# SEMI E179-0822
# SPECIFICATION FOR PROTOCOL BUFFERS COMMON COMPONENTS

This Standard was technically approved by the Information & Control Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on January 4, 2022. Available at www.semiviews.org and www.semi.org in August 2022; originally published March 2020; previously published March 2022.

**NOTICE:** This Document contains one or more restricted biased terms originating from an external source. While the biased term is necessary to use and/or comply with this Document, use of such biased term is to be avoided whenever possible in the SEMI Standards Program and SEMI is working to eliminate biased terms from its own Standards and Safety Guidelines.

**NOTICE:** Paragraphs entitled 'NOTE:' are not an official part of this Standard or Safety Guideline and are not intended to modify or supersede the official Standard or Safety Guideline. These have been supplied by the global technical committee to enhance the usage of the Standard or Safety Guideline.

**NOTICE:** The section symbol (§) is used to reference a numbered section, paragraph, or header and all subordinate paragraphs, headers, and embedded material (i.e., EXCEPTIONs, lists) therein. The paragraph symbol (¶) is used to reference a particular numbered paragraph and embedded material therein. When duplicated (i.e., §§ and ¶¶) the symbols are used to reference more than one section or paragraph, respectively.

## 1 Purpose

1.1 This Specification is a central location for the definition of common Protocol Buffers elements that are not specific to a single standard, but are common across multiple standards.

## 2 Scope

2.1 This Specification defines the representation of Error, Data Types, Data Value Types, Unit and Operator which are considered common components. The common components are used in association with other Protocol Buffers documents generated as part of the Protocol Buffers based SEMI Standard Interface Specifications.

**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 The purpose of this Specification is to define basic components and to allow for future additions as new common components are discovered and required in conjunction with a Protocol Buffers based interface.

## 4 Referenced Standards and Documents

4.1 *SEMI Standards and Safety Guidelines*

SEMI E5 — Specification for SEMI Equipment Communications Standard 2 Message Content (SECS-II)

SEMI E125 — Specification for Equipment Self Description (EqSD)

SEMI E132 — Specification for Equipment Client Authentication and Authorization

SEMI E138 — Specification for XML Semiconductor Common Components

SEMI E145 — Classification for Measurement Unit Symbols in XML

SUPERSEDED



Documents Document Sharing Hub

Full document details: SEMI E179-0822

4.2 *IETF® Documents*[1]

RFC 7540[2] — Hypertext Transfer Protocol Version 2 (HTTP/2)

4.3 *CNCF® Documents*[3]

Wire Format over HTTP2[4] — gRPC®[5] over HTTP2

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

## 5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *DCP* — Data Collection Plan

5.1.2 *EDA* — Equipment Data Acquisition

5.1.3 *gRPC* — Remote Procedure Call system

5.1.4 *HTTP/2* — Hypertext Transfer Protocol version 2

5.1.5 *protobuf* — Protocol Buffers

5.1.6 *protobuf3* — Protocol Buffers version 3

5.1.7 *SECS* — SEMI Equipment Communications Standard

5.1.8 *SECS-II* — SEMI Equipment Communications Standard 2

5.1.9 *TCP/IP* — Transmission Control Protocol/Internet Protocol

5.1.10 *UML®* — Unified Modeling Language

5.1.11 *URI* — Uniform Resource Identifier

## 6 Resources

6.1 Important resources and organizations used in the generation and definition of this Specification are found in:

- *Protocol Buffers* — https://developers.google.com/protocol-buffers/
- *gRPC* — https://grpc.io/docs/

6.2 For an overview of Protocol Buffers and gRPC, refer to:

- Related Information 1, Technology Overview
- Related Information 2, Protocol Buffers and gRPC Implementation Consideration

## 7 Conventions

7.1 *Requirements Identification*

7.1.1 The following notation specifies the structure of requirement identifiers.

7.1.1.1 The following requirements prefix format is used at the beginning of requirement text. Refer to Table 1 for the format notation of the requirements prefix: [Esss.ss-RQ-nnnnn-nn]

7.1.1.2 To mark the end of the requirement text, the following suffix format is used: [/RQ]

---

[1] Internet Engineering Task Force, IETF Administration LLC, 1000 N West Street, Suite 1200, Wilmington, DE 19801, USA; Telephone: +1.510.492.4080, Fax: +1.510.492.4001, http://www.ietf.org. IETF trademark is owned by the IETF Trust.

