European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 9: Metamodel and XSD Schemas Reference Manual

BAE Ref No: IAWG-ECOA-TR-011
Dassault Ref No: DGT 144486-B

Issue: 2

Prepared by
BAE Systems (Operations) Limited and Dassault Aviation

Note: *This specification represents the output of a research programme and contains mature high-level concepts, though low-level mechanisms and interfaces remain under development and are subject to change. This standard of documentation is recommended as appropriate for limited lab-based evaluation only. Product development based on this standard of documentation is not recommended.*

1 Table of Contents

2 List of Figures

3 List of Tables

4 Abbreviations

API	Application Programming Interface
ECOA	European Component Oriented Architecture
UTC	Coordinated Universal Time
XML	eXtensible Markup Language

5 Introduction

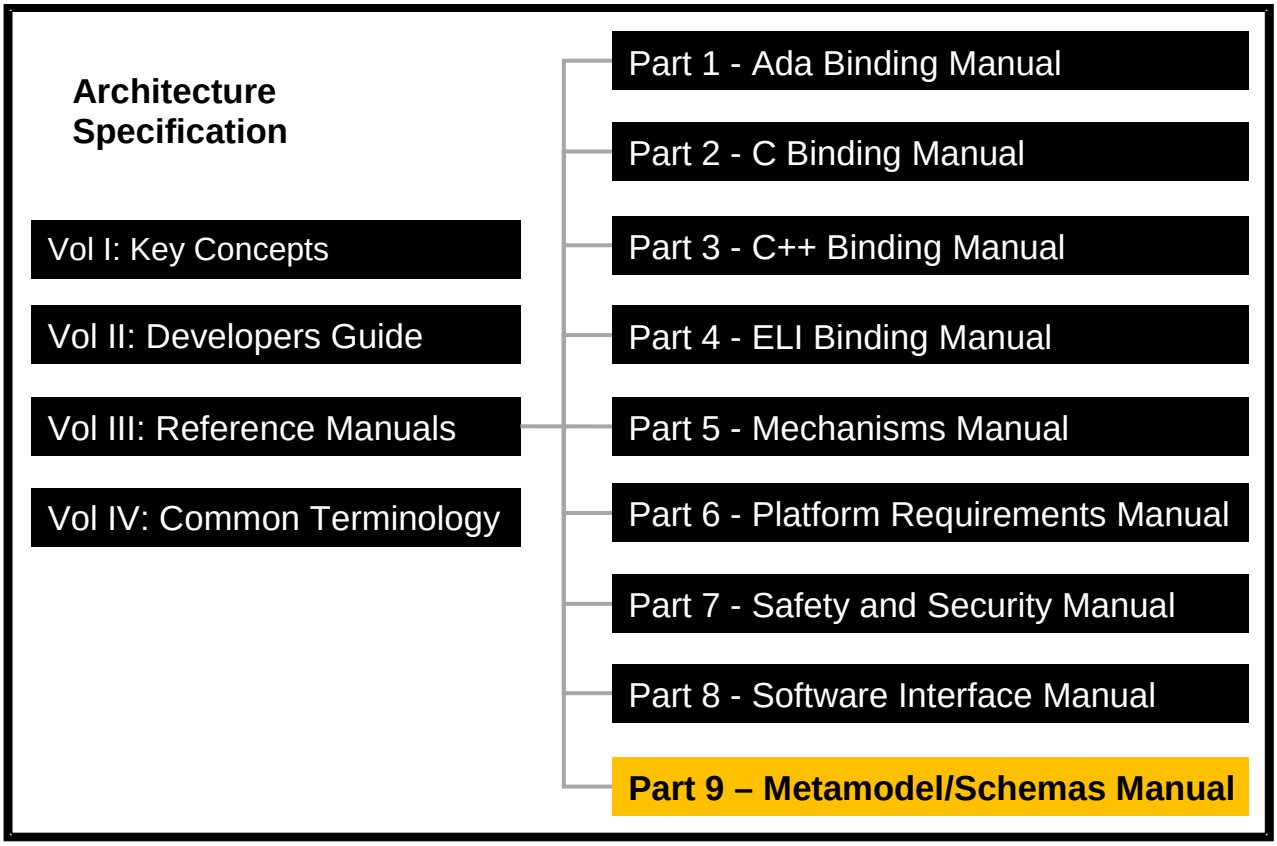

Figure 1 – ECOA Documentation

The Architecture Specification provides the definitive specification for creating ECOA-based systems. It describes the standardised programming interfaces and data-model that allow a developer to construct an ECOA-based system. It is introduced in Key Concepts (Reference 1) and uses terms defined in the Common Terminology (Reference 11). For this reason, the reader should read these documents, prior to this document. The details of the other documents comprising the rest of the Architecture Specification can be found in Section 8.

The Architecture Specification consists of four volumes, as shown in Figure 1:

- Volume I: Key Concepts
- Volume II: Developer's Guide
- Volume III: Reference Manuals
- Volume IV: Common Terminology

This document comprises Volume III Part 9 of the ECOA Architecture Specification, and contains the Metamodel and XML schema definitions for an ECOA system.

The document is structured as follows:

- Section 6 describes the ECOA metamodel;
- Section 7 details of v1.10.1 of the schemas.

6 ECOA Meta Models

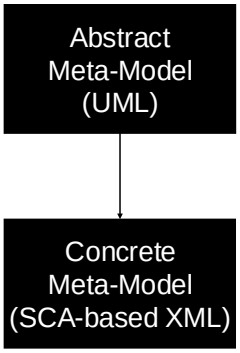

Figure 2 – ECOA Meta-Models

The structure of an ECOA system has been specified as an abstract meta-model which describes the ECOA system's data elements and their relationships. UML entity-relationship diagrams have been used to present the model information which can be found in Section 0.

Ultimately the requirements to exchange components, and automatically instantiate systems from them, require a precisely-specified and machine-readable version of the model. This is known as the concrete meta-model and the implementation is based on the open standard Service Component Architecture (SCA). Section 6.2 defines the concrete meta-model.

It is envisaged that the ECOA implementers will ultimately develop tool support that enables ECOA information to be captured in high-level design tools that support, for example, UML. However, the SCA-based concrete meta-model will remain the standard for exchange of information (e.g. between component suppliers and system integrators).

6.1 Abstract Meta-Model

6.1.1 Overview

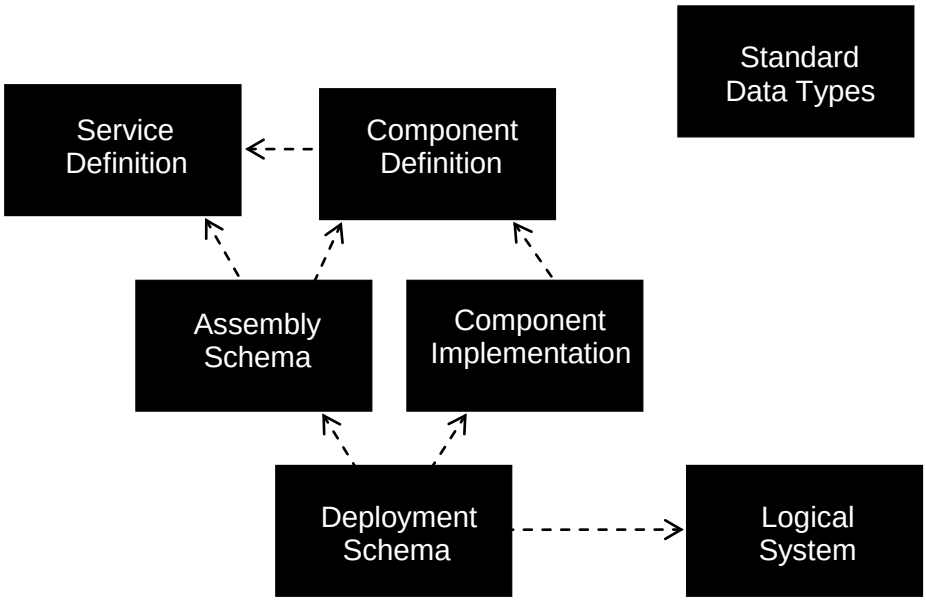

Figure 3 – Overview of Meta-Model

The following sections detail the ECOA abstract meta-model. They aim to provide a definition of all the concepts and objects that need to be formalized to describe an ECOA system. This abstract meta-model is designed using a set of self-sufficient views of a UML model; each view of the meta-model is describing one given concept. An overview of this model is shown in Figure 3 which describes overarching dependencies between main ECOA concepts.

An Assembly Schema (see section 6.1.4) describes the structure of the ECOA system, in a hardware independent fashion. It does this by describing service links between the Application Software Components. These are uni-directional connections between one service provided by an Application Software Component and another service required by an Application Software Component. Both services share the same full Service Definition and have a compatible Quality of Service (QoS). The service is offered or required as a whole, where a client has access to the whole set of service operations of the Service Definition.

An ECOA system is fully formalized within a Deployment Schema (see section 6.1.8) which details how the Application Software Components of the Assembly Schema are deployed on a Logical System: it specifies how modules of each Application Software Component are mapped onto logical processor nodes. A Logical System (see also section 6.1.8) describes a set of hardware computing resources and their physical connections.

The client defines an expected QoS for each required service (service-level and operation-level), and the provider defines an actual QoS for each provided service (service-level and operation-level). The expected and actual QoS need to be compatible for a service link to be established between both services. As a service is a collection of operations, a service link is, at technical level, implemented by a collection of module-level operation links between component modules. These operation links fulfil the rules of direction and multiplicity implied by the service link.

The Component Definition (Section 6.1.3) captures the interface of an Application Software Component and is formalized in terms of the services that are required and the services that are provided by an Application Software Component.

The interface of an Application Service, called Service Definition, is described as a set of operation signatures (see section 6.1.2).

Each Data Type (see section 6.1.7) used at the Service Definition or Component Definition level must be described.

A Component Instance is the software instantiation within the ECOA System of a given Component Implementation. A Component Implementation is the software realization of a Component Definition to which it conforms.

A Component Implementation (see section 6.1.5) is described within the ECOA System in terms of:

- Its Component Definition, which is, in SOA terms, the “Component Contract” to which it conforms,
- Its internal design, which is made of Modules, and Operation Links between Modules or Service Definitions of the component.

The concept of Module is defined (section 6.1.5) as a software entity implementing a given part of the ECOA Component Implementation. Operations in one Module may interact with Operations in another module via standard ECOA mechanisms.

Within a Component Implementation, Modules are linked together, at operation level, and are linked to the operations of the Services of the Component Implementation using Operation Links (see section 6.1.6). The Modules are the software entities that have to be deployed in the Deployment Schema.

The rules specified in Section 6.2.1.1 apply to the names for operations, component implementations and module implementations.

Note: The abstract meta-model is described with UML class diagrams which use the following conventions:

- The default multiplicity for any link between two entities is 1.
- A grey-filled class indicates a reference to another class diagram in which the mentioned entity is described.

6.1.2 Service Definition

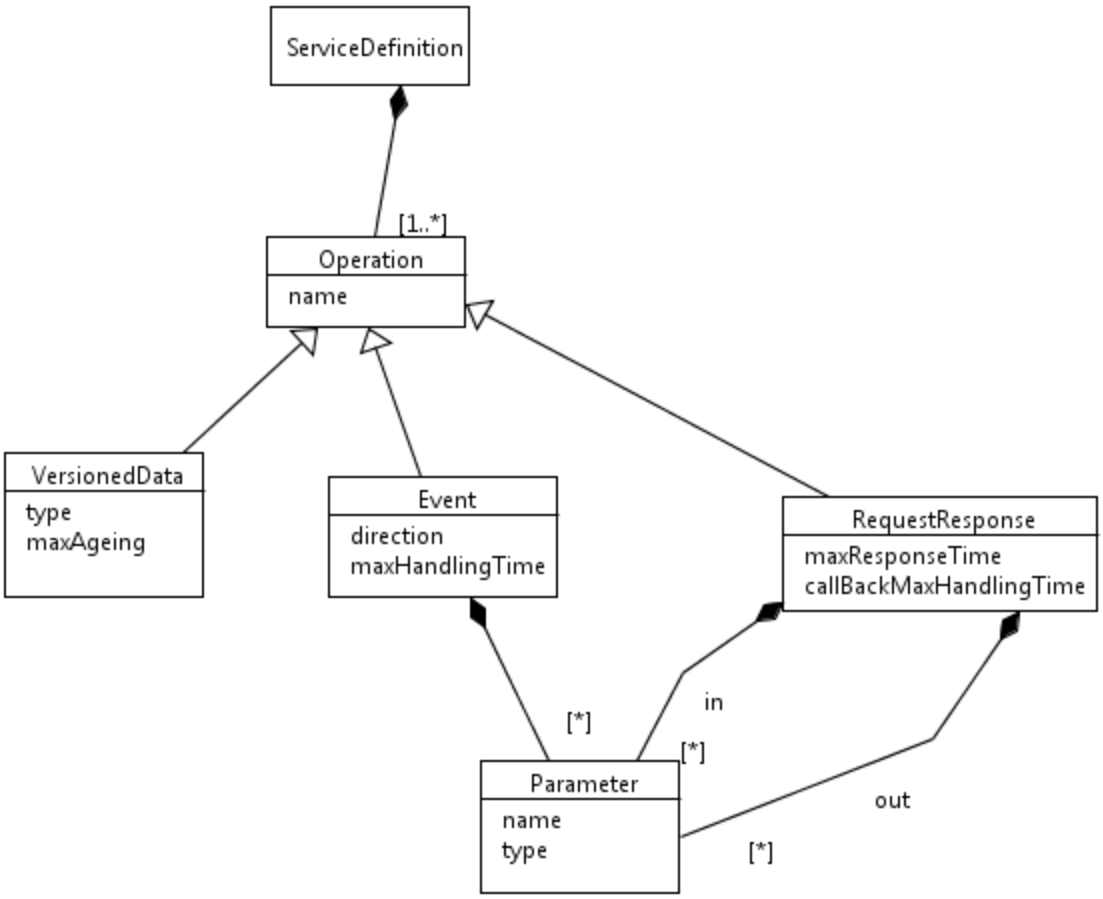

Figure 4 – ServiceDefinition meta-model

A **ServiceDefinition** is a set of **Operations** and **QoS Specifications**. An **Operation** is either, a **VersionedData Operation**, an **Event Operation** or a **RequestResponse Operation**. A ServiceDefinition contains at least one Operation.

For a VersionedData Operation in a ServiceDefinition, the data is published by the Application Software Component that provides the service.

An Event Operation in a ServiceDefinition has a direction: either received_by_provider or sent_by_provider.

A RequestResponse Operation in a ServiceDefinition is initiated by the Application Software Component that requires the service.

An initial proposal of specific QoS attributes for each operation type is given hereafter. Unit is in seconds (s).

	Provided service	Required service
Event	**IN** **MaxHandlingTime** = maximum duration between event receipt and end of related processing	
	OUT **MaxHandlingTime** = specifies an intent on receivers for maximum duration between event receipt and end of related processing	
Request-Response	**IN (request_received)** **MaxResponseTime** = maximum duration between request receipt and response sent	**OUT (request_sent)** **MaxResponseTime** = maximum duration between request sent and response receipt In case of an asynchronous request-response, **callBackMaxHandlingTime** = maximum duration between response receipt and end of related processing
Versioned Data	**Published** **MaxAgeing** = maximum duration between data production (from the source) and the end of the writing process	**Consumed** **MaxAgeing** = maximum duration between data production (from the source) and the end of the reading process

Table 1 – Specific QoS attributes on operations

Data ageing will take into account all the transformations processed through the components chain by analysing components behaviours (see section 6.1.9.1)

In addition to specific attributes, two common QoS attributes are specified and applicable for each operation (independent of its type):

- HighestRate: specifies the maximum number of occurrences of the operation within a specified time frame. If the number of occurrences is 1, the time frame corresponds to the minimum inter-arrival time between operations.
- LowestRate: specifies the minimum number of occurrences of the operation within a specified time frame. If the number of occurrences is 1, the time frame corresponds to the maximum inter-arrival time between operations.

For data, these rates express refreshment period requirements.

For an output event, an occurring rate can be defined to compare the receivers accepted input rates: this allows consistency checking between production and consumption rates.

For R/R replies, it is assumed that the R/R reply follows the same laws as the R/R request.

A comment can be added to describe each operation.

A single Module Operation may invoke many Container Operations during its execution, which introduces dependencies between the rates identified for those operations. Behaviour specifications at Service and Module level have been identified to allow these dependancies to be captured - see section 6.1.9.1.

All QoS attributes on operations are currently optional.

A ServiceDefinition also includes service-level specifications of Quality of Service (QoS) parameters (such as encryption level). These QoS specifications are used when matching up provided and required ServiceDefinition variants when creating ServiceLinks (section 6.1.6).

Note: the definition of service-level attributes requires additional work.

6.1.3 Component Definition

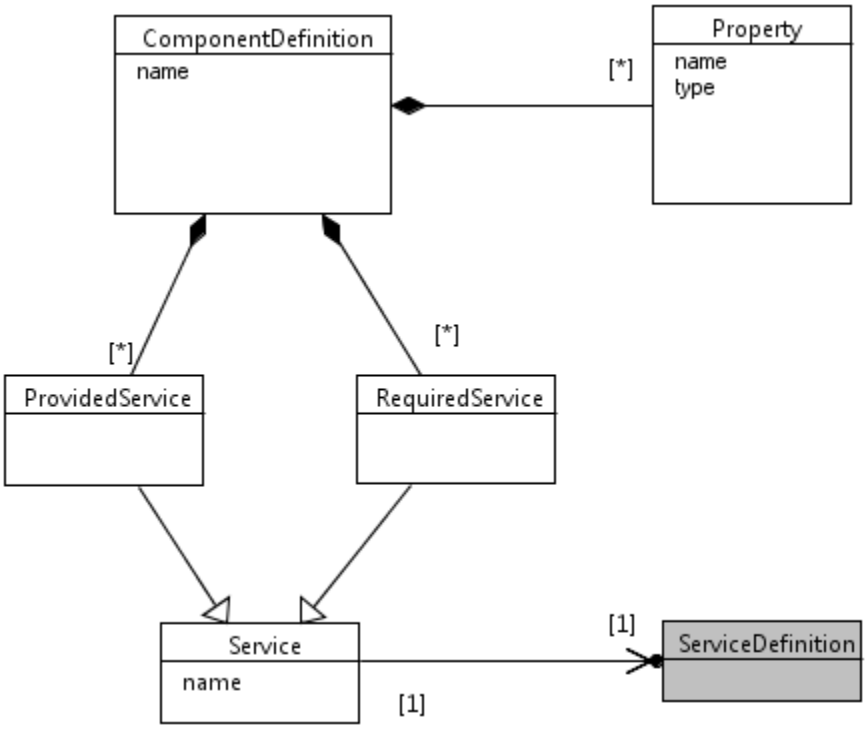

Figure 5 – ComponentDefinition meta-model

A **ComponentDefinition** is a set of **Properties**, **ProvidedServices** and **RequiredServices**. A ProvidedService or a RequiredService references a ServiceDefinition shared by ComponentDefinitions.

