SunSpec Alliance Interoperability Specification

Common Models

By: SunSpec Alliance Common Models Workgroup

John Blair, John Nunneley, Karl Lambert, Pat Adamosky, Ronnie Petterson, Lynn Linse, Bill Randle, Bob Fox, Aaron Parker, Stephen Lapointe

Version 1.6

The SunSpec Alliance Interoperability Specifications describe the information models and MODBUS register mappings for devices used in Renewable Energy systems. A physical device, such as an inverter, is represented by a collection of one or more of these logical models. This document describes the Common Models and model design conventions. All SunSpec devices must contain the device information common model that describes this particular device as the first logical model of the device.

Change History

1.3: Added change history

1.3: Allow device aggregation via multiple common models

1.3 Manufacturer ID should be registered with SunSpec

1.3 Added M/O/C column to xls map.

1.3 Require Manufacturer, Model, and Serial Number to be supplied

1.4 Add Copyright

1.5 Updated Logo

1.5 Corrected Assigned ID range for String Combiner and Module

1.5 Added Assigned ID ranges for Inverter model, version 1.2, Module model, version 1.1, Inverter Controls and Network Interface.

1.5 Added concept of a scale factor as a basic SunSpec type.

1.5 Added concept of SunSpec Device type to Common Model

1.5 Changed over to new names for points

1.5 Added definition for acc64, ipaddr, ipv6addr types

1.5 Scale factor clarification wrt unimplemented

1.5 Align the document with the new short names

1.5 Update Network Configuration, new model ids and some descriptions

1.5 Add recommendation on default serial port settings (9600,8,N,1)

1.5 Removed model table and replaced with reference to the PICS

1.5 Added descriptions for bitfield and enum

1.5 John Blair's contribution: document new types, terminology, model structure, best practices, other conventions. Included Control Procedural Requirements

1.6 Updated Title Page

1.6 Updated Copyright

1.6 Updated Abstract

1.6 Added "Protocol Mapping" section

Introduction

The SunSpec Alliance Interoperability Specifications describe the information models and MODBUS register mappings for devices used in Renewable Energy systems. A physical device, such as an inverter, is represented by a collection of one or more of these logical models. This document describes the Common Models and model design conventions. All SunSpec devices must contain the device information common model that describes this particular device as the first logical model of the device.

The data models can be easily moved by XML and other technology, so these specifications are not tied to the 30-year old MODBUS protocol; MODBUS is the initial implementation specified because MODBUS is well understood and supported within renewable energy systems.

Protocol Mapping

The device specifications outlined in this document support Modbus and XML protocol.

Terminology

Device	**A collection of models including and starting with the Common model.**
Model	A complete SunSpec data model, starting with the model ID (DID), the length and the data.
Block	A section of Modbus registers. A canonical SunSpec model consists of two blocks, each of which is optional.
Instance	The repeating sub-section of a repeating, variable-length block. The total length of the repeating block must be a multiple of the fixed length instance.
Point	A value encoded in one or more Modbus registers using one of the SunSpec datatypes specified in the Common Model Specification. To define a machine readable format for model definition we need to formally specify the structure of a SunSpec model.

SunSpec Device Types

Each type of device is defined in a separate document that details the fields and values specific to that type. Current SunSpec Device Types defined are

1. Inverter
2. Meter

3. Weather Station
4. String Combiner
5. PV Module
6. Tracker
7. MPPT Module
8. Charge Controller

Additional device models will be taken up and defined as directed by the SunSpec Alliance membership.

SunSpec Common Models and Conventions

This document focuses on the elements of the specification that are common to all device types.

- Common data model
- Standard data formats
- MODBUS register mappings
- Device Specification Format
- Extensibility
- Device Aggregation model
- Network configuration models
- Optionality

Common Model

All SunSpec devices support a consecutive collection of well defined models, each prefaced with a well-known Model Identity and length field. This allows a remote client to browse the contents skipping models with unrecognized Model Identity (ID) values. The Common Model specifically allows for extensibility in the following ways:

1. Definition of new Device Specific Models. New devices are defined and assigned a new model id.
2. Revisions of Device Specific Models. Revisions to device models are defined and assigned a new model id.
3. Vendor specific extensions. Models may include fields that have vendor specific values. An example of this is the status and event code fields. Vendors may also define a Vendor Extension Model that includes fields and values specific to the vendor. These are assigned a model type id. This model may be then concatenated to the associated Device Specific Model.
4. Composite device types. Device Specific Models may be concatenated together to form a composite device. An example of this is a Weather Station represented by a collection of environmental models.
5. Device aggregations. A map may contain multiple Common Models. Each Common Model marks the start of a new device. This is useful when a map represents an aggregation of devices. (See the Module/Panel specification for an example of this usage.)