[2] https://tools.ietf.org/html/rfc7540

[3] Cloud Native Computing Foundation c/o The Linux Foundation®, 548 Market St, PMB 57274, San Francisco, CA 94104-5401, USA; Telephone: +1.415.723.9709, https://www.cncf.io. CNCF trademark is owned by The Linux Foundation.

[4] https://github.com/grpc/grpc/blob/master/doc/PROTOCOL-HTTP2.md

[5] Trademark is owned by The Linux Foundation.



SUPERSEDED

Documents
Document Sharing Hub

Full document details: SEMI E179-0822

7.1.1.3 Requirements in the body text are highlighted with a border and light green background (may appear gray in black and white printouts) as shown below.

[Esss.ss-RQ-nnnnn-nn] Requirement text. [/RQ]

**Table 1 Requirement Identifiers**

| *Format Notation* | *Purpose* |
|---|---|
| Esss.ss | SEMI Standards Specification identifier. Examples: E87.00, E87.01, E134.00. |
| RQ | Indicates this is a requirement identifier. |
| Nnnnn | Unique five-digit number within this Specification. 90000–99999 are reserved for use by SEMI. |
| Nn | Two-digit number that indicates version level of the requirement. A value of .00 is used for the first version of a requirement. |
| /RQ | Indicates the end of a requirement. |

7.1.1.4 Note that the requirements in § 14 use the reserved requirement numbers noted in Table 1 because they are repeated in other SEMI Standards that use this convention.

7.1.2 Requirements in tables are delimited in one of two ways.

7.1.2.1 Where the requirement occupies an entire row in the table, the requirement identifier (RequirementID) is placed in column 1. No '[/RQ]' is used to mark the end of the requirements text in this case. The table may also contain rows that are not requirements. In this case, the column 1 cell is left blank.

7.1.2.2 Non-requirements text related to a requirement row may be included in an adjacent row. The relationship between the rows is indicated by a broken line separating the two.

7.1.2.3 Where the requirement occupies only one cell, the text in the cell includes the RequirementID prefix and suffix similar to requirements in the body text.

7.1.2.4 No cell in a table will contain both requirement and non-requirement text.

7.1.2.5 Cells that contain requirements are shaded light green.

7.1.2.6 Table 2 provides an example of requirements in a table. Note that in this example, the same requirement is presented in two alternative formats. In practice, mixing the two approaches in the same table is not typically done.

**Table 2 Example Table with Requirements**

| *RequirementID* | *Statement* | *Additional Detail* |
|---|---|---|
| Esss.ss-RQ-nnnnn-nn | The light shall be the sole blue light. | There shall be no similar color in the light panel. |
| | The blue light is typically placed at the leftmost position in the light panel. | |
| | [Esss.ss-RQ-nnnnn-nn] The light shall be the sole blue light. There shall be no similar color in the light panel. [/RQ] | The blue light is typically placed at the leftmost position in the light panel. |

7.1.3 Only text marked with the RequirementID is a requirement of this Specification.

7.1.3.1 Clarifications, examples, and related recommendations may be provided near a requirement, but are not part of the requirement.

7.1.3.2 Note that the word 'should' is used in some non-requirements text, where it denotes a recommendation or a best practice, not a requirement.

SUPERSEDED



Documents Document Sharing Hub
Full document details: SEMI E179-0822

7.1.3.3 Note that the words 'must' and 'will' are used in the Implementation Overview section to illustrate the logical result of requirements specified in the prerequisites and requirements section. Such statements should not be considered requirements.

