SEMI E139.3-1108
XML/SOAP BINDING FOR RECIPE AND PARAMETER MANAGEMENT

This standard was technically approved by the global Information & Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on August 29, 2008. It was available at www.semi.org in October 2008 and on CD-ROM in November 2008.

1 Purpose

1.1 The purpose of this specification is to provide an implementation mapping of the SEMI E139 specification for Recipe and Parameter Management to XML format and the SOAP 1.1 protocol.

2 Scope

2.1 The scope of this specification is the representation of the RaP services and related parameters in an XML Schema and corresponding WSDL port types and bindings. It provides the following:

- The XML Schema data types used to support the services defined in SEMI E139.
- The WSDL port type and binding definitions used to support the operations defined by the interfaces specified in SEMI E139.

2.2 This specification does not add new domain information or concepts to the SEMI E139 specification. The only additions made are those needed to render useful WSDL or XML Schemas.

2.3 This specification builds upon SEMI E132 Specification for Client Authentication and Authorization. It defines requirements for the application of SEMI E132 concepts to the SEMI E139 specification.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 This specification addresses implementation of RaP services in the absence of other XML/SOAP-based communications. The intent of this specification is to be compatible with other capabilities that might share SEMI E132 services on an equipment. However, it does not attempt to identify or address potential interactions and conflicts should other similar communications interfaces exist (e.g., SEMI E134 services).

4 Referenced Standards and Documents

4.1 *SEMI Standards*

SEMI E121 — Guide for Style & Usage of XML for Semiconductor Manufacturing Applications

SEMI E128-0706 — Provisional Specification for XML Message Structures

SEMI E132 — Specification for Equipment Client Authentication and Authorization

SEMI E134 — Specification for Data Collection Management

SEMI E138 — XML Semiconductor Common Components

4.2 *W3C Standards*[1]

Extensible Markup Language (XML) 1.0 (Second Edition) — W3C, 6 October 2000 (http://www.w3.org/TR/2000/REC-xml-20001006/)

XML Schema Part 0: Primer — W3C, 2 May 2001 (http:// www.w3.org/TR/xmlschema-0/)

XML Schema Part 1: Structures — W3C, 2 May 2001 (http:// www.w3.org/TR/xmlschema-1/)

1 World Wide Web Consortium, Massachusetts Institute of Technology (MIT), Computer Science and Artificial Intelligence Laboratory (CSAIL), 32 Vassar Street Room 32-G515, Cambridge, MA 02139, USA, Telephone: 617.253.2613, Fax: 617.258.5999; http://www.w3.org

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E139.3-1108

XML Schema Part 2: Datatypes — W3C, 2 May 2001 (http:// www.w3.org /TR/xmlschema-2/)

Web Services Description Language (WSDL) 1.1 — W3C Note (http://www.w3.org/TR/wsdl)

Simple Object Access Protocol (SOAP) 1.1 — W3C Note (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *SOAP* — Simple Object Access Protocol

5.1.2 *UML* — Unified Modeling Language

5.1.3 *W3C* — World Wide Web Consortium

5.1.4 *WSDL* — Web Service Description Language

5.1.5 *XML* — eXtensible Markup Language

5.1.6 *SSL* — Secure Sockets Layer

5.1.7 *RaP* — Recipe and Parameter Management. Refers to the capabilities defined in SEMI E139.

5.2 *Definitions*

5.2.1 *Port Type* — a port type is a named set of abstract operations and the abstract messages involved.

5.2.2 *WSDL Binding* — a definition of how a port type definition is bound to a concrete network protocol and message format (e.g., SOAP).

6 Conventions

6.1 *Translating UML to XML Schema*

6.1.1 The reader is expected to have a working knowledge of the XML, WSDL, SOAP, and Schema specifications (see ¶ 4.2). This document does not provide tutorial information on these subjects.

6.1.2 The Guide for Style and Usage of XML (SEMI E121) was consulted in the creation of the conventions for mapping UML to XML.

