Using a Union Data Type

You can use a union data type for a datapoint or property. To support automatic correction of the byte order, the IzoT Interface Interpreter changes each union to a struct. The automatic transcoding between network and application data presentation handles this correctly so that the data appears on the network as a union.

To write to a member of a structure which originally is defined as a union, call **IzotUpdateUnion()** to distribute the updated field correctly among other members of the union. Failure to do so will not only prevent the other members of the union in reporting the modified values, but will also generally lead to incorrect data on the network.

Example

The following code is generated for a **user** datapoint type that includes a union with two members.

```
typedef struct {
        unsigned char select ;
        struct {  // originally a union
                unsigned short field_a ;
                double field_b ;
        } u;
} ExampleType;
```

The following code assigns a new value to the **field_b** member.

```
ExampleType(nvoExample) nvoExample; //@IzoT datapoint

void example(void) {
        nvoExample.data.u.field_b = 1234E-5;
        IzotUpdateUnion (
                &nvoExample.data,
                offsetof(ExampleType, u.field_b),
                sizeof(ExampleType .data.u.field_b)
        );
}
```

Because network types containing unions are presented to your applications with embedded structures in place of the union, you cannot generally use the standard C **sizeof()** operator to determine the network size of a given type. However, the **sizeof()** operator will correctly report the application size of the given type. To determine a given datapoint's initial network size use **IzotGetDeclaredNetworkSize()** or **IzotGetDeclaredNetworkSizeByIndex()**. The actual size of a changeable-type datapoint may be different.