7.1.4 Parent-child relationships of requirements are noted in § 14, Table A1-1. Where a parent requirement includes conditions or selection criteria, these are passed to their children. For example, if a parent requirement is stated to apply only to photolithography equipment, any child requirements also apply only to such equipment. The condition need not be restated for each child requirement.

7.1.4.1 Compliance to a parent requirement includes compliance to all of its child requirements.

7.1.4.2 Some cells or rows in a table may contain headings that are assigned a RequirementID. These are parent requirements for other requirements in the table—typically for the rows that immediately follow that heading.

## 8 General Requirements

8.1 The following is the top-level parent requirement of all others in this Specification.

[E179.00-RQ-00001-00] Compliance to this Specification shall require compliance to all other identified requirements in this Specification, including complementary files, as stated conditions apply. [/RQ]

8.2 *Protocol Buffers File*

8.2.1 This Specification includes a .proto file, 'SEMIE179-0320.proto', that provides the normative definitions of the types described here.

[E179.00-RQ-00002-00] Implementations shall use the Complementary File 'SEMIE179-0320.proto' file that accompanies this Specification. The Complementary File is an official part of and accompanies this Specification. The contents of the .proto file constitutes a core part of this Specification. [/RQ]

8.2.2 The .proto file can be compiled into your implementation-specific programming language, generating the appropriate classes for all specified types.

[E179.00-RQ-00003-00] Implementations shall integrate with Protocol Buffers v3.7.1 at a minimum. Implementations may use newer versions of Protocol Buffers, but they shall be compatible with this stated version. [/RQ]

8.2.3 This Specification also includes Various Materials. These Various Materials are not an official part, but accompany this Specification. Use of these Various Materials is optional and is not required for compliance with this Specification.

8.2.3.1 The file SEMIE179-0320-helper.proto is a marked up version of the .proto file for the Primary Standard. It has comments prefixed by 'INTERNAL –' that describe how the Protocol Buffers messages map to XML definitions. Most of these items were specified in the SEMI E138 schema file. These comments can be helpful to implementers familiar with the XML objects. They will not be included in the official .proto file included with the SEMI Standard to keep that document readable.

NOTE 1: Implementers should not use the SEMIE179-0320-helper.proto file as its messages are the same as the ones defined in the official .proto file that accompanies the Specification.

## 9 Errors

9.1 *Data Type for Error Handling* — Error handling in most cases even with standard interface specifications is implementation-specific. It is not practical for one specification to define all possible error conditions, error codes, descriptions and supplementary information that is needed to properly recover from the error. Regardless, a common approach to communicating error conditions can simplify error management for standards implementers and application developers.

9.2 *Error Data Structure* — The following data structure defines a common error structure.

SUPERSEDED



Documents
Document Sharing Hub
Full document details: SEMI E179-0822

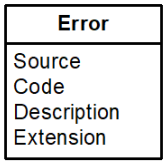


**Figure 1**
**Error Structure**

9.2.1 *Source* — This string attribute represents the document from where the attribute code value is defined.

[E179.00-RQ-00004-00] For error codes defined by SEMI Standards, the Source shall be the urn representation of the standard number. For error codes defined by a supplier, the Source shall be the urn representation of the supplier's name. [/RQ]

9.2.1.1 For example, for SEMI E132 errors, the content of 'Source' is urn:semi-org:E132; for supplier ACME errors the content is: urn:ACME-com). Additional information may be included by the supplier by adding such information after the ':' colon symbol and separating the fields with a '.' period symbol.

**Table 3 Example of Error Sources**

| # | *Source* | *Comment* |
|---|---|---|
| 1 | urn:semi-org:E005 | Standard Error Codes for Equipment – SEMI E5. |
| 2 | urn:semi-org:E087 | Standard Error Codes from Carrier Management. |
| 3 | urn:semi-org:E125 | Standard Error Codes from Equipment Self-Description. |
| 4 | urn:<Supplier>-com[:<SupplierFields>] | Identification of the source of error codes. |

9.2.2 *Code* — This integer attribute directly maps to a particular specification error code list. The following is an example of a list of code ranges. This Specification does not require the allocation of any code range. Other specifications that use SEMI E179 should define code ranges that apply to them.