6.1.3 The key conventions for mapping the SEMI E139 UML to XML are summarized here:

- Attributes of UML classes from SEMI E139 are represented as XML elements.
- Composition associations are mapped to contained elements.

6.1.4 The translation of a UML class to XML Schema types is documented using the table format illustrated by Table 1. Operations defined for a UML class are translated into WSDL operations. These mappings link conceptual requirements to their implementation form. Therefore, they are considered extensions to those requirements.

6.1.5 *Translation Table Column Header Description*

- *Attribute or Role Name* — If an attribute, the name of the attribute is placed here. If an association (including aggregation or composition), the role name from the UML diagram is placed here. Compositions are often not assigned role names. In that case "none" is placed here.
- *UML Name/Type* — If an attribute, the data type of the UML attribute is placed here. If an association, the type of association is placed here. The possible association types are "Composition," "Aggregation," or the basic "Association." UML defines these three types of association.
- *XML Element or Attribute* — Lists the type of XML construct used to represent the UML attribute or association.
- *XML Name/Type* — Provides the name and data type of the resulting XML construct. The type may be a built-in type (e.g., xs:string), or a named type defined within the XML Schema.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139.3-1108

Table 1 Example Translation Table

Attribute or Role Name	*Format/Type*	*XML Element or Attribute*	*XML Name/Type*
Friend	association	element	Friend: HumanReferenceArray
employees	aggregation	element	Employees: HumanArray
Family	composition	element	Family: HumanArray
Name	string	element	Name: xs:string

6.1.6 *Translating Operations to XML Schema*

6.1.6.1 The translation of an operation defined for UML interface classes to XML Schema types is documented using a table format illustrated in Table 2. One row is provided for each possible input/output argument for the operation being described.

Table 2 Example Translation Table for Operation Input/Output Arguments

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
<argument name from abstract specification>	<argument format from abstract specification>	<whether the argument is modeled as an Element or Attribute>	<name of the attribute/element>

6.1.7 *Documenting XML Schema and WSDL Files*

6.1.7.1 Associated with this document are XML Schema and WSDL files that are core components of the specification. Each such document is described using a table format illustrated by Table 3.

Table 3 Example Schema/WSDL Document Description Table

File Name	<.xsd or .wsdl file name>
Target Namespace	<target namespace defined in the schema or WSDL file>
Imported/Referenced Namespaces	<namespaces imported or used by the schema or WSDL file>
Description	<brief description of the purpose/function of the schema or WSDL file>

6.2 *Documenting XML Schema with Diagrams*[2]

6.2.1 This document provides graphical representations of the included XML Schema data type and element definitions and their relationships. They are included to aid the reader in understanding the schema. These diagrams are illustrations of the requirements contained in the XML files and should not be interpreted as additional requirements.

6.2.2 Although no standard graphical notation for XML Schema could be found, various XML tools have their own notation. This document will use the notation provided by XMLSpy from Altova Corporation. Figure 1 shows a sample XML Schema diagram that will be used to provide a basis for explanation of the schema graphical notation used in the rest of the document.

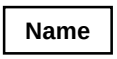
6.2.3 In the diagram, rectangular boxes represent XML element definitions. The owner elements are on the left side while the elements contained within the owner element are on the right side. In the sample diagram, ParentType contains Child1, Child2, Child 3, and Child 4. In turn, Child2 contains Child2a, Child2b, and Child2c. The additional symbols (8-sided boxes) represent sequences or choices. See Table 4 for an explanation of these symbols.

2 All images/graphics representing XML were created using Altova's XMLSPY®. Copyright 2003 Altova GmbH and reprinted with permission of Altova.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E139.3-1108

Table 4 Altova XMLSPY® Schema Diagram Symbols

	Denotes a required ordered sequence of the right hand elements with a cardinality of one for each element. (sequence)
	Denotes an optional ordered sequence of the right hand elements with a cardinality of one for each element. (sequence)
	Denotes a required, but unordered, sequence of the right hand elements with a cardinality of one for each element. (all)
	Denotes a required choice of the right hand elements. Exactly one or two of the right hand elements must be present. (choice)
	Denotes an element. Optional element if outline is dashed line.
	Notation in top left corner denotes an element that contains parsed character data.
	Notation in the bottom left corner denotes a reference to a group or element defined elsewhere.

