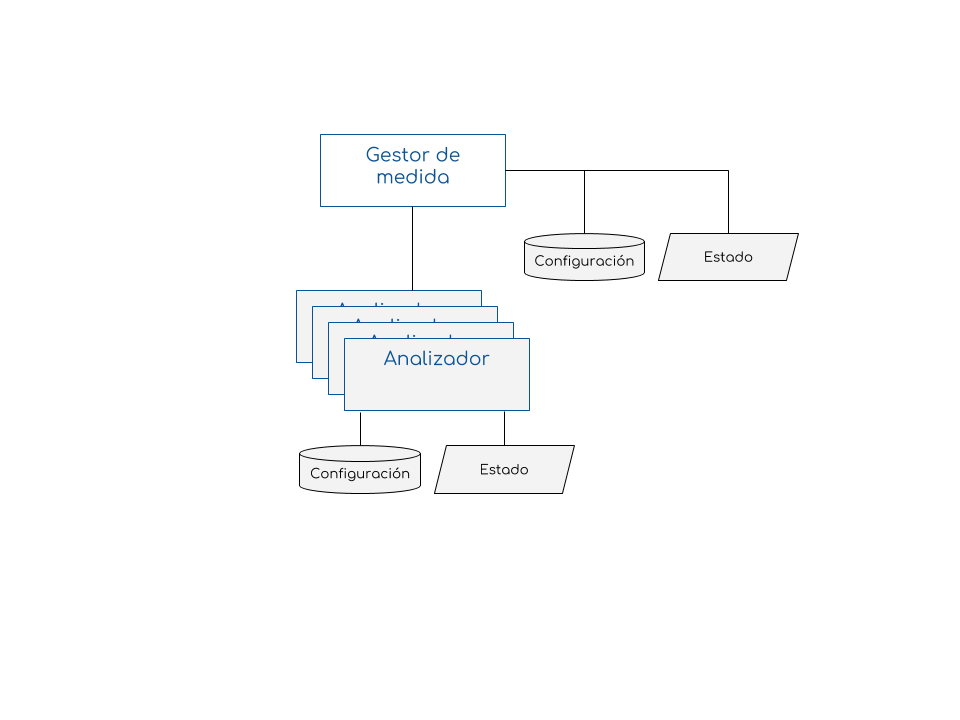
**Modelos de datos relativos al**

**Componente de Medida de parámetros eléctricos**

El diagrama general que representa al Gestor de medida junto con sus analizadores integrados es el siguiente:

****

**Objetos, esquemas e identificación de versiones**

Los modelos de datos se pueden organizar en base a un esquema común formado por varios ítems separados por ‘**:**’, del tipo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ámbito | Funcionalidad | Parámetro | Argumento\_1 | Argumento\_2 | … | Argumento\_N |

Así por ejemplo, en el caso de los analizadores eléctricos, el esquema para definir las medidas instantáneas, se podría definir como:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ámbito | Funcionalidad | Parámetro | Argumento\_1 | Argumento\_2 | … | Argumento\_N |
| **ppl** | **energy** | **analyzer** | **stat** | **measure** |  |  |

**-------------------**

Objeto resultante: **ppl**:**energy**:**analyzer**:**stat:measure**

**-------------------**

Los objetos se representarán en formato JSON, por lo que para su identificación se utilizará un **UID** (uint32) que incluya información sobre el tipo de objeto y la versión del mismo. Este UID será utilizado en el lado del Servidor para poder catalogar el tipo de datos recibidos antes de su registro en la base de datos.

Una posible nomenclatura del UID al ser un uint32 (0xaabbccdd) podría ser ésta (que permite identificar más de 1 millón de objetos distintos con 4096 versiones diferentes para cada uno de ellos):

|  |  |
| --- | --- |
| Bits 31- 20 | Bits 19 – 0 |
| Identificador de la versión | Identificador único del tipo de objeto |

Así por ejemplo si el objeto **ppl**:**energy**:**analyzer**:**stat:measure** anterior, tiene UID = 1 (0x00000001), entonces está en su versión 0 (versión 0 🡪 0x**000**00001).

Mientras que ese mismo objeto, en su versión 1, tendría un UID = 1048577 (0x**001**00001).