A ComponentDefinition must contain at least either a ProvidedService or a RequiredService.

Note: Elements of Behaviour modelling need to be added to the ComponentDefinition: rules for each operation, complexity/processing time estimations, required service usage rules, relationship between provided and required services, definition of states/modes, etc. See section 6.1.9.1.

6.1.4 Assembly Schema

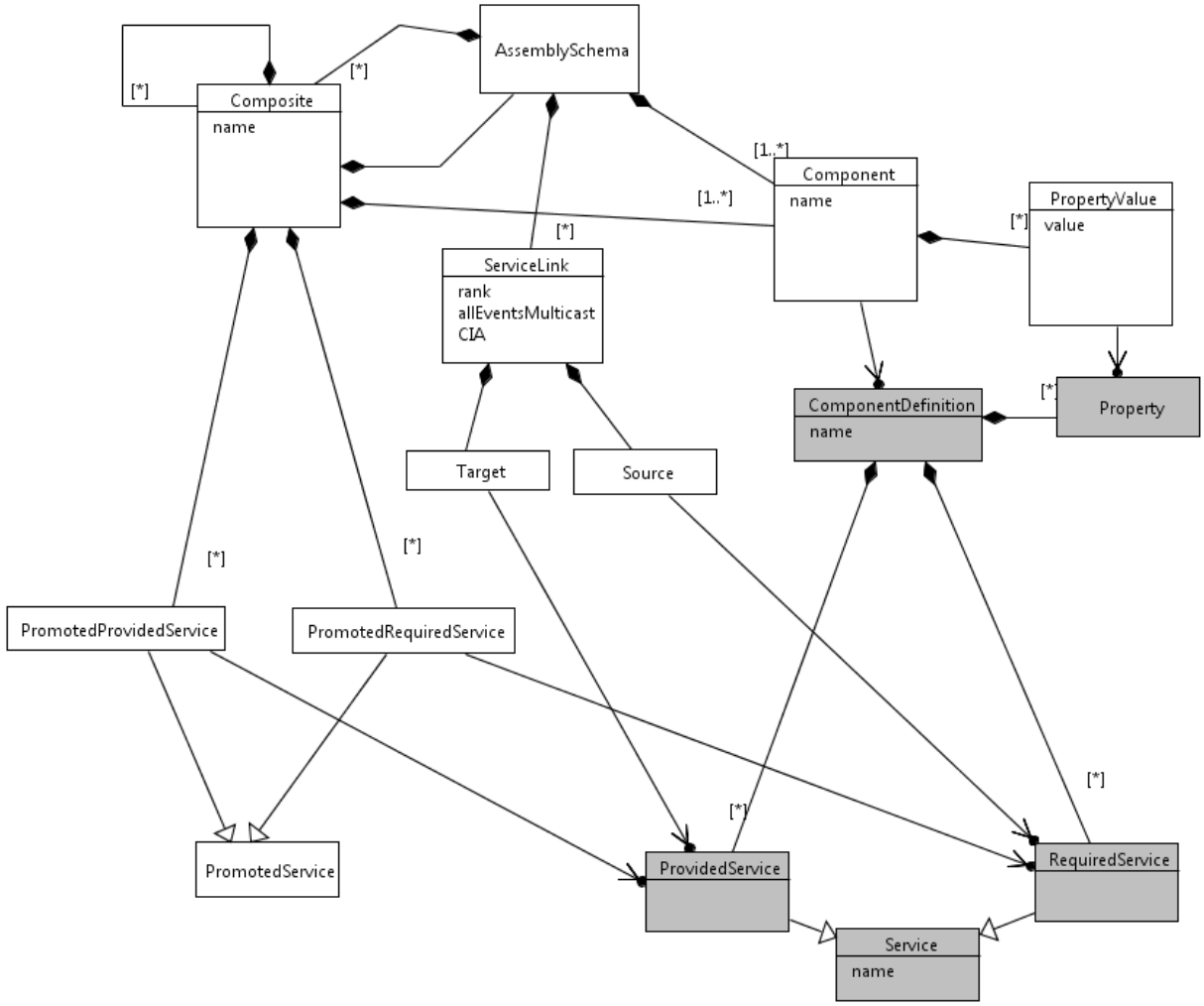

Figure 6 – AssemblySchema and ServiceLink meta-model

An **AssemblySchema** describes the structure of an ECOA system, independently of its physical deployment on hardware platforms. Note that grey in figure above means that the greyed element is more precisely defined in another subsection.

An Assembly is made of **Composites**, **Components** and **ServiceLinks**.

A Composite looks like a component; its definition is provided by a dedicated AssemblySchema and promotion links between its provided or required services and the provided or required services of its internal components. Promotion links are a logical concept to master complexity through hierarchical assembly schemas. They have no existence at IT or technical levels: the assembly schema actually deployed is the one containing only components.

A Component instantiates a ComponentDefinition in a given system. It has a set of instantiation parameters known as Properties defined at ComponentDefinition level.

A ServiceLink connects Components together via provided and required service references. Each ServiceLink connects one ProvidedServiceReference (target) to one RequiredServiceReference (source) each of which refers to a Component and to a RequiredService or ProvidedService of this component's ComponentDefinition (targets and

sources are “technical” objects which are introduced to model ternary associations, without introducing the specific UML notation).

An attribute named **rank** can be associated to a ServiceLink. It is the way the System Integrator indicates a preference for an instance of a service provider over another when both are connected to the same service required. The lower the numerical value of rank, the higher the preference for the link. The rank is mandatory. An additional Boolean attribute **allEventsMulticast** indicates if all event operations of the service definitions are sent on this ServiceLink in a systematic way.

6.1.5 Component Implementation

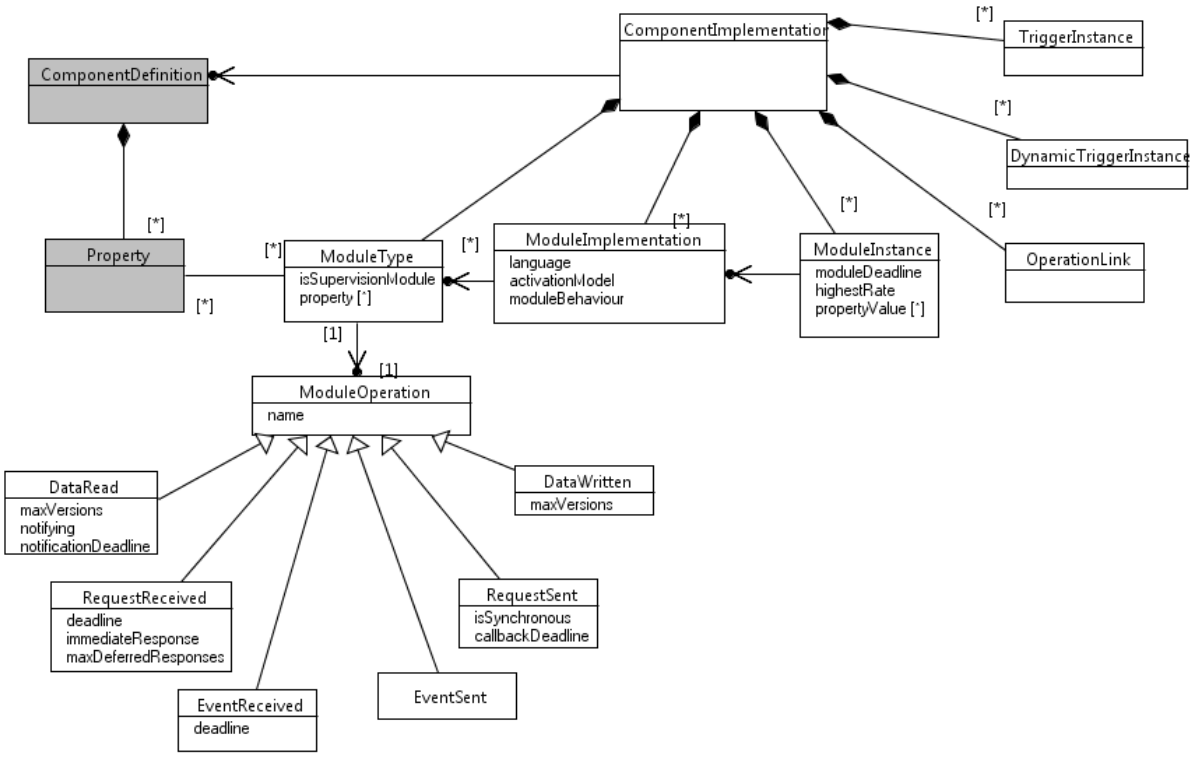

Figure 7 – ComponentImplementation meta-model

A software realization of a ComponentDefinition is described by a **ComponentImplementation**.

A ComponentImplementation gives information to secure the integration of Application Software Components in a system, possibly sharing execution platforms, and to make early verification of the system possible.

A ComponentImplementation is made of **ModuleTypes**, **ModuleInstances**, **TriggerInstances**, **DynamicTriggerInstances** and **OperationLinks**.

A ModuleImplementation corresponds to a piece of software implementing in a certain programming language a given part of the Component Implementation that must be executable in a single thread (no parallelism, no internal synchronisation). A behaviour describing worst case execution paths may be associated to a ModuleImplementation (see 6.1.9.3). A ModuleInstance corresponds to the instantiation of a given **ModuleImplementation**.

A ModuleType defines the interface of a ModuleImplementation in terms of ModuleOperations at module-level or in terms of properties. These operations correspond to the same exchange mechanisms used by Application Software Components (i.e. data, event; request-response), but

have a direction (reflecting the module's point of view), and have additional attributes over and above the service defined ones (e.g. notifying for DataRead or immediateResponse for RequestReceived). These operations' names will appear in the module's container API. Each operation name shall be unique for a given moduleType definition. By annotating a module as a supervision one, a module may support system-level capabilities such as error handling or module lifecycle management.

An additional QoS attribute defines the deadline for each operation (event, request, callback, notification). This deadline allows analyzing the schedulability of operations at design time. It also allows the container to check that the temporal behaviour of the operation at execution time is the expected one.

A ModuleInstance corresponds to an instance of a ModuleImplementation which itself is of a defined ModuleType. A ModuleInstance has its own internal state and has an assigned module deadline (used to determine its execution priority). The module deadline attribute is defined in ModuleInstance which is calculated through considering the response times of operations. The default activation model of the Module Instance is the reactive model; meaning the container activates it as long as there are incoming operations for it. An alternative activation model is the rhythmic model; meaning the container activates it every unit of its deadline, and only if there is an incoming operation. This will be further described in the final release of this document and is included here for completeness.

The notion of ModuleInstance provides the ability to instantiate, a number of times, the same software code in multiple execution contexts (e.g. different module deadline or execution node) inside an ECOA Application Software Component.

A TriggerInstance is similar to a ModuleInstance, except that it is dedicated to producing periodic events: it has no module type; it does not need to be implemented, as the periodic events will be generated automatically by the infrastructure.

A DynamicTriggerInstance is a trigger that generates non periodic events. The delay between the generation of two events can be dynamically be set at runtime. As for the TriggerInstance, the DynamicTriggerInstance is generated by the infrastructure.

Each Property at ComponentDefinition level is implemented within one or more ModuleTypes of the ComponentImplementation.

The internal structure of a given ComponentImplementation must be specified in terms of OperationLinks.

6.1.6 Operation Links

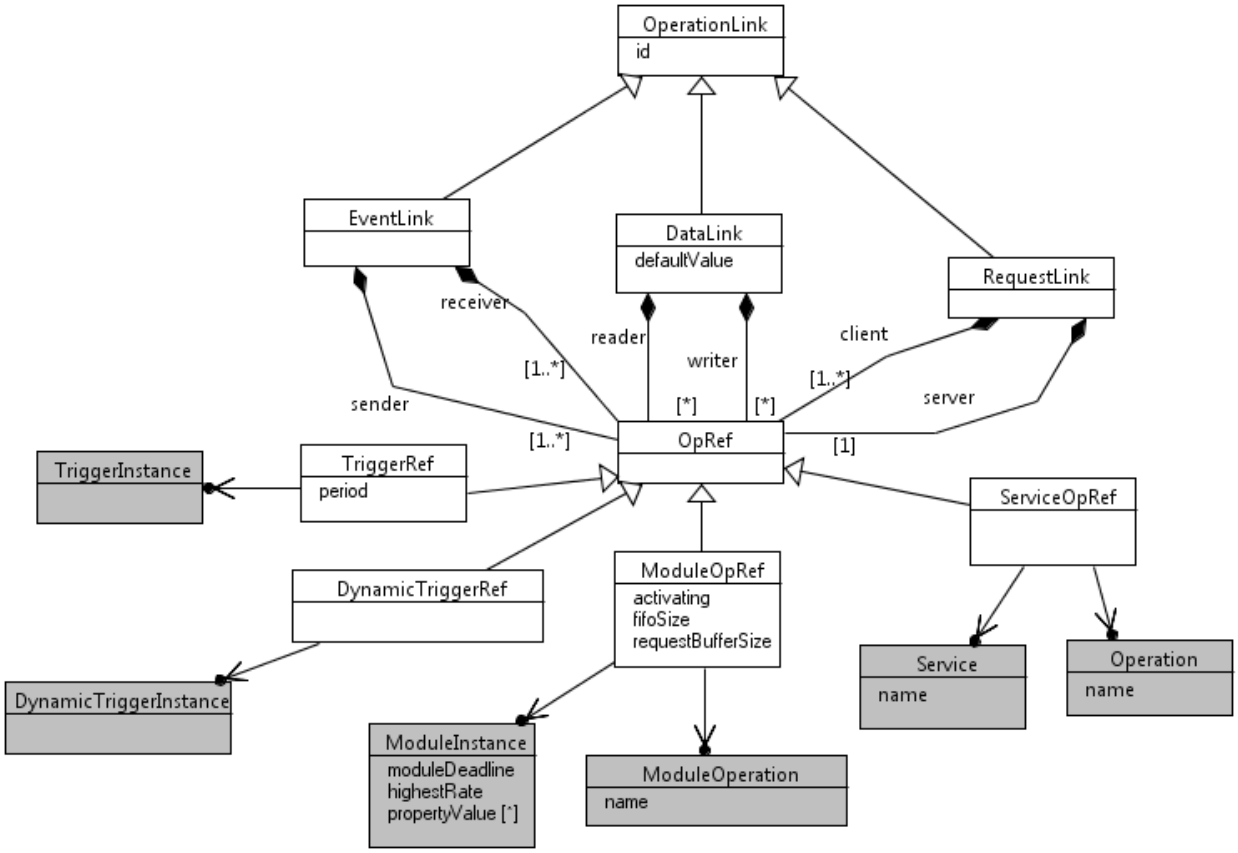

Figure 8 – OperationLink meta-model

The **OperationLinks** describe the interactions/synchronisations between Modules within the same Application Software Component. An OperationLink is a “star-like” connection linking “internal” (module-level) operations. Inter-Module interactions are specified using **DataLink**, **EventLink** or **RequestLink**, depending on the kind of module operations that are linked. These three kinds of links are oriented and have different possible multiplicities:

- A DataLink may have n writers and p readers (a unique data-writer is recommended but not mandatory),
- A RequestLink has one server and p possible clients,
- An EventLink can have n possible senders and p possible receivers of the event.

Note that the cardinality of the requestLink only concerns module internals. To implement redundant servers at component level, multiple service links can be defined between multiple client components and multiple server components.

An internal module operation is referred to by a **ModuleOpRef**, which refers to a ModuleInstance and one of its ModuleOperations. Attributes are associated to a ModuleOpRef to define if the operation is activating or not (activating), the maximum number of waiting operation calls for this operation (fifoSize), and the maximum number of pending requests if the operation is considered as a deferred request-response (requestBufferSize).

Internally to the component, an operation of the service is referred to by a **ServiceOpRef**, which refers to a ProvidedService or RequiredService, and one of the Operations of its ServiceDefinition.

The purpose of the TriggerInstance is to define a periodic event generator internally to the component scope: the TriggerInstance will act as the sender of the event, at the specified period. The generator is handled by the container (e.g. an OS watchdog or an auto-generated invisible module which sends an event). This avoids the creation of event generation components which will break the inversion of control principle, as they will need to access to the OS to generate periodic events. This system allows the creation of several flows of periodic events in a synchronised way (if all events come from the same TriggerInstance), or in a non-synchronised way (if they come from different TriggerInstances). It also allows combining a periodic source of events with other, non-periodic sources.

The purpose of the DynamicTriggerInstance is to define a one-shot event generator internally to the component scope: the DynamicTriggerInstance will act as a sender of a valued event, within a given delay specified at runtime. The principle is to receive an event, named “in” event hereafter, and to send after a given delay an associated event, named “out” event. The first parameter of the “in” event is the delay. Other parameters can be any of the ECOA types. Multiple occurrences of the same event can be queued waiting for the delays to expire. A « reset » operation can purge all waiting event occurrences.

Note that:

- The same RequestReceived, EventReceived or EventSent operation of a module can be part of different ModuleOperationLinks at the same time. All other module operations (DataRead, DataWritten and RequestSent operations) exist in only one OperationLink.
- Each DataLink is associated to a Data that it represents and that is shared within the Application Software Component. It may connect several writers that can be component-internal writes or references. In any case, a reader gets access to the most recent value accessible on the platform.

6.1.7 Data Types

Data types are “portable types” and are only used to describe information transmitted on wires between components and operation links between modules. By using these types, information can be then serialized for transmission with the help of the ELI (see Reference 6). The way they are physically bound to a given processor is left to the platform provider based on language bindings (see References 3, 4, 5 and 10).

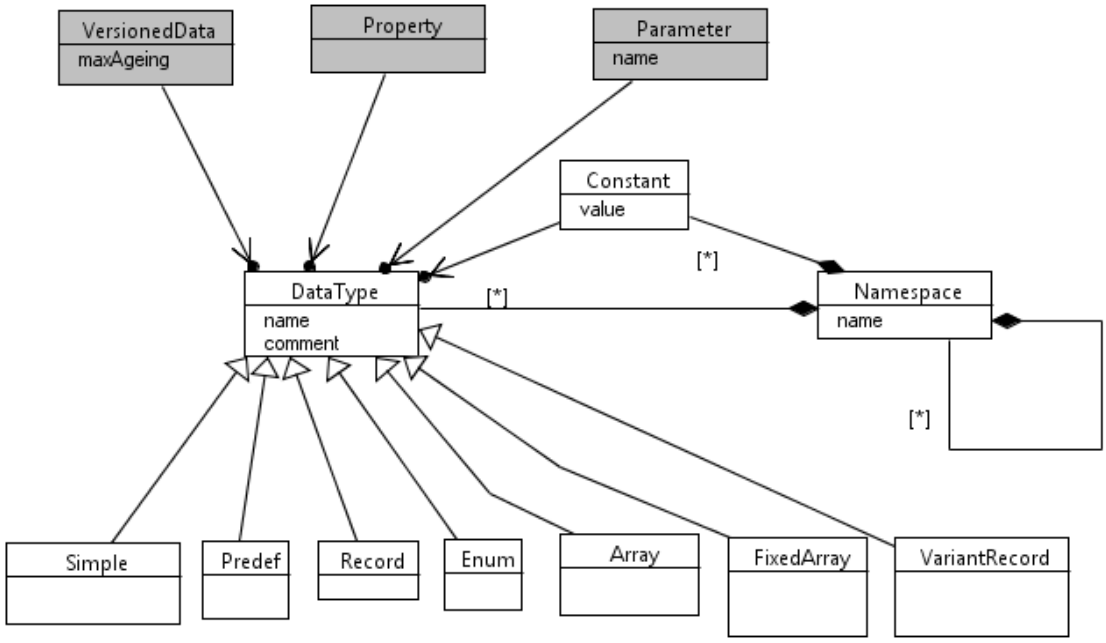

Figure 9 – DataType definition

A **DataType** is a language-neutral type definition. It is used as a shared definition, to help define ServiceDefinitions: it is referenced by VersionedData, and by Parameters of Events and Operations. It is also used to type Properties. A comment can describe the DataType.

