

SEMI A4.1-0821

SPECIFICATION FOR HTTP JSON PROTOCOL IMPLEMENTATION FOR TESTER EVENT MESSAGING FOR SEMICONDUCTORS (TEMS)

This Standard was technically approved by the Automation Technology Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on May 27, 2021. Available at www.semiviews.org and www.semi.org in August 2021.

Table of Contents

1 Purpose	2
2 Scope	2
3 Limitations	2
4 Referenced Standards and Documents	2
5 Terminology	2
6 Message Coding	2
6.1 Coding System	2
6.2 Reasons for Using JSON	2
6.3 JSON Syntax Rules	3
6.4 JSON Data Syntax	3
6.5 Data Type	3
6.6 Fields Rules	7
6.7 Non-Mandatory Fields	7
6.8 Mandatory Fields Without Value Associated	7
7 HTTP Code and Actions	8
Related Information 1	11

1 Purpose

1.1 The purpose of this Specification is to provide an implementation of SEMI A4, Tester Event Messaging for Semiconductors (TEMS), by using HTTP JSON protocol.

2 Scope

2.1 This Specification describes the way to implement the TEMS protocol by using HTTP JSON.

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 None.

4 Referenced Standards and Documents

4.1 *SEMI Standards and Safety Guidelines*

SEMI A4 — Specification for the Automated Test Equipment Tester Event Messaging for Semiconductors (TEMS)

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *DAS* — data application server

5.1.2 *HTTP* — Hypertext Transfer Protocol

5.1.3 *JSON* — JavaScript Object Notation

5.1.4 *STDF* — Standard Test Data Format

5.1.5 *TCS* — tester control system

5.1.6 *TEMS* — Tester Event Messaging for Semiconductors

5.2 *Definitions*

5.2.1 *data application server (DAS)* — software acting at the TEMS message server.

5.2.2 *heartbeat* — periodic status message sent at a controlled rate.

5.2.3 *message broker* — wide area message distribution system.

5.2.4 *tester control system (TCS)* — TEMS software running on a tester.

6 Message Coding

6.1 *Coding System*

6.1.1 TEMS sends messages in JSON format:

- JSON stands for JavaScript Object Notation.
- JSON is a lightweight data interchange format.
- JSON is self-describing and easy to understand.
- JSON is language independent.

6.2 *Reasons for Using JSON*

6.2.1 The exchange of data between a TCS and a DAS is only through text. JSON is text and a JavaScript object can be converted into JSON; therefore, it can be sent to the DAS.

6.2.2 Any JSON received from the TCS can be converted into a language object.

6.3 JSON Syntax Rules

- JSON syntax is derived from JavaScript object notation syntax.
- Data is name-value pairs.
- Data is separated by commas.
- Curly braces hold objects.
- Square brackets hold arrays.

6.4 JSON Data Syntax

6.4.1 JSON data is written as name-value pair, which consists of a filled name (in double quotes), followed by a colon and then, by a value.