**Table 4 Example of Error Code Range**

| # | *Code Range* | *Comment* |
|---|---|---|
| 1 | 1–63, 32768–65535 | Standard Error Codes for Equipment – SEMI E5. |
| 2 | 10000–10500 | Equipment Supplier Specific enumerated error codes (Example). |
| 3 | 5000–5999 | Common Standardized Core Components Errors. |

9.2.3 *Description* — This string attribute is a human-readable description of the error code being reported. It maps directly to the definition of the error code for each of the expected results of the services as defined by the specific standard. Additional information may be supplied after this string to provide more detail about the error. Use the ':' colon symbol as delimiter (i.e., Common Component, Code 5001, 'Insufficient Arguments Provided:Argument "SessionID" missing').

**Table 5 Example of Error Descriptions**

| # | *Description* | *Comment* |
|---|---|---|
| 1 | Unknown Object in Object Specifier | Error description for Code '1' as defined in SEMI E5. |
| 2 | Loadport does not exist | Error code '49' description from SEMI E87 and defined in SEMI E5. |
| 3 | Operation Not Supported | Error description for Error Code '5000'. |

SUPERSEDED



Documents Document Sharing Hub
Full document details: SEMI E179-0822

9.2.4 *Extension* — This string attribute allows the supplier to include additional information regarding the error code encountered. This field is used when the description does not explicitly cover the problem and more specific information is required. If the extension information is not needed, an empty string should be specified.

[E179.00-RQ-00005-00] All errors shall be reported using an error structure that includes the Source (string), Code (integer), Description (string), and Extension (string). [/RQ]

9.3 *Common Standardized Errors* — This section defines common errors to be reported by any application processing a service. Most of these errors are related to the validation of the service or availability of the system.

**Table 6 Common Standardized Error Codes**

| *Code* | *Description Field* | *Comment* |
|---|---|---|
| 5000 | Operation Not Supported [:Operation specific detail] | Description of the actual error encountered. Additional information may be included. |
| 5001 | Insufficient Arguments Provided [:Operation specific detail] | One or more arguments are missing in the request. |
| 5002 | Invalid Arguments Provided [:Operation specific detail] | One or more arguments in the request are invalid or incompatible. |
| 5003 | Specified configuration is invalid [:Operation specific detail] | An argument in the request is valid but not together with the rest of the specified arguments or with the current system state. |
| 5004 | System Not Ready | The system is not ready to perform work. |
| 5005 | System Shutting Down | The system is shutting down and cannot perform work. |

[E179.00-RQ-00006-00] If the equipment encounters any of the common errors defined in Table 6, it shall report the error using the specific error code and description listed in Table 6, and populate the 'Operation specific detail' part of the description appropriately. [/RQ]

9.4 *Raising Exceptions with Trailer gRPC Metadata*

9.4.1 When working with a streaming gRPC method, there is no way to return an error result to the gRPC client to let it know there was a problem processing the streaming gRPC method call. (For example, the gRPC server rejects the streaming gRPC method because it is not valid when the method call was made.)

9.4.2 In this situation, the gRPC server implementation can throw an RPC exception and use the trailer gRPC metadata infrastructure to provide error information to the gRPC client that is normally specified with SEMI E179's ErrorType message.

9.4.3 The trailer gRPC metadata infrastructure sends key-value information back to the calling code after the RPC method call is done. The MetaData.Key values are lowercase. The MetaData.Value is a string data type. For more information about how to raise gRPC exceptions with trailer gRPC metadata, refer to Related Information 2, § R2-8.

