Document #:	X99999
Status:	Draft
Version:	1.0

SunSpec Device Web Service API Specification

SunSpec Specification

Abstract

This specification presents a standardized way for SunSpec devices to communicate with remote data centers. The standard specifies Representational State Transfer (REST) interface semantics to read and write device data that conforms to the SunSpec Device Information Models. The intent is that software developers can use the specification to implement both the server (SunSpec device) and client (data center) parts of a web service to communicate using the Application Programming Interface (API) defined in this specification. The API exposes the Device Information Models implemented on a device as a set of resources addressable using hierarchical endpoints, or Uniform Resource Locators (URLs). The JSON-based, RESTful web service is a widely-used and proven technology that permits implementation in a wide range of production and operational environments.

License Agreement and Copyright Notice

Prepared by the SunSpec Alliance

4040 Moorpark Avenue, Suite 110

San Jose, CA 95117

Website: sunspec.org

Email: info@sunspec.org

Revision History

Version	Date	Comments
1.0	8-1-2019	Initial release

About the SunSpec Alliance

The SunSpec Alliance is a trade alliance of developers, manufacturers, operators, and service providers together pursuing open information standards for the distributed energy industry. SunSpec standards address most operational aspects of PV, storage, and other distributed energy power plants on the smart grid, including residential, commercial, and utility-scale systems, thus reducing cost, promoting innovation, and accelerating industry growth.

Over 100 organizations are members of the SunSpec Alliance, including global leaders from Asia, Europe, and North America. Membership is open to corporations, non-profits, and individuals. For more information about the SunSpec Alliance, or to download SunSpec specifications at no charge, visit sunspec.org.

Field Code Changed

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 workgroup 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 considered to be 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 recommendation with regard to 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/.

Field Code Changed

Table of Contents

Index of Tables

1 Introduction

This specification presents a Representational State Transfer (REST)-ful web service that standardizes reading and writing information on devices that implement SunSpec Device Information Models.

A REST Application Programming Interface (API) is defined that represents the information model as a set of hierarchically related resources that can be referenced individually or as a collection using a Uniform Resource Locator (URL). The API further defines the HTTP access methods supported by each resource, the method parameters, and request and response message encoding.

The objective of this specification is to standardize the interface to facilitate communication between remote devices and data centers that implement SunSpec Device Information Models. The interface, based on the widely-adopted REST architectural style, relies on proven industry technologies. This facilitates implementation, deployment, and operation over a wide range of network and installation environments.

This specification applies to device manufacturers and software developers implementing a device information model web service. Both client and server implementations, including the data model and read/write operations on web service resources, can be designed from this specification and interoperate. By defining the standard at the API level, developers are unconstrained in hardware or system software choices.

1.1 Document Organization

Chapter 2 lists standards documents that are normative references used in this document.

Chapter 3 provides an overview of the RESTful web service.

Chapter 4 describes the REST API used to access Device Information Model resources, including examples that show the semantics and intended use of the API.

1.2 Terminology

Device Information Model	Device Information Models are used to structure device data for exchange across communications interfaces. The models provide a mechanism for specifying the data set supported by a device, which consists of a set of standardized definition elements.
JSON	JavaScript Object Notation is a lightweight format used for data exchange. The canonical form of Device Information Model definitions is specified using JSON encoding. This document specifies JSON encoding for Device Information Model instances.
Model	A Device Information Model *model* element defines a logical grouping of *points*. Each *model* has a unique model ID.
MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL	The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification, are to be interpreted as described in IETF RFC 2119.
Point	A Device Information Model *point* element defines a device data point and has a value.
Resource	A resource is a model entity or collection of entities addressable using a URI. A resource can process model information requests and send responses containing model information.
REST, RESTful	A RESTful web service is an architectural style that uses Representational State Transfer (REST) for web applications to access web service resources. REST specifies HTTP access methods that include GET, PUT, POST, and DELETE.
URI	From RFC 3986, a Universal Resource Identifier uniquely identifies a resource or a collection of resources. URIs are used in this specification to identify a resource. For example, /model. See URL
URL	From RFC 3986. A subset of URI. Identifies the resource and the mechanism for locating the resource. For example, http(s)://models. See URI.
UTF-8	UTF-8 is a method for encoding Unicode characters using 8-bit sequences that can include one or more bytes.

1.3 API Description Notation

The curly braces (`{ }`) in URIs indicate that a value must be substituted for the named variable.