**Gestor de la medida**

En la siguiente tabla se definen los diferentes modelos de datos relativos al módulo que gestiona la medida y que puede integrar uno o varios analizadores de diferente tipo.

|  |  |
| --- | --- |
| **Objetos JSON** | **Descripción** |
| ***ppl:energy*** | *Objeto que incluye toda la información asociada a un gestor de medida de parámetros eléctricos. Incluye información como:*   * *Identificador único que incluye información sobre el tipo de objeto y la versión del mismo* * *Lista de analizadores* * *Parámetros de configuración del gestor* * *Variables de estado del gestor* |
| ***ppl:energy:cfg*** | *Objeto que incluye todos los parámetros de configuración del gestor de medida, como por ejemplo:*   * *Cadencia de envío de medidas instantáneas al servidor* * *Nivel de generación de trazas de depuración de la operativa del gestor y los analizadores integrados* * *Etc...* |
| ***ppl:energy:stat*** | *Objeto que incluye todas las variables de estado del gestor, incluyendo las variables de estado de los analizadores integrados (alarmas, eventos, medidas instantáneas, totales de energía, etc.)* |

**Analizadores eléctricos**

En la siguiente tabla se definen los diferentes modelos de datos relativos únicamente a los analizadores eléctricos.

|  |  |
| --- | --- |
| **Objetos JSON** | **Descripción** |
| ***ppl:energy:analyzer*** | *Objeto que incluye toda la información asociada a un analizador eléctrico, como:*   * *Identificador del analizador (número de serie o similar)* * *Medidas instantáneas* * *Totales acumulados (energía, ...)* * *Parámetros de configuración, calibración, etc., del analizador* * *Etc...* |
| ***ppl:energy:analyzer:cfg*** | *Objeto que incluye todos los parámetros de configuración de un analizador eléctrico:*   * *Rangos min-max para la generación de alarmas* ***ppl:energy:analyzer:cfg:minmax*** * *Parámetros de calibración* ***ppl:energy:analyzer:cfg:calib*** * *Etc...* |
| ***ppl:energy:analyzer:stat*** | *Objeto que incluye todas las variables de estado de un analizador eléctrico, como:*   * *Totales (energía,...):* ***ppl:energy:analyzer:stat:totals*** * *Medidas instantáneas:* ***ppl:energy:analyzer:stat:measure*** * *Etc...* |

**Resumen de objetos. Asignación de UIDs**

En la siguiente tabla se enumeran todos los objetos relativos al Gestor de Medida (***ppl:energy***), asignando sus rangos UID, para su identificación en la parte del Servidor.

|  |  |
| --- | --- |
| **Objetos JSON** | **Rangos UIDs (hasta 4096 versiones por UID)** |
| ***ppl:energy*** | *0x00000001 – 0xfff00001* |
| ***ppl:energy:cfg*** | *0x00000002 – 0xfff00002* |
| ***ppl:energy:stat*** | *0x00000003 – 0xfff00003* |
| ***ppl:energy:analyzer*** | *0x00000004 – 0xfff00004* |
| ***ppl:energy:analyzer:cfg*** | *0x00000005 – 0xfff00005* |
| ***ppl:energy:analyzer:cfg:minmax*** | *0x00000006 – 0xfff00006* |
| ***ppl:energy:analyzer:cfg:calib*** | *0x00000007 – 0xfff00007* |
| ***ppl:energy:analyzer:stat*** | *0x00000008 – 0xfff00008* |
| ***ppl:energy:analyzer:stat:totals*** | *0x00000009 – 0xfff00009* |
| ***ppl:energy:analyzer:stat:measure*** | *0x0000000A – 0xfff0000A* |
| *Otros Objetos externos requeridos:* | |
| ***range:minmaxthres:double*** | *0x00010000 – 0xfff10000* |

