian Program Functions
Figure: 13.1.-10 3-Pedestrian Program Functions
Figure: 13.1.-11 4-Pedestrian Program Functions
Figure: 13.2.-1 Simulation exercise

Table of Tables

Table: 2.1.-1 Normative References
Table: 3.2.-1 Definitions
Table: 3.3.-1 Abbreviations
Table: 4.1.-1
Table: 11.-1
Table: 11.5.1.-1 LwM2M Object: Program Manager object definition
Table: 11.5.1.-2 LwM2M Object: Program Manager Resource definitions
Table: 11.5.2.-1 LwM2M Object: Program Scheduler object definition
Table: 11.5.2.-2 LwM2M Object: Program Scheduler Resource definitions
Table: 11.5.3.-1 LwM2M Object: Calendar Rule object definition
Table: 11.5.3.-2 LwM2M Object: Calendar Rule Resource definitions
Table: 11.5.4.-1 LwM2M Object: Program Function object definition
Table: 11.5.4.-2 LwM2M Object: Program Function Resource definitions
Table: 11.6.-1 LwM2M Object: Input Filter object definition
Table: 11.6.-2 LwM2M Object: Input Filter Resource definitions
Table: 11.7.-1 LwM2M Object: Output Filter object definition
Table: 11.7.-2 LwM2M Object: Output Filter Resource definitions
Table: 11.8.-1 LwM2M Object: Program Monitor object definition
Table: 11.8.-2 LwM2M Object: Program Monitor Resource definitions
Table: A.1-1 Approved Version History

1. Scope

This document describes the uCIFI Schedule Framework and can be considered a comprehensive guide to the published data model. Its purpose is to provide OMA members with a more in-depth explanation of the details related to such functionality, show real examples, and simplify the following integration steps.

2. References

2.1. Normative References

[LwM2M-TS_1.0]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification, Version 1.0.2", February 2018
[LwM2M-TS_1.1]	Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification, Version 1.1.1", June 2019
[DMREPPRO]	"OMA Device Management Representation Protocol, Version 1.3", Open Mobile Alliance™, OMA-TS-DM_RepPro-V1_3, [URL:http://www.openmobilealliance.org/](http://www.openmobilealliance.org/)
[OMADICT]	"Dictionary for OMA Specifications", Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, [URL:http://www.openmobilealliance.org/](http://www.openmobilealliance.org/)
[OMNA]	"OMNA Lightweight M2M (LwM2M) Object & Resource Registry", [*URL:http://www.openmobilealliance.org/*] (http://www.openmobilealliance.org/)
[CoAP]	Z. Shelby, K. Hartke, C. Bormann, B. Frank, "The Constrained Application Protocol (CoAP)", June 2014, [URL:http://www.ietf.org/rfc/rfc7252.txt](http://www.ietf.org/rfc/rfc7252.txt)
[FLOAT]	ANSI/IEEE Std 754-2019, “IEEE Standard for Floating-Point Arithmetic”, IEEE Microprocessor Standards Committee, July 2019
[SENML]	C. Jennings, Z. Shelby, J. Arkko, A. Keranen, C. Bormann, "Sensor Measurement Lists (SenML)", July 2018, [URL:https://tools.ietf.org/html/rfc8428.txt](https://tools.ietf.org/html/rfc8428.txt)
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[RFC3986]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", January 2005[URL:http://www.ietf.org/rfc/rfc3986.txt](http://www.ietf.org/rfc/rfc3986.txt)
[RFC6690]	Shelby, Z. "Constrained RESTful Environments (CoRE) Link Format", August 2012, [URL:http://www.ietf.org/rfc/rfc6690.txt] (http://www.ietf.org/rfc/rfc6690.txt)
[RFC7049]	AC. Bormann, P. Hoffman, "Concise Binary Object Representation (CBOR)", October 2013, [URL:https://tools.ietf.org/rfc/rfc7049.txt](https://tools.ietf.org/rfc/rfc7049.txt)

Table: 2.1.-1 Normative References

3. Terminology and Conventions

As of 7 January, 2025, the uCIFI Specifications were transferred to the Open Mobile Alliance where the uCIFI Specifications continue to be updated. The terms “uCIFI”, “Open Mobile Alliance”, and “OMA” may be used interchangeably in the Specifications. uCIFI is a registered mark of the Open Mobile Alliance.

3.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except "Scope" and "Introduction", are normative, unless they are explicitly indicated to be informative.

3.2. Definitions

[Object	An Object is a collection of logically related resources.
[Object Instance	An Object Instance is one occurrence of an Object.
[Resource	A Resource is an atomic unit of information.
[Resource Instance	A Resource Instance is one occurrence of a Resource.
[LwM2M Client	Component running on a device implementing the LwM2M protocol for interacting with the LwM2M Server and the LwM2M Bootstrap-Server. LwM2M Clients are often implemented in IoT devices and gateways.
[LwM2M Server	Component implementing the server-side functionality of the LwM2M protocol for interacting with an LwM2M Client. Typically, the LwM2M Server software is running on a non-IoT device, such as an on-premise server or in a cloud-based infrastructure.
[LwM2M Bootstrap-Server	The LwM2M Bootstrap-Server is a server responsible for provisioning essential information, including credentials, into the LwM2M Client to enable the LwM2M Client to perform the "Register" operation with one or more LwM2M Servers. Very often, the LwM2M Bootstrap-Server is the first LwM2M entity an LwM2M Client interacts with. The Bootstrap Interface is the only interface used between the LwM2M Client and the LwM2M Bootstrap-Server.
[LwM2M Bootstrap-Server Account	LwM2M Security Object Instance with Bootstrap-Server Resource true
[LwM2M Server Account	LwM2M Security Object Instance with Bootstrap-Server Resource false and associated LwM2M Server Object Instance
[LwM2M Registration Session	An LwM2M Registration Session is an association between an LwM2M Client and an LwM2M Server. It starts with the successful LwM2M register operation and ends when the registration lifetime expires, with a successful LwM2M de-register operation or when an error is detected.
[LwM2M Bootstrap Session	An LwM2M Bootstrap Session is an association between an LwM2M Client and an LwM2M Bootstrap-Server. It starts with the successful Bootstrap-Request or Bootstrap-Pack-Request operation and ends respectively with the Bootstrap-Finish or Bootstrap-Pack operation or when an error is detected.

[Transport Session	A Transport Session is an application-independent association between the LwM2M Client and LwM2M Server or LwM2M Bootstrap-Server to enable application layer communications.

Table: 3.2.-1 Definitions

Kindly consult [OMADICT] for more definitions used in this document.

3.3. Abbreviations

Table: 3.3.-1 Abbreviations

AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation #1
CA	Certification Authority
CBOR	CBOR Concise Binary Object Representation
CMS	Central Management System
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments
COSE	CBOR Object Signing and Encryption
D2D	Device To Device
DTLS	Datagram Transport Layer Security
GW	Gateway
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
&IAN	Internet Assigned Numbers Authority
ID	Identity/Identifier
IE	Information(al) Element
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
IPSO	Internet Protocol for Smart Objects
JSON	JavaScript Object Notation
kB	Kilobyte; one kilobyte is 1000 bytes

KiB	Kibibyte; one kibibyte is 1024 bytes
LoRaWAN	LOng RAnge Wide Area Network
LQI	Link Quality Indicator
LSB	LSB Least Significant Bit
LTE	Long Term Evolution
LwM2M	Lightweight M2M
M2M	Machine to Machine
MQTT	Message Queuing Telemetry Transport
MSB	Most Significant Bit
NB-IoT	NarrowBand Internet of Things
ODF	Object Directory File
OID	Object ID
OMA	Open Mobile Alliance
OMNA	Open Mobile Naming Authority
OSCORE	Object Security for Constrained RESTful Environments
PDN	Packet Data Network
PDU	Protocol Data Unit Data Network
PKI	Public Key Infrastructure
PKCS	Public Key Cryptography Standard
PSK	Pre-Shared Key
RSSI	Received Signal Strength Indicator
RTT	Round Trip Time
RX	Receive
SenML	Sensor Measurement Lists
SHA	Secure Hask Algorithm
TCP	Transmission Control Protocol
TLS	Transmission Layer Security
TLV	Type-Length-Value
TX	Transmit
TZ	Timezone
TLS	Transport Layer Security
UDP	User Datagram Protocol

URI	Uniform Resource Identifier
URN	Uniform Resource Name
UTC	Universal Time Coordinated
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

4. Introduction

4.1. Version 1.0

Documentation aligned to the following LwM2M object versions:

LwM2M Object	Object Number	Object Version
Program Scheduler	3452	1.0
Calendar Rule	3453	1.0
Program Function	3454	1.0
Input Filter	3455	1.0
Output Filter	3456	1.0
Program Manager	3457	1.0
Program Monitor	3458	1.0

Table: 4.1.-1

Objective of the uCIFI Alliance

The objective of the uCIFI Alliance is to unlock the Smart City and Smart Utility markets by providing interchangeability, thus interoperability, between IoT devices from various vendors connected to standardized LPWAN networks such as LoRaWAN, NB-IoT, Wi-SUN/6LowPAN, WiFi.

This mission is achieved by defining a unified data model for constrained devices that allows cities and utilities to deploy IoT devices from any vendor on these LPWAN networks without the need to develop, deploy, run and manage connectors/decoders/translators to integrate proprietary data payloads and standardize data meanings into their IoT platforms, their central management software and their other smart city applications.

Today, outdoor lighting controllers, air quality sensors or smart meters from two different vendors communicating on a LoRaWAN, NB-IoT or Wi-SUN mesh networks have their own proprietary data payload, making it difficult for integrators to provide an interoperable solution without specific integration services and the impossibility to replace a device from supplier "A" by an equivalent device from supplier "B" without additional integration effort.

With the uCIFI device data model, IoT devices will use the same data dictionary. They will provide data with the same agreed-upon payload format and attributes (i.e., the words in a language) so that integrators can provide interoperable smart city and smart utility solutions to their customers. Thanks to the uCIFI device data model, IoT devices can be replaced by other functionally equivalent devices from a competitor, so that end customers are not dependent on a single supplier to deploy their IoT solutions at scale.

The uCIFI Alliance is funded by industrial companies and supported by end-customers (Cities and Utilities). The uCIFI Alliance designs and promotes efficient and open-source solutions to prevent end-customers from being locked into proprietary systems. The uCIFI Alliance is committed to providing easy-to-implement solutions based on existing, proven and widely implemented technologies such as LwM2M from the Open Mobile Alliance.

Specifications and use cases to be addressed

5. Objective of the uCIFI Schedule Framework

The Schedule Framework devised by uCIFI aims to provide Smart City devices with configuration instructions that define control actions to be taken by a device in response to time-based and/or sensor events.

Once configured under the proposed uCIFI Control Program framework, a field device can trigger actions autonomously based on predefined events without relying on external commands from local Gateways or centralized control servers in the datacentre.

6. Target vertical smart city businesses

uCIFI control Programs can be used by Smart City operators to address a wide variety of use cases, amongst which:

1. Control Light Dimming level based on time of day, movement, weather conditions, etc..
2. Control of water pressure in park irrigation based on time of day, movement, weather conditions, etc..
3. Control of air humidifiers based on time of day, movement, temperature, etc..
4. Gate access control based on time of day, beacon detection, etc…

The framework is developed so that it can be applied both to high-performing edge devices with little communication restriction (like edge computers or gateways) and embedded devices operating under constrained communication scenarios, where it is imperative to reduce the size of the communications payload and make the procedure resilient to packet drop.

