Status: Approved

Version: 1.4

SunSpec Modbus Conformance Test Procedures

SunSpec Specification

Abstract

This document specifies the conformance test procedures for compliance with the requirements specified in the *SunSpec Device Information Model Specification* and the associated specific SunSpec information model specifications.

License Agreement and Copyright Notice

Prepared by the SunSpec Alliance

Website: sunspec.org

Email: info@sunspec.org

Revision History

Version	Date	Comments
1.0	10-9-2021	Initial release
1.1	5-11-2022	Added point write validation test case and promoted status to Approved.
1.2	6-24-2024	Added Broadcast test and Device Address write for RTU, single register write test, single register read test, and rejection tests. Added the allowable delay time of 1000 ms on read after write.
1.3	10-15-2025	Removed the allowable delay time of 1000ms on read after write. (Section 2.4.3)

About the SunSpec Alliance

The SunSpec Alliance is a California-based non-profit trade alliance that develops open information standards to support Distributed Energy Resource (DER) interoperability, cybersecurity, and grid resiliency. With over 180 member organizations from North America, Europe, Asia, Australia, and the Middle East, SunSpec serves manufacturers, software developers, utilities, and service providers across the DER industry.

SunSpec standards enable seamless integration of DER systems at residential, commercial, and utility scales, helping reduce cost, ensure compliance, and accelerate innovation. Membership is open to corporations, non-profits, and individuals.

About the SunSpec Specification Process

SunSpec Alliance specifications are initiated by SunSpec members to establish an industry standard for mutual benefit. Any SunSpec member can propose a technical work item. Given sufficient interest and time to participate, and barring significant objections, a work group is formed, and its charter is approved by the board of directors. The workgroup meets regularly to advance the agenda of the team.

The output of the workgroup is generally in the form of a SunSpec Interoperability Specification. These documents are normative, meaning that there is a matter of conformance required to support interoperability. The revision and associated process of managing these documents is tightly controlled. Other documents are informative, or make some recommendations about best practices, but are not a matter of conformance. Informative documents can be revised more freely and more frequently to improve the quality and quantity of information provided.

SunSpec Interoperability Specifications follow a lifecycle pattern of: DRAFT, TEST, APPROVED, and SUPERSEDED.

For more information or to download a SunSpec Alliance specification, go to https://sunspec.org/about-sunspec-specifications/.

Table of Contents

Index of Tables

Table of Figures

No table of figures entries found.

1 Introduction

This document specifies the common SunSpec Modbus conformance test procedures that are used for all SunSpec Modbus profile conformance testing. These test procedures form the basis of all profile conformance testing. Each SunSpec Modbus profile may add additional testing requirements to the test procedures specified here.

2 Test Procedures

This section provides the test procedures for SunSpec Modbus compliance.

2.1 Test Inputs

SunSpec Modbus compliance testing for a device is based on the functionality supported by the device. The device protocol implementation conformance statement (PICS) is used to specify the SunSpec Modbus functional content.

The device PICS is an Excel workbook that specifies the SunSpec models and points that are supported in the implementation along with the supported value ranges associated with each adjustable point. A SunSpec Modbus PICS template can be obtained from the SunSpec Alliance.

Due to the detailed nature of the PICS content, the SunSpec SVP Dashboard software, available at no cost to SunSpec member companies, can be used to generate an initial PICS for a device implementation. The SVP Dashboard probes the contents of the device through the SunSpec Modbus interface and creates an initial PICS workbook based on the content found in the device. The generated PICS workbook must then be updated to provide details that were not able to be discovered during the initial discovery process.

Only SunSpec standardized models are considered during certification testing. Any additional vendor-specific models must conform to SunSpec information modeling rules to the extent that they do not inhibit discovery and use of the standardized model supported in the device.

SunSpec Modbus certification profiles specify conformance criteria for the specific profile including required models, points, and value ranges that must be supported. Profiles are developed for specific use cases. For example, the SunSpec Modbus profile for IEEE 1547-2018 is used to create an inverter interface that complies with the IEEE 1547-2018 standard. The standard SunSpec Modbus certification testing is performed, and profile requirements are then matched to testing results to determine which profiles the device is in conformance with.

2.2 Test Categories

The SunSpec Modbus conformance tests fall into the following general categories: general device tests, general model tests, curve tests, reversion tests, and Modbus protocol tests.

General device tests are performed once for a device.

General model tests are performed once for each SunSpec model implemented in the device.