**Descripción de los diferentes objetos**

|  |  |
| --- | --- |
| ***ppl:energy*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"stat": ppl:energy:stat,*  *"cfg": ppl:energy:cfg*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00001) y versión*  *Valores*  *0x00000001 – 0xFFF00001 siendo XYZ el identificador de la versión* |
| *cfg : ppl:energy:cfg*  *Read-only* | *Descripción*  *Objeto que contiene los parámetros de configuración del gestor energético, y de los analizadores gestionados.* |
| *stat : ppl:energy:stat*  *Read-only* | *Descripción*  *Objeto que contiene el estado actual del gestor energético, incluyendo el estado de los analizadores gestionados.* |

|  |  |
| --- | --- |
| ***ppl:energy:cfg*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"updFlags": u32,*  *"measPeriod": u16,*  *"verbosity": u8,*  *"analyzers": [ppl:energy:analyzer:cfg]*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00002) y versión*  *Valores*  *0x00000002 – 0xFFF00002 siendo XYZ el identificador de la versión* |
| *updFlags : u32*  *Read-Write* | *Descripción*  *Conjunto de flags, que habilitan o no, la notificación de los cambios hechos en algún parámetro de configuración.*  *Valores*  *EnableAMCfgUpdNotif* = (1 << 0), /// notifica cambios en cualquier parámetro de la configuración del objeto  *[default = EnableAMCfgUpdNotif]* |
| *measPeriod : u32*  *Read-Write* | *Descripción*  *Cadencia de envío de las medidas eléctricas, expresada en segundos.*  *Valores*  *15 – maxU32 [default = 900]* |
| *verbosity : u8*  *Read-Write* | *Descripción*  *Nivel de visualización de las trazas de depuración.*  *Valores*  *ESP\_LOG\_NONE* /\*!< No log output \*/  *ESP\_LOG\_ERROR* /\*!< Critical errors, software module can not recover on its own \*/  *ESP\_LOG\_WARN* /\*!< Error conditions from which recovery measures have been taken \*/  *ESP\_LOG\_INFO* /\*!< Information messages which describe normal flow of events \*/  *ESP\_LOG\_DEBUG* /\*!< Extra information which is not necessary for normal use (values, pointers, …). \*/  *ESP\_LOG\_VERBOSE* /\*!< Bigger chunks of information, or messages which can flood the terminal \*/    *[default = ESP\_LOG\_WARN]* |
| *analyzers : [ppl:energy:analyzer:cfg]*  *Read-Write* | *Descripción*  *Objeto que contiene un array con la configuración de los diferentes analizadores integrados. Nota: En el caso de incluir un único analizador con medida trifásica, se representarán como 3 analizadores independientes.* |