7. General requirements

To be able to cater to the stringent requirements of constraint devices uCIFI Control Programs are designed around the following principles:

1. The uCIFI Program Scheduler was devised as a modular framework of configuration objects and resources, where the content and relationships between the object instances define a program schedule of all devices being managed within the device
2. Should cover both lighting and a generic smart city scenario (water sprinkler, city information panel)
3. The framework is meant to be fitted both on constrained devices and more capable edge devices
4. Data model should be in a human-readable format to be easily understandable
5. Data model should be compact to reduce network bandwidth utilization during commissioning
6. Data model should be able to scale well also for complex configurations (e.g. with many actions)
7. Easy mapping from TALQ format shall be taken into consideration

8. Use cases to be supported by the uCIFI Schedule Framework

Several use cases have been defined within the uCIFI alliance:

8.1. Device to perform action on a local actuator at a specific time

Example 1:

Device A switches lamp actuator 100% at Sunset + 5 minutes

Then dims it to 50% at 9 PM (if 9 PM is after Sunset + 5 minutes)

Switch OFF at Sunrise −5 minutes

8.2. Device to perform action on a local actuator upon local sensor change

Example 1:

An illuminance sensor on device A detects that illuminance is above the threshold

Light is turned OFF on the lamp actuator from the same device A

Example 2:

A smoke sensor on device A detects a potential fire

Water valve is turned ON from the same device A

8.3. Device(s) to send notification(s) and perform action(s) on remote actuators

Example 1:

The illuminance sensor on device A detects that the illuminance is below a threshold (sunset)

A notification is sent on the mesh network

Consequently, devices B, C and D turn the light ON at 100%

The very same devices B, C and D dim the light to 50% at 9 PM (if light has been previously turned on)

The illuminance sensor on device A detects that illuminance is above a threshold (sunrise)

A notification is sent on the mesh network

Consequently, devices B, C and D turn the light OFF

Example 2:

A smoke sensor on device A detects a potential fire

A notification is sent on the mesh network

Remote device B turns the water valve ON

Example 3:

Parking occupancy sensors A, B and C detect the presence/absence of a vehicle on the place

For each change, a notification is sent on the mesh network by the devices

A display panel receives all these notifications and computes them to display the total number of free places

Example 4:

Devices B, C and D turn ON the light at Sunset + 5 minutes

Devices B, C and D dim the light to 50% at 9 PM (if 9 PM is after Sunset + 5 minutes)

A presence sensor on device A detects a presence

A notification is sent on the mesh network

If the notification is received between 10 PM and 5 AM, devices B, C and D temporarily increase the light level to 90% (with 10 seconds fading)

After 3 minutes, the light level is restored to the previous dimming level (50% in this case)

Devices B, C and D turn OFF the light at Sunrise - 5 minutes

8.4. Device to control dimming and color temperature on local actuators based on local and remote sensors

This use case should provide a comprehensive example of a relatively complex scenario that combines many aspects covered explicitly by other simpler use cases.

The device:

- One Light Point Controller (physical device) controlling two separate LED drivers with possible color temperature changing, one for the roadside and one for the pedestrian side
- Two sensors related to the pedestrian side:
  - local (near)
  - remote (far)[^1]
- Two "uCIFI Outdoor Lamp Controller" objects
  - 3416/0/1 = command for pedestrian side
  - 3420/0/39 = color change for pedestrian side
  - 3416/1/1 = command for roadside
  - 3416/1/39 = color change for roadside
  - 3302/0/5500 = digital input state on sensor (local)

- 3302/1/5500 = digital input state on sensor (virtual, remote)

Expected behavior:

- Every day / every year:
  - Roadside:
    - Switch ON 100% at sunset + 5 minutes at 4500K
    - Dim down to 50% from sunset + 2 hours to 5AM and change color to 3500K
    - Switch OFF 100% at sunrise – 15 minutes
  - Pedestrian side:
    - Switch ON 100% at sunset + 5 minutes at 4500K
    - From 10PM to 5AM
      - Dim down to 30% and change color to 2900K
      - When sensor #1 (local) detects someone, increase to 100% for 1 minute
      - When sensor #2 (remote) detects someone, increase to 50% for 1 minute
    - Switch OFF 100% at sunrise – 15 minutes
- Every Saturday-to-Sunday night:
  - Roadside:
    - Same as usual, but dimming to 50% is done at midnight instead of sunset + 2 hours
  - Pedestrian side:
    - Same as usual, but dimming to 30% is done at midnight instead of 10 PM

Disclaimer

The current documentation only covers to the following functionalities:

- Local schedule management based on time and sensors (remote sensors to be managed similarly to local ones)
- No Device to Device (D2D) functionality has been finalized yet

9. Key Concepts

9.1. Reference Time

LwM2M requires that timestamps must be referred to UTC, as stated in the Data Types appendix[^2]

Time	Unix Time. A signed integer representing the number of seconds since Jan 1st, 1970 in the UTC time zone.

Figure: 9.1.-1 timestamps must be referred to UTC

To reuse the same schedules among different parts of the world, without having to convert them before commissioning, all date and time references used within the Schedule framework must be intended in **local time**

Since the Schedule Framework is primarily working with string representations or time offsets, this would still be compliant with the standard.

LwM2M provides UTC offset within Device object 3 to allow converting from one to another in case of need.

13	Current Time	RW	Single	Optional	Time			Current UNIX time of the LwM2M Client. The LwM2M Client should be responsible to increase this time value as every second elapses. The LwM2M Server is able to write this Resource to make the LwM2M Client synchronized with the LwM2M Server.
14	UTC Offset	RW	Single	Optional	String			Indicates the UTC offset currently in effect for this LwM2M Device. UTC+X [ISO 8601].

Figure: 9.1.-2 LwM2M provides UTC offset within Device object 3

Additionally, uCIFI already includes a DST change mechanism in the Device Extension object 3410, allowing for automatic DST adjustment across years.

11	Name DST start	Operations RW	Type Time	Unit s
	Description The precise date and UTC time during the year when Daylight Saving Time begins, represented as Unix Time.		Instances Multiple	Mandatory Optional
12	Name DST end	Operations RW	Type Time	Unit s
	Description The precise date and UTC time during the year when Daylight Saving Time ends, represented as Unix Time.		Instances Multiple	Mandatory Optional
13	Name DST offset	Operations RW	Type String	Unit s
	Description The amount by which the UTC offset changes when Daylight Saving Time is in effect, represented as ISO 8601 time offset format. A DST offset of one hour is represented as " 01:00".		Instances Multiple	Mandatory Optional

Figure: 9.1.-3 DST change mechanism in Device Extension object 3410

9.2. Day and Night Calendars

Let's assume there are two different **Program Functions**, one expected to run on Friday and the other on Saturday. The common understanding would probably accept the swap happens at midnight between Friday and Saturday.

Still, there are use cases (mostly related to Smart Lighting) for which program functions must be executed overnight and swapped at noon (instead of midnight).

The uCIFI Schedule Framework supports both types of **Programs Functions**, by adding a special attribute to the corresponding **Calendar**.

In addition, from a **Program Function** point of view, there is little to no impact, and in the device 00:00 would always be assumed to be midnight.

Instead, from a user perspective, the frontend application could easily translate from one syntax to the other by simply adjusting the graphical representation[ ]{#_Toc70508038 .anchor}

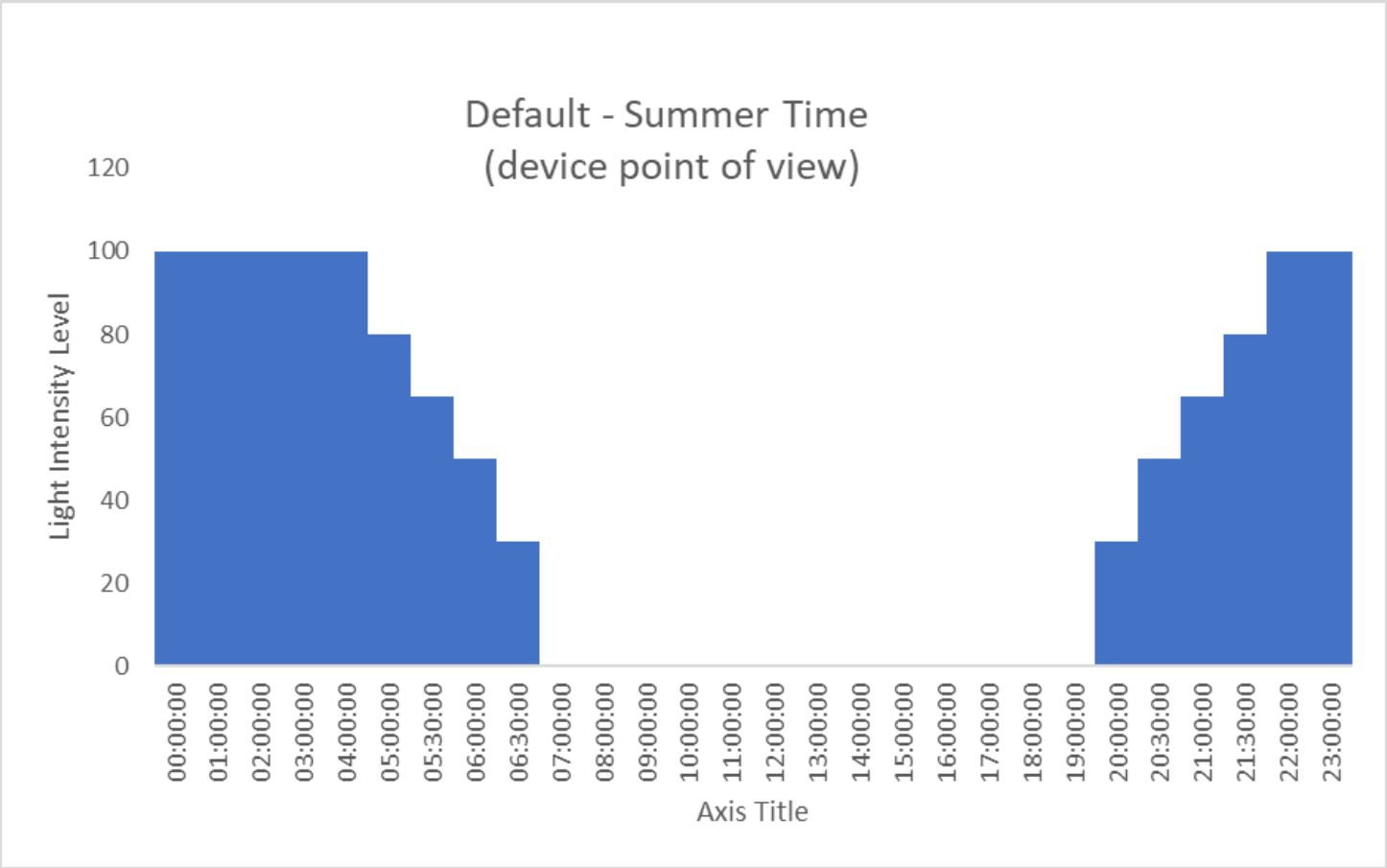


Figure: 9.2.-1 adjusting the graphical representation

Day representation (Midnight-to-Midnight)