A DataType definition describes the **NameSpace** in which it is located. A NameSpace is composed of DataTypes and NameSpaces, the different types are described in Figure 10 and Figure 11.

A **Constant** is a remarkable integer or floating-point value, identified with a name and located in a given NameSpace. A Constant may be of any of the Simple or Predef DataTypes.

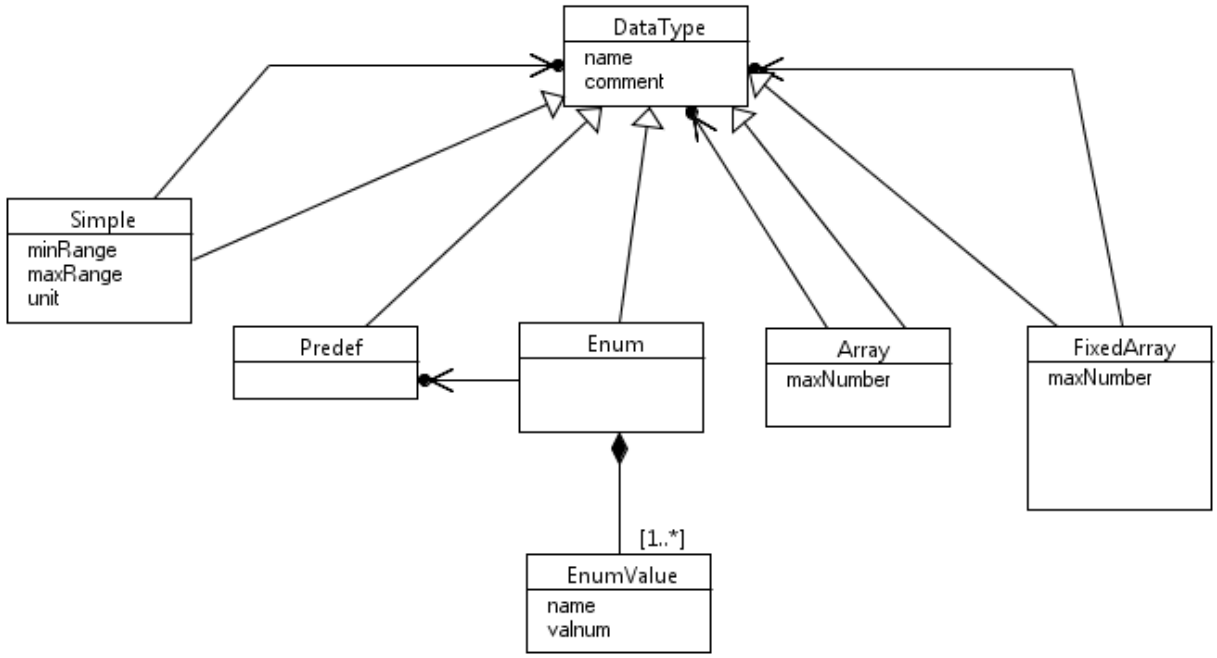

Figure 10 – Supported Data Types

A **Predef** type belongs to a fixed list of predefined types[1]: boolean8, char8, byte, int8, int16, int32, int64[2], uint8, uint16, uint32, float32, double64. 8-bit characters are encoded in ASCII. For boolean8, the value 1 means TRUE while the value 0 FALSE. float32 and double64 are IEEE754 compatible.

A **Simple** type is defined to give a meaningful name to a predef or another simple type. It can define range limits and a unit. Each limit can be a literal numeric, or a reference to a symbolic constant. The unit is functional and expressed as a string (e.g. 'second').

An **Enum** type is based on a predef type and defines the list of authorized values, **EnumValue**, each of which has a symbolic name. Each value can be a literal numeric, or a reference to a symbolic constant.

An **Array** defines a variable-capacity array, whose maximum capacity is fixed. All elements are of the same type. The maximum capacity can be a literal numeric, or a reference to a symbolic constant.

A **FixedArray** defines a fixed-capacity array. All elements are of the same type. The capacity can be a literal numeric, or a reference to a symbolic constant.

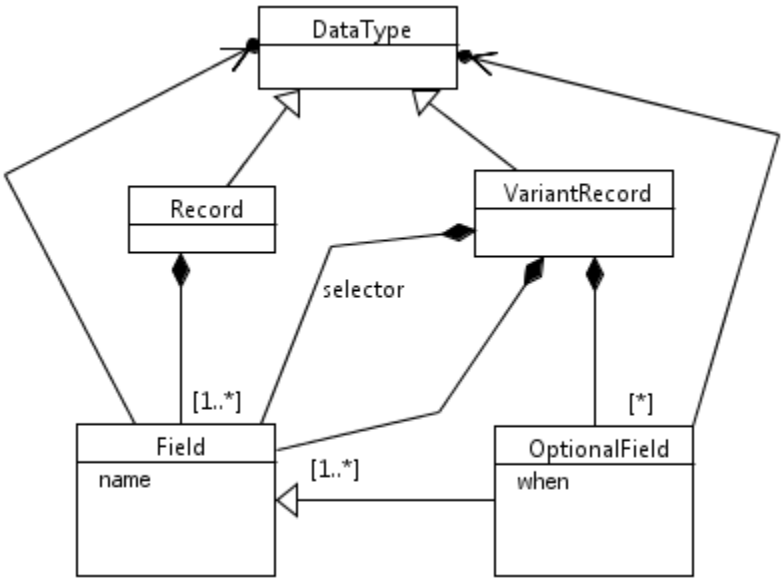

Figure 11 – Records and VariantRecords

A **Record** is a structure with named **Fields**, of any type.

A **VariantRecord** is like a Record, with a special field called the selector (of boolean, integer or enum type). Some of the fields, **OptionalField**, of a VariantRecord are optional: they are valued only when the selector has a certain value (given by the attribute "when").

Note that nested types are not allowed; i.e. it cannot be possible to define local types specific to a given field. All types used at field level must be defined prior to the record definition.

[1] The list of available types may be extended in the future as requirements evolve. For example, fixed-point types may be required.
[2] Uint64 is not supported for the moment since Java does not support it.

6.1.8 Deployment Schema and Logical System

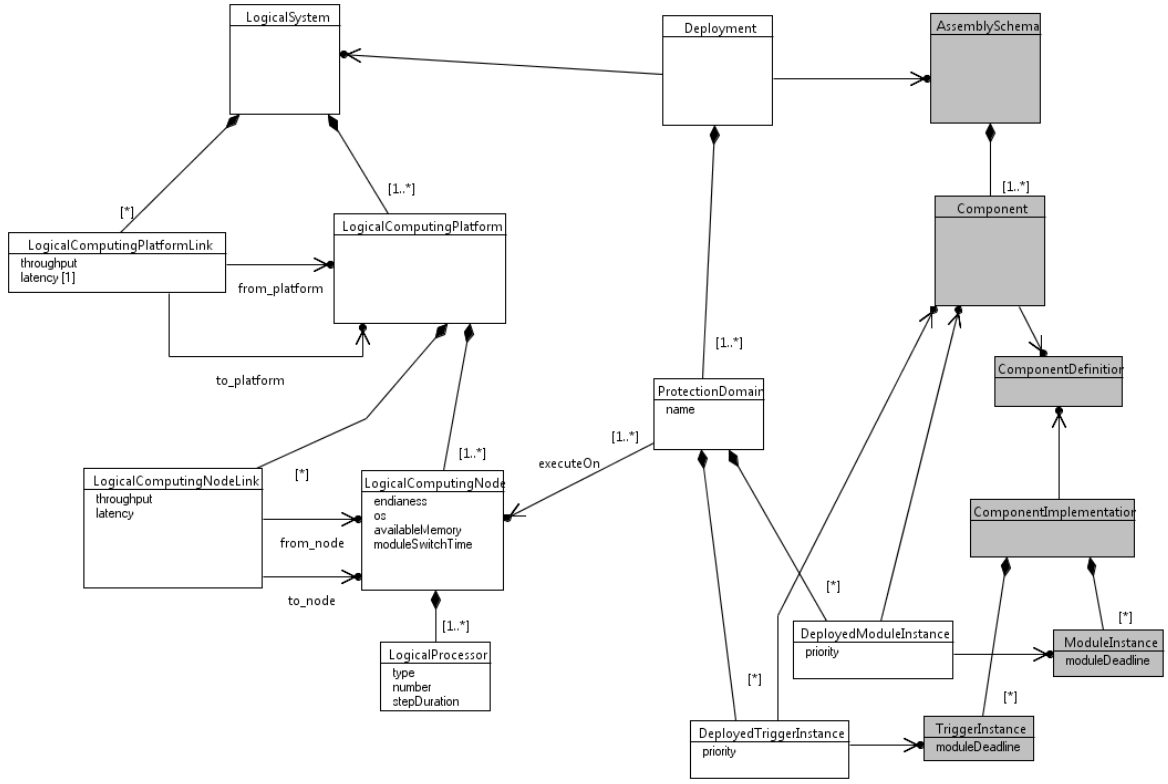

Figure 12 – Deployment Schema

A **Deployment Schema** refers to an Assembly Schema.

It contains the mapping of **ProtectionDomains** on **LogicalComputingNodes**.

A ProtectionDomain offers spatial isolation (memory protection), and possibly also temporal isolation (e.g. ARINC 653 partition scheduling), on a given LogicalComputingNode. It corresponds to the concept of process or partition, depending on the OS used.

Each ProtectionDomain hosts a number of ModuleInstances (which are referenced by the DeployedModuleInstance objects). All names of ModuleInstances hosted by a ProtectionDomain shall be unique within the ProtectionDomain scope.

The **priority** attribute is defined in DeployedModuleInstances which is calculated based on ModuleInstance ModuleDeadline and rate through considering the response times of operations. This will be further described in the Developer's Guide and is included here for completeness.

A LogicalComputingNode allows early verification of the performance of a system by providing an idealised model of a set of processors. This ideal processing resource is parameterised by a number of key attributes such as computing step, memory capacity, module switch time and number/standard of processors.

The initial model of a logical computing node chosen at this stage of the architecture definition is as a symmetric multiprocessor hosting one single OS image. It contains sets of **LogicalProcessors**; these may be heterogeneous if the OS provides an abstract interface. LogicalComputingNodes are linked together through **LogicalComputingNodeLinks** and they

constitute a **LogicalComputingPlatform**. LogicalComputingPlatforms may then be linked together through **LogicalComputingPlatformLinks** and they constitute a **LogicalSystem**.

The mapping from logical computing nodes to actual physical processors (or cores) is not defined in this issue of the document. It may be that it is not the same as the mapping from LogicalComputingNodes to LogicalProcessors. For example one physical processor or core might be used instead of multiple LogicalProcessors or vice-versa. The mapping of logical links onto actual physical buses is not also addressed. These mappings are provided through specific artefacts supplied by the platform provider.

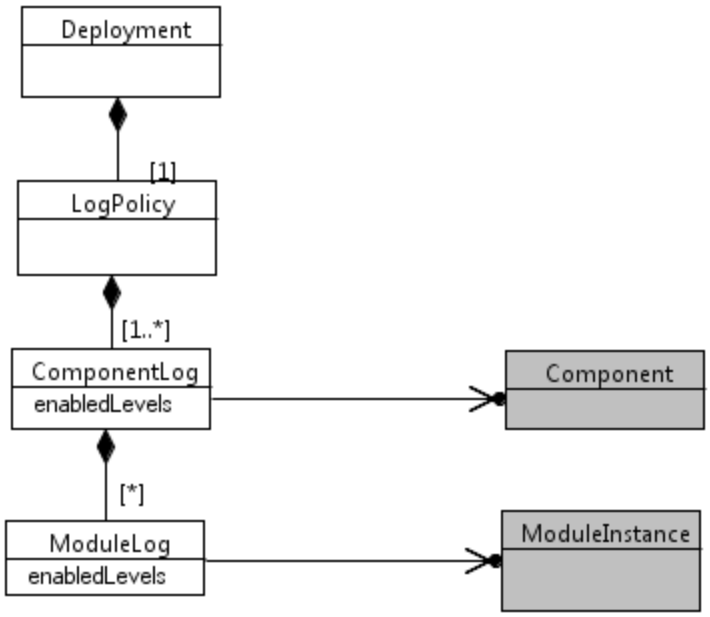

Figure 13 – Log Policy Definition

6.1.9 Behaviour Specification

This section is an initial proposal to specify the behaviours at component level, service level and at module level. Its maturity is low. It will be exercised and refined in future stages.

6.1.9.1 At component level

Component behaviour is used to define how a component behaves within a system. It will describe how it makes use of its required services based upon how its own provided services are used. For example the use of a service operation on a provided service may cause multiple service operations to be invoked on one or more of its required service.

Component behaviour will be captured during the design phase using UML notation.

6.1.9.2 At service level

Service behaviour deals with how service operations within services are meant to be used. It will define how to use one or more service operations and in which order. For example a service may

provide service operation A, B and C, but these must be used by firstly invoking operation A, followed by either operation B or C, but not both.

Service behaviour will be captured during the design phase using UML notation.

6.1.9.3 At module level

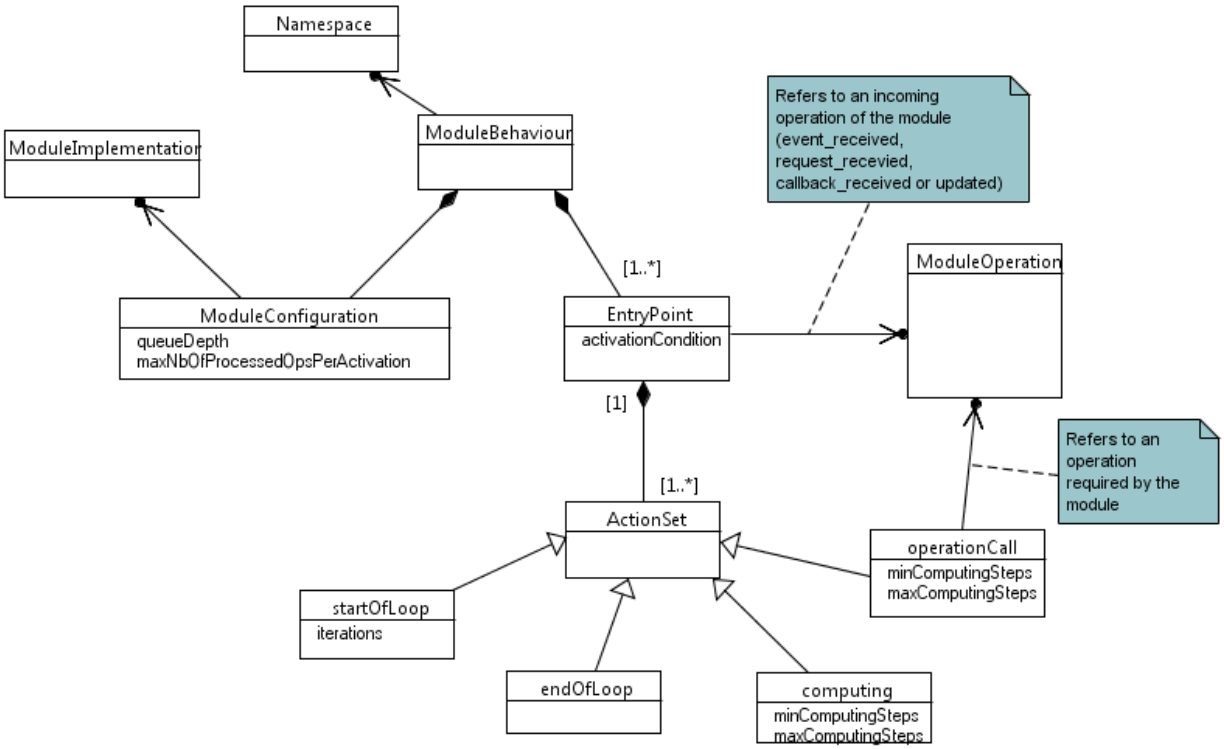

Figure 14 – Module Behaviour

A **moduleBehaviour** allows describing characteristics of a **ModuleImplementation** item, which will be used for deployment and early verification analyses (for example: consistency with the component level behaviour). It mainly gives a decomposition of module treatments, allowing an assessment of CPU resources requirements.

A **moduleBehaviour** is composed of a **moduleConfiguration** and a set of **entryPoints**.

The **moduleConfiguration** refers to the **ModuleImplementation** whose behaviour is described. It defines the maximum number of incoming operations (queueDepth) that can be queued for an instance of this moduleImplementation: note that only “activating” operations (i.e. event_received, request_received or callback of an asynchronous request_sent) are queued. Operations queue management behaviour is FIFO.

NB: a configuration attribute is also created as a provision to specify the maximum number of incoming operations that can be processed within a single module activation (in case an ECOA future model would allow to parameterize the “activating/not activating” property of incoming operations – in the current version, the value is always 1).

An **entryPoint** corresponds to a sequence of **Actions** to be processed on the reception of an activating incoming module operation (according to the rank of the operation in the module operations queue, and according to the scheduling policy of the execution platform). So, each **entryPoint** refers to its associated activating **ModuleOperation**.

The different types of **Actions** that can be specified in this “high level” entry-point behaviour are the following ones:

- startOfLoop (with the number of iterations to be processed),
- endOfLoop,
- computing : it is an internal treatment,
- operationCall : refers to a required module operation (to run a treatment on another module)

For the last two **Actions**, it is possible to define a minimum number of computing steps and a maximum number of computing steps, representing processing complexities for the best case and for the worst case execution scenarios of the related entryPoint. Complexity assessments, through “step” units, have to be consistent with the definition of “step” used to value the **LogicalComputingNode** “stepDuration” parameter (in the **LogicalSystem**).

This way, the “high level” entry-point behaviour allows assessing the Best Case Execution Time (BCET) and the Worst Case Execution Time (WCET) of each module entry-point.

It is important to notice that, in the case of an operation call, values of computing steps correspond to the internal cost of the operation call, and not to the cost of actions to be processed by the called module.

6.2 Concrete Meta-Model

6.2.1 Mapping onto Service-Component Architecture (SCA)

The Service-Component Architecture (SCA) is a standardised model for building applications and (software) systems using a Service-Oriented Architecture (SOA), developed by a set of industry partners. Initially developed as an industrial collaboration, this open standard is now reaching maturity and is maintained by the OASIS (http://www.oasis-open.org/) organisation. Using some SCA concepts and implementations avoids unnecessary re-implementation and potentially leverages existing tool support.