Curve tests are performed for each curve-based model implemented in the device.

Reversion tests are performed for each reversion timer implemented in models contained in the device.

Modbus protocol tests are performed once for a device.

2.3 General Device Tests

SunSpec general device tests verify functionality that is required at the device (e.g., inverter, energy storage, tracker product type) level.

The following device tests must be performed.

Test	Description
DEV-1	General Discovery
DEV-2	Model 1 Support

Table 1 - General Device Tests

2.3.1 DEV-1 – General Discovery

This test validates SunSpec model organization and location within the device.

2.3.1.1 Procedure

1. Verify that supported SunSpec models specified in the PICS can be found using the standard SunSpec discovery procedure.
2. Verify the SunSpec Modbus content is located at one of the SunSpec standard Modbus start addresses (0, 40000, or 50000) and begins with the two-register standard SunSpec Modbus start marker.
3. Verify that the SunSpec Modbus end model (ID 65535) is present with a length of 0.

2.3.2 DEV-2 – Model 1 Support

This test validates that SunSpec Model 1 is supported by the device.

2.3.2.1 Procedure

1. Verify that SunSpec Modbus model 1 is present in the implementation and that the content contained in the model matches that declared in the PICS.

2.4 General SunSpec Model Tests

The following tests must be performed for each SunSpec supported standardized model specified in the PICS.

Test	Description
MOD-1	Model Implementation
MOD-2	Model Read
MOD-3	Point Write

Table 2 - General SunSpec Model Tests

For reporting purposes, the label associated with each test instance is the test name concatenated with the model ID for each model tested with a '.' used as the separator. For example, the label for test MOD-1 for model 701 would be MOD-1.701.

2.4.1 MOD-1 – Model Implementation

This test validates that a model specified in the PICS is present in the implementation.

2.4.1.1 Procedure

1. Verify model is discovered and present using the standard SunSpec discovery procedure.
2. Verify model has the correct length for the contents.
3. Verify each SunSpec mandatory point (ID, Length) is implemented.
4. Verify the implemented points match the PICS.
5. Verify all points in the model can be read as a single point. Each value should be verified based on the expected value range. If a range for the point is specified in the PICS, the value should be validated against the range.

2.4.2 MOD-2 – Model Read

This test validates that the model can be read in a single read. Multiple reads are permitted for model lengths that exceed the Modbus maximum read length.

2.4.2.1 Procedure

1. Verify the model can be read in a single read (or several reads if the length exceeds 125 registers) and that the values are in the specified range based on the datatype.

2.4.3 MOD-3 – Point Write

This test validates that all implemented adjustable points in the model can be written individually and as a group. This test also validates that all implemented adjustable points in the model can be written and then subsequently read. The value read must match the value that was written and any additional processing (e.g., updates to AdptCrvRslt) must also be completed at this time.

2.4.3.1 Procedure

1. Verify all implemented adjustable points can be written based on the value range specified in the PICS. The point must be written with the minimum value, maximum value, and three intermediate values. If an adjustable point has fewer possible values, allthe possible values must be tested.
2. Verify all implemented adjustable points can be written and then read. An automated methodology must be used to perform the write and read sequence to accomplish these requirements and to model typical device-to-device communication.
    3. For adjustable enumerated points, verify each supported value in the PICS can be written.

2.5 Curve Tests

Curve tests verify the curve update functionality. These tests are only applicable to models that contain curve or curve-like control functionality. General curve point value updates are verified as part of the general model testing. These curve tests are only applicable to models in the 7xx range.

The following additional tests must be performed for models containing curve functionality.

Test	Description
CRV-1	Curve 1
CRV-2	Apply Settings
CRV-3	Apply Settings Error

Table 3 - General Curve Tests

For reporting purposes, the label associated with each test instance is the test name concatenated with the model ID for each model tested with a '.' used as the separator. For example, the label for test CRV-1 for model 705 would be CRV-1.705.

2.5.1 CRV-1 – Curve 1 Support

This test validates that curve 1, which represents the curve settings, is supported and all the points in curve 1 are implemented as read-only.

2.5.1.1 Procedure

1. Verify curve 1 is supported in the model.
2. Verify that curve 1 is marked as read-only.
3. Verify that the points associated with curve 1 cannot be written.

2.5.2 CRV-2 – Apply Settings

This test validates that the current curve settings can be updated for each of the supported curves.