6.4.1.1 JSON data example:

`"name": "John"`

6.5 Data Type

6.5.1 The figure below describes the type of data supported by JSON.

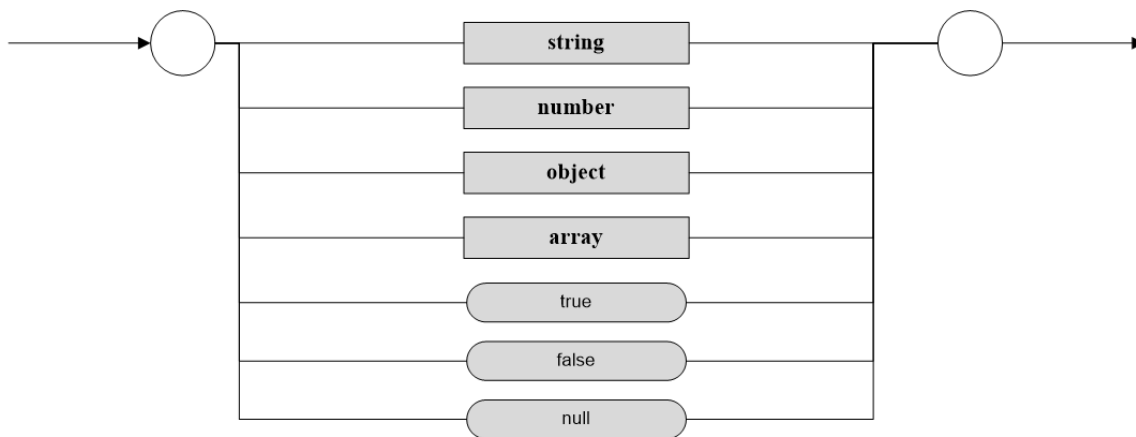


Figure 1
Data Type Supported by JSON

6.5.2 String Type

6.5.2.1 A string will respect the following rules:

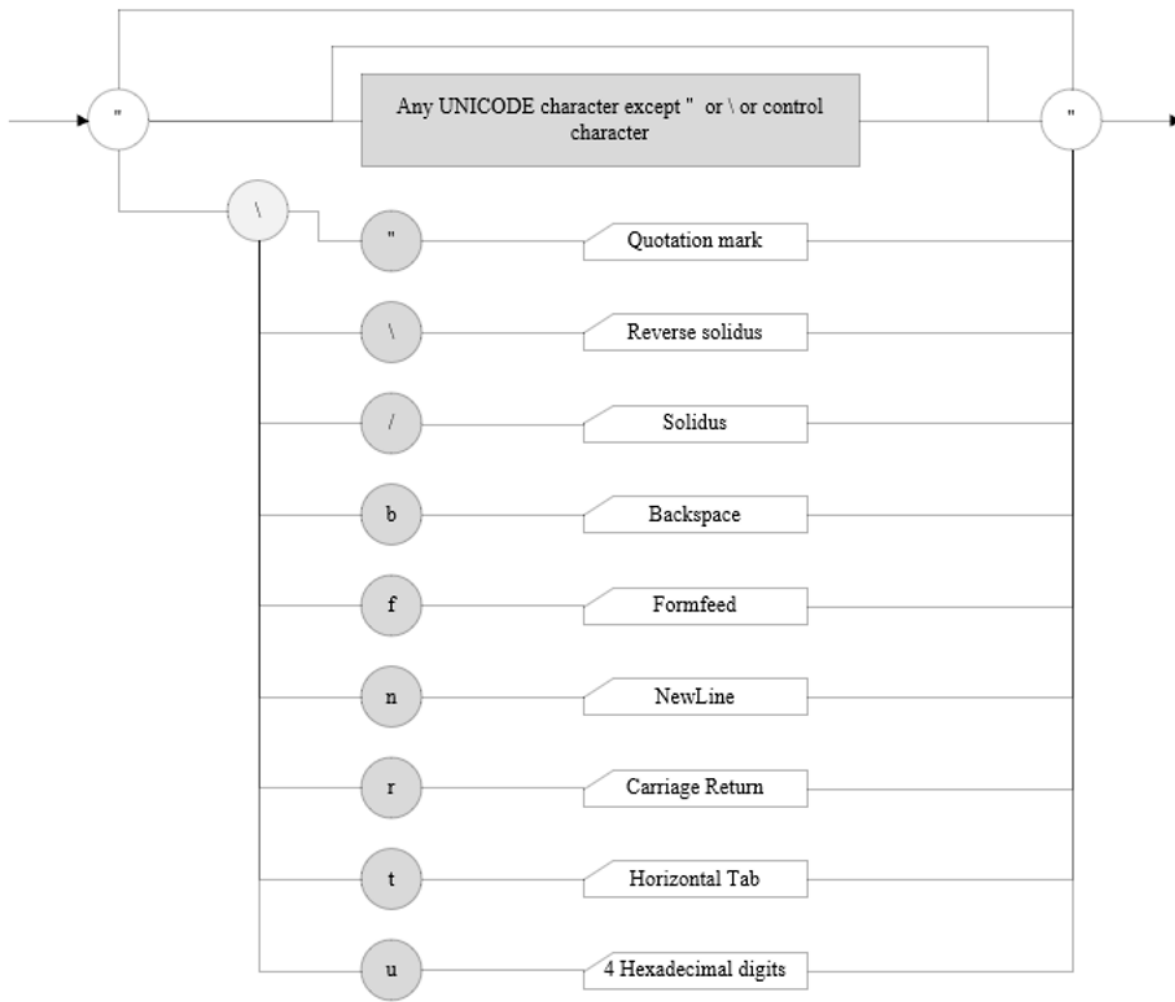


Figure 2
String Format

6.5.3 DateTime Type

6.5.3.1 Datetime is represented as a string. Date and Time-Stamp are codified according to ISO 8601. For example, the field EVENT_DDTM is coded as follows:

"EVENT_DDTM":"2018-06-07T03:25:29.6538690Z"

6.5.4 For each field, minimal precision is required. The different possibilities are:

- us: The value will contain at least the microsecond value.
- ms: The value will contain at least the millisecond value.
- s: The value will contain at least the second value.
- d: The value will contain at least the day value.