This section describes the translation of the abstract meta-model described in the previous section onto an XML meta-model based on the SCA assembly model. In fact, the XML meta-model re-describes all ECOA artefacts already described by the abstract meta-model but in a way usable by software tooling.

6.2.1.1 Rules on XML writing

Certain rules need to be followed to ensure that the XML is consistent and correct. The following rules are in addition to the normal validation requirements of the XML relative to its schema.

- Information names used within XML files are case sensitive. If the name of one item is used many times, character strings used for that name shall use the same case sensitivity.
- The parsing of XML files is done in one pass; i.e. items need to be defined before they are used. For example, the type for a field in a structure shall be defined before the definition of the structure.
- Each component implementation name must be unique within the protection domain hosting it
- Each component instance name must be unique within the assembly schema

- Each module implementation name must be unique across the assembly
- Each module instance name must be unique within the component implementation
- Each module implementation name must be unique within the protection domain hosting it
- Each operation name must be unique within each module definition
- Operation and module names must follow the naming conventions for identifiers used in the most common programming languages: a name being a sequence of letters, figures and underscores, beginning with a letter.
- The order of Operation Parameters in the Component Definition and Implementaiton must match the order declared in the Service Definition.

6.2.1.2 XPath Syntax

The syntax used to identify an element is the XPath one. XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:

Expression	Description
Nodename	Selects all child nodes of the named node
@	Selects attributes of the current node

Table 2 – XPath Expressions

More information can be found at http://www.w3schools.com/xpath/xpath_syntax.asp.

Nodename in XPath should be a NCName (a name which does not contain colon character) or a QName (prefix:localName where prefix is defined as a reference to a namespace elsewhere).

To avoid a prefix definition we add a new syntax: {namespace}localName where 'namespace' is equal to 'ecoa-sca' in the table below.

6.2.1.3 ECOA to SCA Mapping

ECOA Abstract item	SCA item
ServiceDefinition and service-level operations	Interface extension See Section 7.3 (ecoa-interface-1.0.xsd)
ComponentDefinition	componentType See Section 7.9(ecoa-interface-1.0.xsd)
Property	Property
Property / @name	property/@name
Property / @type	property/@{ecoa-sca}type
ProvidedService	Service
ProvidedService/@name	service/@name
ProvidedService / @ServiceDefinition	service/{ecoa-sca}interface/@syntax
RequiredService	Reference
RequiredService / @name	reference/@name
RequiredService / @ServiceDefinition	reference / {ecoa-sca}interface / @syntax
ComponentImplementation, module artefacts, module-level operations and promotion links	Implementation extension See Section 7.8 (ecoa-implementation-1.0.xsd)
Component	Component
Component / @name	component/@name
Component / @ComponentDefinitionRef	component/{ecoa-sca}instance/ @componentType
Component / @ComponentImplementationRef	component/{ecoa-sca}instance/ {ecoa-sca}implementation/@path
ServiceLink	Wire
ServiceLink / @ProvidedServiceRef	wire / @target
ServiceLink / @RequiredServiceRef	wire / @source
AssemblySchema	Composite See Section 7.4 (ecoa-sca-instance-1.0.xsd)
DeploymentSchema	Refinement of a composite. Computing nodes are described in separate XML files.
Data types	Specific description See Section 7.13 (ecoa-types-1.0.xsd)

Table 3 – Relations between the ECOA abstract meta-model and the SCA Assembly model

6.2.2 Schemas

The ECOA concrete meta-model references the following files produced by the OASIS organisation. Currently the ECOA metamodel is defined against version 1.1 of the SCA.

SCA (sca-policy-1.1-cd04.xsd) which is available from:

http://docs.oasis-open.org/opencsa/sca-assembly/sca-1.1-cd06.xsd

SCA core (sca-core-1.1-cd06.xsd) which is available from:

http://docs.oasis-open.org/opencsa/sca-assembly/sca-core-1.1-cd06.xsd

SCA contributions (sca-contribution-1.1-cd06.xsd) which is available from:

http://docs.oasis-open.org/opencsa/sca-assembly/sca-contribution-1.1-cd06.xsd

The ECOA metamodel only refers to a subset of the SCA concepts (see 6.2.1.3). It is so possible to comment out unused XSD entries in SCA schemas to validate the ECOA XML files.

The following table describes the ECOA schemas, which are presented in full in Section 7.

Filename	Description	Section
ecoa-sca-1.0.xsd	Required for compatibility with SCA. Defines SCA extension schemas.	7.1
ecoa-sca-attributes-1.0.xsd	Required for compatibility with SCA. Defines SCA attribute extensions.	7.2
ecoa-sca-interface-1.0.xsd	Describes reference to service definition at component level within the assembly	7.3
ecoa-sca-instance-1.0.xsd	Describes reference to component implementation description at component level within the assembly	7.4
ecoa-bin-desc-1.0.xsd	Defines the links between module implementations and binary objects.	7.5
ecoa-common-1.0.xsd	Declares the use of a library of data types.	7.6
ecoa-deployment-1.0.xsd	Defines how Modules are mapped onto a logical architecture (ie. protection domains and processing nodes)	7.7
ecoa-implementation-1.0.xsd	Describes all the information needed to integrate the software implementation of an ECOA component in an ECOA system.	7.8
ecoa-interface-1.0.xsd	Describes an ECOA service, including a set of operations.	7.9
ecoa-interface-qos.xsd	Describes the provided and required quality of service associated with a component	7.10

Filename	Description	Section
	definition.	
ecoa-logicalsystem-1.0.xsd	Describes a logical computing architecture consisting of computing nodes and protection domains connected by a network. This architecture description is intended to support early verification.	7.11
ecoa-module-behaviour-1.0.xsd	Describes the behaviour of module operations.	7.12
ecoa-types-1.0.xsd	Describes the syntax for defining ECOA types constructed from the basic ECOA predefined types.	7.13
ecoa-project-1.0.xsd	Describes directories used for one given ECOA application	7.14
ecoa-interface-behaviour-1.0.xsd	Describes the behaviour of a service.	7.15
ecoa-udpbinding-1.0.xsd	Describes the binding for UDP communication using ELI.	7.16
ecoa-uid-1.0.xsd	Describes directories used for one given ECOA application	7.17
sca-1.1-cd06.xsd	Service Component Architecture Schema Version 1.1	7.18
sca-contribution-1.1-cd06.xsd	Service Component Architecture Contribution Schema Version 1.1	7.19
sca-core-1.1-cd06.xsd	Service Component Architecture Core Schema Version 1.1	7.20
xml-schema.xsd	Describes the Schema for XML Schemas	7.21
xml.xsd	Describes the XML namespace, in a form suitable for import by other schema documents	7.22

Table 4 – ECOA Defined Schemas

6.2.3 Filename Conventions

Table 5 specifies standard filenames for the different instances of the ECOA concrete meta-model as generated by an ECOA toolset. It also defines the main XSD file associated to the kind of file and it can be used as an entry point within the concrete meta-model.

	ECOA Standard Filename	Comments	XSD
Project definition	`#filename#.project.xml`	General information about one ECOA application - Optional in current stages	ecoa-project-1.0.xsd
Type definitions	`#filename#.types.xml`	Data types used by operations within service or module definitions	ecoa-types-1.0.xsd
Service definition	`#filename#.interface.xml`	List of service operations Name required for conformance to SCA	ecoa-interface-1.0.xsd
Component definition	`#filename#.componentType`	List of services provided and required by the component and its properties. Name required for conformance to SCA	sca-1.1-cd06.xsd
Service QoS definition	`#filename#.interface.qos.xml`	Service operation-level QoS	ecoa-interface-qos-1.0.xsd
Component implementation	`#filename#.impl.xml`	Description of component architecture: modules, triggers, module operation links, etc	ecoa-implementation-1.0.xsd
Module behaviour	`#filename#.behaviour.xml`	File defining a macro temporal behaviour of the module	ecoa-module-behaviour-1.0.xsd
Initial assembly schema	`#filename#.composite`	Application architecture connecting component instances through wires. Decorrelated from any implementation Name required for conformance to SCA	sca-1.1-cd06.xsd
Final assembly schema	`#filename#.impl.composite`	This file adds to `#filename#.composite` pointers to component implementations. Name required for conformance to SCA	sca-1.1-cd06.xsd
Deployment schema	`#filename#.deployment.xml`	Mapping of modules onto computing nodes	ecoa-deployment-1.0.xsd

	ECOA Standard Filename	Comments	XSD
Mapping onto binary files	`#filename#.bin-desc.xml` `bin-desc.xml`	Mapping of logical module implementation names onto actual physical binary files Useful for packaging	ecoa-bin-desc-1.0.xsd
Logical System	`#filename#.logical-system.xml`	Description of the computing platforms: computing nodes, links between them and performance characteristics.	ecoa-logicalsystem-1.0.xsd

Table 5 – ECOA Standard Filenames

6.2.4 Interim data organisation

Within the development toolset all data describing the example are organized into files and directories.

Figure 15 shows an intermediate organisation used during early stages of the programme. This organisation might then evolved based on the optional ecoa-project file in future stages.

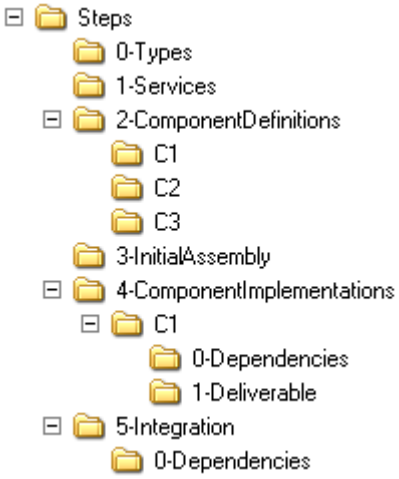

Figure 15 – Directories

Data types used for every definition are defined by "***.types.xml" located in the directory named "0-Types".

Service definitions are defined by "***.interface.xml" located in the directory named "1-Services".

Each Component Definition is described by a "***.componentType" file located in a sub-directory of the directory named "2-ComponentDefinitions". The name of the sub-directory is the name of the component definition itself. For each Component Definition, "***.interface-qos.xml" files describe the initial QoS expected for each service provided or required by an instance of this component definition.

The initial Assembly Schema is defined by a "***.composite" file located in the directory named "3-InitialAssembly".

Each Component Implementation is described by a "***.impl.xml" file located in a sub-directory of the directory named "4-ComponentImplementations". The name of the sub-directory is the name of the component implementation itself. The component supplier may also overload the expected with a new QoS; however, the new QoS shall be compatible with the expected one (e.g. an overloaded data maxageing can be lesser than the expected one). For each Component Implementation, "***.behaviour.xml" files describe the behaviour of each module operation. The file bin-desc.xml describes the list of binary objects associated to modules. computingPlaform and computingNode attributes of the element executeOn in the deployment XML file shall match id attributes of logicialComputingPlatform element and one of its logicalComputingNode child elements in the logical-system XML file. The values for these attributes are free character strings. It is not required to use fixed prefixes.

The directory "5-Integration" describes associations and mappings of software onto a logical system. The logical system is described by the file "logical-system.xml": it defines logical computing nodes and logical links between them. The association between the component instances and the component implementations is described by the "***.impl.composite". The grouping of modules into partition domains and the mapping of partition domains onto logical computing nodes is described by the file "deployment.xml". The actual deployment (fine grain deployment) is described by platform-specific files and shall be documented by the platform provider. These files are not described in this example. The file "sca-contribution.xml" is only there for compatibility with the SCA standard.

The following table summarizes text above.

Directory	**Sub-directory 1**	**Sub-directory 2**	**Files**
0-Types	N/A	N/A	***.types.xml
1-Services	N/A	N/A	***.interface.xml
2-ComponentDefinitions	<name_of_component_definition>		<name_of_component>.componentType ***.interface.qos.xml By example: required_<service_name>.qos.xml and provided_<service_name>.qos.xml
3-InitialAssembly	N/A		***.composite
4-ComponentImplementations	<name_of_implementation>		<name_of_implementation>.impl.xml ***.interface.qos.xml (e.g. new_required_<service_name>.qos.xml) bin-desc.xml ***.behaviour.xml Binary files (e.g. *.o or *.dll)
		0-Dependencies	Data type, service and component definitions if "0-Types", "1-Services" and "2-ComponentDefinitions" directories are not available.
		1-Deliverable	Zipped file of the upper directory
5-Integration	N/A		***.impl.composite logical-system.xml deployment.xml sca-contribution.xml
	0-Dependencies	N/A	Set of directories containing component implementations if 4-ComponentImplementations is not available

Table 6 – Model Data Organisation

Tools developed for the ECOA Phase 1 demonstrations may rely on this data organisation.

7 Schemas

7.1 ecoa-sca-1.0.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:ecoa-
sca="http://www.ecoa.technology/sca" xmlns:sca="http://docs.oasis-
open.org/ns/opencsa/sca/200912" targetNamespace="http://www.ecoa.technology/sca"
elementFormDefault="qualified">
      <import namespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
schemaLocation="sca-core-1.1-cd06.xsd"/>
      <include schemaLocation="extensions/ecoa-sca-instance-1.0.xsd"/>
      <include schemaLocation="extensions/ecoa-sca-interface-1.0.xsd"/>
      <include schemaLocation="ecoa-sca-attributes-1.0.xsd"/>

</schema>
```

7.2 ecoa-sca-attributes-1.0.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ecoa.technology/sca" elementFormDefault="qualified">
      <attribute name="rank" type="xs:int"/>
      <attribute name="allEventsMulticasted" type="xs:boolean" default="false">
        <annotation>
        <documentation>Boolean indicating if all events provided by the sender are
multicast or not</documentation>
      </annotation>
      </attribute>
      <attribute name="type" type="xs:string"/>
      <attribute name="deployment" type="xs:string"/>
      <complexType name="cia">
        <xs:annotation>
          <xs:documentation>PROVISIONAL Defines level of CIA required for a
wire</xs:documentation>
        </xs:annotation>
        <xs:attribute name="confidentiality" type="xs:string" use="required" />
        <xs:attribute name="integrity" type="xs:string" use="required" />
        <xs:attribute name="availability" type="xs:string" use="required" />
      </complexType>
</schema>
```

7.3 ecoa-sca-interface-1.0.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ecoa="http://www.ecoa.technology/sca" xmlns:sca="http://docs.oasis-
open.org/ns/opencsa/sca/200912"
 targetNamespace="http://www.ecoa.technology/sca"
  elementFormDefault="qualified"
    xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
            jxb:version="1.0"
  >

  <import namespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
schemaLocation="../sca-core-1.1-cd06.xsd" />

  <element name="interface" type="ecoa:Interface" substitutionGroup="sca:interface">
```

```
    <annotation>
      <appinfo>
        <jxb:class name="ecoaInterfaceElement" />
      </appinfo>
    </annotation>
  </element>

  <complexType name="Interface">
    <annotation>
      <appinfo>
        <jxb:class name="EcoaInterface" />
      </appinfo>
    </annotation>
    <complexContent>
      <extension base="sca:Interface">
        <attribute name="syntax" type="anyURI" use="required" />
        <attribute name="qos" type="anyURI" use="optional" />
        <attribute name="behaviour" type="anyURI" use="optional" />
      </extension>
    </complexContent>
  </complexType>

</schema>
```

7.4 ecoa-sca-instance-1.0.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ecoa="http://www.ecoa.technology/sca" xmlns:sca="http://docs.oasis-
open.org/ns/opencsa/sca/200912" targetNamespace="http://www.ecoa.technology/sca"
  elementFormDefault="qualified">

  <import namespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
schemaLocation="../sca-core-1.1-cd06.xsd" />

  <element name="instance" type="ecoa:Instance" substitutionGroup="sca:implementation"
/>

  <complexType name="Instance">
    <complexContent>
      <extension base="sca:Implementation">
           <sequence>
          <element name="implementation" minOccurs="0" maxOccurs="1">
            <complexType>
              <attribute name="name" type="string" use="required" />
            </complexType>
          </element>
        </sequence>
        <attribute name="componentType" type="anyURI" use="required" />
        <attribute name="version" type="string" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

</schema>
```

7.5 ecoa-bin-desc-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsd:schema
  targetNamespace="http://www.ecoa.technology/bin-desc-1.0"
  xmlns="http://www.ecoa.technology/bin-desc-1.0"
  xmlns:tns="http://www.ecoa.technology/bin-desc-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >

  <xsd:include schemaLocation="ecoa-common-1.0.xsd" />
  <xsd:element name="binDesc" type="BinDesc" />

  <xsd:complexType name="BinDesc">
    <xsd:annotation>
      <xsd:documentation>Links between module implementations and binary
objects</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="processorTarget" type="ProcessorTarget" />
      <xsd:element name="binaryModule" type="BinaryModule" maxOccurs="unbounded" />
    </xsd:sequence>
    <!-- the following attribute points to a logical name -->
    <xsd:attribute name="componentImplementation" type="NameId" use="required" />
  </xsd:complexType>

  <xsd:complexType name="ProcessorTarget">
    <xsd:annotation>
      <xsd:documentation>"Identification of the processor for which modules have been
compiled"</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="type" use="required">
      <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="x86_32" />
            <xsd:enumeration value="x86_64" />
            <xsd:enumeration value="e500_32" />
            <xsd:enumeration value="e500_64" />
            <xsd:enumeration value="ppc-eabi" />
          </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="BinaryModule">
    <xsd:annotation>
      <xsd:documentation>A set of named operations.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="reference" type="xsd:NCName" use="required">
      <xsd:annotation>
            <xsd:documentation>Name of the module implementation</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="object" type="xsd:anyURI" use="required" >
      <xsd:annotation>
            <xsd:documentation>Filename of the binary implementing the referenced
module</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="userContextSize" type="xsd:int" use="required" >
      <xsd:annotation>
```

```
            <xsd:documentation>Size in bytes of the module user
context</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="stackSize" type="xsd:int" use="required" >
      <xsd:annotation>
            <xsd:documentation>maximum size in bytes of the stack used by any module
entry point (including all sub-function calls)</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="heapSize" type="xsd:int" use="required" >
      <xsd:annotation>
            <xsd:documentation>maximum size in bytes of the heap (memory dynamically
allocated by the module binary itself: malloc or object instances)</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="checksum" type="xsd:string" use="required" >
      <xsd:annotation>
            <xsd:documentation>MD5sum of the binary</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>