For each method defined for the API, the method is described in terms of the following fields:

Request	*Request* consists of the REST method and resource URL and shows the valid resource request syntax.
Request Header	*Request Header* lists the required and optional HTTP header keys supported by the method.
Request Parameters	*Request Parameters* lists required and optional HTTP parameters supported by the method.
Response	If applicable, *Response* describes the data returned by the method.
Status Codes	*Status Codes* lists the HTTP status codes supported by the method.
Example	Non-normative, the example is intended only to show the semantics and intended use of the method. The example(s) are representative REST method calls. The *{myhostdevice:1313}* part of the example URL must be replaced with the actual device IP address and port number.

2 Normative References

[ISO.8601.2004] International Organization for Standardization, "Data elements and interchange formats – Information interchange -- Representation of dates and times," 2004.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, DOI 10.17487/RFC1738, December 1994, <https://www.rfc-editor.org/info/rfc1738>.

[RFC1808] Fielding, R., "Relative Uniform Resource Locators", RFC 1808, DOI 10.17487/RFC1808, June 1995, <https://www.rfc-editor.org/info/rfc1808>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, DOI 10.17487/RFC2396, August 1998, <https://www.rfc-editor.org/info/rfc2396>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, DOI 10.17487/RFC2617, June 1999, <https://www.rfc-editor.org/info/rfc2617>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, DOI 10.17487/RFC2279, January 1998, <https://www.rfc-editor.org/info/rfc2279>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/info/rfc5789>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

SunSpec X99999; SunSpec Device Information Model Specification, version 0.1, May 2019.

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

Field Code Changed

3 Overview

The following table lists the resources that SHALL be supported by the Device Information Model API:

			HTTP Method			Auth
Resource URI	Description	Required	GET	POST	PATCH	Req'd
/models	Get models content.	YES	YES			YES
/models/{modelId}/instances/{index}	Get content for specified model and instance.	YES	YES		YES	YES
/models/report	Schedule report.	NO		YES		YES

Table 1~~2~~: REST API Summary

For each resource, the supported HTTP methods and authentication requirements are listed. Conforming implementations SHALL implement all of the interfaces, HTTP methods, and authentication requirements shown in the table.

3.1 Protocol

3.1.1 Hypertext Transport Protocol (HTTP)

The RESTful interface specified in this document is implemented using HTTP. The HTTP-defined verbs, GET, POST, and PATCH are the basic operations between a web service client and server.

The interface SHALL expose Device Information Models using HTTP, version 1.1.

The interface SHALL be implemented using HTTP over TCP/IP.

3.1.2 URIs

Universal Resource Identifiers (URIs) identify resources. The service SHALL use `/models` as the root resource that represents the supported Device Information Models. Sub-resources within the `/models` resource SHALL be represented hierarchically in the URI. For example, `v1/models/report`.

URIs SHALL include the API version. For example, `v1/models`. The current version of this API is `v1`.

URIs SHALL be encoded as described in RFC 1738, Section 2.2, and RFC 2396, Section 2.0.

Field Code Changed

Field Code Changed

3.1.3 Supported Methods

The API SHALL use the following HTTP method subset:

GET

The GET method reads model information.

If the endpoint represents a collection, the GET request SHALL return a list of members of the collection. Otherwise, the endpoint is a terminal node in the resource hierarchy and the GET request SHALL get the current data associated with the resource.

POST

The POST method creates a resource. The API defines a POST method to schedule reports.

PATCH

The PATCH method updates model information for an existing resource. A PATCH request SHALL update only the specified data for the resource.

See the **Error! Reference source not found.**~~REST API Summary~~ section for the list of methods supported for each resource.

3.1.4 Data Encoding

Resource data representation SHALL be based on the JSON schema.

HTTP requests and responses containing a body SHALL use the `application/json` media type in the HTTP header.

3.1.5 Versioning

The protocol version SHALL be indicated in the URI. The root URI for this version of the model SHALL be `v1/model`.

3.1.6 Error Reporting

HTTP status codes, as described in RFC 2616, Section 14, SHALL be returned in response to each HTTP request to indicate the success or failure of an API request to indicate the success or failure of the request.

Field Code Changed

A successful read (GET), write (POST), or update (PATCH) request SHALL return an HTTP status code of 200.

If a request fails, the server SHALL return an error HTTP status code and the response message body SHALL include the following information:

Field	Description
`errCode`	Error code that can be used programmatically.
`errMessage`	Concise error message suitable for display.
`errReason`	Detailed error information describing the cause of the error in addition to the HTTP status code cause.
`debug`	(Optional) Very detailed debug information.
`TBD`	Possibly add request information such as target resource, target values, and HTTP method.

Table 2~~1~~: Error Response Body

Example error response body:

```
{"errCode": "VAL-001",
 "errMessage": "Value is out of range",
 "errReason": "Maximum voltage range for Volt-Var curve is 160% of Vref",
 "debug": null,

}
```

3.2 Resources

The `/models` resource SHALL be the root node and initial entry point into a server and SHALL represent the Device Information Models published by the server.