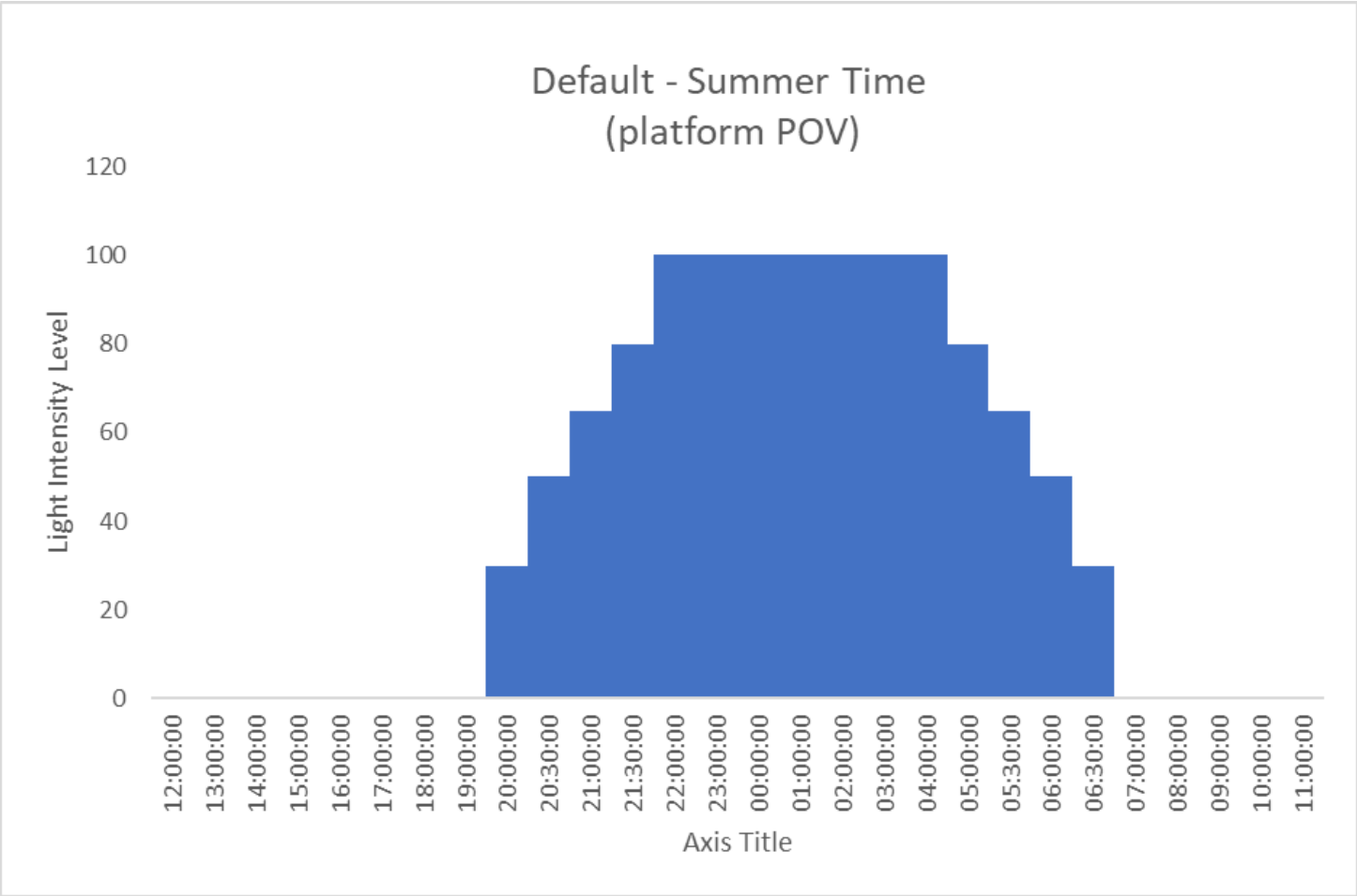


Figure: 9.2.-2 Day representation (Midnight-to-Midnight)

Night representation (Noon-to-Noon)

The device may also easily check if a specific **Calendar** rule matches by comparing the current time T in case of day calendars or the time T - 12 hours[^3] in case of night calendars

9.3. Compound Operations

Compound operation is a powerful functionality that can provide advanced functionalities and simplify some specific operations.

The concept is that a single function does not provide the overall control value, but by combining more functions in a sort of *daisy chain* operation.

If the compound operation of a specific entry is false (or if the compound operation is not supported at all), all the targets referenced shall not be subsequently altered by lower priority functions.

There are a few examples that could clarify how such a feature could be used:

Example 1:

It is possible to use the compound operation to define separate and smaller Program Functions that, combined, define

the overall behavior of the actuator based on time and the status of some sensors:

Rule 1: time-based **Program Function** acting on actuator #1, compound = true

Rule 2: sensor-based **Program Function** acting on the same actuator #1 and checking sensor A, compound = true

Rule 3: sensor-based **Program Function** acting on the same actuator #1 and checking sensor B, compound = false

Example 2:

Let's now consider that a network has already been deployed, each device with its own independent Schedule.

The user would now like to override the current behaviour of all the devices for a specific time and day (e.g., turn off the light from 21:00 to 23:30 on Sunday).

Without compound operation, it would be required to:

1. Retrieve all the various programs in effect on Sunday evening for each lamp of each device
2. Modify the various programs
3. Recommission them together with a specific calendar that would be assigned the highest priority

Instead, using the compound operation, it would be sufficient to:

1. Create a single program function that only covers the Sunday evening
2. Commission it together with a specific calendar that would be assigned the highest priority, and a compound operation enabled

In this way, executing the original schedule outside of that specific time will still be possible.

> Note
>
> Please also note that if a temporary override is needed for an Outdoor Lamp Controller, the Manual Override functionality provided by object 3416 would achieve a similar behaviour.

9.4. Multiple Sensor Inputs

A vital aspect relates to how the various sensors shall be evaluated to trigger actions.

Given that sensors can be of many different kinds, the current model is based on the following assumptions:

1. Sensors are ultimately represented via LwM2M Objects
2. For each object, a specific resource will be considered as the reference value to be checked
3. A separate input filter (described later) can be provided to better manage the polling mechanism or any special operation that could be required on the input (value hold, de-bouncing, etc)
4. Sensor evaluation is performed utilizing a **positional identifier**
5. Comparison is performed **by range** or **by value**
6. Time can also be considered

Now let's consider the following scenario, where we have

- Time of day (t)
- Soil moisture sensors (w1 and w2)
- Motion/presence sensor (m)
- Precipitation sensor (p)

And we would like to express the following piecewise function that controls the pressure of the irrigation valve depending on:

- time: no irrigation to be done in the late morning and early afternoon
- soil humidity: if the soil is wet enough, there would be no need to provide additional water
- precipitation sensor: like the soil humidity
- motion: if movement is detected in the distance, irrigation should be performed with low pressure

$$ F(t,w1,w2,p,m) = \left\{ \begin{aligned} 10\ bar,\ \ & 0 \leq t \leq 11,\ w_{2} \leq 30%,\ p \leq 0.1,\ m = 0 \ 5\ bar,\ \ & 0 \leq t \leq 11,\ w_{1} \leq 30%,\ p \leq 0.1 \ 0\ bar,\ \ & 11 < t \leq 15 \ 5\ bar,\ \ & 15 < t \leq 24,\ w_{1} \leq 30%,\ p \leq 0.1 \end{aligned} \right.
$$

Now, let's consider the following:

- Time is always the first argument
- Subsequent arguments may be any sensor, but the order, whatever it is, shall be kept in mind (how to change such ordering will be explained later)

And consider the following assumptions:

- Each expression is composed by n fields, where the first is always the time and each other field refers to the corresponding sensor (based on the positional identifier)
- An expression evaluates to true if all the contained fields evaluate to true
  - An asterisk * or an empty field shall be interpreted as "don't care" and continually evaluates to true
  - Trailing fields can be omitted and are automatically considered as "don't care"
  - When using range expressions ~ both leftmost and rightmost limits are inclusive
- Multiple expressions can be provided, with the first matching expression taking precedence and causing the subsequent expressions to be ignored

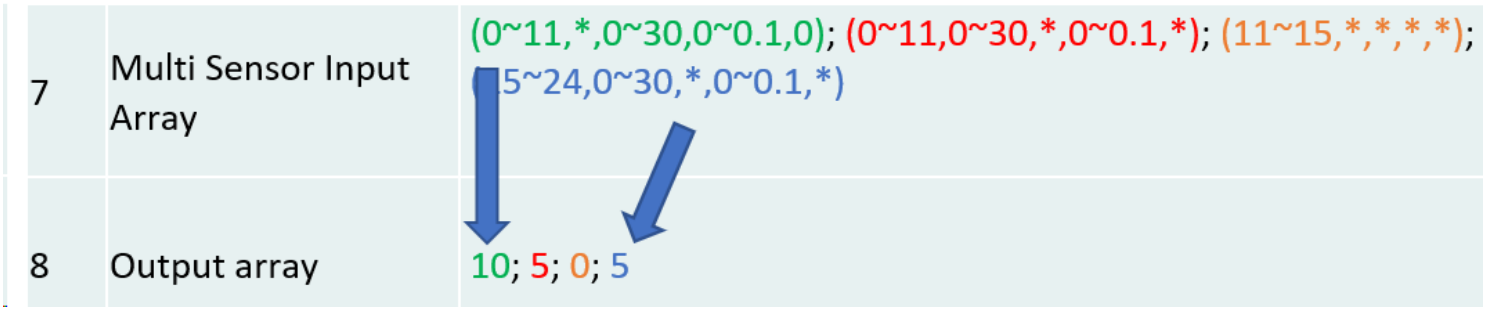


Figure: 9.4.-1 multiple expressions

9.5. Off when inactive

Run conditions are quite convenient for mixing astronomical and fixed-time commands without the risk of overlap across the year.

Consider a rule that states to switch on at 40% at 6 am and to switch off at 0% at sunrise.

If sunrise occurs at 06:30, the light will remain at 40% for half an hour before switching off, ensuring a smooth operation.

But if sunrise were to happen at 05:30, the light would stay at 0% for half an hour and then switch to 40% for the rest of the day!

Run conditions would avoid this situation by enforcing that a specific program is active, for example, from sunset to sunrise.

If sunrise happens before 6 am, the 40% rule would be skipped.

If the resource **Off when inactive** is set to true in a specific **Program Scheduler**, a default state of **Off** would be enforced when the run condition is not met, rather than applying the default state of the device being controlled.

The current schedule examples have been written using this approach.

10. Syntax Rules

The data model needs to be compact and scale well, also when supporting a high number of actions.

For this reason, the following conventions have been adopted:

1. String resources are being used instead of relying on multiple resources and/or resource instances
2. Time rules are as follows:
    - Fixed times are expressed in hh:mm format (24 hours), e.g. 04:00 for 4 am, 19:30 for 7:30 pm, where 00:00 is always the local midnight
    - Fixed time can also be specified with seconds (if the device supports this capability). In this case, we could have 04:00, 04:00:25, 04:00:25~05:00
    - Astronomical events are expressed with a letter (normally uppercase, although lowercase is also acceptable), eventually followed by a +/- digit and a number that represents the **minutes** offset:
        - **S** = sunset
        - **R** = sunrise
        - **M** = astronomical midnight

e.g. S (sunset), S+30 (30 minutes after sunset), R-10 (10 minutes before sunrise)

3. Fields are concatenated by means of separators
    - fields that refer to different conditions or outputs and have to be considered alternatively (e.g. time conditions) are separated by semicolons: 04:00**;**S+10;R-30
    - fields that refer to the same condition (e.g. run condition, multi-sensor input array) are instead separated by commas: time,sensor_1,sensor_2,sensor_3
    - If multiple conditions have to be concatenated (e.g. in multi-sensor input array) an additional set of brackets and semicolon separators is used as follows **(time,sensor_1,sensor_2,sensor_3); (time,sensor_1,sensor_2,sensor_3)**
    - If a field is a don't care input, an asterisk or an empty character can be used
    - If trailing fields are omitted, they must be assumed to be "don't care"
    - No assumption has to be made about the ordering of time rules. It could be acceptable to write S+30;R-20;S+10

11. Data Model

The current Schedule Framework is composed of the following LWM2M objects:

Name	Mandatory	Purpose
Program Manager	Mandatory	Single Instance describing the capabilities of the device
Program Scheduler	Mandatory	Summary of the mapping between the various functions, eventual inputs and the target actuators, spanning multiple days or years
Program Function	Mandatory	Described the behavior that controls the associated target based on time or sensor status
Program Monitor	Optional	Exports some useful information related to program execution
Calendar	Optional	Allows to specify when to execute a particular function
Input Filter	Optional	Used to fine tune the sensor evaluation
Output Filter	Optional	Used to fine tune the output commands

Table: 11.-1

11.1. Program Function

The first building block worth considering is the Program Function, which contains the main logic driving the associated actuators.