2.5.2.1 Procedure

For each supported additional curve:

1. For a writable curve, apply a different set of curve settings to the curve being tested than are present in curve 1. For a read-only curve, update curve 1 to contain a different set of settings than the read-only curve being tested.
2. Adopt the updated curve.
3. Verify that the update curve status indicates success.
4. Verify that the values in curve 1 match the values in the curve used for update.
5. Verify that the current curve settings (curve 1) can be updated from each of the additional supported curves.
6. If additional read-only curves are present, verify they are located after the writable curves.

2.5.3 CRV-3 – Apply Settings Error

This test validates the behavior on an adopt curve settings error.

1. Perform an adopt curve request for a curve index that is out of range.
2. Verify that the update curve status indicates failure.
3. Verify that all curve 1 settings are in the same state as before performing the test.

2.6 Reversion Tests

Reversion tests verify the reversion timer functionality. If this functionality is not implemented in a model, the tests are not performed.

The following tests must be performed for each reversion timer that is implemented.

Test	Description
REV-1	Reversion Timeout
REV-2	Reversion Time Update
REV-3	Reversion Cancel

Table 4 - Reversion Tests

For reporting purposes, the label associated with each test instance is the test name concatenated with the model ID for each model tested with a '.' used as the separator. For example, the label for test REV-1 for model 705 would be REV-1.705.

2.6.1 REV-1 – Reversion Timeout

This test validates that the set of required reversion points are implemented for the reversion timer and that the reversion operation is performed on reversion timeout. Some models may define more than one reversion timer, but an implementation may choose to implement any specific reversion timer independently from other reversion timers in the same model.

2.6.1.1 Procedure

For each set of points in the model that support reversion:

1. Verify all reversion points are implemented for the reversion timer specified in the PICS.
2. Set the points to values that are different than current settings.
3. Set the reversion timer.
4. Verify the reversion time remaining point is updated and is within two seconds of the remaining reversion time as the reversion timer counts down. Check at least three times during countdown duration.
5. Verify the reversion timer times out and the timeout is within 2 seconds of the reversion time setting.
6. Verify the reversion settings are applied on reversion timeout.

2.6.2 REV-2 - Reversion Time Update

This test validates the ability to update the reversion timer value during reversion timer operation.

For each set of points in the model that support reversion:

1. Set the reversion points to values that are different than current settings.
2. Set the reversion timer.
3. Verify the reversion time remaining point is updated and is within two seconds of the remaining reversion time as the reversion timer counts down.
4. After at least half of the reversion time has elapsed, rewrite the reversion time register to the full reversion time value.
5. Verify the reversion time remaining point is updated and is within two seconds of the remaining reversion time based on the updated reversion time and elapsed time since the update.
6. Allow the test to run longer than the original reversion timeout period.
7. Verify the reversion timer is still running based on the time update.
8. Verify the reversion settings are not applied on original reversion timeout period.
9. Repeat steps 4 – 8 at least two more times and verify that each test pass is successful.

2.6.3 REV-3 – Reversion Cancel

This test validates the timeout behavior of the reversion timer.

2.6.3.1 Procedure

For each set of points in the model that support reversion:

1. Set the reversion points to values that are different than current settings.
2. Set the reversion timer.
3. Verify the reversion time remaining point is updated and is within two seconds of the remaining reversion time as the reversion timer counts down.
4. After at least half of the reversion time has elapsed, set the reversion time to 0.
5. Verify the reversion time remaining point is updated to 0.
6. Allow the test to run longer than the original reversion timeout period.
7. Verify the reversion settings are not applied on original reversion timeout period.

2.7 Modbus Protocol Tests

Modbus protocol tests are performed to verify correct operation at the Modbus protocol layer.

Modbus RTU

The following tests must be performed for devices supporting a Modbus RTU interface.

Test	Description
RTU-1	RTU Interface
RTU-2	Baud Rate
RTU-3	Partial Request
RTU-4	Broadcast Test
RTU-5	Device Address Write

Table 5 – Modbus RTU Tests

2.7.1 RTU-1 – RTU Interface

This test validates the support of a Modbus RTU interface. This test can be run concurrently with other required conformance tests.

2.7.1.1 Procedure

1. Perform all the conformance tests over the Modbus RTU interface.

2.7.2 RTU-2 – Baud Rate

This test validates each of the additional supported baud rates for the Modbus RTU interface.1. Perform at least three of the specified conformance test cases for each supported baudrate.