  </xsd:complexType>

</xsd:schema>
```

7.6 ecoa-common-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >

    <!--
    The following regexps define what is allowed/forbidden for each kind of names used
in ECOA.
    They must take into account the contraints of different programming languages
    and development environments supported by ECOA (characters allowed in file names,
identifiers, etc.)

    NOTE: XML character classes (\i, \c, etc.) are intentionally avoided, because of
the complexity
    of their definition. -->

      <!--  Name of a library containing data types -->
      <!--  Note: The '.' character is used to structure libraries into hierarchical
namespaces
      (like Java packages). -->
      <xsd:simpleType name="LibraryName">
          <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Za-z][A-Za-z0-9_\.]*"></xsd:pattern>
          </xsd:restriction>
      </xsd:simpleType>

      <!--  Name that can be used as an identifier in ECOA models and in the source
code of ECOA components -->
      <!--  Note: Names starting with '_' are excluded from ECOA models. -->
      <xsd:simpleType name="NameId">
```

```
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[A-Za-z][A-Za-z0-9_\-]*"></xsd:pattern>
          </xsd:restriction>
        </xsd:simpleType>

        <!--  Name of a data type inside a library -->
        <xsd:simpleType name="TypeName">
            <xsd:restriction base="NameId">
            </xsd:restriction>
        </xsd:simpleType>

        <!--  Name of a type, possibly prefixed by the name of the library that defines
it. -->
        <!--  The prefix may be omitted only for predefined types. -->
        <!--  A type T defined in a library L will be denoted "L:T". -->
        <xsd:simpleType name="TypeQName">
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="([A-Za-z][A-Za-z0-9_\.]*:)?[A-Za-z][A-Za-z0-
9_]*"></xsd:pattern>
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:element name="use">

          <xsd:annotation>
                    <xsd:documentation>Declares the use of a library of data types. A
type T defined in a library L will be denoted "L:T". </xsd:documentation>
              </xsd:annotation>

          <xsd:complexType>
                    <xsd:attribute name="library" type="LibraryName" use="required" />
              </xsd:complexType>
      </xsd:element>

</xsd:schema>
```

7.7 ecoa-deployment-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/deployment-1.0"
  xmlns="http://www.ecoa.technology/deployment-1.0"
  xmlns:tns="http://www.ecoa.technology/deployment-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >
    <xsd:include schemaLocation="ecoa-common-1.0.xsd"/>

        <xsd:element name="deployment" type="Deployment">
              <xsd:key name="execnamekey">
                    <xsd:selector xpath="tns:protectionDomain"/>
                    <xsd:field xpath="@name"/>
              </xsd:key>
        </xsd:element>
        <xsd:complexType name="Deployment">
              <xsd:sequence>
                    <xsd:choice minOccurs="1" maxOccurs="unbounded">
```

```
                        <xsd:element name="logPolicy" type="LogPolicy"
minOccurs="0" maxOccurs="unbounded"/>
                      <xsd:element name="protectionDomain" type="ProtectionDomain"
maxOccurs="unbounded"/>
                  </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="finalAssembly" type="NameId" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Name of the composite referenced by this
deployment</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="logicalSystem" type="NameId" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Name of the logical system this
deployment is made on</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
      </xsd:complexType>
      <xsd:complexType name="ProtectionDomain">
            <xsd:annotation>
                  <xsd:documentation>Defines an OS executable, offering memory (and
possibly also temporal) protection</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="executeOn">
                        <xsd:complexType>
                              <xsd:attribute name="computingNode"
                                    type="NameId"  use="required"/>
                              <xsd:attribute name="computingPlatform" type="NameId"
use="optional" >
                     <xsd:annotation>
                                       <xsd:documentation>Id of a logical
system.</xsd:documentation>
                                 </xsd:annotation>
                              </xsd:attribute>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="deployedModuleInstance" minOccurs="0"
                        maxOccurs="unbounded">
                        <xsd:complexType>
                              <xsd:attribute name="componentName"
                                    type="NameId" use="required" />
                              <xsd:attribute name="moduleInstanceName"
                                    type="NameId" use="required" />
                                 <xsd:attribute name="modulePriority"
type="ModulePriority" use="required">
                     <xsd:annotation>
                           <xsd:documentation>abstract module priority that can be
used by the platform to map the module on an actual OS priority</xsd:documentation>
                        </xsd:annotation>
                     </xsd:attribute>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="deployedTriggerInstance" minOccurs="0"
                        maxOccurs="unbounded">
                        <xsd:complexType>
                              <xsd:attribute name="componentName"
```

```
                                        type="NameId" use="required" />
                                    <xsd:attribute name="triggerInstanceName"
                                        type="NameId" use="required" />
                                <xsd:attribute name="triggerPriority"
type="ModulePriority" use="required">
                                <xsd:annotation>
                                    <xsd:documentation>abstract trigger priority that can be
used by the platform to map the trigger on an actual OS priority</xsd:documentation>
                                </xsd:annotation>
                                    </xsd:attribute>
                                </xsd:complexType>
                            </xsd:element>
                            </xsd:choice>
                    </xsd:sequence>
                    <xsd:attribute name="name" type="NameId" use="required"/>
            </xsd:complexType>

        <xsd:simpleType name="ModulePriority">
            <xsd:restriction base="xsd:decimal">
              <xsd:minInclusive value="0" />
              <xsd:maxInclusive value="255"/>
            </xsd:restriction>
        </xsd:simpleType>

            <xsd:complexType name="LogPolicy">
              <xsd:annotation>
                  <xsd:documentation>Defines the log policy for deployed components and
modules</xsd:documentation>
              </xsd:annotation>
              <xsd:sequence>
                  <xsd:element name="componentLog" type="ComponentLog" minOccurs="0"
maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>

            <xsd:complexType name="ComponentLog">
              <xsd:annotation>
                <xsd:documentation>Defines default level of logging for a given
component</xsd:documentation>
                  </xsd:annotation>
                  <xsd:sequence>
                        <xsd:element name="moduleLog" type="ModuleLog" minOccurs="0"
maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="instanceName"     type="xsd:string" use="required"
/>
                  <xsd:attribute name="enabledLevels"    type="xsd:string" use="required"
/>
            </xsd:complexType>

            <xsd:complexType name="ModuleLog">
                  <xsd:annotation>
                        <xsd:documentation>Defines level of logging for a deployed module
instance</xsd:documentation>
                  </xsd:annotation>
                  <xsd:attribute name="instanceName" type="xsd:string" use="required" />
                  <xsd:attribute name="enabledLevels"    type="xsd:string" use="required"
/>
            </xsd:complexType>
```

```
</xsd:schema>
```

7.8 ecoa-implementation-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.ecoa.technology/implementation-1.0"
  xmlns:tns="http://www.ecoa.technology/implementation-1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ecoa.technology/implementation-1.0"
  elementFormDefault="qualified">
  <xsd:include schemaLocation="ecoa-common-1.0.xsd" />
  <xsd:element name="componentImplementation" type="ComponentImplementation">
    <!-- keys: name unicity constraints -->
    <xsd:key name="moduleTypekey">
      <xsd:selector xpath="tns:moduleType" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="moduleImplementationkey">
      <xsd:selector xpath="tns:moduleImplementation" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="moduleInstancekey">
      <xsd:selector xpath="tns:moduleInstance" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="triggerInstancekey">
      <xsd:selector xpath="tns:triggerInstance" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="dynamicTriggerInstancekey">
      <xsd:selector xpath="tns:dynamicTriggerInstance" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <!-- triggers, dynamicTriggers and ordinary modules must have distinct
      names -->
    <xsd:key name="moduleOrTriggerInstancekey">
      <xsd:selector
        xpath="tns:moduleInstance|tns:triggerInstance|tns:dynamicTriggerInstance" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <!-- The same operation shall appear only one time if present in the element
clients -->
    <xsd:key name="moduleInstanceClientRequestLinkkey">
      <xsd:selector xpath="tns:requestLink/tns:clients/tns:moduleInstance" />
      <xsd:field xpath="@instanceName" />
      <xsd:field xpath="@operationName" />
    </xsd:key>
    <xsd:key name="serviceClientRequestLinkkey">
      <xsd:selector xpath="tns:requestLink/tns:clients/tns:service" />
      <xsd:field xpath="@instanceName" />
      <xsd:field xpath="@operationName" />
    </xsd:key>
    <!-- keyrefs: constraints that a reference refers to a name defined in
      a key -->
    <xsd:keyref name="moduleInstancekeyRef" refer="moduleInstancekey">
      <xsd:selector xpath="*/*/tns:moduleInstance" />
      <xsd:field xpath="@instanceName" />
```

```
    </xsd:keyref>
    <xsd:keyref name="triggerInstancekeyRef" refer="triggerInstancekey">
      <xsd:selector xpath="tns:eventLink/tns:trigger" />
      <xsd:field xpath="@triggerInstance" />
    </xsd:keyref>
    <xsd:keyref name="dynamicTriggerInstancekeyRef" refer="dynamicTriggerInstancekey">
      <xsd:selector xpath="tns:eventLink/*/tns:dynamicTrigger"/>
      <xsd:field xpath="@instanceName"/>
    </xsd:keyref>
    <xsd:keyref name="moduleImplementation_to_moduleType"
      refer="moduleTypekey">
      <xsd:selector xpath="tns:moduleImplementation" />
      <xsd:field xpath="@moduleType" />
    </xsd:keyref>
    <xsd:keyref name="moduleInstance_to_moduleImplementation"
      refer="moduleImplementationkey">
      <xsd:selector xpath="tns:moduleInstance" />
      <xsd:field xpath="@implementationName"/>
    </xsd:keyref>
  </xsd:element>
  <xsd:complexType name="ComponentImplementation">
    <xsd:annotation>
      <xsd:documentation>
        Describes all the information needed to integrate
        the software implementation
        of
        an ECOA component in an ECOA system.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="use" minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="service" type="ServiceQoS" minOccurs="0"
        maxOccurs="unbounded" />
      <xsd:element name="reference" type="ServiceQoS"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="moduleType" type="ModuleType"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="moduleImplementation" type="ModuleImplementation"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="moduleInstance" type="ModuleInstance"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="triggerInstance" type="TriggerInstance"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="dynamicTriggerInstance" type="DynamicTriggerInstance"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="dataLink" type="DataLink" minOccurs="0"
          maxOccurs="unbounded" />
        <xsd:element name="eventLink" type="EventLink"
          minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="requestLink" type="RequestLink"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="componentDefinition" type="NameId"
      use="required" />
  </xsd:complexType>
  <xsd:complexType name="ServiceQoS">
    <xsd:annotation>
```

```
      <xsd:documentation>To define a new QoS for a provided or required
        service
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="NameId" use="required" />
    <xsd:attribute name="newQoS" type="xsd:anyURI" use="required" />
  </xsd:complexType>
  <xsd:complexType name="ModuleType">
    <xsd:annotation>
      <xsd:documentation>Describes a single-threaded ECOA module,
        implemented as
        software, contributing to the implementation of
        an ECOA
        component.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="properties" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Set of module properties. The value of each module
property is set at design time.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="property" type="Parameter" maxOccurs="unbounded" >
              <xsd:annotation>
                <xsd:documentation>The value of each module property is set at design
time at instance definition level.</xsd:documentation>
              </xsd:annotation>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:unique name="propertyname">
          <xsd:selector xpath="tns:property" />
          <xsd:field xpath="@name" />
        </xsd:unique>
      </xsd:element>
      <xsd:element name="operations">
        <xsd:complexType>
          <xsd:choice maxOccurs="unbounded">
            <xsd:element name="dataWritten" type="VersionedData"
              minOccurs="0">
              <xsd:annotation>
                <xsd:documentation>Read+Write access to a versioned data.
                </xsd:documentation>
              </xsd:annotation>
            </xsd:element>
            <xsd:element name="dataRead" minOccurs="0">
              <xsd:annotation>
                <xsd:documentation>Read-only access to a versioned data.
                </xsd:documentation>
              </xsd:annotation>
              <xsd:complexType>
                <xsd:complexContent>
                  <xsd:extension base="VersionedData">
                    <!-- This deadline is associated to the notifying case -->
                    <xsd:attribute name="notificationDeadline" type="xsd:double"
                      use="optional" />
                    <xsd:attribute name="notifying" type="xsd:boolean"
```

```
                    use="optional" default="false" />
                </xsd:extension>
              </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="eventSent" type="Event" minOccurs="0">
            <xsd:unique name="eventparameter_sent">
              <xsd:selector xpath="tns:input" />
              <xsd:field xpath="@name" />
            </xsd:unique>
          </xsd:element>
          <xsd:element name="eventReceived" minOccurs="0">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="Event">
                  <xsd:attribute name="deadline" type="xsd:double"
                    use="optional" />
                </xsd:extension>
              </xsd:complexContent>
            </xsd:complexType>
            <xsd:unique name="eventparameter_received">
              <xsd:selector xpath="tns:input" />
              <xsd:field xpath="@name" />
            </xsd:unique>
          </xsd:element>
          <xsd:element name="requestSent" minOccurs="0">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="RequestResponse">
                  <xsd:attribute name="callbackDeadline" type="xsd:double"
                    use="optional" />
                  <xsd:attribute name="isSynchronous" type="xsd:boolean"
                    use="required" />
                </xsd:extension>
              </xsd:complexContent>
            </xsd:complexType>
            <xsd:unique name="requestparameter_req">
              <xsd:selector xpath="tns:input|tns:output" />
              <xsd:field xpath="@name" />
            </xsd:unique>
          </xsd:element>
          <xsd:element name="requestReceived" minOccurs="0">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="RequestResponse">
                  <xsd:attribute name="deadline" type="xsd:double"
                    use="optional" />
                  <xsd:attribute name="immediateResponse" type="xsd:boolean"
                    use="optional" default="true">
                    <xsd:annotation>
                      <xsd:documentation>Is the response immediatly sent at the
                        end of request entry-point execution ?
                      </xsd:documentation>
                    </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="maxDeferredResponses"
type="xsd:positiveInteger"
                    use="optional" default="10">
                    <xsd:annotation>
```

```
                        <xsd:documentation>Max number of deferred responses kept at
the same time.
                        </xsd:documentation>
                      </xsd:annotation>
                    </xsd:attribute>
                  </xsd:extension>
                </xsd:complexContent>
              </xsd:complexType>
              <xsd:unique name="requestparameter_pro">
                <xsd:selector xpath="tns:input|tns:output" />
                <xsd:field xpath="@name" />
              </xsd:unique>
            </xsd:element>
          </xsd:choice>
        </xsd:complexType>
        <xsd:key name="operationkey">
          <xsd:selector xpath="tns:*" />
          <xsd:field xpath="@name" />
        </xsd:key>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="NameId" use="required" />
    <xsd:attribute name="isSupervisionModule" type="xsd:boolean" use="optional"
default="false"/>
  </xsd:complexType>
  <xsd:complexType name="Event">
    <xsd:sequence>
      <xsd:element name="input" type="Parameter" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="NameId" use="required" />
  </xsd:complexType>
  <xsd:complexType name="RequestResponse">
    <xsd:sequence>
      <xsd:element name="input" type="Parameter" minOccurs="0"
        maxOccurs="unbounded" />
      <xsd:element name="output" type="Parameter" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="NameId" use="required" />
  </xsd:complexType>
  <xsd:complexType name="VersionedData">
    <xsd:attribute name="name" type="NameId" use="required" />
    <xsd:attribute name="type" type="TypeQName" use="required">
      <xsd:annotation>
        <xsd:documentation>Type stored by the versioned data.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="maxVersions" type="xsd:positiveInteger"
      use="optional" default="1">
      <xsd:annotation>
        <xsd:documentation>Max number of versions accessed at the same
          time.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="Parameter">
```

```
    <xsd:annotation>
      <xsd:documentation>A parameter a an operation (Event,
        RequestResponse
        or
        VersionedData)
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="NameId" use="required" />
    <xsd:attribute name="type" type="TypeQName" use="required" />
  </xsd:complexType>
  <xsd:complexType name="ModuleImplementation">
    <xsd:attribute name="name" type="NameId" use="required" />
    <xsd:attribute name="language" use="required">
      <!-- Programming language -->
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="C" />
          <xsd:enumeration value="C++" />
          <xsd:enumeration value="Ada" />
          <xsd:enumeration value="Java" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="moduleType" type="NameId" use="required" />
    <xsd:attribute name="activationModel" default="reactive">
      <!-- Activation model for activating operations management -->
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="reactive" />
          <xsd:enumeration value="rhythmic" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="moduleBehaviour" type="xsd:anyURI"
      use="optional" />
  </xsd:complexType>
  <xsd:complexType name="Instance">
    <xsd:annotation>
      <xsd:documentation></xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="NameId" use="required"/>
    <xsd:attribute name="moduleDeadline" type="xsd:double" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="ModuleInstance">
    <xsd:annotation>
      <xsd:documentation>Describes an instance of a Module (having its
        own
        internal state).
      </xsd:documentation>
    </xsd:annotation>
     <xsd:complexContent>
      <xsd:extension base="Instance">
      <xsd:sequence>
        <xsd:element name="highestRate" type="ModuleRate" minOccurs="0" />
        <xsd:element name="propertyValues" type="PropertyValues" minOccurs="0"
maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="implementationName" type="NameId" use="required" />
      </xsd:extension>
```

```
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ModuleRate">
  <xsd:attribute name="numberOfOccurrences" type="xsd:decimal"
    use="optional">
    <xsd:annotation>
      <xsd:documentation>Max number of received activating operations
        during a specified duration
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="timeFrame" type="xsd:double" use="optional">
    <xsd:annotation>
      <xsd:documentation>Equal to min inter-arrival time between 2
        activating operations when numberOfOccurrences value is 1. In other
        cases, specifies a sizing duration for activating operations
        bursts.Unit is second.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="PropertyValues">
  <xsd:annotation>
    <xsd:documentation>set of module property values</xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="propertyValue" type="PropertyValue" maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PropertyValue">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="name" type="xsd:string" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="DataLink">
  <xsd:annotation>
    <xsd:documentation>Link between DATA operations.</xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="writers">
      <xsd:complexType>
        <xsd:sequence maxOccurs="unbounded">
          <xsd:choice>
            <xsd:element name="reference" type="OpRef" />
            <xsd:element name="moduleInstance" type="OpRef" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="readers">
      <xsd:complexType>
        <xsd:sequence maxOccurs="unbounded">
          <xsd:choice>
```

```
            <xsd:element name="service" type="OpRef" />
            <xsd:element name="moduleInstance" type="OpRefActivatingFifo" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="defaultvalue" type="xsd:string"
      minOccurs="0">
      <xsd:annotation>
        <xsd:documentation>The value that consumers will read, if they
          read
          before any production has occured
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:int" use="optional" />
</xsd:complexType>
<xsd:complexType name="EventLink">
  <xsd:annotation>
    <xsd:documentation>Link between EVENT operations.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="senders" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence maxOccurs="unbounded">
          <xsd:choice>
            <xsd:element name="service" type="OpRef" />
            <xsd:element name="reference" type="OpRef" />
            <xsd:element name="moduleInstance" type="OpRef" />
            <xsd:element name="trigger" type="OpRef_Trigger" />
            <xsd:element name="dynamicTrigger" type="OpRef" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="receivers">
      <xsd:complexType>
        <xsd:sequence maxOccurs="unbounded">
          <xsd:choice>
            <xsd:element name="service" type="OpRef" />
            <xsd:element name="reference" type="OpRef" />
            <xsd:element name="moduleInstance" type="OpRefActivatingFifo" />
            <xsd:element name="dynamicTrigger" type="OpRef" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:int" use="optional" />
</xsd:complexType>
<xsd:complexType name="RequestLink">
  <xsd:annotation>
    <xsd:documentation>Link between RR operations. Must have exactly one
      server. Can have many clients.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
```

```
    <xsd:element name="clients">
      <xsd:complexType>
        <xsd:sequence maxOccurs="unbounded">
          <xsd:choice>
            <xsd:element name="service" type="OpRef" />
            <xsd:element name="moduleInstance" type="OpRefActivatingFifo">
              <xsd:annotation>
                <xsd:documentation>Note: attributes 'activating' and 'fifoSize'
                  concern the response, and are applicable to asynchronous RR
                  operations only.
                </xsd:documentation>
              </xsd:annotation>
            </xsd:element>
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="server">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element name="reference" type="OpRef" />
          <xsd:element name="moduleInstance" type="OpRefServer">
            <xsd:annotation>
              <xsd:documentation>Note: optional attributes concern the request
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:int" use="optional" />
</xsd:complexType>
<xsd:complexType name="OpRef">
  <xsd:attribute name="instanceName" type="NameId" use="required">
    <xsd:annotation>
      <xsd:documentation>Reference to a module instance, a service, or a
        reference
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="operationName" type="NameId" use="required" />
</xsd:complexType>
<xsd:complexType name="OpRefActivating">
  <xsd:complexContent>
    <xsd:extension base="OpRef">
      <xsd:attribute name="activating" type="xsd:boolean"
        use="optional" default="true">
        <xsd:annotation>
          <xsd:documentation>Does the reception of the event/data/rr cause
            the activation of the receiver module ?
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="OpRefServer">
  <xsd:complexContent>
```

```
    <xsd:extension base="OpRefActivatingFifo">
      <xsd:attribute name="requestBufferSize" type="xsd:positiveInteger"
        use="optional" default="1">
        <xsd:annotation>
          <xsd:documentation>Maximum number of requests that can be
            simultaneously processed (when immediateResponse is false)
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="OpRefActivatingFifo">
  <xsd:complexContent>
    <xsd:extension base="OpRefActivating">
      <xsd:attribute name="fifoSize" type="xsd:int" use="optional" default="8">
        <xsd:annotation>
          <xsd:documentation>Max number of events that can be stored in the
            receiver module's FIFO queue, before the activation of the
            corresponding entrypoint.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="OpRef_Trigger">
  <xsd:attribute name="instanceName" type="NameId" use="required" />
  <xsd:attribute name="period" type="xsd:double" use="required">
    <xsd:annotation>
      <xsd:documentation>period in seconds</xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="TriggerInstance">
  <xsd:complexContent>
    <xsd:extension base="Instance">
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DynamicTriggerInstance">
  <xsd:complexContent>
    <xsd:extension base="Instance">
  <xsd:sequence>
    <xsd:element name="parameter" type="Parameter" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="size" type="xsd:int" use="optional"
    default="1">
    <xsd:annotation>
      <xsd:documentation>Max number of events waiting for delay expiration
        in the trigger
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="delayMin" type="xsd:double" use="optional"
    default="0.0">
    <xsd:annotation>
      <xsd:documentation>The trigger will not accept delays lower that
```

```
          this value (in seconds)
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="delayMax" type="xsd:double" use="optional">
      <xsd:annotation>
        <xsd:documentation>
          The trigger will not accept delays higher that
          this value (in  seconds)
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

7.9 ecoa-interface-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/interface-1.0"
  xmlns="http://www.ecoa.technology/interface-1.0"
  xmlns:tns="http://www.ecoa.technology/interface-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >
       <xsd:include schemaLocation="ecoa-common-1.0.xsd"/>
       <xsd:element name="serviceDefinition" type="ServiceDefinition"/>
       <xsd:complexType name="ServiceDefinition">
              <xsd:annotation>
                     <xsd:documentation>The definition of an ECOA service, including a
set of operations.</xsd:documentation>
              </xsd:annotation>
              <xsd:sequence>
                     <xsd:element ref="use" minOccurs="0" maxOccurs="unbounded"/>
                     <xsd:element name="operations" type="Operations"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="Operations">
              <xsd:annotation>
                     <xsd:documentation>A set of named operations.</xsd:documentation>
              </xsd:annotation>
              <xsd:choice minOccurs="0" maxOccurs="unbounded">
                     <xsd:element name="data" type="Data"/>
                     <xsd:element name="event" type="Event"/>
                     <xsd:element name="requestresponse" type="RequestResponse"/>
              </xsd:choice>
       </xsd:complexType>
    <xsd:complexType name="Operation" abstract="true">
      <xsd:attribute name="name" type="NameId" use="required"/>
      <xsd:attribute name="comment" type="xsd:string" use="optional"/>
    </xsd:complexType>
       <xsd:complexType name="Data">
              <xsd:annotation>
                     <xsd:documentation>Use of the "versionned data" exchange
mechanism.</xsd:documentation>
              </xsd:annotation>
```

```
	<xsd:complexContent>
	<xsd:extension base="Operation">
		<xsd:attribute name="type" type="TypeQName" use="required"/>
	</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	<xsd:complexType name="Event">
		<xsd:annotation>
			<xsd:documentation>Use of the "event" exchange
mechanism.</xsd:documentation>
		</xsd:annotation>
	    <xsd:complexContent>
	      <xsd:extension base="Operation">
			<xsd:sequence>
				<xsd:element name="input" type="Parameter" minOccurs="0"
maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="direction" type="E_EventDirection"
use="required"/>
		</xsd:extension>
	</xsd:complexContent>
		</xsd:complexType>
	<xsd:complexType name="RequestResponse">
		<xsd:annotation>
			<xsd:documentation>Use of the "request-response" exchange
mechanism.</xsd:documentation>
		</xsd:annotation>
	<xsd:complexContent>
		<xsd:extension base="Operation">
				<xsd:sequence>
				<xsd:element name="input" type="Parameter" minOccurs="0"
maxOccurs="unbounded"/>
				<xsd:element name="output" type="Parameter" minOccurs="0"
maxOccurs="unbounded"/>
				</xsd:sequence>
		</xsd:extension>
	</xsd:complexContent>
	</xsd:complexType>
	<xsd:simpleType name="E_EventDirection">
		<xsd:restriction base="xsd:string">
			<xsd:enumeration value="SENT_BY_PROVIDER"/>
			<xsd:enumeration value="RECEIVED_BY_PROVIDER"/>
		</xsd:restriction>
	</xsd:simpleType>
	<xsd:complexType name="Parameter">
		<xsd:attribute name="name" type="NameId" use="required"/>
		<xsd:attribute name="type" type="TypeQName" use="required"/>
	</xsd:complexType>
</xsd:schema>
```