All SunSpec devices must include at least three models:

- The first model is the Common Model, which supplies vendor and model information for the device.
- The second (and subsequent) models will be device specific – for example a meter model, or a meter model followed by a GPS Location model. Any vendor extension models would be included here.
- The final End Model formally marks the end of the Device Specific Models.

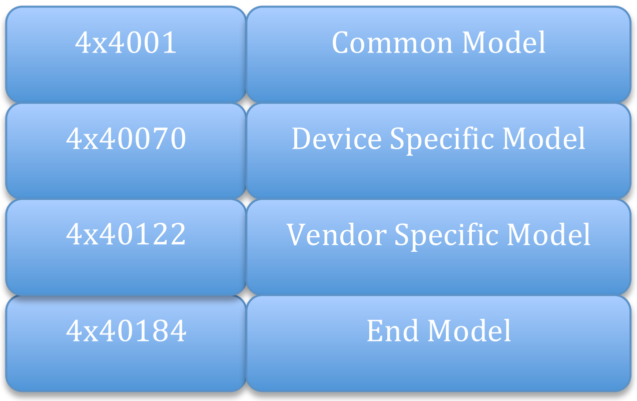

Figure 1 - Example SunSpec Device with two device-specific models

Data Element Optionality

Data element support may be designated as Mandatory, or Optional. Mandatory means the attribute is required to be supported. Optional means the element is not required. Implementations shall return the "Not Supported" value if the element is not used in this case.

Common Model

The following data elements shall be provided to uniquely identify a SunSpec device.

- **SunSpec_ID –** A well-known value that confirms to the client that this device has implemented a collection of SunSpec Alliance models.
- **ID –** A well-known value that uniquely identifies the type of SunSpec model. The IDs are assigned by SunSpec. For the Common Model, this value is "1".
- **L –** A value that is the length of the device model in Modbus 16-bit registers. This value does NOT include the ID or L. I.e. it is the number of registers that follow the length. For the Common Model, this value is 66 but may be 65. All SunSpec models should be of even length. Common Models of length 66 shall contain the SunSpec pad16 value as the final register.
- **Mn** – A unique value that identifies the Manufacturer of this device. Manufacturers are encouraged to register their ID with SunSpec to guarantee uniqueness. SunSpec maintains a database of certified products by Manufacturer. The Mn should be concise and constrained to simple alphanumeric text void of spacing.
- **Md** – A manufacturer specific value that identifies the model of this device. SunSpec maintains a database of certified products by Model. The Md should be concise and constrained to simple alphanumeric text void of spacing.
-
- **Opt** - A manufacturer specific value that identifies any model options for this device.
- **Vr** - A manufacturer specific value that identifies the firmware version of this device.
- **SN** - A manufacturer specific value that uniquely identifies this device within the manufacturer name space.
- **DA** - Protocol specific value to address this device instance. For MODBUS devices, this is the MODBUS device ID.
- **Pad –** SunSpec pad16 register . Must be included for Common Models of length 66. Not included if length is 65.

Note: SunSpec requires that the result of concatenating the three strings **Mn, Md, and SN** returns a globally unique string for all of a manufacture's products. If the optional **Md** value is not implemented, then concatenating the two strings **Mn** and **SN** must be a globally unique string. This string may be used (for example) by logging and uploading functions.

Device Specific Models

The SunSpec Common Model is then followed by one or more device specific models of data values. The Device Specific Model begins like the Common Model.

- **ID –** A well-known value that uniquely identifies the specific type of SunSpec model this is.
- **L –** A value that is the length of the model in registers. The length should be even and may contain a rounding pad register. The length does not include the ID or L register. It is the number of registers that follow.
- Model data fields would follow as required. If the ID is not recognized by the client, the L value may be used to skip over the model data and move to the next model.