|  |  |
| --- | --- |
| ***ppl:energy:stat*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"analyzers": [ppl:energy:analyzer:stat]*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00003) y versión*  *Valores*  *0x00000003 – 0xFFF00003 siendo XYZ el identificador de la versión* |
| *analyzers : [ppl:energy:analyzer:stat]*  *Read-only* | *Descripción*  *Objeto que contiene un array con los estados de los diferentes analizadores integrados. Nota: En el caso de incluir un único analizador con medida trifásica, se representarán como 3 analizadores independientes.* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"serial": char[16],*  *"cfg": ppl:energy:analyzer:cfg,*  *"stat": ppl:energy:analyzer:stat*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00004) y versión*  *Valores*  *0x00000004 – 0xFFF00004 siendo XYZ el identificador de la versión* |
| *serial : char[16]*  *Read-only* | *Descripción*  *Número de serie (o similar) del analizador. Texto en formato ASCII de hasta 15 caracteres* |
| *cfg : ppl:energy:analyzer:cfg*  *Read-Write* | *Descripción*  *Objeto que contiene los parámetros de configuración del analizador.* |
| *stat : ppl:energy:analyzer:stat*  *Read-only* | *Descripción*  *Objeto que contiene el estado actual del analizador.* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:cfg*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"updFlags": u32,*  *"evtFlags": u32,*  *"minmaxData": ppl:energy:analyzer:cfg:minmax,*  *"calibData": ppl:energy:analyzer:calib*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00005) y versión*  *Valores*  *0x00000005 – 0xFFF00005 siendo XYZ el identificador de la versión* |
| *updFlags : u32*  *Read-Write* | *Descripción*  *Conjunto de flags, que habilitan o no, la notificación de los cambios hechos en algún parámetro de configuración.*  *Valores*  *EnableAMCfgUpdNotif* = (1 << 0), /// Habilita notif de cambios en cualquier parámetro  *EnableAMMinMaxCfgUpdNotif* = (1 << 1), /// Habilita notif de cambios en ppl:energy:analyzer:cfg:minmax  *EnableAMCalibCfgUpdNotif* = (1 << 2), /// Habilita notif de cambios en ppl:energy:analyzer:cfg:calib  *[default = EnableAMCfgUpdNotif]* |
| *evtFlags : u32*  *Read-Write* | *Descripción*  *Conjunto de flags, que habilitan las notificaciones de los diferentes eventos/alarmas. Para que una alarma sea notificada, su flag debe estar activado, en caso contrario, aunque la condición de alarma se produzca, no se generará ninguna notificación.*  *Valores*  *AMNoEvents* = 0, //!< Flag para indicar que no hay eventos  *AMVoltageOverLimitEvt* = (1 << 0), //!< Evento al superar el límite superior (voltaje)  *AMVoltageBelowLimitEvt* = (1 << 1), //!< Evento al superar el límite inferior (voltaje)  *AMVoltageInRangeEvt* = (1 << 2), //!< Evento al volver a entrar en rango (voltaje)  *AMCurrentOverLimitEvt* = (1 << 3), //!< Evento al superar el límite superior (corriente)  *AMCurrentBelowLimitEvt* = (1 << 4), //!< Evento al superar el límite inferior (corriente)  *AMCurrentInRangeEvt* = (1 << 5), //!< Evento al volver a entrar en rango (corriente)  *AMPhaseOverLimitEvt* = (1 << 6), //!< Evento al superar el límite superior (fase)  *AMPhaseBelowLimitEvt* = (1 << 7), //!< Evento al superar el límite inferior (fase)  *AMPhaseInRangeEvt* = (1 << 8), //!< Evento al volver a entrar en rango (fase)  *AMPFactorOverLimitEvt* = (1 << 9), //!< Evento al superar el límite superior (factor de potencia)  *AMPFactorBelowLimitEvt* = (1 << 10), //!< Evento al superar el límite inferior (factor de potencia)  *AMPFactorInRangeEvt* = (1 << 11), //!< Evento al volver a entrar en rango (factor de potencia)  *AMCosPhyOverLimitEvt* = (1 << 12), //!< Evento al superar el límite superior (coseno phy)  *AMCosPhyBelowLimitEvt* = (1 << 13), //!< Evento al superar el límite inferior (coseno phy)  *AMCosPhyInRangeEvt* = (1 << 14), //!< Evento al volver a entrar en rango (coseno phy)  *AMActPowOverLimitEvt* = (1 << 15), //!< Evento al superar el límite superior (potencia activa)  *AMActPowBelowLimitEvt* = (1 << 16), //!< Evento al superar el límite inferior (potencia activa)  *AMActPowInRangeEvt* = (1 << 17), //!< Evento al volver a entrar en rango (potencia activa)  *AMReactPowOverLimitEvt* = (1 << 18), //!< Evento al superar el límite superior (potencia reactiva)  *AMReactPowBelowLimitEvt* = (1 << 19), //!< Evento al superar el límite inferior (potencia reactiva)  *AMReactPowInRangeEvt* = (1 << 20), //!< Evento al volver a entrar en rango (potencia reactiva)  *AMFrequencyOverLimitEvt* = (1 << 21), //!< Evento al superar el límite superior (frecuencia)  *AMFrequencyBelowLimitEvt* = (1 << 22), //!< Evento al superar el límite inferior (frecuencia)  *AMFrequencyInRangeEvt* = (1 << 23), //!< Evento al volver a entrar en rango (frecuencia)  *AMThdAOverLimitEvt* = (1 << 24), //!< Evento al superar el límite superior (THD-A)  *AMThdABelowLimitEvt* = (1 << 25), //!< Evento al superar el límite inferior (THD-A)  *AMThdAInRangeEvt* = (1 << 26), //!< Evento al volver a entrar en rango (THD-A)  *AMThdVOverLimitEvt* = (1 << 27), //!< Evento al superar el límite superior (THD-V)  *AMThdVBelowLimitEvt* = (1 << 28), //!< Evento al superar el límite inferior (THD-V)  *AMThdVInRangeEvt* = (1 << 29), //!< Evento al volver a entrar en rango (THD-V)  *AMInstantMeasureEvt* = (1 << 30), //!< Evento al realizar una medida  *AMInvalidEvt* = (1 << 31)  *[default = AMInstantMeasureEvt]* |