The concept behind it is to define the behavior of the target resources based on time and/or dynamic input from sensors.

An example could be that at noon, the value should be set to 50, or at sunset plus 30 minutes, it should be set to 24, or again, when the first sensor is active, it should be set to 100.

The object is agnostic about what it will be required to control (if targeting a lamp driver, this value would correspond to a dimming value in percentage; if targeting an irrigation valve, instead, it could represent a pressure value, just to make an example—clearly, there would be some relationship between the argument value and the resource it is meant to control).

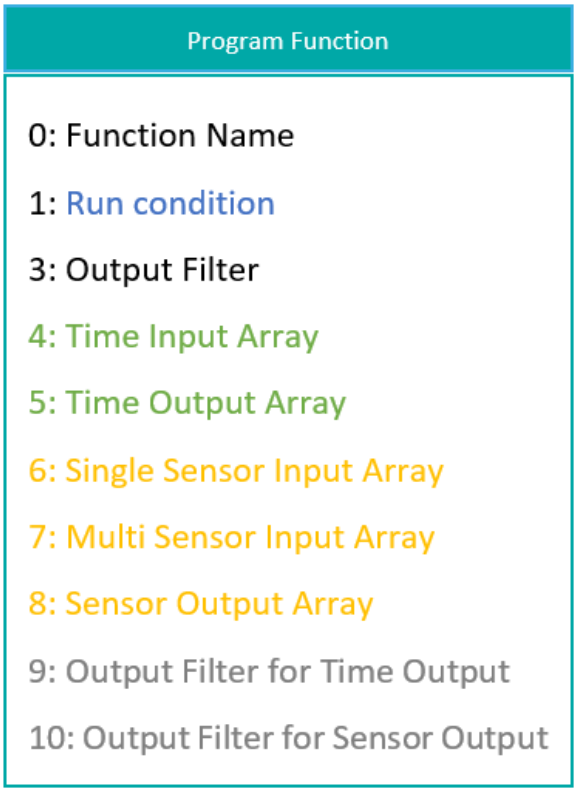


Figure: 11.1.-1 relationship between the argument value and the resource

Time-related behavior is represented by two different resources:

4. Time Input Array
5. Time Output Array

Whilst the sensor-related behavior is represented by three different resources:

6. Single Sensor Input Array
7. Multi-Sensor Output Array
8. Sensor Output Array

Where the single sensor and the multi-sensor are mutually exclusive

The idea is to establish a 1:1 relationship between the input values and the corresponding output values, identifying a status in the input and selecting the corresponding status for the output.

A single Program may contain logic to drive multiple actuators in output. We want to manage both the dimming level and the colour temperature. In this case, each target will be associated a separate resource instance index, so that the first instance of the Time Input Array, Time Output Array, Multi Sensor Input Array, etc will refer to the dimming level, whilst the second instance of the Time Input Array, Time Output Array, Multi Sensor Input Array, etc will refer to the color temperature.

If we would like to translate the example mentioned above using the proper syntax, we would end up with something

like this:

4.0: 12:00, S+30

5.0: 50, 24

6.0: 1

8.0: 100

Or, using the multi-sensor approach

4.0: 12:00, S+30

5.0: 50, 24

7.0: (*,1)

8.0: 100

The function may be limited to running only during a specific time frame. This period in which the function is active can be expressed utilizing one or more Run Conditions.

Such run conditions follow the same syntax conventions used by the multisensor input, with the only exception that each instance would be used to map a separate condition.

The various conditions would be evaluated in OR, and if at least one condition is met, the overall run condition would be satisfied.

This approach would allow us to map the following conditions simply:

- One or more absolute time periods
  - 04:00~17:30 from 4 am to half past 5 pm)
- One or more astronomical periods
  - S-10~R+10 from sunset minus 10 minutes to sunrise plus 10 minutes
- Sensor-driven active period
  - *,1 anytime, when the sensor is active
- A combination of time and sensor-based periods
  - S~R,1 from sunset to sunrise AND the sensor is active
  - S~R *,1 from sunset to sunrise OR when the sensor is active (This requires two instances)

Last, it would be worth mentioning that a descriptive label can be associated with the program function to help easily recognize the associated function and simplify its reuse.

The input filter and output filter will be discussed in a later section.

11.2. Calendar

To schedule the execution of specific functions throughout the year, it is possible to define specific calendar rules that limit the execution to specific dates or weekdays.

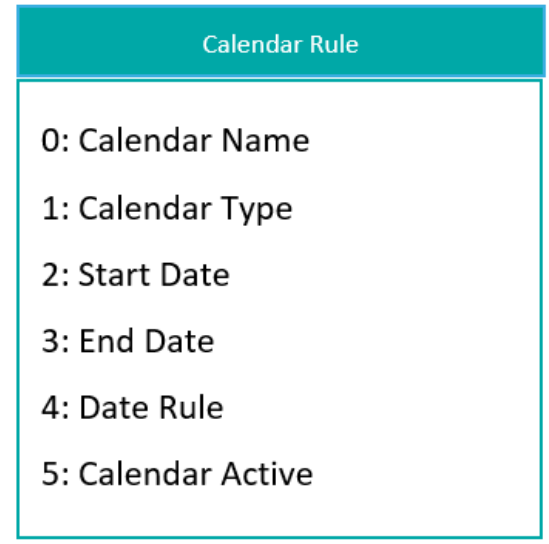


Figure: 11.2.-1 Define specific calendar rules

In this case, it is also possible to associate a specific label with each calendar rule to make them easy to recognize.

Furthermore, the calendar type can be used to differentiate between day calendars and night calendars (and, consequently, between programs).

The start and end dates enable a user to define a date range that can span a specific year or multiple years, while the date rule can be used to restrict the rule to match only specific days of the week (e.g., weekends or working days).

Additional details can be found in subsequent sections.

11.3. Program Scheduler

When discussing the Schedule Framework, the Program Scheduler can be considered the second most important object. It represents the real glue logic between functions, calendars, input sensors, and actuators.

Such an object is composed of multiple instances, each representing an association between a calendar rule and a specific program function that acts on a set of actuators.

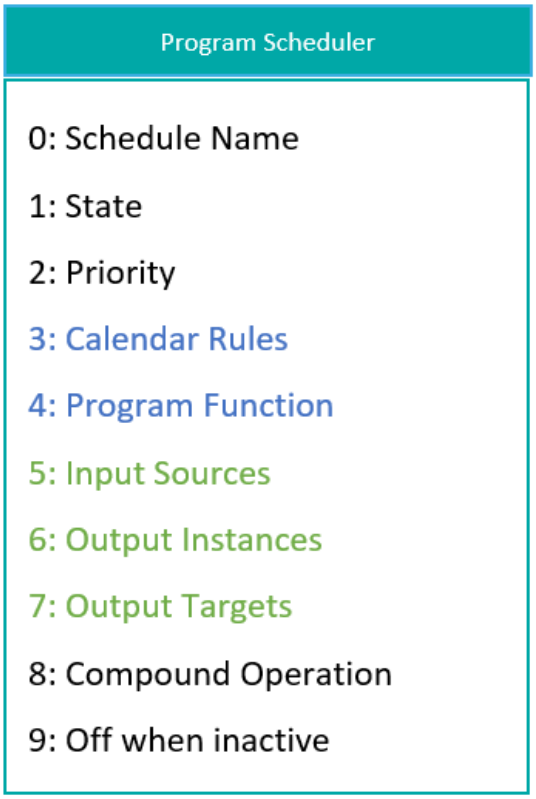


Figure: 11.3.-1 Calendar rule and a specific program function

Here again, a string description can be used to easily recognise a schedule previously configured.

Multiple calendars can be associated so that the rule is matched if at least one of them evaluates to true.

In this case, the associated program function is executed.

The list of input sources can be used to provide multiple sensors as an argument to the program function (to either the sensor input array or the multi-sensor input array)

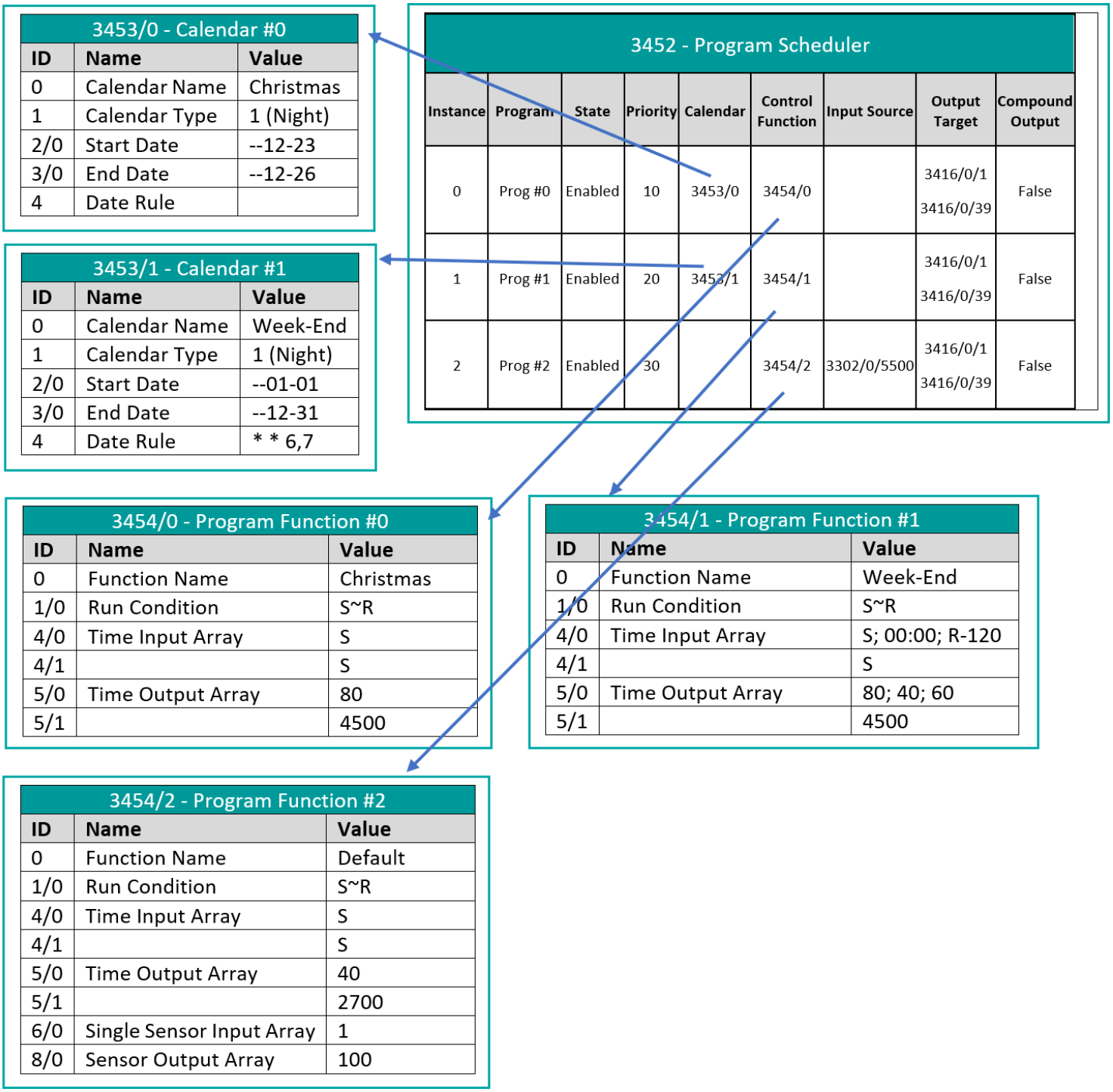
The output target instead provides the list of actuators driven by the associated program function.