End of SunSpec Device Marker

The overall SunSpec Device is punctuated by a termination ID ((0xFFFF)) followed by a length (L) value of zero.

Network Configuration Models

A series of network configuration models are provided to aid in the configuration of various types of interfaces that may be present on a device. A full communications interface description consists of an interface header model and a simple network model, a serial interface model, or a complex model consisting of a link model and zero or more IPv4 or IPv6 models, and zero or more proxy server models. A counter model is also provided to capture basic interface statistics.

Standard Data Formats

The MODUBS specification is not explicit on how to encode numbers other than 16 bit integers. Differences do exist between one manufacturer's implementation and another's. Not all implementations of a specific model will support all of the values defined for that model. Unsupported values are indicated by supplying the "NOT_IMPLEMENTED" type specific value in response. This specification restricts values to the following standard data formats.

int : signed integer value.

uint: unsigned integer value

pad: reserved field, used to round a model to an even number of registers

acc: accumulated value, used for ever increasing values that may roll over

enum: enumerated value, used for status and state

bitfield: a collection of bits, multi-valued alarms or state

ip: internet protocol formatted network address

16 bit Integer Values

Values are stored in big-endian order per the MODBUS specification and consist of a single register. All integer values are documented as signed or unsigned. All signed values are represented using two's-compliment.

Modbus Register	1															
Byte	0								1							
Bits	15	14	13	12	10	11	9	8	7	6	5	4	3	2	1	0

int16 Range: -32767 ... 32767	Not Implemented: 0x8000
uint16 Range: 0 ... 65534	Not Implemented: 0xFFFF
acc16 Range: 0 ... 65535	Not Accumulated: 0x0000
enum16 Range: 0 ... 65534	Not Implemented: 0xFFFF
bitfield16 Range: 0 ... 0x7FFF	Not Implemented: 0xFFFF
pad Range: 0x8000	Always returns 0x8000

NOTE: it is up to the master to detect rollover of accumulated values.

NOTE: if the most significant bit in a bitfield is set, all other bits shall be ignored.

32 bit Integer Values

32 bit integers are stored using two registers in big-endian order

Modbus Register	1		2	
Byte	0	1	2	3
Bits	31 ... 24	23 ... 16	15 ... 8	7 ... 0

int32 Range: -2147483647 ... 2147483647	Not Implemented: 0x80000000
uint32 Range: 0 ... 4294967294	Not Implemented: 0xFFFFFFFF
acc32 Range: 0 ... 4294967295	Not Accumulated: 0x00000000
enum32 Range: 0 ... 4294967294	Not Implemented: 0xFFFFFFFF
bitfield32 Range: 0 ... 0x7FFFFFFF	Not Implemented: 0xFFFFFFFF
ipaddr 32 bit IPv4 address	Not Configured: 0x00000000

NOTE: it is up to the master to detect rollover of accumulated values.

NOTE: if the most significant bit in a bitfield is set, all other bits shall be ignored.

64 bit Integer Values

64 bit integers are stored using four registers in big-endian order.

Modbus Register	1		2	
Byte	0	1	2	3
Bits	63 … 56	55 … 48	47 … 40	39 … 32

Modbus Register	3		4	
Byte	4	5	6	7
Bits	31 … 24	23 … 16	15 … 8	7 … 0

int64 Range: -9223372036854775807 … 9223372036854775807

Not Implemented: 0x8000000000000000

acc64 Range: 0 ... 9223372036854775807 Not Accumulated: 0

NOTE: Only positive values in the int64 range are allowed. Values outside of this range shall be considered invalid.

NOTE: This value shall rollover after the highest positive value in the int64 range (0x7fffffffffffffff). It is up to the reader to detect rollover of accumulated values.

128 Bit Integer Values

128 bit integers are stored using eight registers in big-endian order.

ipv6addr 128 bit IPv6 address Not Configured: 0

String Values

Store variable length string values in a fixed size register range using a NULL (0 value) to terminate or pad the string. For example, up to 16 characters can be stored in 8 contiguous registers as follows

Modbus Register	1		2		3		4		5		6		7		8	
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Character	E	X	A	M	P	L	E	spc	S	T	R	I	N	G	!	NULL

NOT_IMPLEMENTED value: all registers filled with NULL or 0x0000

Floating Point Values

Floating point values are 32 bits and encoded according to the IEEE 754 floating point standard.