7.10 ecoa-interface-qos-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp2 (http://www.altova.com) by DA (DASSAULT AVIATION) --
>
<xsd:schema xmlns="http://www.ecoa.technology/interface-qos-1.0"
xmlns:tns="http://www.ecoa.technology/interface-qos-1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
targetNamespace="http://www.ecoa.technology/interface-qos-1.0"
elementFormDefault="qualified">
	<xsd:include schemaLocation="ecoa-common-1.0.xsd"/>
	<xsd:element name="serviceInstanceQoS" type="ServiceInstanceQoS"/>
	<xsd:complexType name="ServiceInstanceQoS">
		<xsd:annotation>
			<xsd:documentation>The definition of an ECOA service, including a
set of operations.</xsd:documentation>
		</xsd:annotation>
		<xsd:sequence minOccurs="1" maxOccurs="1">
			<xsd:element name="operations" type="Operations"/>
		</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="Operations">
		<xsd:annotation>
			<xsd:documentation>A set of named operations.</xsd:documentation>
		</xsd:annotation>
		<xsd:choice minOccurs="1" maxOccurs="unbounded">
				<xsd:element name="data" type="Data"/>
				<xsd:element name="event" type="Event"/>
				<xsd:element name="requestresponse"
type="RequestResponse"/>
		</xsd:choice>
	</xsd:complexType>
	<xsd:complexType name="Data">
		<xsd:annotation>
			<xsd:documentation>Use of the "versionned data" exchange
mechanism.</xsd:documentation>
		</xsd:annotation>
		<xsd:sequence>
			<xsd:element name="highestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
				<xsd:annotation>
					<xsd:documentation>Max number of occurrences within a
reference time frame</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
			<xsd:element name="lowestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
				<xsd:annotation>
					<xsd:documentation>Min number of occurrences within a
reference time frame</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
		</xsd:sequence>
		<xsd:attribute name="name" type="NameId" use="required"/>
		<xsd:attribute name="maxAgeing" type="xsd:double" use="optional">
			<xsd:annotation>
				<xsd:documentation>Operation Provided : max duration
between Data production (from the source) and the end of writing process. Operation
Required : max duration between Data production (from the source) and the end of
reading process. Unit is second.</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
		<xsd:attribute name="notificationMaxHandlingTime" type="xsd:double"
use="optional">
			<xsd:annotation>
				<xsd:documentation>Notifying data case: maxHandlingTime for
notification event. Unit is second.</xsd:documentation>
```

```
                </xsd:annotation>
            </xsd:attribute>
      </xsd:complexType>
      <xsd:complexType name="Event">
            <xsd:annotation>
                  <xsd:documentation>Use of the "event" exchange
mechanism.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="highestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
                        <xsd:annotation>
                              <xsd:documentation>Max number of occurrences within a
reference time frame</xsd:documentation>
                        </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="LowestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
                        <xsd:annotation>
                              <xsd:documentation>Min number of occurrences within a
reference time frame</xsd:documentation>
                        </xsd:annotation>
                  </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="NameId" use="required"/>
            <xsd:attribute name="maxHandlingTime" type="xsd:double" use="optional">
                  <xsd:annotation>
                        <xsd:documentation>Event Sent : specifies an intent on
receivers for
          maximal duration between Event Reception and end of related processing
          Event Received : maximal duration between Event Received and end of
          related processing. Unit is second.
        </xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
      </xsd:complexType>
      <xsd:complexType name="RequestResponse">
            <xsd:annotation>
                  <xsd:documentation>Use of the "request-reply" exchange
mechanism.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="highestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
                        <xsd:annotation>
                              <xsd:documentation>Max number of occurrences within a
reference time frame</xsd:documentation>
                        </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="LowestRate" type="OperationRate" minOccurs="0"
maxOccurs="1">
                        <xsd:annotation>
                              <xsd:documentation>Min number of occurrences within a
reference time frame</xsd:documentation>
                        </xsd:annotation>
                  </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="NameId" use="required"/>
            <xsd:attribute name="maxResponseTime" type="xsd:double" use="optional">
```

```
                        <xsd:annotation>
                              <xsd:documentation>Operation Provided : maximal duration
between Request Reception and Callback Sent
                              Operation Required : maximal duration between Request Sent
and Callback reception. Unit is second.</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="callbackMaxHandlingTime" type="xsd:double"
use="optional">
                        <xsd:annotation>
                              <xsd:documentation>maxHandlingTime to execute the callback
entry-point. Unit is second.</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
            </xsd:complexType>
            <xsd:complexType name="OperationRate">
                  <xsd:attribute name="numberOfOccurrences" type="xsd:decimal"
use="optional">
                        <xsd:annotation>
                              <xsd:documentation>Min or max number of operations
occurring during a specified duration</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="timeFrame" type="xsd:double" use="optional">
                        <xsd:annotation>
                              <xsd:documentation>Equal to min or max inter-arrival time
when NumberOfOccurrences value is 1.
                              In other cases, specifies a sizing duration for operations
bursts. Unit is second.</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
            </xsd:complexType>
</xsd:schema>
```

7.11 ecoa-logicalsystem-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/logicalsystem-1.0"
  xmlns="http://www.ecoa.technology/logicalsystem-1.0"
  xmlns:tns="http://www.ecoa.technology/logicalsystem-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >
  <xsd:element name="logicalSystem">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="logicalComputingPlatform" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
              <xsd:element name="logicalComputingNode" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="endianess">
                      <xsd:complexType>
                        <xsd:attribute name="type" use="required">
                          <xsd:simpleType>
                            <xsd:restriction base="xsd:string">
                              <xsd:enumeration value="BIG"/>
```

```
                        <xsd:enumeration value="LITTLE"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="logicalProcessors" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="stepDuration">
                      <xsd:complexType>
                        <xsd:attribute name="nanoSeconds" type="xsd:integer"
use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="type" use="required">
                    <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="x86_32"/>
                        <xsd:enumeration value="x86_64"/>
                        <xsd:enumeration value="e500_32"/>
                        <xsd:enumeration value="e600_32"/>
                        <xsd:enumeration value="powerpc"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
                  <xsd:attribute name="number" type="xsd:integer"
use="required"/>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="os">
                <xsd:complexType>
                  <xsd:attribute name="name" use="required">
                    <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="windows"/>
                        <xsd:enumeration value="linux"/>
                        <xsd:enumeration value="fastos"/>
                        <xsd:enumeration value="ims-vxworks"/>
                        <xsd:enumeration value="ima-integrity"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
                  <xsd:attribute name="version" type="xsd:string"
use="optional"/>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="availableMemory">
                <xsd:complexType>
                  <xsd:attribute name="gigaBytes" type="xsd:integer"
use="required"/>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="moduleSwitchTime">
                <xsd:complexType>
                  <xsd:attribute name="microSeconds" type="xsd:integer"
use="required"/>
                </xsd:complexType>
```

```
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="id" type="xsd:string" use="required"/>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="logicalComputingNodeLinks" minOccurs="0"
maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="link" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="throughput">
                            <xsd:complexType>
                              <xsd:attribute name="megaBytesPerSecond"
type="xsd:integer" use="required"/>
                            </xsd:complexType>
                          </xsd:element>
                          <xsd:element name="latency">
                            <xsd:complexType>
                              <xsd:attribute name="microSeconds" type="xsd:integer"
use="required"/>
                            </xsd:complexType>
                          </xsd:element>
                        </xsd:sequence>
                        <xsd:attribute name="id" type="xsd:ID"/>
                        <xsd:attribute name="to" type="xsd:string" use="required"/>
                        <xsd:attribute name="from" type="xsd:string" use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:ID" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="logicalComputingPlatformLinks" minOccurs="0"
maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="link" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="throughput">
                      <xsd:complexType>
                        <xsd:attribute name="megaBytesPerSecond" type="xsd:integer"
use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="latency">
                      <xsd:complexType>
                        <xsd:attribute name="microSeconds" type="xsd:integer"
use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="id" type="xsd:ID"/>
                  <xsd:attribute name="to" type="xsd:string" use="required"/>
```

```
              <xsd:attribute name="from" type="xsd:string" use="required"/>
            </xsd:complexType>
           </xsd:element>
         </xsd:sequence>
       </xsd:complexType>
     </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="required"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

7.12 ecoa-module-behaviour-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema
  xmlns="http://www.ecoa.technology/module-behaviour-1.0"
  xmlns:tns="http://www.ecoa.technology/module-behaviour-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ecoa.technology/module-behaviour-1.0"
  elementFormDefault="qualified">

   <!-- Behaviour of module operations -->

      <xsd:include schemaLocation="ecoa-common-1.0.xsd"/>
      <xsd:element name="moduleBehaviour" type="ModuleBehaviour"/>
      <xsd:complexType name="ModuleBehaviour">
            <xsd:sequence>
                  <xsd:element ref="use" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="moduleConfiguration" type="ModuleConfiguration"
minOccurs="1" maxOccurs="1"/>
                  <xsd:element name="entryPoint" type="EntryPoint" minOccurs="1"
maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="ModuleConfiguration">
            <xsd:annotation>
                  <xsd:documentation>
                     Module implementation characteristics
               </xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="moduleImplementationName" type="NameId"
use="required">
                  <xsd:annotation>
                        <xsd:documentation>
                        Reference to a moduleImplementation name
                  </xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="queueDepth" type="xsd:positiveInteger"
use="optional" default="1">
                  <xsd:annotation>
                        <xsd:documentation>
                        depth of the incoming operations queue (Warning : to be placed in
another XML file for next stages)
                  </xsd:documentation>
                  </xsd:annotation>
```

```
            </xsd:attribute>
            <xsd:attribute name="maxNbOfProcessedOpsPerActivation"
type="xsd:positiveInteger" default="1">
                  <xsd:annotation>
                        <xsd:documentation>
                  Max number of processed operations per activation (Not for stage 1
: to be used later, once activating property on incoming operations will be taken into
account)
               </xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
      </xsd:complexType>
      <xsd:complexType name="ActionSet">
            <xsd:annotation>
                  <xsd:documentation>A set of actions to be sequentially executed by
the module</xsd:documentation>
            </xsd:annotation>
            <xsd:choice maxOccurs="unbounded">
                  <xsd:element name="Loop" type="Loop"/>
                  <xsd:element name="computing" type="Computing"/>
                  <xsd:element name="operationCall" type="OperationCall"/>
            </xsd:choice>
      </xsd:complexType>
      <xsd:complexType name="EntryPoint">
        <xsd:complexContent>
      <xsd:extension base="ActionSet">
            <xsd:attribute name="name" type="NameId" use="required"/>
            <xsd:attribute name="activatingCondition" type="NameId" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Reference to an incoming
operation</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
      </xsd:extension>
     </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="Loop">
            <xsd:complexContent>
            <xsd:extension base="ActionSet">
            <xsd:attribute name="Iterations" type="xsd:integer" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Number of iterations</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            </xsd:extension>
      </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="Computing">
            <xsd:attribute name="minComputingSteps" type="xsd:long" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Minimum number of computing
steps</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="maxComputingSteps" type="xsd:long" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Maximum number of computing
steps</xsd:documentation>
                  </xsd:annotation>
```

```
            </xsd:attribute>
      </xsd:complexType>
      <xsd:complexType name="OperationCall">
            <xsd:attribute name="moduleOperationRef" type="NameId" use="required">
                  <xsd:annotation>
                        <xsd:documentation>Reference to a required
operation</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="minComputingSteps" type="xsd:long" use="optional"
default="0">
                  <xsd:annotation>
                        <xsd:documentation>Minimum number of computing steps due to
the operation call itself (not including computing steps to execute the operation in
the called module)</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="maxComputingSteps" type="xsd:integer" use="optional"
default="0">
                  <xsd:annotation>
                        <xsd:documentation>Maximum number of computing steps due to
the operation call itself (not including computing steps to execute the operation in
the called module)</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
      </xsd:complexType>
</xsd:schema>
```

7.13 ecoa-types-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/types-1.0"
  xmlns="http://www.ecoa.technology/types-1.0"
  xmlns:tns="http://www.ecoa.technology/types-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >
      <xsd:include schemaLocation="ecoa-common-1.0.xsd"/>
      <xsd:simpleType name="E_predef">
            <xsd:annotation>
                  <xsd:documentation>Predefined ECOA types</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="boolean8"/>
                  <xsd:enumeration value="int8"/>
                  <xsd:enumeration value="int16"/>
                  <xsd:enumeration value="int32"/>
                  <xsd:enumeration value="int64"/>
                  <xsd:enumeration value="uint8"/>
                  <xsd:enumeration value="uint16"/>
                  <xsd:enumeration value="uint32"/>
                  <xsd:enumeration value="uint64"/>
                  <xsd:enumeration value="char8"/>
                  <xsd:enumeration value="byte"/>
                  <xsd:enumeration value="float32"/>
                  <xsd:enumeration value="double64"/>
            </xsd:restriction>
```

