Using a Bitfield Data Type

You can use a bitfield data type for a datapoint or property. To access a bitfield within a given data structure, access the affected bits within the container byte which contains the bitfield. The IzoT Interface Interpreter generates utility macros that provide the name of the container byte within the data structure or union, the size of the bit field in bits, and the offset of the least-significant bit of the bit field within the container byte, where the offset counts down from the most significant bit within the byte.

Example

The **UNVT_iot_status_flags** datapoint type is defined as a structure with four one-bit bitfields. The container is a byte named ___**bf00**, for bitfield 00. The f ault bitfield is defined with a size of 1 and an offset of 1. The fault bit has a numeric value of 0x40 within the ___**bf00** byte. The size, offset and container byte are available from the **UNVT_iot_status_flags__fault_SIZE**, **UNVT_iot_status_flags__fault_OFFSET**, and **UNVT_iot_status_flags__fault_CONT AINER** definitions.

To set this bit, perform a logical or of the container byte as shown in the following example.

```
myFlags.UNVT_iot_status_flags__fault_CONTAINER |= 1 << unvt_iot_status_flags__fault_offset
```

To clear this bit, perform a logical and with the inverse of the bit on the container byte as shown in the following example.

```
myFlags.UNVT_iot_status_flags__fault_CONTAINER &= ~(1 << unvt_iot_status_flags__fault_offset)
```