Modbus Register	1															
Byte	0								1							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEEE 754	sign	Exponent							Fraction							

Modbus Register	2															
Byte	2								3							
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEEE 754	Fraction least															

float32 Range: see IEEE 754   Not Implemented: 0x7FC00000 (NaN)

Scale Factors

As an alternative to floating point format, values are represented by integer values with a signed scale factor applied. The scale factor explicitly shifts the decimal point to the left (negative value) or the right (positive value). Scale factors may be fixed and specified in the documentation of a value, or may have a variable scale factor associated with it. E.g. A value "Value" may have an associated value "Value_SF" of type "sunssf" that is a 16 bit two's compliment integer.

sunssf signed range: -10 ... 10 Not Implemented: 0x8000 or 0x0000

NOTE: Any value outside of the sunssf range shall be considered to be "Not Implemented".

Defined Units

No units are defined as part of the Common Model. Units are defined as needed by specific models. Where units are shared across models, care will be taken to ensure a common definition of those units.

Assigned Values

Values that are well-known and part of the Common model are defined here.

SunSpec Device ID : 0x53756e53 ("SunS")

All devices are prefixed with this ID to uniquely identify them as SunSpec devices.

SunSpec Model ID : Assigned Model IDs

The following model identifiers have been assigned

00X Common Models

001 - Common Model

002 – Aggregator Model

003 – Secure Dataset Read Model

004 – Secure Dataset Write Model

01X Configuration Models

010 – Interface Configuration Model

011 – Ethernet Link Configuration

012 – IPv4 Protocol Configuration

013 – IPv6 Protocol Configuration

014 – Proxy Server Configuration

015 – Interface Packet Counters

016 – Simplified Network Configuration

017 – Serial Interface Configuration

018 – Cellular Link Configuration

019 – PPP Link Configuration

1XX Inverter Models

101 - Single Phase Inverter Model (Integer+SF)

102 - Split Phase Inverter Model (Integer+SF)

103 - 3 Phase Inverter Model (Integer+SF)

111 - Single Phase Inverter Model (Floats)

112 - Split Phase Inverter Model (Floats)

113 - 3 Phase Inverter Model (Floats)

120 – Inverter Controls Nameplate Ratings

121 – Inverter Controls Basic Settings

122 – Inverter Controls Measurements and Status

123 – Inverter Controls Immediate Controls

124 – Inverter Controls Basic Storage Control

125 – Inverter Controls Pricing Signal

126 – Inverter Controls Static Volt-VAR Arrays

127 – Inverter Controls Frequency Watt Control

128 – Inverter Controls Dynamic Reactive Current

129 – Inverter Controls LVRT Arrays

130 – Inverter Controls HVRT Arrays

131 – Inverter Controls Watt-Power Factor Arrays

132 – Inverter Controls Voltage-Watt Arrays

133 – Inverter Controls Basic Scheduling

134

135 - 138

160 – Inverter Multiple Power Point Tracker (MPPT)

2XX Meter Models

201 - Single Phase Meter Model

202 - Split Phase Meter Model

203 - Wye-Connect Meter Model

204 - Delta-Connect Meter Model

211 - Single Phase Meter Model (float)

212 - Split Phase Meter Model (float)

213 - Wye-Connect Meter Model (float)

214 - Delta-Connect Meter Model (float)

3XX Environmental Models

301 - Base Meteorological Model (DEPRECATED)

302 - Irradiance Model

303 – Back of Module Temperature Model

304 - Inclinometer Model

305 - Location Model

306 - Reference Point Model

307 – Base Meteorological Model (Corrected)

308 – Mini Meteorological Model

4XX String Combiner Models

401 – Basic String Combiner (superseded)

402 – Advanced String Combiner (superseded)

403 – Basic String Combiner 1.1

404 – Advanced String Combiner 1.1

5XX PV Module (Panel) Models

501 – Panel Model (Float)

502 – Panel Model (Integer)

601 - Tracker Model

64XXX Vendor Extension Models

See http://SunSpec.org/AssignedIDs for the latest

65XXX SunSpec Reserved

65535 - End of SunSpec Map (0xFFFF)

C_Status: Assigned Status Codes

Models that support a status code value in the Device Specific Model shall support the following status values.