Adding instances to the table is equivalent to associating new behavior with specific actuators, either in addition to or replacing the current behavior.

The remaining resources are described in more detail in a later section.

11.4. General View

Before providing an in-depth description of each specific object, it is beneficial to clarify the overall concept with a visual representation of the main architecture of the Framework, at least for a minimal subset of objects.



3453/0 - Calendar #0

ID	Name	Value
0	Calendar Name	Christmas
1	Calendar Type	1 (Night)
2/0	Start Date	--12-23
3/0	End Date	--12-26
4	Date Rule	

3452 - Program Scheduler

Instance	Program	State	Priority	Calendar	Control Function	Input Source	Output Target	Compound Output
0	Prog #0	Enabled	10	3453/0	3454/0		3416/0/1 3416/0/39	False
1	Prog #1	Enabled	20	3453/1	3454/1		3416/0/1 3416/0/39	False
2	Prog #2	Enabled	30		3454/2	3302/0/5500	3416/0/1 3416/0/39	False

3453/1 - Calendar #1

ID	Name	Value
0	Calendar Name	Week-End
1	Calendar Type	1 (Night)
2/0	Start Date	--01-01
3/0	End Date	--12-31
4	Date Rule	* * 6,7

3454/0 - Program Function #0

ID	Name	Value
0	Function Name	Christmas
1/0	Run Condition	S~R
4/0	Time Input Array	S
4/1		S
5/0	Time Output Array	80
5/1		4500

3454/1 - Program Function #1

ID	Name	Value
0	Function Name	Week-End
1/0	Run Condition	S~R
4/0	Time Input Array	S; 00:00; R-120
4/1		S
5/0	Time Output Array	80; 40; 60
5/1		4500

3454/2 - Program Function #2

ID	Name	Value
0	Function Name	Default
1/0	Run Condition	S~R
4/0	Time Input Array	S
4/1		S
5/0	Time Output Array	40
5/1		2700
6/0	Single Sensor Input Array	1
8/0	Sensor Output Array	100

Figure: 11.4.-1 Visual representation of the main architecture

In this representation, it is possible to observe that two calendars and three program functions have been instantiated.

Three different calendar rules have been created, two of which are mapped to specific calendars (Christmas time and Weekends), and the last is mapped to no calendar (meaning every other day, as a default fallback).

Each association has a specific priority, so the first matching would indeed select the program function to be executed (in this example, the instances are ordered by priority just for the sake of simplicity).

A presence sensor is additionally provided as an input argument to the last program function, for dynamic input.

The dimming value and color temperature associated with the outdoor lamp controller are instead being provided as output targets.

11.5. Detailed Description

Follows the detailed description of the various objects.

11.5.1. 3457: Program Manager

Description

The uCIFI Program Manager describes the scheduling capabilities of the device, referring to Program Schedulers, Program Functions, Calendar Rules and Filters.

Object definition

Name	Object ID	Object Version	LWM2M Version
Program Manager	3457	1.0	1.0
Object URN		Instances	Mandatory
urn:oma:lwm2m:ext:3457		Single	Mandatory

Table: 11.5.1.-1 LwM2M Object: Program Manager object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Max number of Program Schedulers	RW	Single	Mandatory	Integer			Maximum number of program schedulers supported by the device. Can be eventually adjusted in order to reserve more or less memory.
1	Current number of Program Schedulers	R	Single	Mandatory	Integer			Number of program schedulers currently allocated.
2	Max number of Program Functions	RW	Single	Mandatory	Integer			Maximum number of program functions supported by the device. Can be eventually adjusted in order to reserve more or less memory.
3	Current number of Program Functions	R	Single	Mandatory	Integer			Number of program functions currently allocated.

4	Max number of Calendars	RW	Single	Mandatory	Integer			Maximum number of calendars supported by the device. Can be eventually adjusted in order to reserve more or less memory.
5	Current number of Calendar	R	Single	Mandatory	Integer			Number of calendars currently allocated.
6	Max number of Actuators	RW	Single	Mandatory	Integer			Maximum number of target actuators supported. Can be eventually adjusted in order to reserve more or less memory.
7	Max number of Input Sources	RW	Single	Optional	Integer			Maximum number of input sources supported. Can be eventually adjusted in order to reserve more or less memory.
8	Max number of Dates in Calendar	RW	Single	Optional	Integer			Maximum number of start/end date in Calendar supported. Can be eventually adjusted in order to reserve more or less memory.
9	Max Flash Memory Allocated	RW	Single	Mandatory	Integer		B	Max flash memory (in bytes) to be allocated. Can be eventually adjusted in order to reserve more or less memory
10	Flash Memory Free	R	Single	Optional	Integer		B	Free flash memory in bytes
11	Error Conditions	R	Single	Mandatory	Integer			Bitmap Bit 0: No valid program Bit 1: Missing configuration objects Bit 2: Missing configuration resources Bit 3: Invalid inputs Bit 4: Invalid outputs Bit 5: Invalid function format Bit 6: Out of storage memory Bit 7: Out of runtime memory Bit 8..31: reserved for future use
12	Clear errors	E	Single	Optional				When executed, clears temporary error conditions. To be noted that some of the errors may require to change the content of other objects to be completely cleared.
13	Supported Capabilities	R	Single	Optional	Integer			Bitmap Bit 0: Astronomical Events Bit 1: Single sensor Inputs Bit 2: Multi Sensor Inputs Bit 3: Input Filters Bit 4: Output Filters Bit 5: Calendar Bit 6: Compound Operations Bit 7: Second granularity (for fixed time expressions) Bit 8: Astronomical Midnight Bit 9: Off when inactive
14	Max Dynamic Memory Allocated	RW	Single	Optional	Integer		B	Max RAM memory (in bytes) to be allocated. Can be eventually adjusted in order to reserve more or less memory
15	Dynamic Memory Free	R	Single	Optional	Integer		B	Free RAM memory in bytes
16	Reset schedule objects	E	Single	Optional				When executed, trigger a complete wipe of schedules already commissioned

Table: 11.5.1.-2 LwM2M Object: Program Manager Resource definitions

11.5.2. 3452: Program Scheduler

Description

Summary of the mapping between the various functions, eventual inputs and the target actuators, spanning multiple days or years.

Object definition

Name	Object ID	Object Version	LWM2M Version
Program Scheduler	3452	1.0	1.0
Object URN		Instances	Mandatory
urn:oma:lwm2m:ext:3452		Multiple	Mandatory

Table: 11.5.2.-1 LwM2M Object: Program Scheduler object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Schedule Name	RW	Single	Optional	String			Unique key identifying the Schedule
1	State	RW	Single	Mandatory	Boolean			If disabled, the instance shall be ignored
2	Priority	RW	Single	Mandatory	Integer			Integer representing the priority of the program within all program object instances. Lower number (with sign) correspond to higher priorities. Higher priority programs have pre-eminence over lower priority program schedule to be running in parallel in same day. For example: Program #11 is scheduled to run every day between Jan 1st and Dec 31st with a priority 7 Program #22 is scheduled to run every Saturday between March 1st and June 30th with priority 3 Program #22 should apply for all Saturdays between March 1st and June 30th Program #11 for all other Saturdays in the year If compound operation is supported, functions are executed in a order based on this priority field
3	Calendar Rules	RW	Multiple	Optional	Integer			Number of Calendar Rule object(s) instance containing information on which days in the week, months, year(s) is the program scheduled to run.
4	Program Function	RW	Single	Mandatory	Integer			Number of Program Function object instance containing information on how to control the output(s) based on time and sensor input(s)

5	Input Sources	RW	Multiple	Optional	String			List of sensors or input filters that can be referenced within the associated program. If more than one sensor/input filter has to be checked, multiple instances shall be used. Each one may be assigned a different input filter Resources are specified using Core Link syntax For example: Instance 0: </3302/0/5500> (presence sensor #1) Instance 1: </3302/1/5500> (presence sensor #2) Or: Instance 0: </3455/0/0/0> (input filter #1)
6	Output Instances	RW	Multiple	Mandatory	String			List of LWM2M </Object/Instance> that will be controlled by the associated program. If more than one </Object/Instance> pair shall be controlled, multiple instances shall be used. The target resource within each instance is specified in Output Targets. Instances are specified using Core Link syntax For example instance 0: </3416/0> (lamp #0) instance 1: </3416/1> (lamp #1)
7	Output Targets	RW	Multiple	Mandatory	String			List of LWM2M resources that will be controlled by the associated program. If more than one resource shall be controlled, multiple instances shall be used. Each one may be assigned a different output filter For example instance 0: 1 (dimming) instance 1: 39 (colour temperature) Each target must be prepended with all the output instances to identify the full resource path
8	Compound Operation	RW	Single	Optional	Boolean			The control Program is expected to be evaluated within a list of multiple control programs, where the evaluation occurs in an orderly fashion following the value set under its Priority resource. The compound operation resource, if set, will allow the output of the current control program acting on a given output#1 to be compounded with and output of a lower priority program acting on the same output #1. For example: 1) CP#1 priority 100, outputs 50% dimming level for light #1 with operation SET (compound flag is enabled) 2) CP#11 priority 500, outputs 10% dimming level for light #1 with operation ADD 3) Final light level on light#1 after CP list is executed is 60%
9	Off when inactive	RW	Single	Optional	Boolean			If true, allows to force an OFF behavior (for actuators that would support it) in case the run condition is not met.

Table: 11.5.2.-2 LwM2M Object: Program Scheduler Resource definitions

11.5.3. 3453: Calendar

Description

Allows to specify when to execute a particular function.

Object definition

Name	Object ID	Object Version	LWM2M Version
Calendar Rule	3453	1.0	1.0
Object URN		Instances	Mandatory
urn:oma:lwm2m:ext:3453		Multiple	Optional

Table: 11.5.3.-1 LwM2M Object: Calendar Rule object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Calendar Name	RW	Single	Optional	String			Mnemonic name of the calendar
1	Calendar Type	RW	Single	Optional	Integer			0 - Day time Gregorian calendar used to reference periods of 24h (00-24h) during the year 1 - Night time calendar, a variation of the Gregorian calendar proposed by uCiFi to reference "night" periods of 24h during a year. Measured from 12:00 of one Gregorian calendar day to 12:00 of the next day. If omitted, 0 shall be assumed
2	Start Date	RW	Multiple	Optional	String			Start date formatted as yyyy-mm-dd. To define the same start date every year: --mm-dd (e.g. --01-01 for Jan 1st every year). Start date is optional. If no start date, no end date should be mentioned, and the scheduler should apply every day of every year.
3	End Date	RW	Multiple	Optional	String			End date formatted as yyyy-mm-dd. To define the same end date every year: --mm-dd (e.g. --03-31 for March 31st every year). End date is optional. If no start date is specified, no end date should be mentioned as well, and the scheduler should apply every day of every year. If the start date is specified but the end date is not, it shall be assumed to be equal to the start date

4	Date Rule	RW	Single	Optional	String			Date string formatted as dom month dow, inspired from the CRON format without minutes and hours, that indicates the occurrence of the rule: dom = day of month (1-31) month = 1-12 dow = day of week (0-7, where 0,7=Sunday) * * * means every day between "Start date" and "End date" 1-10 * * means from the 1st to the 10th day of the month each month between "Start date" and "End date" * * 6,7 means every Saturday and Sunday, between "Start date" and "End date" The date rule is optional. If not present, it means every day between "Start date" and "End date". If present, it would be applied in AND to every instance of Start date and End date
5	Calendar Active	R	Single	Optional	Boolean			True if the calendar rule is currently active

Table: 11.5.3.-2 LwM2M Object: Calendar Rule Resource definitions

11.5.4. 3454: Program Function

Description

Describes the behavior that controls the associated target based on time or sensor status.