**Table 7 Trailer gRPC Details for Exceptions thrown for SEMI E179 Error Condition**

| *Trailer gRPC Metadata* | | |
|---|---|---|
| | *MetaData.Key* | *MetaData.Value* |
| | source | Error's Source value |
| | code | Error's Code value |
| | description | Error's Description value |
| | extension | Error's Extension value |

SUPERSEDED



Documents
Document Sharing Hub
Full document details: SEMI E179-0822

[E179.00-RQ-00014-00] All exceptions thrown by the gRPC server implementation code shall use the trailer gRPC metadata to report error information as specified in Table 7. [/RQ]

9.4.4 If the error does not have a specific error attribute (e.g., extension), it should still be included in the trailer gRPC metadata with an empty string value.

9.4.5 This Specification does not define a specific gRPC status code to be used when the gRPC server code throws the RPC exception. Other SEMI Documents may specify what gRPC status code should be used for specific error conditions. Otherwise the implementation uses the gRPC status code that best matches the error condition.

## 10 Data Types

10.1 The following sections define data types that can be used with a parameter definition (e.g., refer to SEMI E125). A defined parameter in equipment metadata has an associated TypeDefinition data type that provides a defined format. Parameters have references to a defined data type via their typeName. Figure 2 shows a class that describes re-usable data types. TypeDefinitions can be re-used in the metadata wherever a parameter's type can be declared.

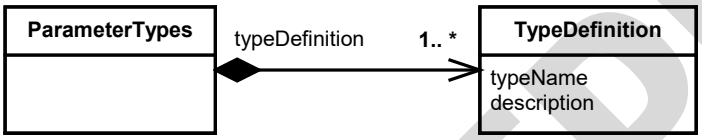


**Figure 2**
**Describing Parameter Type Definitions**

10.2 *ParameterType* — This class defines all available TypeDefinitions which define the data format used when reporting a parameter's value or content.

**Table 8 ParameterType Association Definition**

| *Association Role Name* | *Definition* | *Comments* |
|---|---|---|
| typeDefinition | The type definition being referenced. | One or more TypeDefinition classes used to represent and report a parameter data value or content. |

10.3 *TypeDefinition* — The class from which all type description classes are derived. This class has two attributes: typeName and description. A parameter TypeDefinition can be a named SimpleTypeSpecifier, Array, Structure, Enumeration, or Variable class.

**Table 9 TypeDefinition Attribute Definition**

| *Attribute Name* | *Definition* | *Required* | *Form* |
|---|---|---|---|
| typeName | The name for the type definition. | Y | Text, should be unique across all defined TypeDefinitions. SimpleTypeSpecifier names should use the names defined in Table 9, column 1. The remaining types (i.e., Array, Structure, and Variable) names are defined by the equipment or the standard from where the type originates. |
| description | A description of the meaning or purpose of the type definition. | Y | Text. |

SUPERSEDED



Documents Document Sharing Hub
Full document details: SEMI E179-0822

10.4 *TypeDefinitions* — Figure 3 shows the basic set of classes that can be used to describe parameter types.