C_STATUS_NORMAL	0x00000000	:	Operating Normally
C_STATUS_ERROR	0xFFFFFFFE	:	Generic Failure
C_STATUS_UNK	0xFFFFFFFF	:	Status Unknown

Device specific status codes shall be defined in corresponding device model.

MODBUS Register Mappings

All SunSpec devices will have a well-known base address with an alternate base address that may be used if the base address is unavailable for a particular manufacturer's device. The first register in the map shall be the well-known 32 bit identifier SunSpec_ID. This allows for discovery of SunSpec compatible devices. If the base register does not return this value, the alternate base register shall be checked. If this test fails, the device is not SunSpec compatible. The next value indicates the SunSpec model type. The first device model shall always be the SunSpec Common Model. The third value is the length of the device model, followed by the device model values. Additional device model types are concatenated with their corresponding type, length, and values.

SunSpec Base and Alternate Base Register Addresses (decimal values)

Preferred Base Register: 40001

Alternate Base Register: 50001

Alternate Base Register: 00001

Base registers are actual register offsets that start at 1 – not a function code and not to be confused with the Modicon convention, which would represent these as 4x40001 and 4x50001.

To read register 40001, use the hexadecimal offset of 0x9C40 (40,000) on the wire.

If you read beyond the end of the device, an exception may or may not be thrown according to the implementation. If no exception is thrown, then data that comes after the End Model is invalid and should not be used. It is recommended that masters read the common model to determine the contents of the map.

Fixed model sizes must be the size as defined. It is nonconformant to truncate a fixed size model.

REMINDER: This specification only addresses the format of the data. The data can be moved via Modbus/TCP or RTU – or any other media which can move Modbus data. Implementations are responsible for returning measured values as designed. For example, power and voltage readings may not be in sync depending on the product design.

Default Serial Port Settings

All SunSpec compatible devices that support an RS-232 or RS-485 serial port interface shall support the configuration of the serial port baud rate to 9600. In order to support out-of-the-box interoperability it is strongly recommended that the factory settings be (9600,8,N,1).

- Baud Rate – 9600

- Data Bits - 8
- Parity – None
- Stop Bits – 1

N.B. - This recommendation is NOT the same as in the Modbus specification but rather reflects common industry practice and defacto usage. The method used for configuration of the settings is not specified.

Modbus/TCP Settings

It is recommended that the Modbus Unit Identifier should start at address ‘0’. Additional devices should be numbered as 1, 2, etc. with no gaps in the sequence to facilitate device discovery. Default TCP port shall be 502.

SunSpec MODBUS Model Mapping

MODUBS maps are defined using the SunSpec Model Definition XML (SMDX). Please reference the SMDX file. The SMDX files are also compiled into a spreadsheet called PICS.xls for human consumption. The SMDX and PICS are the definitive models. Where discrepancies between the SMDX and PICS exist, the SMDX shall be authoritative.

SunSpec Model Structure

Canonical Structure

The original common model specification defines common features of a SunSpec data model. It requires every model to start with a fixed 2 register header specifying the ID of the model and the length of the data portion of the Modbus registers (in 16 bit words) following this header.

Out of the need to support a variable number of points the following canonical structure has been developed:

Header (Model ID + Length)
Fixed Length Block (may be zero length)
Repeating Block (may be zero length)

The repeating block is composed of one or more repeating *instance*s, where an instance is a sequence of points defined in the model's specification.

Thus the length of the repeating block is constrained to be a whole number multiple of the length of an individual instance.

We can formalize this structure as follows:

Component	**Size (Modbus Registers)**	**Comments**
DID (Model ID)	1	well known value
Model Length	1	length of data portion (fixed length block + total repeated block length)
Fixed Length Data Block	data block length	may be optional (zero

Component	**Size (Modbus Registers)**	**Comments**
		length)
Repeating Block	number of instances * instance length	may be optional (zero length)

All SunSpec data models must adhere to this format. Reading applications, such as data loggers, depend on this structure to properly parse discovered data models. The structure also allows reading applications to ignore models it does not understand yet continue to parse any models it does understand.