| *minmaxData : ppl:energy:cfg:minmax*  *Read-Write* | *Descripción*  *Objeto que incluye los rangos min-max-thres de diferentes parámetros relativos a la generación de eventos y/o alarmas:*   * *Al salir de rango por encima (value > max)* * *Al salir de rango por debajo (value < min)* * *Al volver al rango tras haber salido (value < [max-thres]) o (value > [min+thres]).* |
| *calibData : ppl:energy:cfg:calib*  *Read-Write* | *Descripción*  *Objeto que incluye los valores de calibración del analizador.* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:cfg:minmax (versión driver EMi10 YTL)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"voltage": range:minmaxthres:double,*  *"current": range:minmaxthres:double,*  *"phase": range:minmaxthres:double,*  *"pfactor": range:minmaxthres:double,*  *"aPow": range:minmaxthres:double,*  *"rPow": range:minmaxthres:double,*  *"msPow": range:minmaxthres:double,*  *"freq": range:minmaxthres:double,*  *"thdA": range:minmaxthres:double,*  *"thdV": range:minmaxthres:double*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00006) y versión*  *Valores*  *0x00000006 – 0xFFF00006 siendo XYZ el identificador de la versión*  *[default = 0x00000006] versión para el driver Emi10 YTL* |
| *voltage : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Rango de valores mínimo-máximo del Voltaje para la generación de alarmas al salir de rango. Incluye “threshold” para detectar la reentrada en el rango. Se mide en Voltios* |
| *current : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Corriente. Se mide en Amperios* |
| *phase : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Fase. Se mide en grados (º)* |
| *pfactor : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Factor de Potencia. No tiene unidades* |
| *aPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia activa. Se mide en KW* |
| *rPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia reactiva. Se mide en KW* |
| *msPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia media aparente. Se mide en KVA* |
| *freq : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Frecuencia. Se mide en Hz* |
| *thdA : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Distorsión armónica total de corriente. No tiene unidades* |
| *thdV : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Distorsión armónica total de tensión. No tiene unidades* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:cfg:minmax (versión driver M90E26)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"voltage": range:minmaxthres:double,*  *"current": range:minmaxthres:double,*  *"phase": range:minmaxthres:double,*  *"pfactor": range:minmaxthres:double,*  *"aPow": range:minmaxthres:double,*  *"rPow": range:minmaxthres:double,*  *"msPow": range:minmaxthres:double,*  *"freq": range:minmaxthres:double,*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00006) y versión*  *Valores*  *0x00000006 – 0xFFF00006 siendo XYZ el identificador de la versión*  *[default = 0x00100006] versión para el driver M90E26* |
| *voltage : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Rango de valores mínimo-máximo del Voltaje para la generación de alarmas al salir de rango. Incluye “threshold” para detectar la reentrada en el rango. Se mide en Voltios* |
| *current : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Corriente. Se mide en Amperios* |
| *phase : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Fase. Se mide en grados (º)* |
| *pfactor : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Factor de Potencia. No tiene unidades* |
| *aPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia activa. Se mide en KW* |
| *rPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia reactiva. Se mide en KW* |
| *msPow : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Potencia media aparente. Se mide en KVA* |
| *freq : range:minmaxthres:double*  *Read-Write* | *Descripción*  *Idem para Frecuencia. Se mide en Hz* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:cfg:calib (versión driver EMi10 YTL)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00007) y versión*  *Valores*  *0x00000007 – 0xFFF00007 siendo XYZ el identificador de la versión*  *[default = 0x00000007] versión para el driver Emi10 YTL* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:cfg:calib (versión driver M90E26)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"meterRegs": u16[16],*  *"measRegs": u16[16]*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00007) y versión*  *Valores*  *0x00000007 – 0xFFF00007 siendo XYZ el identificador de la versión*  *[default = 0x00100007] versión para el driver M90E26* |
| *meterRegs : u16[16]*  *Read-Write* | *Descripción*  *Objeto que incluye los valores de calibración del medidor del driver de medida M90E26.*  *Ver sección 5.3.1 del datasheet:* <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-46002-SE-M90E26-Datasheet.pdf>*.* |
| *measRegs : u16[16]*  *Read-Write* | *Descripción*  *Objeto que incluye los valores de calibración de la medida del driver de medida M90E26.*  *Ver sección 5.3.2 del datasheet:* <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-46002-SE-M90E26-Datasheet.pdf>*.* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:stat*** | |