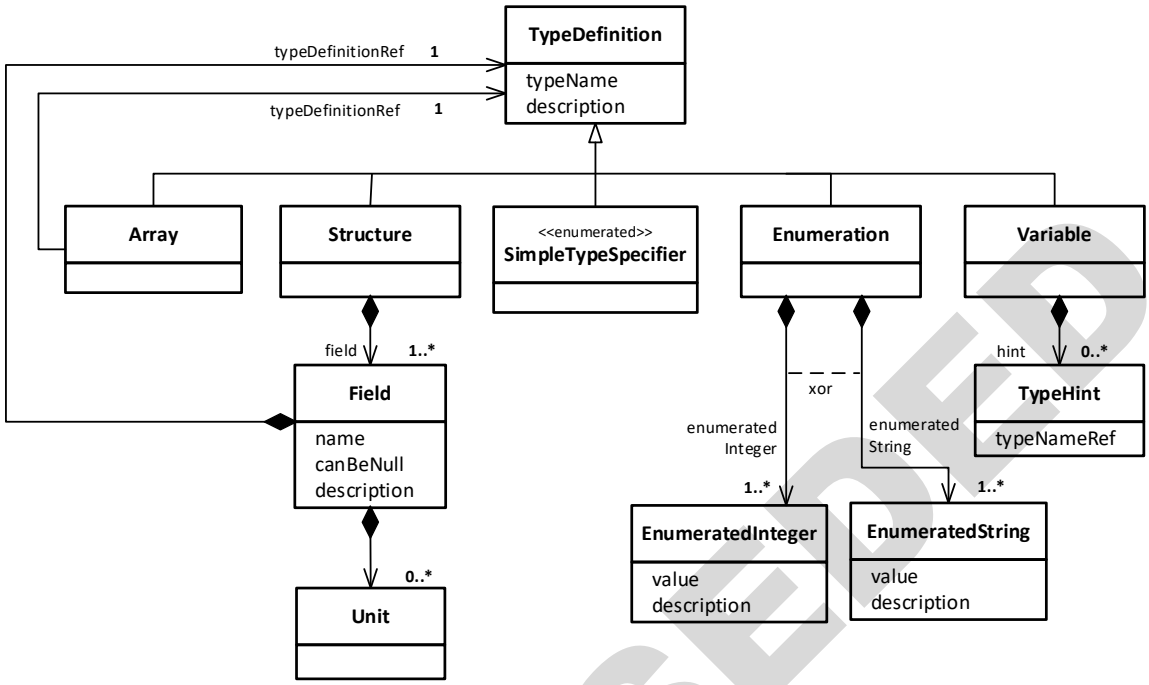


**Figure 3**
**Describing Parameter Types**

[E179.00-RQ-00007-00] Any TypeDefinitions supported by the equipment shall be reported as either a SimpleTypeSpecifier (associated with one data type instance from among the aggregations listed in Table 9), a Variable data type, or a Array, Enumeration, or Structure. [/RQ]

10.4.1 *SimpleTypeSpecifier* — Many equipment suppliers provide data items described with SEMI E125 that are also available through the traditional SECS/GEM control interface. The SimpleTypeSpecifier class type definitions specify what Protocol Buffers data type will be used to hold the data and can be mapped to the common types defined by SECS-II (refer to SEMI E5). Additional mappings may be possible if a specific technology implementation type collection is required.

[E179.00-RQ-00008-00] The typeName defined in Table 9 shall be used when defining the name for the simple data type.[/RQ]

**Table 10 SimpleTypeSpecifier Enumeration Definition**

| *typeName* | *SimpleType Specifier enum* | *Protocol Buffers type equivalent* | *SECS-II type* | *Comments* |
|---|---|---|---|---|
| double | F8 | double | 8 byte floating point | |
| float | F4 | float | 4 byte floating point | |