3.3 Data Model Schema

The data model schema of the `/models` resource SHALL conform to the Device Information Model implementation described in the SunSpec Device Information Model Specification.

3.4 Security

Security MAY be implemented in the HTTP protocol using HTTPS (HTTP over TLS), and in the application layer using HTTP Basic Authentication.

Devices MAY implement the HTTP protocol without TLS, although HTTPS is recommended for communication over an external network.

3.4.1 Authentication

If security is enabled, the API requires HTTP Basic Authentication.

With each HTTP request, the client SHALL provide a base64-encoded username and password to the host device. API requests without authentication SHALL fail.

3.4.2 TLS

If security is enabled, all API requests MUST be made using HTTPS and requests using HTTP SHALL return a failure status code.

4 Model API Specification

This section specifies a RESTful web services API for accessing Device Information Model resources implemented in a SunSpec device. Applications can use the API to authenticate and exchange data with a device or to get reports from a device, using read and write methods on resource endpoints.

The /models resource is the highest level of the Device Information Model hierarchy implemented in the device and includes all device model instance identifiers and data points.

The API consists of the following resources:

- /models
- /models/{modelId}/instances/{index}
- /models/report

4.1 Get Model Information

The API defines two interfaces for getting device data:

- `/models` SHALL return a list of all models supported by the device.
- `/models/{modelId}/instances/{modelIndex}` SHALL return the single device information model specified in the request.

4.1.1 GET Models Collection

The `/models` resource GET method returns Device Information Model instance information for all models implemented by the device. A *summary* parameter SHALL be provided to specify the level of detail of the information returned. Data point name-value pairs SHALL be returned as JSON-formatted data.

Request

```
GET http://{ipAddress:port}/v1/models
```

Request Header

Key	Value
Content-Type	`application/json; charset=utf-8`
Authorization	`Basic` *<base64encodedData>* Example: `Basic dm9yZGVsOnZvcmRlbA==`

Request Parameters

Key	Description	Required
summary	Flag to indicate whether to return the full model information set or a summary of the information. `true` = Returns the full model information set for all models implemented by the device. `false` (default) = Returns a summary of the model information set for all models implemented by the device: ▪ model name ▪ model ID ▪ number of instances of the model	No

Response

A summary response requested using the summary parameter SHALL return JSON-formatted summary data for the collection of all models supported by the device:

- name
- ID
- Count

A full data request SHALL return all JSON-formatted data for the collection of all models supported by the device:

- name
- ID
- full model data

Status Codes

Status Code	Description
200	Successfully returned the full or summary model content for all models implemented in the device.
401	Authentication error. Invalid username and password.

Example: Request Summary Information

Request

```
GET https://myhostdevice:1313/v1/models?summary=true
```

Response

```
{"models": [
   {"name": "Common",
    "ID": 1,
    "count": 1
   },
   {"name": "VoltVar",
    "ID": 174,
    "count": 1
   }
   ]
}
```

Example: Request Full Model Information

Request

```
GET https://myhostdevice:1313/v1/models
```

Response

```
{"models": [
   {"name": "Common",
    "ID": 1,
    … full model information elided …
   },
   {"name": "VoltVar",
    "ID": 174,
    … full model information elided …
   }
      … additional models elided …
   ]
}
```

4.1.2 GET Models Instance

The GET instance method SHALL get the full data content for the specified model instance of the specified model. The instance `{index}` value SHALL be one-based. The *points* parameter MAY be used to filter the points returned in the response.

Request

```
GET http://{ipAddress:port}/v1/models/{modelId}/instances/{modelIndex}
```

Request Header

Key	Value
Content-Type	`application/json; charset=utf-8`
Authorization	`Basic` *`<base64encodedData>`* Example: `Basic dm9yZGVsOnZvcmRlbA==`

Request Parameters

Key	Description	Required
points	Filter the points returned in the response. The points are specified as a comma-separated list of model point names, and only the specified points that are implemented are returned in the response. Only points or point groups in the top-level model group can be selected. If points in other groups are needed the entire model should be read.	No

Response

```
{"models": [
   {"name": <modelName>,
    "ID": <value>,
    … full model information elided …
   }
}
```

Status Codes

Status Code	Description
200	Successfully returned the full model content for the specified model.
401	Authentication error. Invalid username and password.