6.2.4 A graphic using a solid line is a required element; using a dashed line represents an optional element. Numbers or ranges in the lower right hand corner represent cardinality. The default cardinality is one.

6.2.5 To simplify a diagram or help focus on a particular aspect, detail may be hidden. The 8-sided symbols have a small square on the right end. If a minus sign "–" is in the box, then all detail is shown. If the box contains a plus sign "+", then all detail to the right of that symbol is hidden. The example has no hidden detail.

6.2.6 The yellow (or gray if printed in monochrome) boxes indicate the use of other defined types. So, Child4 is of type "Child4Type." Child4Type defines Child4a and Child4b. This detail may be hidden in the diagram. Object oriented inheritance is typically represented in XML as type extension. In Figure 1, ParentType extends Child1and2Type by adding a sequence that includes Child3 and Child4.

6.2.7 Reading Figure 1 would yield the following additional information:

1. Child1and2Type is an ordered sequence of two items: Child2 and Child1.
2. Child1 contains an optional ordered sequence of Child1a, one or more Child1b, and (optionally) Child1c.
3. Child2 contains a choice of one or two of the following: Child2a, Child2b, and Child2c.
4. Child3 contains an unordered sequence of Child3a and Child3b.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139.3-1108

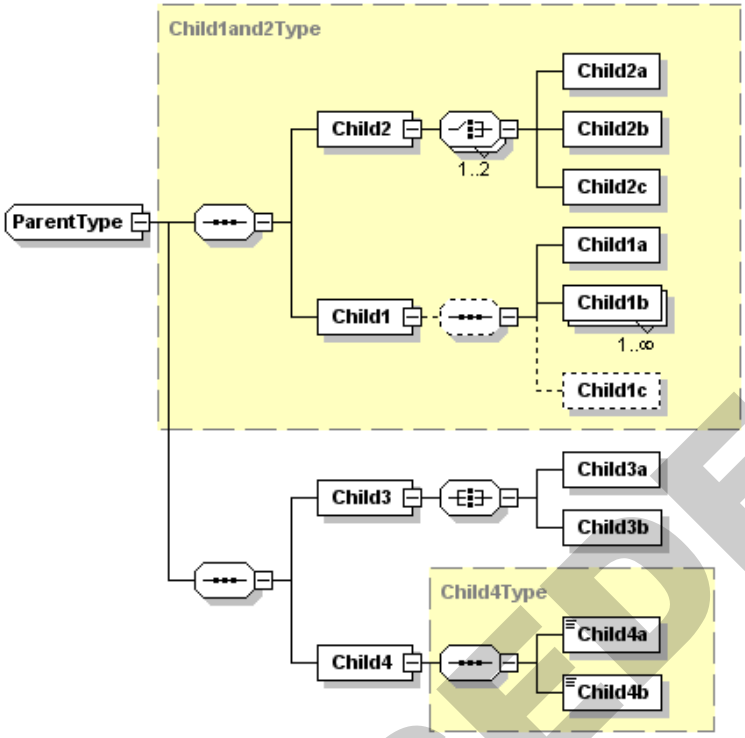


Figure 1
XML Schema Example Diagram

6.2.8 *XML Schema Sample*

6.2.8.1 The sample XML Schema for the example shown in Figure 1 is presented below as Figure 2. Refer to the XML documentation referenced in ¶ 4.2 for a complete description of the syntax and semantics of XML Schema.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E139.3-1108