Determining the Number of Repeating Instances

In all cases the reading application determines the number of repeating instances by using the following formula:

$n = (l - f) / i$

n	number of repeating instances in the repeating block
l	total length of the model, as provided by the device
f	length of the fixed length block, in registers
i	length of the repeated block, in registers

There may be cases where a model also indicates the number of instances in the repeating section in a point, usually named N. The reading application must not rely on this value but should instead compute the number of instances based on the total indicated length.

Cases Derived From the Canonical Structure

The specified structure gives rise to 3 main classes of models, plus 1 degenerate case. We will consider each case in turn along with corresponding examples of existing data models. Length values will be indicated using the n, i, f, l notation described above.

Fixed Length Data Models

The fixed length block is the easiest to understand. The number of registers is fixed. The repeating section is zero length and thus omitted.

Model ID	*id*
Length	*f*
Fixed Length Block	*f* registers

Examples of data models with this structure are 1: Common Model, 101: Single Phase Inverter Model and 201: Single Phase Meter Model.

Since there is no repeating section the length the length indicated by the device must match the specified length for a given model.

Repeating Models

The repeating model contains no fixed length section.

Model ID	*id*
Length	*n* * *i*
Repeating Block	*n* * *i* registers

Examples of this structure are 302: Irradiance Model, 303: Back of Module Temperature Model and 304: Inclinometer Model.

To determine the number of repeated instances the reading application must know, from the specification, the length of an instance in a given model. The reading application divides the indicated total model length by the known instance length to determine the number of instances present.

For example, if the reading application discovers Model 304 with a length of 18. It knows that the length of the repeating instance in Model 304 is 6, so applying the formula from above it determines there are 3 repeated instances.

3 = (18 – 0) / 6

Combined Model with Both a Fixed Length Block and Variable Length Block

A model may contain both a fixed length section and a repeating section.

Model ID	*id*
Length	*f* + (*n* * *i*)
Fixed Length Block	*n* registers
Repeating Block	*n* * *i* registers

For example, Model 403, the String Combiner, is composed of a fixed length block of 16 registers followed by a repeating block with an instance length of 8 registers. If a device reports the total length is 112 registers then there must be 12 instances in the repeating section.

12 = (112 – 16) / 8

Zero Length Model

The zero length model is included for the sake of completeness. The only zero length model allowed is the SunSpec end marker, with ID 0xFFFF. In the canonical model, both the fixed length block and the repeated block are length zero.

ID	0xFFFF
Length	0

Best Practices for Model Design

We've spent several years developing data models for multiple classes of devices. In addition to the model design described previously, we have come up with the following list of best practices. These are not hard-and-fast rules; in fact, most of them are violated by one or more models. However, going forward we would like all data models to follow these guidelines.

Group scale factors

Scale factors should be grouped together in a block rather than scattered throughout the model. Grouping the scale factors instead of intermixing them with datapoints reduces the number of registers a Modbus master will need to read.

For example, model 403, the String Combiner Model shows the scale factors grouped in the fixed-length section at the start of the mode. Actually, there are two groups, since InDCA_SF and InDCAhr_SF were added when 401 was deprecated.

Model 101, the single phase inverter model, puts the scale factors in the model close to the datapoints they modify. While mitigated by the fact the entire model fits into a single Modbus read, this design requires the scale factors to be read to retrieve all the measurements.

Use Instance Scale Factors When Appropriate

Keep in mind that for a device with many subcomponents represented by instance variables, it may not be appropriate to use the same scale factor for aggregated values and instance values. This mistake was made with the first versions of the String Combiner (401 and 402). The same scale factor was used, for example, for aggregated current and individual string current. This meant the dynamic range of the string current value was limited to the range required by the entire combiner. This issue was corrected in models 403 and 404.

Use PAD to keep 32 and 64 bit alignment

As a consideration to low resource devices model designers should align all 32 and 64 bit values on even numbered offsets. A PAD register should be added at the end of the fixed length block and/or at the end of a repeating block to ensure both add up to an even number of registers. Arrange the other points as needed to place the PAD at the end of the block.

Official SunSpec models should not have more than two PAD registers: one for the fixed length block and one for the repeating block. Vendor models may use PAD to mark ranges of registers as reserved for future use.

Reuse point IDs from other models

Use the same ID as existing models to represent the same physical quantity. This will aid implementation and interpretation of the data model.