| *{*  *"uid": u32,*  *"flags": u32,*  *"energyValues": ppl:energy:analyzer:stat:totals,*  *"measureValues": ppl:energy:analyzer:stat:measure*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00008) y versión*  *Valores*  *0x00000008 – 0xFFF00008 siendo XYZ el identificador de la versión* |
| *flags : u32*  *Read-only* | *Descripción*  *Conjunto de flags, que indican los eventos y/o alarmas activos en el instante actual*  *Valores*  *AMNoEvents* = 0, //!< Flag para indicar que no hay eventos  *AMVoltageOverLimitEvt* = (1 << 0), //!< Evento al superar el límite superior (voltaje)  *AMVoltageBelowLimitEvt* = (1 << 1), //!< Evento al superar el límite inferior (voltaje)  *AMVoltageInRangeEvt* = (1 << 2), //!< Evento al volver a entrar en rango (voltaje)  *AMCurrentOverLimitEvt* = (1 << 3), //!< Evento al superar el límite superior (corriente)  *AMCurrentBelowLimitEvt* = (1 << 4), //!< Evento al superar el límite inferior (corriente)  *AMCurrentInRangeEvt* = (1 << 5), //!< Evento al volver a entrar en rango (corriente)  *AMPhaseOverLimitEvt* = (1 << 6), //!< Evento al superar el límite superior (fase)  *AMPhaseBelowLimitEvt* = (1 << 7), //!< Evento al superar el límite inferior (fase)  *AMPhaseInRangeEvt* = (1 << 8), //!< Evento al volver a entrar en rango (fase)  *AMPFactorOverLimitEvt* = (1 << 9), //!< Evento al superar el límite superior (factor de potencia)  *AMPFactorBelowLimitEvt* = (1 << 10), //!< Evento al superar el límite inferior (factor de potencia)  *AMPFactorInRangeEvt* = (1 << 11), //!< Evento al volver a entrar en rango (factor de potencia)  *AMCosPhyOverLimitEvt* = (1 << 12), //!< Evento al superar el límite superior (coseno phy)  *AMCosPhyBelowLimitEvt* = (1 << 13), //!< Evento al superar el límite inferior (coseno phy)  *AMCosPhyInRangeEvt* = (1 << 14), //!< Evento al volver a entrar en rango (coseno phy)  *AMActPowOverLimitEvt* = (1 << 15), //!< Evento al superar el límite superior (potencia activa)  *AMActPowBelowLimitEvt* = (1 << 16), //!< Evento al superar el límite inferior (potencia activa)  *AMActPowInRangeEvt* = (1 << 17), //!< Evento al volver a entrar en rango (potencia activa)  *AMReactPowOverLimitEvt* = (1 << 18), //!< Evento al superar el límite superior (potencia reactiva)  *AMReactPowBelowLimitEvt* = (1 << 19), //!< Evento al superar el límite inferior (potencia reactiva)  *AMReactPowInRangeEvt* = (1 << 20), //!< Evento al volver a entrar en rango (potencia reactiva)  *AMFrequencyOverLimitEvt* = (1 << 21), //!< Evento al superar el límite superior (frecuencia)  *AMFrequencyBelowLimitEvt* = (1 << 22), //!< Evento al superar el límite inferior (frecuencia)  *AMFrequencyInRangeEvt* = (1 << 23), //!< Evento al volver a entrar en rango (frecuencia)  *AMThdAOverLimitEvt* = (1 << 24), //!< Evento al superar el límite superior (THD-A)  *AMThdABelowLimitEvt* = (1 << 25), //!< Evento al superar el límite inferior (THD-A)  *AMThdAInRangeEvt* = (1 << 26), //!< Evento al volver a entrar en rango (THD-A)  *AMThdVOverLimitEvt* = (1 << 27), //!< Evento al superar el límite superior (THD-V)  *AMThdVBelowLimitEvt* = (1 << 28), //!< Evento al superar el límite inferior (THD-V)  *AMThdVInRangeEvt* = (1 << 29), //!< Evento al volver a entrar en rango (THD-V)  *AMInstantMeasureEvt* = (1 << 30), //!< Evento al realizar una medida  *AMInvalidEvt* = (1 << 31) |
| *energyValues : ppl:energy:stat:totals*  *Read-only* | *Descripción*  *Objeto que incluye las medidas totales de energía (activa, reactiva, etc...).* |
| *measureValues : ppl:energy:stat:measure*  *Read-only* | *Descripción*  *Objeto que incluye los valores de medida instantánea.* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:stat:totals*** | |
| *{*  *"uid": u32,*  *"active": double,*  *"reactive": double*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (00009) y versión*  *Valores*  *0x00000009 – 0xFFF00009 siendo XYZ el identificador de la versión* |
| *active : double*  *Read-only* | *Descripción*  *Lectura de los totales de energía activa. En KW* |
| *reactive : double*  *Read-only* | *Descripción*  *Lectura de los totales de energía reactiva. En KW* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:stat:measure (versión driver EMi10 YTL)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"voltage": double,*  *"current": double,*  *"phase": double,*  *"pfactor": double,*  *"aPow": double,*  *"rPow": double,*  *"msPow": double,*  *"freq": double,*  *"thdA": double,*  *"thdV": double*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (0000A) y versión*  *Valores*  *0x0000000A – 0xFFF0000A siendo XYZ el identificador de la versión*  *[default = 0x0000000A] versión para el driver Emi10 YTL* |
| *voltage : double*  *Read-only* | *Descripción*  *Voltaje en Voltios* |
| *current : double*  *Read-only* | *Descripción*  *Corriente en Amperios* |
| *phase : double*  *Read-only* | *Descripción*  *Fase en grados (º)* |
| *pfactor : double*  *Read-only* | *Descripción*  *Factor de Potencia. No tiene unidades* |
| *aPow : double*  *Read-only* | *Descripción*  *Potencia activa en KW* |
| *rPow : double*  *Read-only* | *Descripción*  *Potencia reactiva en KW* |
| *msPow : double*  *Read-only* | *Descripción*  *Potencia media aparente en KVA* |
| *freq : double*  *Read-only* | *Descripción*  *Frecuencia en Hz* |
| *thdA : double*  *Read-only* | *Descripción*  *Distorsión armónica total de corriente. No tiene unidades* |
| *thdV : double*  *Read-only* | *Descripción*  *Distorsión armónica total de tensión. No tiene unidades* |