6.5.5 Object Type

6.5.5.1 An object follows the figure below.

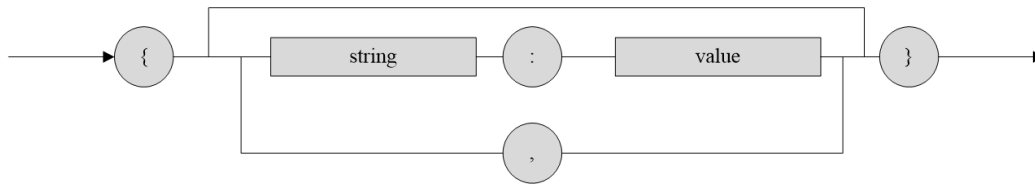


Figure 3
Object Format

6.5.5.2 An object can be defined inside another object. When it says 'Object' in a message or response, it means that it can be any JSON object described above.

6.5.5.3 Object example:

```
"employee":
{
  "name" : "Keith",
  "age"  : 61,
  "city" : "TonyPandy"
}
```

6.5.6 Boolean Values

6.5.6.1 For Boolean, the value will be without quotes and will be lowercase. It will follow the figure below.

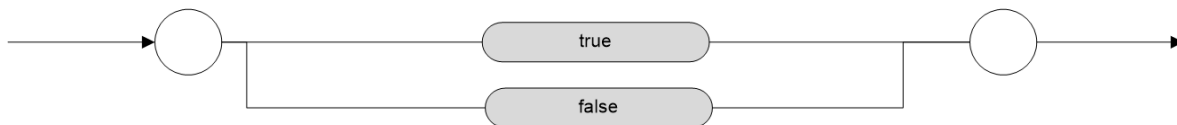


Figure 4
Boolean Format

6.5.6.2 Boolean example:

```
"Walshe": true
```

6.5.7 Null Values

6.5.7.1 For null the value will be without quotes and will be lowercase. It will follow the schema shown below:



Figure 5
Null Format

6.5.7.2 Null example:

```
"MiddleName": null
```

6.5.8 Number for TEMS

6.5.8.1 TEMS respects the coding defined in the JavaScript specification. Unlike other programming languages, JSON does not define types of numbers, such as integers, short, long floating-point, etc.

6.5.8.2 The numbers follow the IEEE 750 standard.

6.5.8.3 Any numbers contained in the messages are shown below:

Table 1 Number Format Under TEMS

Type	Precision
Integers (numbers without a period or exponent notation)	Accurate up to 15 digits
Floating (number with a period)	Maximum of decimals is 17