SUPERSEDED



Documents
Document Sharing Hub
Full document details: SEMI E179-0822

| *typeName* | *SimpleType Specifier enum* | *Protocol Buffers type equivalent* | *SECS-II type* | *Comments* |
|---|---|---|---|---|
| long | I8 | sint64,<br>fixed64,<br>sfixed64 | 8 byte integer (signed),<br>4 byte integer (unsigned) | Protocol Buffers supports:<br>• a signed integer data type that may be better for positive and negative data values.<br>• a fixed64 data type that may be better for positive values greater than $2^{56}$<br>• a signed fixed64 data type that may be better for positive and negative values greater than $2^{56}$ |
| int | I4 | sint32,<br>fixed32,<br>sfixed32 | 4 byte integer (signed),<br>2 byte integer (unsigned) | Protocol Buffers supports:<br>• a signed integer data type that may be better for positive and negative data values.<br>• a fixed32 data type that may be better for positive values greater than $2^{28}$<br>• a signed fixed32 data type that may be better for positive and negative values greater than $2^{28}$ |
| short | I2 | sint32 | 2 byte integer (signed),<br>1 byte integer (unsigned) | Protocol Buffers does not have a datatype equivalent to small integers (e.g., short). |
| byte | I1 | sint32 | 1 byte integer (signed)<br>1 byte integer (unsigned) | Protocol Buffers does not have a datatype equivalent to small integers (e.g., byte). |
| unsignedLong | U8 | uint64,<br>fixed64 | 8 byte integer (unsigned) | Protocol Buffers supports:<br>• an unsigned integer data type that may be better for positive data values.<br>• a fixed64 data type that may be better for positive values greater than $2^{56}$ |
| unsignedInt | U4 | uint32<br>fixed32 | 4 byte integer (unsigned) | Protocol Buffers supports:<br>• an unsigned integer data type that may be better for positive data values.<br>• a fixed32 data type that may be better for positive values greater than $2^{28}$ |
| unsignedShort | U2 | uint32 | 2 byte integer (unsigned) | Protocol Buffers does not have a datatype equivalent to small integers (e.g., ushort). |
| unsignedByte | U1 | uint32 | 1 byte integer (unsigned) | Protocol Buffers does not have a datatype equivalent to small integers (e.g., byte). |
| string | String | string | ASCII, JIS-8, 2 byte character | |
| dateTime | DateTime | google.protobuf.Timestamp | ASCII, JIS-8, 2 byte character | DateTime would be converted to a string if exposed as a SECS-II type. |
| boolean | Boolean | bool | Boolean | |

SUPERSEDED



Documents Document Sharing Hub
Full document details: SEMI E179-0822

| *typeName* | *SimpleType Specifier enum* | *Protocol Buffers type equivalent* | *SECS-II type* | *Comments* |
|---|---|---|---|---|
| base64Binary | Binary64 | bytes | Binary,<br>8 byte integer (unsigned) | 8-byte unsigned integers should be described as an 8-byte signed integer if possible, or sent as a (big-endian) byte array. |
| noValue | NoValue | n/a | n/a | To be used if there are any problems obtaining the requested value(s). |
| anyURI | URI | string | ASCII, JIS-8, 2 byte character | String value identifies a specific resource. |

NOTE 2: The SECS-II List format has no Protocol Buffers equivalent. Multiple element items are typically represented as Arrays or Structures.

10.4.2 *Array* — Arrays are container types that can hold zero or more values of the same type. Figure 3 shows how to describe the type of the elements in a given array. An array can refer to only one of the primitive, composite types or defined type. For example, an array description that refers to the 'F8' primitive type represents an array of double numbers that has no limitation on the number of values it can contain.

**Table 11 ArrayType Association Definition**

| *Association Role Name* | *Definition* | *Comments* |
|---|---|---|
| typeDefinitionReference | Describes a type that has been previously defined. | A TypeDefinitionReference (can be a recursive reference from within the same TypeDefinition). |

10.4.2.1 *TypeDefinitionReference* — Holds a reference to a defined TypeDefinition. This allows the reuse of TypeDefinitions in defining more complex TypeDefinitions, such as structures and arrays.

10.4.2.1.1 For example, an array of two-byte integers would be represented by an Array class with a typeDefinitionReference specifying 'I2', which is the mandated typeName for a two-byte integer value, according to Table 9.

**Table 12 TypeDefinitionReference Association Definition**

| *Association Role Name* | *Definition* | *Comments* |
|---|---|---|
| typeNameRef | The type name being referenced. | One and only one type definition name derived from an existing TypeDefinition's typeName. |

10.4.3 *Structure* — Structures are composite types that can hold one or more fields that reference primitive or composite types. Figure 3 shows how a structure is described as a collection of one or more typed fields.

**Table 13 StructureType Association Definition**

| *Association Role Name* | *Definition* | *Comments* |
|---|---|---|
| field | The fields that comprise this data structure. | Ordered list of one or more structured data elements of type Field. |

10.4.3.1 *Field* — Represents a named field that has one and only one type, primitive or composite. This class is used for describing fields within structures. Refer to § 10.3 for a description of the TypeDefinition class.

SUPERSEDED



Documents
Document Sharing Hub
Full document details: SEMI E179-0822