```
      </xsd:simpleType>
      <xsd:complexType name="Simple">
            <xsd:annotation>
                  <xsd:documentation>A type based on a predefined type (simple or
E_predef) with a name, min/max constraints, and a unit.</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="type" type="TypeQName" use="required"/>
            <xsd:attribute name="name" type="TypeName" use="required"/>
            <xsd:attribute name="minRange" type="ConstantReferenceOrFloatValue"
use="optional"/>
            <xsd:attribute name="maxRange" type="ConstantReferenceOrFloatValue"
use="optional"/>
            <xsd:attribute name="unit" type="xsd:string" use="optional">
                  <xsd:annotation>
                        <xsd:documentation>Use of International System units is
recommended.</xsd:documentation>
                  </xsd:annotation>
            </xsd:attribute>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Constant">
            <xsd:annotation>
                  <xsd:documentation>Constant definition</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="name" type="TypeName" use="required"/>
            <xsd:attribute name="type" type="TypeQName" use="required"/>
            <xsd:attribute name="value" type="xsd:decimal" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
    <xsd:simpleType name="ConstantReferenceOrFloatValue">
            <xsd:annotation>
                  <xsd:documentation>Use of a constant or of a (float or integer)
value.</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
          <xsd:pattern value="%([A-Za-z][A-Za-z0-9_\.]*:)?[A-Za-z][A-Za-z0-9_]*%|(\+|-
)?([0-9]+(\.[0-9]*)?|\.[0-9]+)([Ee](\+|-)?[0-9]+)?"></xsd:pattern>
          </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="ConstantReferenceOrPositiveIntegerValue">
      <xsd:annotation>
        <xsd:documentation>Use of a constant or of a positive integer
value.</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
          <xsd:pattern value="%([A-Za-z][A-Za-z0-9_\.]*:)?[A-Za-z][A-Za-z0-9_]*%|[0-
9]+"></xsd:pattern>
      </xsd:restriction>
    </xsd:simpleType>
      <xsd:simpleType name="ConstantReferenceOrIntegerValue">
      <xsd:annotation>
        <xsd:documentation>Use of a constant or of an integer
value.</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
          <xsd:pattern value="%([A-Za-z][A-Za-z0-9_\.]*:)?[A-Za-z][A-Za-z0-
9_]*%|(\+|-)?[0-9]*"></xsd:pattern>
      </xsd:restriction>
    </xsd:simpleType>
```

```
      <xsd:complexType name="Enum">
            <xsd:annotation>
                  <xsd:documentation>Enumerated type</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="value" type="EnumValue" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="TypeName" use="required"/>
            <xsd:attribute name="type" type="TypeQName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="EnumValue">
            <xsd:annotation>
                  <xsd:documentation>A possible value of an enumerated
type</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="name" type="TypeName" use="required"/>
            <xsd:attribute name="valnum" type="ConstantReferenceOrIntegerValue"
use="optional"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="FixedArray">
            <xsd:annotation>
                  <xsd:documentation>Fixed-size array (size is always equal to max
capacity)</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="itemType" type="TypeQName" use="required"/>
            <xsd:attribute name="maxNumber"
type="ConstantReferenceOrPositiveIntegerValue" use="required"/>
            <xsd:attribute name="name" type="TypeName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Array">
            <xsd:annotation>
                  <xsd:documentation>Variable-size (bounded capacity)
array</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="itemType" type="TypeQName" use="required"/>
            <xsd:attribute name="maxNumber"
type="ConstantReferenceOrPositiveIntegerValue" use="required"/>
            <xsd:attribute name="name" type="TypeName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Record">
            <xsd:annotation>
                  <xsd:documentation>A record with named fields (Ada record, C
struct)</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="field" type="Field" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="TypeName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Field">
            <xsd:attribute name="name" type="NameId" use="required"/>
            <xsd:attribute name="type" type="TypeQName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
```

```
      <xsd:complexType name="VariantRecord">
            <xsd:annotation>
                  <xsd:documentation>A record with variable parts: each "union"
exists only if the selector has the value given by the "when"
attribute.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="field" type="Field" minOccurs="0"
maxOccurs="unbounded"/>
                  <xsd:element name="union" type="Union" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="TypeName" use="required"/>
            <xsd:attribute name="selectName" type="NameId" use="required"/>
            <xsd:attribute name="selectType" type="TypeQName" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Union">
            <xsd:attribute name="name" type="NameId" use="required"/>
            <xsd:attribute name="type" type="TypeQName" use="required"/>
            <xsd:attribute name="when" type="xsd:string" use="required"/>
    <xsd:attribute name="comment" type="xsd:string" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="DataTypes">
            <xsd:annotation>
                  <xsd:documentation>A set of data type
definitions</xsd:documentation>
            </xsd:annotation>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="simple" type="Simple"/>
                  <xsd:element name="record" type="Record">
                        <xsd:unique name="field">
                              <xsd:selector xpath="tns:field"/>
                              <xsd:field xpath="@name"/>
                        </xsd:unique>
                  </xsd:element>
                  <xsd:element name="constant" type="Constant"/>
                  <xsd:element name="variantRecord" type="VariantRecord">
                        <xsd:unique name="fieldunion">
                              <xsd:selector xpath="tns:field|tns:union"/>
                              <xsd:field xpath="@name"/>
                        </xsd:unique>
                  </xsd:element>
                  <xsd:element name="array" type="Array"/>
                  <xsd:element name="fixedArray" type="FixedArray"/>
                  <xsd:element name="enum" type="Enum">
                        <xsd:unique name="value">
                              <xsd:selector xpath="tns:value"/>
                              <xsd:field xpath="@name"/>
                        </xsd:unique>
                        <xsd:unique name="valnum">
                              <xsd:selector xpath="tns:value"/>
                              <xsd:field xpath="@valnum"/>
                        </xsd:unique>
                  </xsd:element>
            </xsd:choice>
      </xsd:complexType>
      <xsd:complexType name="Library">
            <xsd:annotation>
```

```
                  <xsd:documentation>A set of data types in a
library</xsd:documentation>
              </xsd:annotation>
              <xsd:sequence>
                    <xsd:element ref="use" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="types" type="DataTypes">
               <xsd:unique name="typename">
                 <xsd:selector xpath="*"/>
                 <xsd:field xpath="@name"/>
               </xsd:unique>
             </xsd:element>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element name="Library" type="Library"/>
</xsd:schema>
```

7.14 ecoa-project-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/project-1.0"
  xmlns="http://www.ecoa.technology/project-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >
  <xsd:include schemaLocation="ecoa-common-1.0.xsd"/>

  <xsd:element name="ECOAProject" type="EcoaProject"/>


  <xsd:complexType name="EcoaProject">
    <xsd:annotation>
      <xsd:documentation>
             Describes a whole ECOA project
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
       <xsd:choice minOccurs="0" maxOccurs="unbounded">
             <xsd:element name="serviceDefinitions" type="Files"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="componentDefinitions" type="Files"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="types" type="Files"
                    maxOccurs="unbounded" minOccurs="0"/>
             <xsd:element name="initialAssembly" type="xsd:anyURI"
                    maxOccurs="unbounded" minOccurs="0"/>
             <xsd:element name="componentImplementations" type="Files"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="logicalSystem" type="xsd:anyURI"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="deploymentSchema" type="xsd:anyURI"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="outputDirectory" type="xsd:anyURI"
                    minOccurs="0" maxOccurs="unbounded" />
             <xsd:element name="implementationAssembly" type="xsd:anyURI"
                    minOccurs="0" maxOccurs="unbounded" />
       </xsd:choice>
    </xsd:sequence>
```

```
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="Files">
    <xsd:annotation>
      <xsd:documentation>List of files</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
       <xsd:element name="file" type="xsd:anyURI" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>


</xsd:schema>
```

7.15 ecoa-interface-behaviour-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.ecoa.technology/interface-behaviour-1.0"
xmlns:tns="http://www.ecoa.technology/interface-behaviour-1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ecoa.technology/interface-behaviour-1.0"
elementFormDefault="qualified">
	<xsd:include schemaLocation="ecoa-common-1.0.xsd"/>
	<xsd:element name="serviceInstanceBehaviour" type="ServiceInstanceBehaviour"/>
	<xsd:complexType name="ServiceInstanceBehaviour">
		<xsd:annotation>
			<xsd:documentation>The definition of an ECOA service, including a
set of operations.</xsd:documentation>
		</xsd:annotation>
		<xsd:sequence minOccurs="1" maxOccurs="unbounded">
			<xsd:element name="mode" type="Mode"/>
		</xsd:sequence>
		<xsd:attribute name="name" type="NameId" use="required"/>
	</xsd:complexType>
	<xsd:complexType name="Mode">
		<xsd:annotation>
			<xsd:documentation>A set of named operations.</xsd:documentation>
		</xsd:annotation>
		<xsd:sequence minOccurs="0" maxOccurs="unbounded">
			<xsd:element name="automata" type="Automata"/>
		</xsd:sequence>
		<xsd:attribute name="name" type="NameId" use="required"/>
	</xsd:complexType>

	<xsd:complexType name="Automata">
		<xsd:annotation>
			<xsd:documentation>A set of named operations.</xsd:documentation>
		</xsd:annotation>
		<xsd:choice minOccurs="1" maxOccurs="unbounded">
				<xsd:element name="state" type="State"/>
				<xsd:element name="startState" type="State"/>
				<xsd:element name="endState" type="State"/>
				<xsd:element name="link" type="Link"/>
		</xsd:choice>
		<xsd:attribute name="name" type="NameId" use="required"/>
	</xsd:complexType>
```

```
	<xsd:complexType name="Link">
		<xsd:annotation>
			<xsd:documentation>Link between states of the
automata)</xsd:documentation>
		</xsd:annotation>
		<xsd:sequence>
			<xsd:element name="delay" type="Delay" minOccurs="0"
maxOccurs="1">
				<xsd:annotation>
					<xsd:documentation>Max number of occurrences within a
reference time frame</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
			<xsd:element name="operations" type="Operations" minOccurs="0"
maxOccurs="1">
				<xsd:annotation>
					<xsd:documentation>Set of operations called during
the transition</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
		</xsd:sequence>
		<xsd:attribute name="name" type="NameId" use="required"/>
		<xsd:attribute name="initialState" type="NameId" use="required">
			<xsd:annotation>
				<xsd:documentation>Initial State</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
		<xsd:attribute name="finalState" type="NameId" use="required">
			<xsd:annotation>
				<xsd:documentation>Final State</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
	</xsd:complexType>

	<xsd:complexType name="State">
		<xsd:annotation>
			<xsd:documentation>The guard delay associated to each
link</xsd:documentation>
		</xsd:annotation>
		<xsd:attribute name="name" type="NameId" use="required"/>
	</xsd:complexType>

	<xsd:complexType name="Delay">
		<xsd:annotation>
			<xsd:documentation>The guard delay associated to each
link</xsd:documentation>
		</xsd:annotation>
		<xsd:attribute name="value" type="xsd:double" use="optional">
			<xsd:annotation>
				<xsd:documentation>delay in seconds</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
	</xsd:complexType>


	<xsd:complexType name="Operations">
		<xsd:annotation>
```

```
                  <xsd:documentation>Use of the "event" exchange
mechanism.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                  <xsd:element name="operation" type="Operation" minOccurs="1"
maxOccurs="unbounded">
                        <xsd:annotation>
                              <xsd:documentation>Max number of occurrences within a
reference time frame</xsd:documentation>
                        </xsd:annotation>
                  </xsd:element>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Operation">
            <xsd:annotation>
                  <xsd:documentation>The guard delay associated to each
link</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="name" type="NameId" use="required"/>
      </xsd:complexType>


</xsd:schema>
```

7.16 ecoa-udpbinding-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.ecoa.technology/udpbinding-1.0"
            xmlns:tns="http://www.ecoa.technology/udpbinding-1.0"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.ecoa.technology/udpbinding-1.0"
            elementFormDefault="qualified">
      <xsd:element name="platform">
            <xsd:complexType>
                  <xsd:attribute name="platformId" type="PlatformID"
use="required"/>
                  <xsd:attribute name="name" type="xsd:string" use="required"/>
                  <xsd:attribute name="maxChannels" type="xsd:positiveInteger"
use="optional" default="256"/>
                  <xsd:attribute name="receivingPort" type="xsd:positiveInteger"
use="required"/>
                  <xsd:attribute name="receivingMulticastAddress" type="xsd:string"
use="required"/>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="UDPBinding">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element ref="platform" maxOccurs="unbounded"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
    <xsd:simpleType name="PlatformID">
    <xsd:annotation>
      <xsd:documentation>
      PlatformID is used to identify uniquely each platform within ELI-UDP exchanges.
      It is assumed that no more than 16 platforms will be connected together.
```

```
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:unsignedInt">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="15"/>
    </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

7.17 ecoa-uid-1.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://www.ecoa.technology/uid-1.0"
  xmlns="http://www.ecoa.technology/uid-1.0"
  xmlns:tns="http://www.ecoa.technology/uid-1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >

  <xsd:element name="ID_map" type="ID_map">
    <!-- each "key" attribute shall be unique -->
    <xsd:key name="key">
      <xsd:selector xpath="tns:ID"/>
      <xsd:field xpath="@key"/>
    </xsd:key>
    <!-- each "value" attribute shall be unique -->
    <xsd:key name="value">
      <xsd:selector xpath="tns:ID"/>
      <xsd:field xpath="@value"/>
    </xsd:key>
  </xsd:element>

  <xsd:complexType name="ID_map">
    <xsd:sequence>
      <xsd:element name="ID" type="ID" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ID">
    <xsd:attribute name="key" type="xsd:string" use="required" />
    <xsd:attribute name="value" type="xsd:int" use="required" />
  </xsd:complexType>

</xsd:schema>
```

7.18 sca-1.1-cd06.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved.
     OASIS trademark, IPR and other policies apply.  -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912">

   <include schemaLocation="sca-core-1.1-cd06.xsd"/>
```

```
<!--    <include schemaLocation="sca-interface-java-1.1-csd05.xsd"/> -->
<!--    <include schemaLocation="sca-interface-wsdl-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-interface-cpp-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-interface-c-1.1-cd06.xsd"/> -->

<!--    <include schemaLocation="sca-implementation-java-1.1-csd03.xsd"/> -->
<!--    <include schemaLocation="sca-implementation-composite-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-implementation-cpp-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-implementation-c-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-implementation-bpel-1.1-cd03.xsd"/> -->
<!--    <include schemaLocation="sca-implementation-spring-1.1-csd01.xsd"/> -->

<!--    <include schemaLocation="sca-binding-ws-1.1-cd04-rev2.xsd"/> -->
<!--    <include schemaLocation="sca-binding-ws-callback-1.1-cd04-rev1.xsd"/> -->
<!--    <include schemaLocation="sca-binding-jms-1.1-csd05.xsd"/> -->
<!--    <include schemaLocation="sca-binding-jca-1.1-cd04-rev2.xsd"/>    -->
<!--    <include schemaLocation="sca-binding-sca-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-binding-ejb-1.1-cd02-rev2.xsd"/> -->

<!--    <include schemaLocation="sca-definitions-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-policy-1.1-cd04.xsd"/> -->

   <include schemaLocation="sca-contribution-1.1-cd06.xsd"/>
<!--    <include schemaLocation="sca-contribution-cpp-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-contribution-c-1.1-cd06.xsd"/> -->
<!--    <include schemaLocation="sca-contribution-java-1.1-csd03.xsd"/> -->

</schema>
```

7.19 sca-contribution-1.1-cd06.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved.
     OASIS trademark, IPR and other policies apply.  -->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
   xmlns:ecoa="http://www.ecoa.technology/sca"
   targetNamespace="http://docs.oasis-open.org/ns/opencsa/sca/200912"
   elementFormDefault="qualified">

   <import namespace="http://www.ecoa.technology/sca" schemaLocation="ecoa-sca-
1.0.xsd" />

   <include schemaLocation="sca-core-1.1-cd06.xsd"/>

   <!-- Contribution -->
   <element name="contribution" type="sca:ContributionType"/>
   <complexType name="ContributionType">
      <complexContent>
         <extension base="sca:CommonExtensionBase">
            <sequence>
               <element name="deployable" type="sca:DeployableType"
                        minOccurs="0" maxOccurs="unbounded"/>
<!--               <element ref="sca:importBase" minOccurs="0"  -->
<!--                        maxOccurs="unbounded"/> -->
<!--               <element ref="sca:exportBase" minOccurs="0"  -->
<!--                        maxOccurs="unbounded"/> -->
<!--               <element ref="sca:extensions" minOccurs="0" maxOccurs="1" /> -->
```

```
            </sequence>
         </extension>
      </complexContent>
   </complexType>

   <!-- Deployable -->
   <complexType name="DeployableType">
      <complexContent>
         <extension base="sca:CommonExtensionBase">
            <sequence>
               <any namespace="##other" processContents="lax" minOccurs="0"
                    maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="ecoa:deployment"/>
            <attribute name="composite" type="QName" use="required"/>
         </extension>
      </complexContent>
   </complexType>

   <!-- Import -->
<!--    <element name="importBase" type="sca:Import" abstract="true" /> -->
<!--    <complexType name="Import" abstract="true"> -->
<!--       <complexContent> -->
<!--          <extension base="sca:CommonExtensionBase"> -->
<!--             <sequence> -->
<!--                <any namespace="##other" processContents="lax" minOccurs="0"  -->
<!--                     maxOccurs="unbounded"/> -->
<!--             </sequence> -->
<!--          </extension> -->
<!--       </complexContent>    -->
<!--    </complexType> -->

<!--    <element name="import" type="sca:ImportType"  -->
<!--             substitutionGroup="sca:importBase"/> -->
<!--    <complexType name="ImportType"> -->
<!--       <complexContent> -->
<!--          <extension base="sca:Import"> -->
<!--             <attribute name="namespace" type="string" use="required"/> -->
<!--             <attribute name="location" type="anyURI" use="optional"/> -->
<!--          </extension> -->
<!--       </complexContent> -->
<!--    </complexType> -->

   <!-- Export -->
<!--    <element name="exportBase" type="sca:Export" abstract="true" /> -->
<!--    <complexType name="Export" abstract="true"> -->
<!--       <complexContent> -->
<!--          <extension base="sca:CommonExtensionBase"> -->
<!--             <sequence> -->
<!--                <any namespace="##other" processContents="lax" minOccurs="0"  -->
<!--                     maxOccurs="unbounded"/> -->
<!--             </sequence> -->
<!--          </extension> -->
<!--       </complexContent>       -->
<!--    </complexType> -->

<!--    <element name="export" type="sca:ExportType"  -->
<!--             substitutionGroup="sca:exportBase"/> -->
<!--    <complexType name="ExportType"> -->
```

```
<!--       <complexContent> -->
<!--          <extension base="sca:Export"> -->
<!--             <attribute name="namespace" type="string" use="required"/> -->
<!--          </extension> -->
<!--       </complexContent> -->
<!--    </complexType> -->

</schema>
```