6.5.8.4 The codification of any number is shown in the figure below:

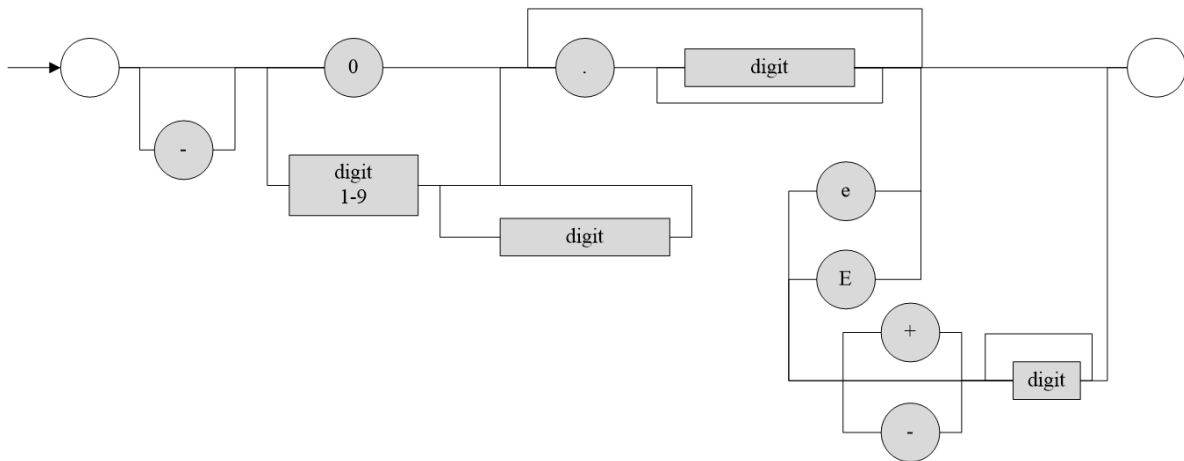


Figure 6
Number Format

6.5.8.5 For TEMS, the data type used is float. In this case, there is no differentiation between float and double.

6.5.8.6 Number example:

"mynumber": 15.58969

6.5.9 Array Type

6.5.9.1 Any array inside TEMS is shown in the figure below.

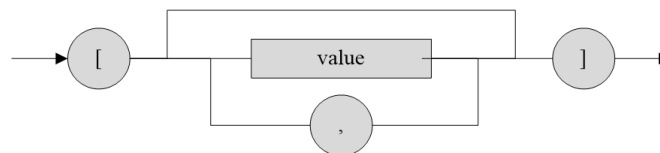


Figure 7
Array Type

6.5.9.2 Array example:

```
"first_names":["Leo","Thomas","Julien"]
```

6.5.10 Array of Objects

6.5.10.1 An array of objects can be defined.

6.5.10.2 Array of Objects example:

```
"employees":  
[  
  { "name" : "Leo",      "age" : 3,  "city" : "Aubagne"  
    { "name" : "Thomas", "age" : 45, "city" : "Marseille"  
    { "name" : "Julien", "age" : 48, "city" : "San Etienne"  
  ]
```

6.6 Fields Rules

6.6.1 Rules concerning the field names.

6.6.1.1 Fields names shall only use letters, digits, underscore (“_”), dot (“.”) and hyphen (“-”) from the first 128 characters of UTF-8.