Object definition

Name	Object ID	Object Version	LWM2M Version
Program Function	3454	1.0	1.0
Object URN		Instances	Mandatory
urn:oma:lwm2m:ext:3454		Multiple	Mandatory

Table: 11.5.4.-1 LwM2M Object: Program Function object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Function Name	RW	Single	Optional	String			Mnemonic name of the function

1	Run Condition	RW	Multiple	Optional	String			Conditions that would cause the function to be effective (if at least one condition is met). Syntax of each condition is as follows: <time>,<sensor #1>, <sensor #2>,...,<sensor #n> hh:mm or S(unset), sun(R)ise /- minute offset Ranges are separated with a tilde character "~~" Open ranges can be written as start~~ (value >= start) or ~end (value <= end) Time should be a range, eg 18:00~~04:00 or S~~R And each sensor can be either a specific value or a range of values If omitted or not supported, function can be assumed to be effective 24/24
3	Output Filter	RW	Multiple	Optional	Integer			Number of Output Filter object instances, each one being applied to the corresponding target output
4	Time Input Array	RW	Multiple	Mandatory	String			Sequence of fixed time and/or astronomical events separated by semicolons Each resource instance is specified as a sequence that will target the corresponding output hh:mm = fixed time S+30 = sunset plus 30 minutes R-20 = sunrise minus 20 minutes
5	Time Output Array	RW	Multiple	Mandatory	String			Sequence of arguments to be provided to the target. If strings have to be provided, they must be enclosed in double quotes e.g. 10;50;24 "parking empty";"parking full" Any spacing provided outside double quotes can be safely ignored
6	Single Sensor Input Array	RW	Multiple	Optional	String			Allows to define multiple conditions based on sensor value. Only the first sensor is considered (in case multiple sensors are configured). This resource shall be used in mutual exclusion with the multi sensor input array.
7	Multi Sensor Input Array	RW	Multiple	Optional	String			Allows to define multiple conditions based on time and sensor values with a syntax similar to the one used for the Run condition Each condition shall be enclosed in brackets and separated by semicolons (<time>, <sensor #1>,<sensor #2>,...,<sensor #n>);(<time>, <sensor #1>,<sensor #2>,...,<sensor #n>);... The first condition matching would take precedence over the ones following Any spacing provided outside double quotes can be safely ignored When multiple output targets have to be managed, additional conditions will be provided in a separate resource instance

8	Sensor Output Array	RW	Multiple	Optional	String			Sequence of arguments to be provided to the target in a similar way of the time output array The sensor arrays can be used in conjunction of the time arrays and, in this case, would take precedence in case of matching This would allow to define both fixed time control and sensor based control in a single function If both single sensor input array and multi sensor input array are both specified, the multi sensor one would take precedence
9	Output Filter for Time Output	RW	Multiple	Optional	String			Sequence of arguments representing Output Filter instances to be applied to the corresponding target output e.g. 0;;1 means /3456/0;default;/3456/1 If resource or specific entry is not specified, the default output filter, if present, would be applied
10	Output Filter for Sensor Output	RW	Multiple	Optional	String			Sequence of arguments representing Output Filter instances to be applied to the corresponding target output e.g. 0;;1 means 3456/0;default;3456/1 If resource or specific entry is not specified, the default output filter, if present, would be applied

Table: 11.5.4.-2 LwM2M Object: Program Function Resource definitions

11.6. 3455: Input Filter

Description

The uCIFI Input Filter can be used to fine tune the sensor evaluation, in order to provide hold functionality or custom filtering. To be used in conjunction with a Program Scheduler.

Object definition

Name	Object ID	Object Version	LWM2M Version
Input Filter	3455	2.0	1.0
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:ext:3455:2.0		Multiple	Optional

Table: 11.6.-1 LwM2M Object: Input Filter object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Input State	R	Multiple	Optional	String			Current state captured on the corresponding input source instance

1	Input Source	RW	Multiple	Optional	String			Resources are specified using Core Link syntax Example: Instance #0: </3302/0/5500> Instance #1: </3302/1/5500>
2	Data Type	RW	Single	Optional	Integer			0 – opaque 1 – counter integer 2 – gauge integer 3 – gauge float 4 – Time 5 – String
3	Evaluation Interval	RW	Single	Optional	Float		s	Sets the frequency at which the LwM2M resource identified as Source should be evaluated. Example, if set to 5000, the Program should evaluate the value reported by the LwM2M resource pointed by Source every 5 seconds.
4	Step Filter	RW	Single	Optional	Float			Sets a step interval, making the Program evaluate the LwM2M resource pointed by Source, only when its value changes by more than Step Filter. Example, if set to 1000, the Program should evaluate the value reported by the LwM2M resource pointed by Source only if it changes by more than 1000 units from the last evaluation.
5903	Busy to Clear delay	RW	Single	Optional	Integer		ms	Delay from the detection state to the clear state in ms.
5904	Clear to Busy delay	RW	Single	Optional	Integer		ms	Delay from the clear state to the busy state in ms.

Table: 11.6.-2 LwM2M Object: Input Filter Resource definitions

11.7. 3456: Output Filter

Description

The uCIFI Output Filter can be used to fine tune the output commands in order, for example, to achieve a prolonged fade. To be used in conjunction with a Program Scheduler.

Object definition

Name	Object ID	Object Version	LWM2M Version
Output Filter	3456	1.0	1.0
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:ext:3456		Multiple	Optional

Table: 11.7.-1 LwM2M Object: Output Filter object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description

0	Operation	RW	Single	Mandatory	Integer			This resource defines an operation to be performed between the value of the Control Program output and the value currently store in the LwM2M resource pointed by 'Target'. Possible operations: 1 – set 2 – min 3 – max 4 – add 5 – subtract 6 – multiply When operation or filter is not defined the default output operation is set
1	Delay	RW	Single	Optional	Float		s	When defined the control output is not immediately applied and transaction time = delay is implemented between the current target resource value and the value of the control program output. This can be also associated to Fade Time in smart lighting scenario
2	Transaction	RW	Single	Optional	Integer			Defines the type of progression to occur to transition the current target resource value to the value defined by the of the control program output. Current options: 0 – Immediate 1 – linear 2 – exponential 3 – Device internal mechanism
3	Expiration Time	RW	Single	Optional	Float			Limit the time period for which the output command should be considered valid. This is to be used in scenarios where Control Programs are not under continuous re-evaluation and for which the output control action is not continuously re-calculated. In such scenarios, the expiration time resource can be used to limit the time frame that the control output should be applied to the target resource.

Table: 11.7.-2 LwM2M Object: Output Filter Resource definitions

11.8. 3458:Program Monitor

Description

The uCIFI Program Monitor exports some useful information related to program execution.

Object definition

Name	Object ID	Object Version	LWM2M Version
Program Monitor	3458	1.0	1.0
Object URN		**Instances**	**Mandatory**
urn:oma:lwm2m:ext:3458		Multiple	Optional

Table: 11.8.-1 LwM2M Object: Program Monitor object definition

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Program Name	R	Single	Mandatory	String			Name of the program
1	Status	R	Single	Mandatory	Integer			Identifies the current status of the control program, taking the values: 0 – Disabled 1 – Inactive 2 – Running where 'Disabled' should be present whenever the CP has been administratively disabled. The state should be set to 'Inactive' if the calendar associated with the CP makes is NOT applicable for the current day. If the CP is administratively enabled, and either the associated calendar makes it valid for the current day or no Calendar is associated with it.
2	Error Conditions	R	Single	Mandatory	Integer			Bitmap Bit 0: Sensor syntax error Bit 1: Time syntax error Bit 2: Invalid program function Bit 3..31: reserved for future use
3	Clear errors	E	Single	Optional				When executed, clears temporary error conditions. To be noted that some of the errors may require to change the content of the corresponding program function object to be completely cleared.
4	Evaluation Counter	R	Single	Optional	Integer			Cumulative counter for the number of times the current control program has been evaluated
5	Execution Counter	R	Single	Optional	Integer			Cumulative counter for the number of times the current control program has ben executed after evaluation, triggering the corresponding action and output. With 'Execution counter' less than or equal to 'Evaluation counter'
6	Counter Reset	E	Single	Optional				Reset all counters under the current object
7	Last Execution	R	Single	Optional	Time			Timestamp for the latest execution of the current CP
8	Last Control Output	R	Multiple	Optional	String			String identifying the control output enforced during last program execution. If the control program supports multiple ouputs, one instance of this resource should be created for each output.

Table: 11.8.-2 LwM2M Object: Program Monitor Resource definitions

12. Examples

In the following sections, a few examples will be provided as a reference for the implementors.

12.1. Complex Lighting Use Case – Compact Form

This complex scenario reflects the use case mentioned in section 3.4.

The Light Point Controller (physical device) controls two separate LED drivers with possible color temperature changing (resource 39), one for the roadside and one for the pedestrian side and taking input from two different sensors (one near and one far away).

The overall behavior can be summarized as follows (in bold, the differences between weekends and other days):

	Roadside	Pedestrian
Saturday to Sunday Night	Switch ON to 100% at sunset + 5 minutes and change colour to 4500K Dim down to 50% from **midnight** to 5AM and change colour to 3500K Switch OFF at sunrise - 15 minutes	Switch ON to 100% at sunset + 5 minutes and change colour to 4500K From **midnight** to 5AM Dim down to 30% and change colour to 2900K When sensor #1 (local) detects someone, set to 100% for 1 minute When sensor #2 (remote) detects someone, set to 50% for 1 minute Switch OFF at sunrise - 15 minutes
Every other day	Switch ON to 100% at sunset + 5 minutes at 4500K Dim down to 50% from **sunset + 2 hours** to 5AM and change colour to 3500K Switch OFF at sunrise - 15 minutes	Switch ON to 100% at sunset + 5 minutes and change colour to 4500K From **10PM **to 5AM Dim down to 30% and change colour to 2900K When sensor #1 (local) detects someone, set to 100% for 1 minute When sensor #2 (remote) detects someone, set to 50% for 1 minute Switch OFF at sunrise - 15 minutes

First thing to do is to define the Calendar representation that would represent weekends (Saturday and Sunday, identified by week day 6 and 7 respectively). Since it shall be executed every day of every year, it is possible to use the 1^st^ January to 31^st^ December range.

Finally, since it is associated to a Smart Lighting application, we can assume it do be a Night calendar (thus being active from Saturday noon to Monday noon)

</3453/0> - Calendar #0		
ID	**Name**	**Value**
0	Calendar Name	Weekend
1	Calendar Type	1 (Night)
2/0	Start Date	--01-01
3/0	End Date	--12-31
4	Date Rule	* * 6,7
5	Calendar Active	True

Figure: 12.1.-1 Associated to a Smart Lighting application

Then it would be required to define 4 different Program Functions for:

- Pedestrian lamp behavior during weekends
- Pedestrian lamp behavior during working days
- Roadside lamp behavior during weekends
- Roadside lamp behaviour during working days

Let's first describe the pedestrian program associated with weekends in more detail.