Example: Request All Points

Request all points

```
GET https://myhostdevice:1313/v1/models/174/instances/1
```

Response

```
{"models": [
   {"name": "VoltVar",
    "ID": 174,
    … full model information elided …
   }
}
```

Example: Request Specific Points in the DERmeasurementAC Model

Request

```
GET https://myhostdevice:1313/v1/models/701/instances/1?points=W,Var
```

Response

```
{"models": [
   {"name": "DERMeasurementAC",
    "ID": 701,
    "W": 2300,
    "Var": 145
   }
}
```

Val has no signif (reporting)

4.2 Update Model Information

A single PATCH method SHALL be provided to update the Device Information Model.

4.2.1 PATCH Models Instance

The PATCH method SHALL update all or partial Device Information Model data for the specified model instance of the specified model. All model updates are considered partial updates to an existing resource.

Only one model instance can be updated at a time. The body of the PATCH request SHALL be encoded as a model populated with the points to be updated. Any point that is not present in the model provided in the body SHALL NOT be affected by the update.

If a point that is marked as read-only or unimplemented is part of the update, the request SHALL fail and return an HTTP status code of 405 `Method Not Allowed`.

If a {`modelIndex`} is not provided in the URI and there is more than one instance of the model id in the device, the request SHALL fail and return an HTTP status code of 400 `Bad Request`.

If a requested update cannot be performed, the request SHALL fail with no updates performed on the information model.

Model data that are not being updated SHALL be represented as empty objects.

Request

```
PATCH http://{ipAddress:port}/v1/models/{modelId}/instances/{modelIndex}
```

Request Header

Key	Value
`Content-Type`	`application/json; charset=utf-8`
`Authorization`	`Basic` *`<base64encodedData>`* Example: `Basic dm9yZGVsOnZvcmRlbA==`

Request Parameters

None

Request Body

The request body SHALL be a JSON information model structure that specifies the model parameters to be updated and the updated values.

Response

The response SHALL be the full Device Information Model for the specified model and index, in JSON format.

Status Codes

Status Code	Description
200	Successful operation with the full representation of the updated model returned in the body of the response.
400	Bad Request. Request could not be processed.
401	Authentication error. Invalid username and password.
405	Method Not Allowed. Request not supported by the target resource.

Example

Update points in a volt-var curve where points have a scale factor of -1, assuming the following scale factor for the curve:

```
      "Pt": [
         {"V": 950, "Var": 500},
         {"V": 990, "Var": 500},
         {"V": 1010, "Var": -1000},
         {"V": 1050, "Var": -1000},
      ]
```

Request

```
PATCH https://myhostdevice:1313/v1/models/174/instances/1
```

Request Body

```
{"ID: 174 {
   "Crv": [
     {},
     {
       "Pt": [
         {"V": 950, "Var": 1000},
         {"V": 990, "Var": 1000},
         {"V": 1010, "Var": -1000},
         {"V": 1050, "Var": -1000},
         ]
       }
     ]
  }
}
```
https://myhostdevice:1313/v1/models/174/instances/1 -d instance=174

Field Code Changed

Response

```
{"ID: 174 {
   "Crv": [
     {},
     {
       "Pt": [
         {"V": 950, "Var": 1000},
         {"V": 990, "Var": 1000},
         {"V": 1010, "Var": -1000},
         {"V": 1050, "Var": -1000},
         ]
       }
     ]
  }
}
```

4.3 Schedule Model Information Report

An implementation MAY support a request to schedule periodic model information reporting using the `/models/report` API.

4.3.1 POST Models Report

The POST method SHALL schedule periodic information model reporting.

Request

```
POST http://{ipAddress:port}/v1/models/report
```

Request Header

Key	Value
`Content-Type`	`application/json; charset=utf-8`
`Authorization`	`Basic` *`<base64encodedData>`* Example: `Basic dm9yZGVsOnZvcmRlbA==`

Request Parameters

None

Request Body

Key	Description	Required
endpoint	Endpoint URI that the periodic report should be posted to.	YES
period	The period in seconds between report postings.	YES
models	The list of information models to report, specifying the parameters to report. The parameter values are not significant in this request and SHALL be set to zero by convention.	YES

Response

The response body SHALL be the requested list of models and model parameters to report.

Status Codes

Status Code	Description
201	Report successfully scheduled.
401	Authentication error. Invalid username and password.

Example

Request a periodic report of the DERMeasurementAC model power and var parameters.

Request

```
POST https://myhostdevice:1313/v1/models/report
```

Request Body

```
{
   "endpoint":"https://destinationhost/report",
   "period":"300",
   "models":[
      {"name": "DERMeasurementAC",
       "ID": 701,
       "W": 0,
       "Var": 0
   ]
}
```

Response

```
{
   "endpoint":"https://destinationhost/report",
   "period":"300",
   "models":[
      {"name": "DERMeasurementAC",
       "ID": 701,
       "W": 0,
       "Var": 0
   ]
}
```