2.7.3 RTU-3 – Partial Request

This test validates that a device can recover from an incomplete Modbus request message and ignore it.

2.7.3.1 Procedure

1. Send an incomplete partial Modbus request.
2. Send a different complete Modbus request.
3. Verify the successful response to the second Modbus request.

2.7.4 RTU-4 – Broadcast Test

This test verifies the broadcast functionality works on the device as described in the Modbus standard.

2.7.4.1 Procedure

1. Send a write request from the testing client to the SunSpec Server with a Unit ID set to 0.
2. Confirm the device does not respond to the request.
3. Read the point in the device to verify the point has been updated with the ‘write’ value from step 1.

2.7.5 RTU-5 – Device Address Write

This test verifies the device has the capability to change the device address/unit ID between 1 and 247.

2.7.5.1 Procedure

1. Write five different values between 1 and 247 to the Device Address in Model 1. For each value, read and write at least 1 SunSpec register using the updated Unit ID (Device Address). .
2. Verify for each of the 5 Device Addresses in Step 1, the device does not response to either read or write requests for 3 other Unit ID values.
3. Verify the device rejects writing zero to the Device Address field.

Modbus TCP

The following tests must be performed for devices supporting a Modbus TCP interface.

Test	Description
TCP-1	TCP Interface
TCP-2	Partial Request
TCP-3	Multiple TCP Packets

Table 6 – Modbus TCP Tests

2.7.6 TCP-1 – TCP Interface

This test validates the support of a Modbus TCP interface.

2.7.6.1 Procedure

1. Perform all the conformance tests over the Modbus TCP interface.

2.7.7 TCP-2 – Partial Request

This test validates that a device can recover from an incomplete Modbus request message.

2.7.7.1 Procedure

1. Send an incomplete partial Modbus request.
2. Send a different complete Modbus request.
3. Verify the successful response to the second Modbus request.

2.7.8 TCP-3 – Multiple TCP Packets

This test validates that a device can support a Modbus request that is divided across multiple TCP frames.

2.7.8.1 Procedure

1. Send a Modbus TCP request that is broken across two TCP frames.
2. Verify the successful response to the Modbus TCP request.

General Modbus Tests

2.7.9 MB-1 – Modbus Single/Multiple Register Write

This test validates the device under test can perform multiple write (Function Code: 16) and single write (Function Code: 06) operations.

2.7.9.1 Procedure

1. Select two consecutive RW points to be written.
2. Send a multiple write request using function code 16 to write the new values to the points.
3. Verify both the points are written successfully.
4. Send a single write request using function code 6 on the two points one by one.
5. Verify the points are written successfully.

2.7.10 MB-2 – Modbus Single Register Read

This test validates the device under test can perform single register reads which are allowed in function code 3.

2.7.10.1 Procedure

1. Select a register within the modbus map of the device under test.
2. Perform a single register read using function code 3 and register count 1.
3. Verify the register has been read correctly.
4. Repeat steps 1-3 for two other registers.

Note: Some devices may reject a read of a register if it's a part of a value longer than 16 bits. This may be done to avoid reading a part of a value and getting incomplete information. This is not considered non-compliant. In such cases, a 16-bit value should be read.

2.8 Exception Generation Tests

These tests are performed to verify the device under test performs exception handling correctly.

Test	Description
EXC-1	Invalid Value
EXC-2	Writing a Read-Only Register
EXC-3	Illegal Function Code

Table 7 – Exception Generation Tests

2.8.1 EXC-1 – Invalid Value

This test verifies the device throws an exception when an invalid value is written to a point.

2.8.1.1 Procedure

1. For each model, write an invalid value to a RW point. For enum points, write a value outside of the defined enumerations.
2. Verify the device responds with an exception code of 2, 3 or 4.
3. Read the point to verify it was not written.

2.8.2 EXC-2 – Writing a Read-Only Register

This test verifies the device throws an exception when a write request is sent to a read-only register.

2.8.2.1 Procedure

1. Identify three or more read-only registers and send a write request to change the values in them.
2. Verify the device responds with an exception code of 2, 3 or 4.
3. Read each point to verify it was not written in the previous step.

2.8.3 EXC-3 – Illegal Function Code

This test verifies the device correctly throws Exception 1 when it receives an illegal function code.

2.8.3.1 Procedure

1. Select a RW register implemented in the device and send a request with a function code that doesn't exist in the Modbus specification (e.g., 50).
2. Verify the device responds with Modbus Exception 1.