The function name can be used to recognize the program easily.

The run condition should be configured to run from 5 minutes after sunset to 15 minutes before sunrise.

Since it would be required to drive two different outputs (the dimming and the color temperature), some resources will have two instances that will be referred to a specific target

For example, the Time Input Array would contain two instances with the same content[^4]:

- Sunset + 5 minutes
- Midnight
- 5 AM

Whilst the first instance of Time Output Array would contain the corresponding dimming values:

- 100 % (to be executed at Sunset + 5 minutes)
- 30 % (to be executed at Midnight)
- 100 % (to be executed at 5 AM)

And the second instance of the Time Output Array would contain the corresponding color temperature values:

- 4500 K (to be executed at Sunset + 5 minutes)

- 2900 K (to be executed at Midnight)
- 4500 K (to be executed at 5 AM)

Finally, two dynamic actions should be associated to the dimming only:

- From Midnight to 5 AM
- If the local sensor is triggered, set dimming to 100%
- Else if the remote sensor is triggered, set dimming to 50%

It is important to note that:

- The sensors are only being considered within a specific time frame
- There is an explicit precedence within the two sensors

This behavior can be mapped with the following Multi-Sensor Input Array, that would exactly reflect such implications:

(00:00~05:00,1,*); (00:00~05:00,0,1)

The corresponding dimming values would instead be expressed as usual in the Sensor Output Array:

100; 50

</3454/0> - Program Function #0		
ID	**Name**	**Value**
0	Function Name	Pedestrian Weekend
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 00:00; 05:00
4/1		S+5; 00:00; 05:00
5/0	Time Output Array	100; 30; 100
5/1		4500; 2900; 4500
7/0	Multi Sensor Input Array	(00:00~05:00,1,*); (00:00~05:00,0,1)
8/0	Sensor Output Array	100; 50

Figure: 12.1.-2 The corresponding dimming values

</3454/0> - Program Function #0		
ID	**Name**	**Value**
0	Function Name	Pedestrian Weekend
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 00:00; 05:00
4/1		S+5; 00:00; 05:00
5/0	Time Output Array	100; 30; 100
5/1		4500; 2900; 4500
7/0	Multi Sensor Input Array	(00:00~05:00,1,*); (00:00~05:00,0,1)
8/0	Sensor Output Array	100; 50

Figure: 12.1.-3 The sensor input

The sensor input (together with the corresponding filter) will be provided later.

The remaining Program Functions can be created similarly, with the only difference that no dynamic action is required on the Roadside.

</3454/2> - Program Function #2		
ID	**Name**	**Value**
0	Function Name	Roadside Weekend
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 00:00; 05:00
4/1		S+5; 00:00; 05:00
5/0	Time Output Array	100; 50; 100
5/1		4500; 3500; 4500

</3454/3> - Program Function #3		
ID	**Name**	**Value**
0	Function Name	Roadside Default
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; S+120; 05:00
4/1		S+5; S+120; 05:00
5/0	Time Output Array	100; 50; 100
5/1		4500; 3500; 4500

Figure: 12.1.-4 The remaining Program Functions

Since the dynamic control shall be kept for at least one minute, even if the sensor is triggered for a few seconds, the status shall be kept in hold for at least one minute to ensure that the light is kept on for enough time.

Some sensors may allow you to configure the hold time separately (e.g., the IPSO Presence Sensor 3302 includes two distinct resources called Busy to Clear Delay and Clear to Busy Delay). Otherwise, it could be possible to attach an Input Filter to the Program Function to achieve a sample and Hold approach.

</3455/0> – Input Filter #0		
ID	**Name**	**Value**
1/0	Input Source	</3302/0/5500>
3	Evaluation Interval	1000
4	Step Filter	0
5903	Busy to Clear delay	60000
5904	Clear to Busy delay	100

</3455/1> – Input Filter #1		
ID	**Name**	**Value**
1/0	Input Source	</3302/1/5500>
3	Evaluation Interval	1000
4	Step Filter	0
5903	Busy to Clear delay	60000
5904	Clear to Busy delay	100

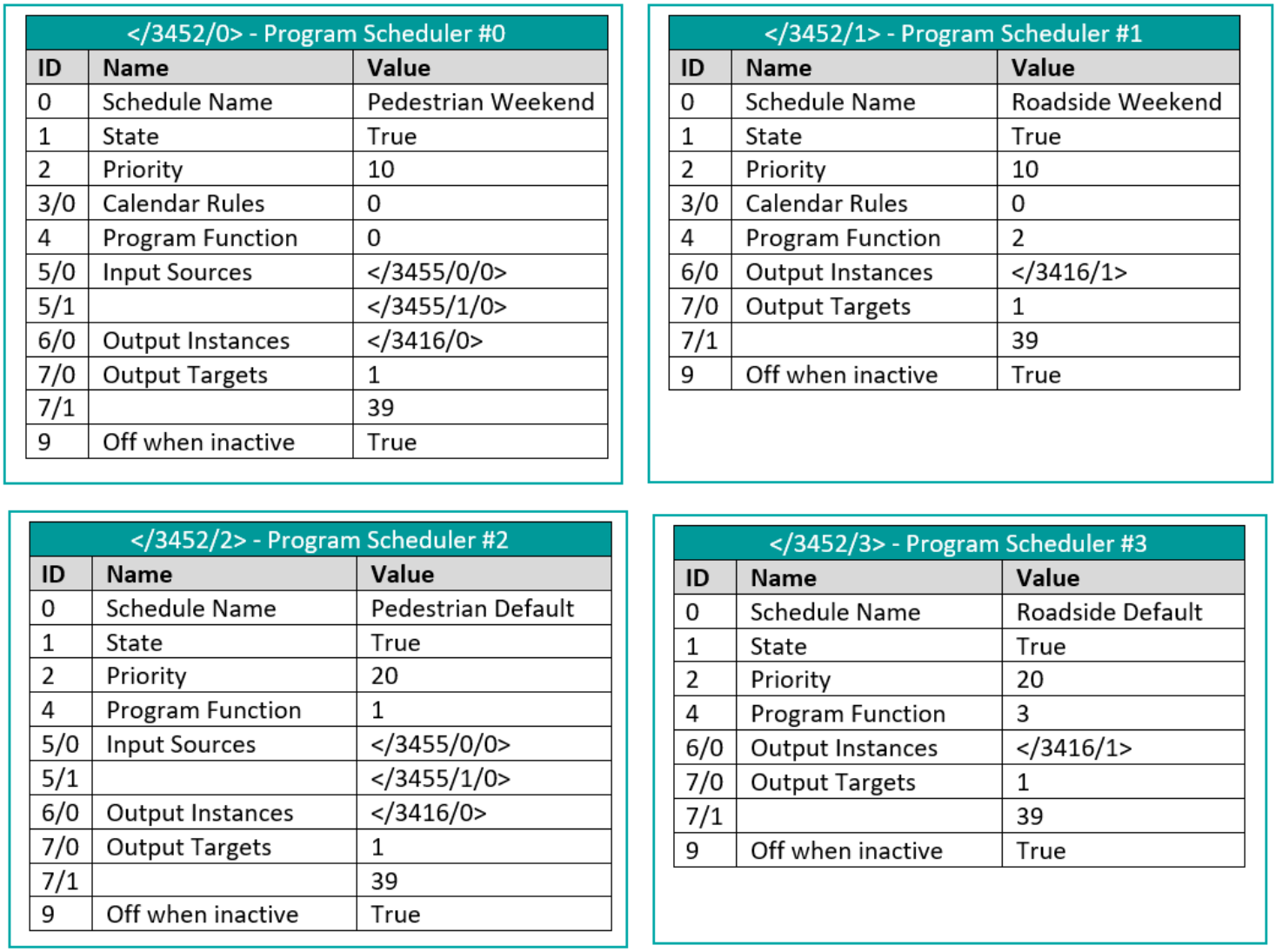
Figure: 12.1.-5 Attach an Input Filter

Finally, everything shall be combined to provide an overall schedule configuration.

It should be noted that the Weekend rule should come first (having a higher priority—lower value) since it should take precedence over the default rule, which, not having any specific calendar rule associated, would match anything else but with a lower priority.

The priority associated to different actuators may be the same since the execution order is meaningless.

The two sensors are referenced within the Program Scheduler's Input Sources and used within the Run Condition and the Multi Sensor Input Array expressions of the associated Program Functions.



</3452/0> - Program Scheduler #0		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Weekend
1	State	True
2	Priority	10
3/0	Calendar Rules	0
4	Program Function	0
5/0	Input Sources	</3455/0/0>
5/1		</3455/1/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
7/1		39
9	Off when inactive	True

</3452/1> - Program Scheduler #1		
ID	**Name**	**Value**
0	Schedule Name	Roadside Weekend
1	State	True
2	Priority	10
3/0	Calendar Rules	0
4	Program Function	2
6/0	Output Instances	</3416/1>
7/0	Output Targets	1
7/1		39
9	Off when inactive	True

</3452/2> - Program Scheduler #2		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Default
1	State	True
2	Priority	20
4	Program Function	1
5/0	Input Sources	</3455/0/0>
5/1		</3455/1/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
7/1		39
9	Off when inactive	True

</3452/3> - Program Scheduler #3		
ID	**Name**	**Value**
0	Schedule Name	Roadside Default
1	State	True
2	Priority	20
4	Program Function	3
6/0	Output Instances	</3416/1>
7/0	Output Targets	1
7/1		39
9	Off when inactive	True

Figure: 12.1.-6 Input Sources of the Program Scheduler

The overall configuration would then consist of:

Object	Number of Instances
Program Scheduler	4
Program Function	4
Calendar Rule	1
Input Filter	2

13. Simulation exercise

As an exercise, it could be worth simulating the behaviour during a Weekend at 2 AM with the local sensor triggered.

For such a task, it is better to resume the alternative representation for the Program Scheduler instances:

	</3452> - Program Scheduler							
Instance	**Name**	**Priority**	**Calendar Rules**	**Program Function**	**Input Source**	**Output Instances**	**Output Targets**	**Compound Operation**
0	Pedestrian Weekend	10	0	0	</3455/0/0> </3455/1/0>	</3416/0>	1 39	False
1	Roadside Weekend	10	0	1		</3416/1>	1 39	False
2	Pedestrian Default	20		2	</3455/0/0> </3455/1/0>	</3416/0>	1 39	False
3	Roadside Default	20		3		</3416/1>	1 39	False

Figure: 13.-1 Program Scheduler instances

The rules shall first be ordered by priority (as already done for the sake of simplicity) and evaluated at each sequential step:

Step	Behaviour	/3416/0/	/3416/0/39	/3416/1/1	/3416/1/39
0	Instance 0 is evaluated. Calendar </3453/0> matches (Weekend) and is executed against targets </3416/0/1> and </3416/0/39>.Since the compound operation is false, such outputs shall be considered finalized	**100 (final)**	**2900 (final)**	-	-
1	Instance 1 is evaluated. Calendar </3453/0> matches (Weekend) and is executed against targets </3416/1/1> and </3416/1/39>. Since the compound operation is false, such outputs shall be considered finalized	no change	no change	**50 (final)**	**3500 (final)**
2	Instance 2 is evaluated. No calendar is associated so that it would match, but the associated outputs are already finalized, so nothing happens	no change	no change	no change	no change
3	Instance 3 is evaluated. No calendar is associated so it would match, but the associated outputs are already finalized, so nothing happens	no change	no change	no change	no change

13.1. Complex Lighting Use Case – Constrained Device Implementation

The very same scenario can also be implemented without requiring some advanced functionalities, such as the Multisensor Output, that some constrained devices may choose not to support.