Reuse status and events from other models

When two models share similar behavior please re-use existing status and event values. Mark unneeded vales as RESERVED. Compare the inverters models (101, 102 and 103), the Smart Panel models (501 and 502) and the Multiple MPPT Inveter Extension (160) for an example.

SunSpec Inverter Control Procedural Requirements

Error Handling

There is a need to handle errors in a consistent way for conformance testing and interoperability. Applications need a known way to detect errors and handle them.

Unimplemented Registers

Optional registers may not be supported by an implementation.

READ: The value returned is the SunSpec unimplemented value.

WRITE: The written value is ignored. No exception is generated.

A setting is assumed to be a field in a SunSpec model that is defined in the specification as read/write.

Invalid Setting Value

A setting is written that is not a supported value for the implementation.

WRITE: An exception “3” Illegal Data Value shall be returned and processing of the write operation shall terminate. I.e. previous registers may have been written but no subsequent registers shall be written.

NOTE: Such limitations may be documented in the PICS for the implementation. There is no other SunSpec defined method for specifying or determining limits at this time.

Read-Only and Write-Only Registers

Attempting to write to a read-only register or read from a write-only register.

WRITE to R: The written value is ignored. No exception is generated.

READ from W: If the register is supported returns 0 else returns the unsupported value.

Incomplete Enumeration

An incomplete enumeration occurs when the values for a setting are expressed as an enumerated value and some of the possible valid enumerations in the specification are not valid in the implementation.

Modelers should include a register to indicate which enumerations are supported.

READ: Returns the current value

WRITE: Handled same as an Invalid Setting. The value is not changed and an exception “3” shall be generated.

Incomplete Operation

SunSpec only defines READ and WRITE operations. Some operations may not take place in time for a Modbus response. If it desired to return an Exception “5” ACKNOWLEDGE, the model must support a completion register. Otherwise, Exception “5” ACKNOWLEDGE shall not be returned. No exception shall be returned.

Organization of Control Read/Write Values

It is desirable to organize related settings values within the Modbus map in a way that minimizes the number of writes necessary to accomplish updating and activating the settings.   Control operations that rely on a group of settings shall have a register to control the activation (enabling / disabling) of the control.

To facilitate that goal, the following guidelines are recommended:

- Related writable settings field are organized in a contiguous block
- Activation fields are located at the end of the settings block
- Activation fields only operate on the behavior related to that settings block

Procedure for Enabling/Changing Settings Requiring Multiple Write Operations

Implementations must re-enable to control for setting changes to take effect. It is not required or recommended to disable the control to update the settings.

Enable Procedure:

1. All settings are written
2. The activation field is enabled

Change Procedure

1. Changed settings are written. Changes do NOT take effect, even if the activation field is already enabled, until the activation field is enabled.
2. The activation field is enabled

Disable Procedure

1. The activation field is disabled

Certification

The SunSpec Certified™ program is designed to instill buyer confidence in SunSpec member product solutions by establishing objective criteria for verifying compliance of communication interfaces to SunSpec specifications. This program has been in operation since 2012 and 25 companies have been awarded certification.

The SunSpec Alliance Certification Test consists of submitting a SunSpec PICS (Protocol Implementation Conformance Statement) and successfully executing the SunSpec Alliance Compliance test. Both certification and interoperability testing can be completed over the Internet. Certification is currently offered to SunSpec Alliance Contributing members free of charge.

The SunSpec Alliance has a program to accommodate companies that would like to offer SunSpec Certified products but can't join the SunSpec Alliance as members. This program, called the SunSpec Certified Partner Program, is for equipment manufacturers and software developers seeking to have their products SunSpec Certified™ without needing to join the SunSpec Alliance.

The SunSpec Certified mark instills buyer confidence in the quality and interoperability of your company's products. Some additional benefits include:

- Rigorous testing of product interface to SunSpec Alliance standards

- In-depth interoperability testing with other SunSpec Certified solutions
- Right to display SunSpec Certified mark on products
- Right to use of SunSpec Alliance logo in marketing materials
- Product listing in SunSpec Certified catalog
- Inclusion in the bidding process for system builders specifying SunSpec compatibility

Support

The Alliance provides ongoing support for members and non-members utilizing SunSpec Alliance specifications. If you have any questions, concerns, or thoughts, please visit http://www.sunspec.org or email us at info@sunspec.org.