|  |  |
| --- | --- |
| ***ppl:energy:analyzer:stat:measure (versión driver M90E26)*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"voltage": double,*  *"current": double,*  *"phase": double,*  *"pfactor": double,*  *"aPow": double,*  *"rPow": double,*  *"msPow": double,*  *"freq": double,*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (0000A) y versión*  *Valores*  *0x0000000A – 0xFFF0000A siendo XYZ el identificador de la versión*  *[default = 0x0010000A] versión para el driver M90E26* |
| *voltage : double*  *Read-only* | *Descripción*  *Voltaje en Voltios* |
| *current : double*  *Read-only* | *Descripción*  *Corriente en Amperios* |
| *phase : double*  *Read-only* | *Descripción*  *Fase en grados (º)* |
| *pfactor : double*  *Read-only* | *Descripción*  *Factor de Potencia. No tiene unidades* |
| *aPow : double*  *Read-only* | *Descripción*  *Potencia activa en KW* |
| *rPow : double*  *Read-only* | *Descripción*  *Potencia reactiva en KW* |
| *msPow : double*  *Read-only* | *Descripción*  *Potencia media aparente en KVA* |
| *freq : double*  *Read-only* | *Descripción*  *Frecuencia en Hz* |

**Otros objetos externos requeridos**

|  |  |
| --- | --- |
| ***range:minmaxthres:double*** | |
| ***Objeto JSON:***  *{*  *"uid": u32,*  *"min": double,*  *"max": double,*  *"thres": double*  *}* | |
|  | |
| *uid : u32*  *Read-only* | *Descripción*  *Identificador del tipo de objeto (10000) y versión*  *Valores*  *0x00010000 – 0xFFF10000 siendo XYZ el identificador de la versión* |
| *min : double*  *Read-Write* | *Descripción*  *Valor mínimo* |
| *max : double*  *Read-Write* | *Descripción*  *Valor máximo* |
| *thres : double*  *Read-Write* | *Descripción*  *Ventana “threshold”* |