The configuration of the Roadside is identical to what was explained earlier.

</3454/0> - Program Function #0		
ID	**Name**	**Value**
0	Function Name	Roadside Weekend
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 00:00; 05:00
4/1		S+5; 00:00; 05:00
5/0	Time Output Array	100; 50; 100
5/1		4500; 3500; 4500

</3454/1> - Program Function #1		
ID	**Name**	**Value**
0	Function Name	Roadside Default
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; S+120; 05:00
4/1		S+5; S+120; 05:00
5/0	Time Output Array	100; 50; 100
5/1		4500; 3500; 4500

Figure: 13.1.-1 Constrained Device Implementation

For the Pedestrian, which also includes dynamic behaviour based on sensor value, it is possible to split the functionality into three separate and smaller Program Functions:

- 1 associated with the time-related actions

</3454/2> - Program Function #2		
ID	**Name**	**Value**
0	Function Name	Pedestrian Weekend 1/3
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 00:00; 05:00
4/1		S+5; 00:00; 05:00
5/0	Time Output Array	100; 30; 100
5/1		4500; 2900; 4500

Figure: 13.1.-2 Constrained Device Implementation - Associated with the time-related actions

- 2 associated with the dynamic sensor actions
  - one attached for the local sensor
  - one attached to the remote sensor

</3454/3> - Program Function #3		
ID	**Name**	**Value**
0	Function Name	Pedestrian Weekend 2/3
1/0	Run Condition	00:00~05:00,1
2	Input Filter	3455/0
4/0	Time Input Array	00:00
5/0	Time Output Array	50

</3454/4> - Program Function #4		
ID	**Name**	**Value**
0	Function Name	Pedestrian Weekend 3/3
1/0	Run Condition	00:00~05:00,1
2	Input Filter	3455/0
4/0	Time Input Array	00:00
5/0	Time Output Array	100

Figure: 13.1.-3 Constrained Device Implementation - 2 Associated to the dynamic sensor actions

In this case, the run condition selects when to evaluate the corresponding program. For this reason, a separate time action shall be provided with the dimming to be applied.

Follow the corresponding three Program Functions for the working days:

</3454/5> - Program Function #5		
ID	**Name**	**Value**
0	Function Name	Pedestrian Default 1/3
1/0	Run Condition	S+5~R-15
4/0	Time Input Array	S+5; 22:00; 05:00
4/1		S+5; 22:00; 05:00
5/0	Time Output Array	100; 30; 100
5/1		4500; 2900; 4500

Figure: 13.1.-4 Constrained Device Implementation - 1. three Program Functions

</3454/6> - Program Function #6		
ID	**Name**	**Value**
0	Function Name	Pedestrian Default 2/3
1/0	Run Condition	22:00~05:00,1
4/0	Time Input Array	22:00
5/0	Time Output Array	50

</3454/7> - Program Function #7		
ID	**Name**	**Value**
0	Function Name	Pedestrian Default 3/3
1/0	Run Condition	22:00~05:00,1
4/0	Time Input Array	22:00
5/0	Time Output Array	100

Figure: 13.1.-5 Constrained Device Implementation - 2. three Program Functions

The same Input Filters shown before would also apply here:

</3455/0> – Input Filter #0		
ID	**Name**	**Value**
1/0	Input Source	</3302/0/5500>
3	Evaluation Interval	1000
4	Step Filter	0
5903	Busy to Clear delay	60000
5904	Clear to Busy delay	100

</3455/1> – Input Filter #1		
ID	**Name**	**Value**
1/0	Input Source	</3302/1/5500>
3	Evaluation Interval	1000
4	Step Filter	0
5903	Busy to Clear delay	60000
5904	Clear to Busy delay	100

Figure: 13.1.-6 Constrained Device Implementation - Input Filters

And for each Program Function, a corresponding Program Schedule instance shall be created as well.

The priorities associated with the Weekend rules shall be lower (higher value) than those of the default rules because, thanks to the Compound Operation, they must be executed after the default ones.

</3452/0> - Program Scheduler #0		
ID	**Name**	**Value**
0	Schedule Name	Roadside Weekend
1	State	True
2	Priority	20
3/0	Calendar	0
4	Program Function	0
6/0	Output Instances	</3416/1>
7/0	Output Targets	1
7/1		39
8	Compound Operation	False
9	Off when inactive	True

</3452/1> - Program Scheduler #1		
ID	**Name**	**Value**
0	Schedule Name	Roadside Default
1	State	True
2	Priority	10
4	Program Function	1
6/0	Output Instances	</3416/1>
7/0	Output Targets	1
7/1		39
8	Compound Operation	True
9	Off when inactive	True

Figure: 13.1.-7 Compound Operation

In addition, for the Pedestrian Program Functions, the overall behaviour is determined by the combined evaluation of all the Program Functions in sequence in order to properly consider both time-based and sensor-based rules.

</3452/2> - Program Scheduler #2		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Weekend 1/3
1	State	True
2	Priority	40
3/0	Calendar	0
4	Program Function	2
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
7/1		39
8	Compound Operation	True
9	Off when inactive	True

</3452/3> - Program Scheduler #3		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Weekend 2/3
1	State	True
2	Priority	50
3/0	Calendar	0
4	Program Function	3
5/0	Input Sources	</3455/1/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
8	Compound Operation	True
9	Off when inactive	True

Figure: 13.1.-8 1-Pedestrian Program Functions

</3452/4> - Program Scheduler #4		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Weekend 3/3
1	State	True
2	Priority	60
3/0	Calendar	0
4	Program Function	4
5/0	Input Sources	</3455/0/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
8	Compound Operation	False
9	Off when inactive	True

Figure: 13.1.-9 2-Pedestrian Program Functions

</3452/5> - Program Scheduler #5		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Default 1/3
1	State	True
2	Priority	10
4	Program Function	5
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
7/1		39
8	Compound Operation	True
9	Off when inactive	True

</3452/6> - Program Scheduler #6		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Default 2/3
1	State	True
2	Priority	20
4	Program Function	6
5/0	Input Sources	</3455/1/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
8	Compound Operation	True
9	Off when inactive	True

Figure: 13.1.-10 3-Pedestrian Program Functions

</3452/7> - Program Scheduler #7		
ID	**Name**	**Value**
0	Schedule Name	Pedestrian Default 3/3
1	State	True
2	Priority	30
4	Program Function	7
5/0	Input Sources	</3455/0/0>
6/0	Output Instances	</3416/0>
7/0	Output Targets	1
8	Compound Operation	True
9	Off when inactive	True

Figure: 13.1.-11 4-Pedestrian Program Functions

The overall configuration would then consist of:

Object	Number of Instances
Program Scheduler	8
Program Function	8
Calendar Rule	1
Input Filter	2

13.2. Simulation exercise

As an exercise, it could be worth simulating the behaviour during a Weekend at 2 AM with the local sensor triggered.

For such a task, it is better to resume the alternative representation for the Program Scheduler instances, where the various instances are **ordered by priority** (from lowest to highest).

This is particularly important when the **Compound Operation** is used because the order in which the various instances are evaluated determines the final status of the outputs.

	</3452> - Program Scheduler							
Instance	**Name**	**Priority**	**Calendar Rules**	**Program Function**	**Input Sources**	**Output Instances**	**Output Targets**	**Compound Operation**
1	Roadside Default	10		1		</3416/1>	1 39	True
5	Pedestrian Default 1/3	10		5		</3416/0>	1 39	True
0	Roadside Weekend	20	0	0		</3416/1>	1 39	False
6	Pedestrian Default 2/3	20		6	</3455/1/0>	</3416/0>	1	True
7	Pedestrian Default 3/3	30		7	</3455/0/0>	</3416/0>	1	True
2	Pedestrian Weekend 1/3	40	0	2		</3416/0>	1 39	True
3	Pedestrian Weekend 2/3	50	0	3	</3455/1/0>	</3416/0>	1	True
4	Pedestrian Weekend 3/3	60	0	4	</3455/0/0>	</3416/0>	1	False

Figure: 13.2.-1 Simulation exercise

And simulate what would happen during a Weekend:

Step	Behaviour	</3416/0/	/3416/0/39	/3416/1/1	/3416/1/39
0	Default rule for Roadside is evaluated. No calendar is associated so it always matches. Associated outputs </3416/1/1> and </3416/1/39> are set to 50 % and 3500 % respectively, but are not finalized, since compound operation is true.	-	-	50 (pending)	3500 (pending)

1	Default rule #1 for Pedestrian is evaluated. No calendar is associated so it always matches. Associated outputs </3416/0/1> and </3416/0/39> are set to 30 % and 2900 % respectively, but are not finalized, since the Compound Operation is true.	**30 (pending)**	**2900 (pending)**	no change	no change
2	Weekend rule for Roadside is evaluated. Calendar </3453/0> matches (Weekend) and is executed against targets </3416/1/1> and </3416/1/39>, changing the values to 50 % and 3500 K respectively (despite being the same values as the default program). Since the Compound Operation is false, such outputs shall be considered finalized	no change	no change	**50 (final)**	**3500 (final)**
3	Default rule #2 for Pedestrian is evaluated. No calendar is associated, so it always matches, but the remote sensor is not triggered, so the function is skipped	no change	no change	no change	no change
4	Default rule #3 for Pedestrian is evaluated. The local sensor is triggered, so the target </3416/0/1> is updated, but not finalized, since the Compound Operation is true.	**100 (pending)****	no change	no change	no change
5	Weekend rule #1 for Pedestrians is evaluated. Calendar </3453/0> matches (Weekend) and is executed against targets </3416/0/1> and </3416/0/39>, updating the values. Since the compound operation is true, such outputs shall not be considered finalized	**30 (pending)**	**2900 (pending)**	no change	no change
6	Weekend rule #2 for Pedestrians is evaluated. Calendar </3453/0> matches (Weekend) but the remote sensor is not triggered, so the function is skipped	no change	no change	no change	no change
7	Weekend rule #3 for Pedestrians is evaluated. Calendar </3453/0> matches (Weekend) and the local sensor is triggered, so the target </3416/0/1> is updated accordingly. Since the Compound Operation is false, such outputs shall be considered finalized	**100 (final)**	no change	no change	no change

At the end of the evaluation, since no further schedule is present, the output value for </3416/0/39> is finalized as well.

14. Join the uCIFI Alliance

The uCIFI® Alliance is an IoT alliance composed of leading companies, cities and utilities committed to developing, promoting and certifying the uCIFI® open unified data model for all smart city connected devices and the uCIFI® mesh implementation. uCIFI® leverages and extends existing standards (e.g. LwM2M) and provides an open-source implementation of the unified data model for Cellular and LoRaWAN protocols, which standardize the physical layer and the messaging protocol but do not specify the data model for smart city devices. The uCIFI® open-source smart city stack on top of Wi-SUN (6LowPan) mesh completes the Alliance's mission to provide full interoperability for all smart city devices and prevent vendor lock-in and expensive proprietary API integrations.

To join the OMA, please go to <https://www.openmobilealliance.org/home>.

[^1]: The remote sensor is currently managed as a standard sensor. In the future, virtual sensors will be introduced to support the device-to-device concept, while the overall sensor input concept will remain unchanged.

[^3]: The 12 hours is a generalization and could even be 11 or 13 in case of DST adjustments happening on specific dates

[^4]: There is no special requirement to have the same time intervals for dimming and colour temperature (it has only been done for the sake of simplicity)

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-uCIFI-Schedule-Framework-V1_0-20250530-A	30 May 2025	Status changed to Approved by SCWG on 25 May 2025 conference call.

Table: A.1-1 Approved Version History