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:complexType name="ParentType" abstract="true">
      <xs:complexContent>
         <xs:extension base="Child1and2Type">
            <xs:sequence>
               <xs:element name="Child3">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="Child3a"/>
                        <xs:element name="Child3b"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
               <xs:element name="Child4" type="Child4Type" nillable="false"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="Child4Type" abstract="true">
      <xs:sequence>
         <xs:element name="Child4a" type="xs:integer" nillable="false"/>
         <xs:element name="Child4b" type="xs:string" nillable="false"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="Child1and2Type" abstract="true">
      <xs:sequence>
         <xs:element name="Child2">
            <xs:complexType>
               <xs:choice maxOccurs="2">
                  <xs:element name="Child2a"/>
                  <xs:element name="Child2b"/>
                  <xs:element name="Child2c"/>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="Child1">
            <xs:complexType>
               <xs:sequence minOccurs="0">
                  <xs:element name="Child1a"/>
                  <xs:element name="Child1b" maxOccurs="unbounded"/>
                  <xs:element name="Child1c" minOccurs="0"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="Parent" type="ParentType" final="#all"/>
</xs:schema>
```

Figure 2
XML for Sample

6.2.9 *Translating UML to WSDL*

6.2.9.1 In general, UML interface classes defined in the abstract specification are translated into WSDL as portType definitions, and each portType definition has a corresponding WSDL binding definition. The following tables show the convention used for documenting the WSDL port type and binding definitions for a given UML interface class.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139.3-1108

Table 5 Example Interface WSDL Port Type Table

Class Name	<UML interface name from abstract specification>
WSDL Port Type Name	<port type name used in WSDL>
SEMI E139 Operation → WSDL Operation	<operation name from UML interface> → <WSDL port type operation name>

Table 6 Example Interface WSDL Binding Table

SEMI E139 Class Name	<UML interface name from abstract specification>
WSDL Binding Name	<binding name used in WSDL>
SOAP Binding Style	<RPC or document>
SOAP Transport	<transport identifying URI>

7 Overview

7.1 To promote better understanding of this specification, this section provides an overview of this specification before moving to the detailed requirements. **This overview section does not specify any requirements**. The descriptions contained may presume certain requirements that are specified in the requirements section below.

7.2 This specification builds upon SEMI E132. SEMI E132 provides a mechanism for making secure connections from client to service provider. Implementations of this specification can coexist with other standards based on SEMI E132, such as SEMI E125 Equipment Self-Description. Thus, the E132 port can be shared where appropriate. The alternative is that the implementation may choose to use separate ports.

7.2.1 This document focuses on the specification of the message definitions as translated from the original SEMI E139 form to one usable for XML/SOAP messages.

7.2.2 Where SEMI E132 refers to "equipment," for the purpose of this specification, that should be interpreted as "service provider."

7.2.3 Privileges are defined for the E139 services and are enforced using the SEMI E132 mechanisms. A separate privilege is defined for each RaP service (excepting requestToSend).

7.3 The XML/SOAP messaging approach is intended to co-exist with other messaging protocols, such as SECS. It is expected that this will provide a transition path. For example, in the case of an existing SECS communication link, the addition of XML/SOAP-based communication would allow an off-tool editor to connect to the tool via XML/SOAP messaging before the factory commits to a change to the SECS host application.

7.4 SEMI E139 defines message parameters for reporting errors that might occur during processing. However, additional possible errors are introduced with the new protocol. The response messages for the RaP services have been structured so that protocol errors may be handled separately. The message content is designed as an XML choice construct that can return either the SEMI E139 defined response or a separate protocol error construct.

7.5 TransferContainer handling in this specification is very simple. TransferContainers are included in the SOAP messages as base64 encoded binary message elements.

7.5.1 Implementers should note that some precaution should be taken to avoid instantiation in memory of messages containing very large TransferContainers. A RaPnode may have a practical size limit for messages received, based on the memory available to the application.

8 Requirements

8.1 Conformance to this specification requires satisfaction of all requirements as stated within this specification.

8.2 This specification builds directly upon SEMI E139. Implementations of this specification shall conform to the requirements of SEMI E139.

8.3 A RaPnode may provide RaP services via more than one defined protocol. For example, both SECS (per SEMI E139.2) and XML/SOAP (per this specification) might be provided.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E139.3-1108