6.6.1.2 Field names described in SEMI A4.

6.6.1.2.1 Letters in the field name shall be uppercase.

6.6.1.3 Custom field names.

6.6.1.3.1 Any field that is not defined in SEMI A4, but is required by the TCS or DAS shall be a custom field.

6.6.1.3.2 Letters in the custom field name shall be lowercase.

6.7 Non-Mandatory Fields

6.7.1 If a field is not mandatory and it does not have a value, the field can be removed from the message.

6.8 Mandatory Fields Without Value Associated

6.8.1 For a string it will be represented by a null value or an empty string.

6.8.2 For a number it will be represented by a null value.

7 HTTP Code and Actions

7.1 This section describes which actions are taken by the TCS, depending on the HTTP code provided by the DAS.

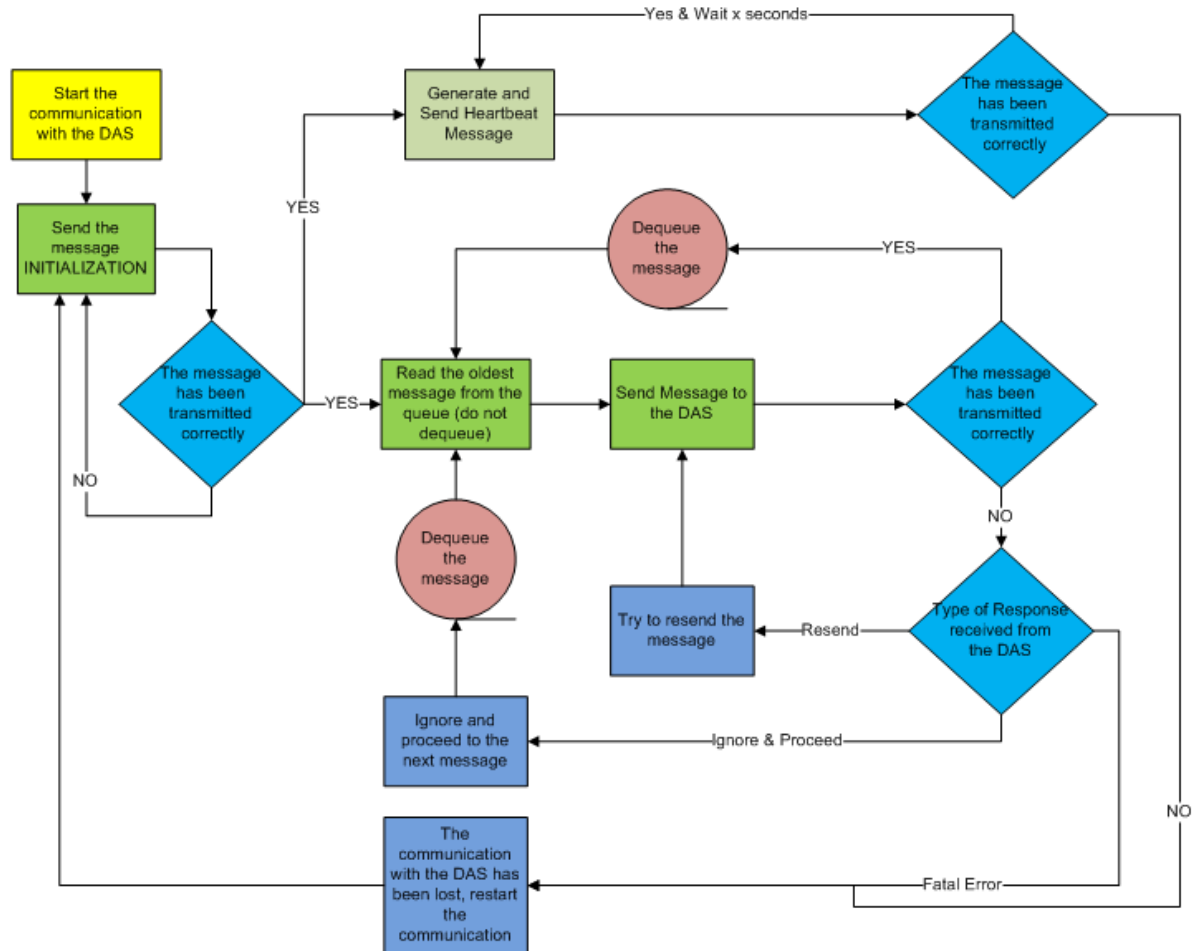


Figure 8
Type of Response

7.2 The TCS defines four types of behavior, depending on the HTTP code provided by the DAS:

- DAS Accepts the Message
- Resend
- Ignore & Proceed
- Fatal Error

7.2.1 DAS Accepts the Message

7.2.1.1 When DAS provides with the following code, TCS considers that the message has been transmitted correctly.

Table 2 HTTP Code for Message Accepted Case

<i>Name</i>	<i>HTTP Code</i>
HttpStatusCode.Accepted	202
HttpStatusCode.Created	201
HttpStatusCode.OK	200

7.2.2 Resend the Message

7.2.2.1 When TCS receives any http code listed below, TCS tries to resend this message. The INITIALIZATION message is not resent.

Table 3 Resend INITIALIZATION Message When TCS Receives the Following HTTP Code

<i>Name</i>	<i>HTTP Code</i>
HttpStatusCode.BadGateway	502
HttpStatusCode.Continue	100
HttpStatusCode.GatewayTimeout	504
HttpStatusCode.LengthRequired	411
HttpStatusCode.RedirectKeepVerb	307
HttpStatusCode.UseProxy	305
HttpStatusCode.Unused	306
HttpStatusCode.RequestTimeout	408

7.2.3 Code for the Case Ignore & Proceed

7.2.3.1 When TCS receives any http code listed below, proceed to the next message, and ignore the current one. The current message is not resent. The INITIALIZATION message is not resent.

Table 4 HTTP Code for Ignore & Proceed Case

<i>Name</i>	<i>HTTP Code</i>
HttpStatusCode.Ambiguous	300
HttpStatusCode.BadRequest	400
HttpStatusCode.Conflict	409
HttpStatusCode.ExpectationFailed	417
HttpStatusCode.Forbidden	403
HttpStatusCode.Found	302
HttpStatusCode.HttpVersionNotSupported	505
HttpStatusCode.Moved	301
HttpStatusCode.NoContent	204
HttpStatusCode.NonAuthoritativeInformation	203
HttpStatusCode.NotAcceptable	406
HttpStatusCode.NotFound	404
HttpStatusCode.NotImplemented	501
HttpStatusCode.NotModified	304
HttpStatusCode.PartialContent	206
HttpStatusCode.PaymentRequired	402

<i>Name</i>	<i>HTTP Code</i>
HttpStatusCode.PreconditionFailed	412
HttpStatusCode.ProxyAuthenticationRequired	407
HttpStatusCode.UnsupportedMediaType	415
HttpStatusCode.Unauthorized	401
HttpStatusCode.SwitchingProtocols	101
HttpStatusCode.ResetContent	205
HttpStatusCode.RequestUriTooLong	414
HttpStatusCode.RequestEntityTooLarge	413
HttpStatusCode.RequestedRangeNotSatisfiable	416

7.2.4 Fatal Error Message

7.2.4.1 When TCS receives any http code listed below, TCS does not resent the message. The INITIALIZATION message is resent.

Table 5 Fatal HTTP Code

<i>Name</i>	<i>HTTP Code</i>
HttpStatusCode.Gone	410
HttpStatusCode.InternalServerError	500
HttpStatusCode.MethodNotAllowed	405
HttpStatusCode.ServiceUnavailable	503