7.20 sca-core-1.1-cd06.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright(C) OASIS(R) 2005,2010. All Rights Reserved. OASIS trademark, IPR and
other policies apply. -->
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:sca="http://docs.oasis-
open.org/ns/opencsa/sca/200912" xmlns:ecoa="http://www.ecoa.technology/sca"
xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
xmlns:xml="http://www.w3.org/XML/1998/namespace" targetNamespace="http://docs.oasis-
open.org/ns/opencsa/sca/200912" elementFormDefault="qualified" jxb:version="1.0">
<!--    <include schemaLocation="sca-policy-1.1-cd04.xsd"/> -->

      <import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="../xml/xml.xsd"/>
      <import namespace="http://www.ecoa.technology/sca" schemaLocation="ecoa-sca-
1.0.xsd"/>
      <!-- Common extension base for SCA definitions -->
      <complexType name="CommonExtensionBase">
            <sequence>
                  <element ref="sca:documentation" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <!-- <anyAttribute namespace="##other" processContents="lax"/> -->
      </complexType>
      <element name="documentation" type="sca:Documentation"/>
      <complexType name="Documentation" mixed="true">
            <sequence>
                  <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="xml:lang"/>
      </complexType>
      <!-- Component Type -->
      <element name="componentType" type="sca:ComponentType"/>
      <complexType name="ComponentType">
            <complexContent>
                  <extension base="sca:CommonExtensionBase">
                        <sequence>
                              <!-- <element ref="sca:implementation"
minOccurs="0"/> -->
                              <choice minOccurs="0" maxOccurs="unbounded">
                                    <!-- <element name="service"
type="sca:ComponentService"> -->
                                    <element name="service">
                                          <complexType>
                                                <complexContent>
                                                      <restriction
base="sca:ComponentService">
                                                            <sequence>
```

```
                                                                  <sequence>
      <element ref="ecoa:interface"/>
                                                                  </sequence>
                                                              </sequence>
                                                          </restriction>
                                                      </complexContent>
                                                  </complexType>
                                              </element>
                                              <element name="reference"
type="sca:ComponentTypeReference"/>
                                              <element name="property" type="sca:Property"/>
                                    </choice>
                                    <!-- <element ref="sca:extensions" minOccurs="0"
maxOccurs="1" /> -->
                              </sequence>
                        </extension>
                  </complexContent>
      </complexType>
      <!-- Composite -->
      <element name="composite" type="sca:Composite"/>
      <complexType name="Composite">
            <complexContent>
                  <extension base="sca:CommonExtensionBase">
                        <sequence>
                              <!-- <element ref="sca:include" minOccurs="0" -->
                              <!-- maxOccurs="unbounded"/> -->
                              <choice minOccurs="0" maxOccurs="unbounded">
                                    <!-- <element ref="sca:requires"/> -->
                                    <!-- <element ref="sca:policySetAttachment"/>
-->
                                    <element name="service" type="sca:Service"/>
                                    <element name="property" type="sca:Property"/>
                                    <element name="component"
type="sca:Component"/>
                                    <element name="reference"
type="sca:Reference"/>
                                    <element name="wire" type="sca:Wire"/>
                              </choice>
                              <!-- <any namespace="##other" processContents="lax"
minOccurs="0" -->
                              <!-- maxOccurs="unbounded"/> -->
                        </sequence>
                        <attribute name="name" type="NCName" use="required"/>
                        <attribute name="targetNamespace" type="anyURI"
use="required"/>
                  </extension>
                  <!-- <attribute name="local" type="boolean" use="optional" -->
                  <!-- default="false"/> -->
                  <!-- <attribute name="autowire" type="boolean" use="optional" -->
                  <!-- default="false"/> -->
                  <!-- <attribute name="requires" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
                  <!-- <attribute name="policySets" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
            </complexContent>
      </complexType>
      <!-- Contract base type for Service, Reference -->
      <complexType name="Contract" abstract="true">
```

```
            <complexContent>
                  <extension base="sca:CommonExtensionBase">
                        <sequence>
                              <!-- <element ref="sca:interface" minOccurs="0"
maxOccurs="1" /> -->
                              <element ref="ecoa:interface" minOccurs="0"/>
                              <!-- <element ref="sca:binding" minOccurs="0" -->
                              <!-- maxOccurs="unbounded" /> -->
                              <!-- <element ref="sca:callback" minOccurs="0"
maxOccurs="1" /> -->
                              <!-- <element ref="sca:requires" minOccurs="0" -->
                              <!-- maxOccurs="unbounded"/> -->
                              <!-- <element ref="sca:policySetAttachment"
minOccurs="0" -->
                              <!-- maxOccurs="unbounded"/> -->
                              <!-- <element ref="sca:extensions" minOccurs="0"
maxOccurs="1" /> -->
                        </sequence>
                        <attribute name="name" type="NCName" use="required"/>
                  </extension>
                  <!-- <attribute name="requires" type="sca:listOfQNames" -->
                  <!-- use="optional" > -->
                  <!-- <annotation> -->
                  <!-- <appinfo> -->
                  <!-- <jxb:property name="requiresAtt"/> -->
                  <!-- </appinfo> -->
                  <!-- </annotation> -->
                  <!-- </attribute> -->
                  <!-- <attribute name="policySets" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
            </complexContent>
      </complexType>
      <!-- Service -->
      <complexType name="Service">
            <complexContent>
                  <extension base="sca:Contract">
                        <attribute name="promote" type="anyURI" use="required"/>
                  </extension>
            </complexContent>
      </complexType>
      <!-- Interface -->
      <element name="interface" type="sca:Interface" abstract="true"/>
      <complexType name="Interface" abstract="true">
            <complexContent>
                  <extension base="sca:CommonExtensionBase"/>
                  <!-- <choice minOccurs="0" maxOccurs="unbounded"> -->
                  <!-- <element ref="sca:requires"/> -->
                  <!-- <element ref="sca:policySetAttachment"/> -->
                  <!-- </choice> -->
                  <!-- <attribute name="remotable" type="boolean" use="optional"/> -
->
                  <!-- <attribute name="requires" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
                  <!-- <attribute name="policySets" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
            </complexContent>
      </complexType>
      <!-- Reference -->
      <complexType name="Reference">
```

```
            <complexContent>
                  <extension base="sca:Contract">
                        <attribute name="multiplicity" type="sca:Multiplicity"
use="required"/>
                        <attribute name="promote" type="sca:listOfAnyURIs"
use="required"/>
                  </extension>
                  <!-- <attribute name="target" type="sca:listOfAnyURIs" -->
                  <!-- use="optional"/> -->
                  <!-- <attribute name="wiredByImpl" type="boolean" use="optional" -
->
                  <!-- default="false"/> -->
            </complexContent>
      </complexType>
      <!-- Property -->
      <complexType name="SCAPropertyBase" mixed="true">
            <sequence>
                  <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                  <!-- NOT an extension point; This any exists to accept the
element-based or complex type property i.e. no element-based extension point under
"sca:property" -->
            </sequence>
            <attribute name="name" type="NCName" use="required"/>
            <!-- mixed="true" to handle simple type -->
            <attribute name="type" type="QName" use="optional"/>
            <!-- <attribute name="element" type="QName" use="optional"/> -->
            <!-- <attribute name="many" type="boolean" use="optional"
default="false"/> -->
            <!-- <attribute name="value" type="anySimpleType" use="optional"/> -->
            <anyAttribute namespace="##other" processContents="lax"/>
      </complexType>
      <complexType name="Property" mixed="true">
            <complexContent mixed="true">
                  <extension base="sca:SCAPropertyBase">
                        <attribute name="mustSupply" type="boolean" use="optional"
default="false"/>
                  </extension>
            </complexContent>
      </complexType>
      <complexType name="PropertyValue" mixed="true">
            <complexContent mixed="true">
                  <extension base="sca:SCAPropertyBase">
                        <attribute name="source" type="string" use="optional"/>
                        <attribute name="file" type="anyURI" use="optional"/>
                  </extension>
            </complexContent>
      </complexType>
      <!-- Binding -->
      <!-- <element name="binding" type="sca:Binding" abstract="true"/> -->
      <!-- <complexType name="Binding" abstract="true"> -->
      <!-- <complexContent> -->
      <!-- <extension base="sca:CommonExtensionBase"> -->
      <!-- <sequence> -->
      <!-- <element ref="sca:wireFormat" minOccurs="0" maxOccurs="1" /> -->
      <!-- <element ref="sca:operationSelector" minOccurs="0" -->
      <!-- maxOccurs="1" /> -->
      <!-- <element ref="sca:requires" minOccurs="0" -->
      <!-- maxOccurs="unbounded"/> -->
```

```
<!-- <element ref="sca:policySetAttachment" minOccurs="0" -->
<!-- maxOccurs="unbounded"/> -->
<!-- </sequence> -->
<!-- <attribute name="uri" type="anyURI" use="optional"/> -->
<!-- <attribute name="name" type="NCName" use="optional"/> -->
<!-- <attribute name="requires" type="sca:listOfQNames" -->
<!-- use="optional"> -->
<!-- <annotation> -->
<!-- <appinfo> -->
<!-- <jxb:property name="requiresAtt"/> -->
<!-- </appinfo> -->
<!-- </annotation> -->
<!-- </attribute> -->
<!-- <attribute name="policySets" type="sca:listOfQNames" -->
<!-- use="optional"/> -->
<!-- </extension> -->
<!-- </complexContent> -->
<!-- </complexType> -->
<!-- Binding Type -->
<!-- <element name="bindingType" type="sca:BindingType"/> -->
<!-- <complexType name="BindingType"> -->
<!-- <complexContent> -->
<!-- <extension base="sca:CommonExtensionBase"> -->
<!-- <sequence> -->
<!-- <any namespace="##other" processContents="lax" minOccurs="0" -->
<!-- maxOccurs="unbounded"/> -->
<!-- </sequence> -->
<!-- <attribute name="type" type="QName" use="required"/> -->
<!-- <attribute name="alwaysProvides" type="sca:listOfQNames" -->
<!-- use="optional"/> -->
<!-- <attribute name="mayProvide" type="sca:listOfQNames" -->
<!-- use="optional"/> -->
<!-- </extension> -->
<!-- </complexContent> -->
<!-- </complexType> -->
<!-- WireFormat Type -->
<element name="wireFormat" type="sca:WireFormatType" abstract="true"/>
<complexType name="WireFormatType" abstract="true">
      <!-- <anyAttribute namespace="##other" processContents="lax"/> -->
</complexType>
<!-- OperationSelector Type -->
<!-- <element name="operationSelector" type="sca:OperationSelectorType" -->
<!-- abstract="true"/> -->
<!-- <complexType name="OperationSelectorType" abstract="true"> -->
<!-- <anyAttribute namespace="##other" processContents="lax"/> -->
<!-- </complexType> -->
<!-- Callback -->
<!-- <element name="callback" type="sca:Callback"/> -->
<!-- <complexType name="Callback"> -->
<!-- <complexContent> -->
<!-- <extension base="sca:CommonExtensionBase"> -->
<!-- <choice minOccurs="0" maxOccurs="unbounded"> -->
<!-- <element ref="sca:binding"/> -->
<!-- <element ref="sca:requires"/> -->
<!-- <element ref="sca:policySetAttachment"/> -->
<!-- <element ref="sca:extensions" minOccurs="0" maxOccurs="1" /> -->
<!-- </choice> -->
<!-- <attribute name="requires" type="sca:listOfQNames" -->
<!-- use="optional"/> -->
```

```
	<!-- <attribute name="policySets" type="sca:listOfQNames" -->
	<!-- use="optional"/> -->
	<!-- </extension> -->
	<!-- </complexContent> -->
	<!-- </complexType> -->
	<!-- Component -->
	<complexType name="Component">
		<complexContent>
			<extension base="sca:CommonExtensionBase">
				<sequence>
					<element ref="sca:implementation"/>
					<choice minOccurs="0" maxOccurs="unbounded">
						<element name="service"
type="sca:ComponentService"/>
						<element name="reference"
type="sca:ComponentReference"/>
						<element name="property"
type="sca:PropertyValue"/>
						<!-- <element ref="sca:requires"/> -->
						<!-- <element ref="sca:policySetAttachment"/>
-->
					</choice>
					<!-- <element ref="sca:extensions" minOccurs="0"
maxOccurs="1" /> -->
				</sequence>
				<attribute name="name" type="NCName" use="required"/>
			</extension>
			<!-- <attribute name="autowire" type="boolean" use="optional"/> --
>
			<!-- <attribute name="requires" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
			<!-- <attribute name="policySets" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
		</complexContent>
	</complexType>
	<!-- Component Service -->
	<complexType name="ComponentService">
		<complexContent>
			<extension base="sca:Contract"/>
		</complexContent>
	</complexType>
	<!-- Component Reference -->
	<complexType name="ComponentReference">
		<complexContent>
			<extension base="sca:Contract">
				<attribute name="multiplicity" type="sca:Multiplicity"
use="optional" default="1..1"/>
			</extension>
			<!-- <attribute name="autowire" type="boolean" use="optional"/> --
>
			<!-- <attribute name="target" type="sca:listOfAnyURIs" -->
			<!-- use="optional"/> -->
			<!-- <attribute name="wiredByImpl" type="boolean" use="optional" -
->
			<!-- default="false"/> -->
			<!-- <attribute name="nonOverridable" type="boolean"
use="optional" -->
			<!-- default="false"/> -->
		</complexContent>
```

```
	</complexType>
	<!-- Component Type Reference -->
	<complexType name="ComponentTypeReference">
		<complexContent>
			<restriction base="sca:ComponentReference">
				<sequence>
					<element ref="sca:documentation" minOccurs="0"
maxOccurs="unbounded"/>
					<!-- <element ref="sca:interface" minOccurs="0"/> -->
					<element ref="ecoa:interface"/>
					<!-- <element ref="sca:binding" minOccurs="0" -->
					<!-- maxOccurs="unbounded"/> -->
					<!-- <element ref="sca:callback" minOccurs="0"/> -->
					<!-- <element ref="sca:requires" minOccurs="0" -->
					<!-- maxOccurs="unbounded"/> -->
					<!-- <element ref="sca:policySetAttachment"
minOccurs="0" -->
					<!-- maxOccurs="unbounded"/> -->
					<!-- <element ref="sca:extensions" minOccurs="0"
maxOccurs="1" /> -->
				</sequence>
				<attribute name="name" type="NCName" use="required"/>
				<attribute name="multiplicity" type="sca:Multiplicity"
use="optional" default="1..1"/>
			</restriction>
			<!-- <attribute name="autowire" type="boolean" use="optional"/> --
>
			<!-- <attribute name="wiredByImpl" type="boolean" use="optional" -
->
			<!-- default="false"/> -->
			<!-- <attribute name="requires" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
			<!-- <attribute name="policySets" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
			<!-- <anyAttribute namespace="##other" processContents="lax"/> -->
		</complexContent>
	</complexType>
	<!-- Implementation -->
	<element name="implementation" type="sca:Implementation" abstract="true"/>
	<complexType name="Implementation" abstract="true">
		<complexContent>
			<extension base="sca:CommonExtensionBase"/>
			<!-- <choice minOccurs="0" maxOccurs="unbounded"> -->
			<!-- <element ref="sca:requires"/> -->
			<!-- <element ref="sca:policySetAttachment"/> -->
			<!-- </choice> -->
			<!-- <attribute name="requires" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
			<!-- <attribute name="policySets" type="sca:listOfQNames" -->
			<!-- use="optional"/> -->
		</complexContent>
	</complexType>
	<!-- Implementation Type -->
	<element name="implementationType" type="sca:ImplementationType"/>
	<complexType name="ImplementationType">
		<complexContent>
			<extension base="sca:CommonExtensionBase">
				<attribute name="type" type="QName" use="required"/>
			</extension>
```

```
                  <!-- <sequence> -->
                  <!-- <any namespace="##other" processContents="lax" minOccurs="0"
-->
                  <!-- maxOccurs="unbounded"/> -->
                  <!-- </sequence> -->
                  <!-- <attribute name="alwaysProvides" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
                  <!-- <attribute name="mayProvide" type="sca:listOfQNames" -->
                  <!-- use="optional"/> -->
            </complexContent>
      </complexType>
      <!-- Wire -->
      <complexType name="Wire">
            <complexContent>
                  <extension base="sca:CommonExtensionBase">
                  <sequence>
                  <any namespace="##other" processContents="lax" minOccurs="0"
                  maxOccurs="unbounded"/>
                  </sequence>
                        <attribute name="source" type="anyURI" use="required"/>
                        <attribute name="target" type="anyURI" use="required"/>
                        <attribute ref="ecoa:rank" use="required"/>
                        <attribute ref="ecoa:allEventsMulticasted" use="optional"/>
                  </extension>
                  <!-- <sequence> -->
                  <!-- <any namespace="##other" processContents="lax" minOccurs="0"
-->
                  <!-- maxOccurs="unbounded"/> -->
                  <!-- </sequence> -->
                  <!-- <attribute name="replace" type="boolean" use="optional" -->
                  <!-- default="false"/> -->
            </complexContent>
      </complexType>
      <!-- Include -->
      <!-- <element name="include" type="sca:Include"/> -->
      <!-- <complexType name="Include"> -->
      <!-- <complexContent> -->
      <!-- <extension base="sca:CommonExtensionBase"> -->
      <!-- <attribute name="name" type="QName"/> -->
      <!-- </extension> -->
      <!-- </complexContent> -->
      <!-- </complexType> -->
      <!-- Extensions element -->
      <element name="extensions">
      <complexType>
      <sequence>
      <any namespace="##other" processContents="lax"
      minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
      </complexType>
      </element>
      <!-- Intents within WSDL documents -->
      <!-- <attribute name="requires" type="sca:listOfQNames"/> -->
      <!-- Global attribute definition for @callback to mark a WSDL port type as
having a callback interface defined in terms of a second port type. -->
      <!-- <attribute name="callback" type="anyURI"/> -->
      <!-- Value type definition for property values -->
      <element name="value" type="sca:ValueType"/>
      <complexType name="ValueType" mixed="true">
```

```
            <sequence>
                  <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <anyAttribute namespace="##any" processContents="lax"/>
            <!-- mixed="true" to handle simple type -->
      </complexType>
      <!-- Miscellaneous simple type definitions -->
      <simpleType name="Multiplicity">
            <restriction base="string">
                  <enumeration value="0..1"/>
                  <enumeration value="1..1"/>
                  <enumeration value="0..n"/>
                  <enumeration value="1..n"/>
            </restriction>
      </simpleType>
      <!-- <simpleType name="OverrideOptions"> -->
      <!-- <restriction base="string"> -->
      <!-- <enumeration value="no"/> -->
      <!-- <enumeration value="may"/> -->
      <!-- <enumeration value="must"/> -->
      <!-- </restriction> -->
      <!-- </simpleType> -->
      <simpleType name="listOfQNames">
            <list itemType="QName"/>
      </simpleType>
      <simpleType name="listOfAnyURIs">
            <list itemType="anyURI"/>
      </simpleType>
      <!-- <simpleType name="CreateResource"> -->
      <!-- <restriction base="string"> -->
      <!-- <enumeration value="always" /> -->
      <!-- <enumeration value="never" /> -->
      <!-- <enumeration value="ifnotexist" /> -->
      <!-- </restriction> -->
      <!-- </simpleType> -->
</schema>
```

7.21 xml-schema.xsd

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- XML Schema schema for XML Schemas: Part 1: Structures -->
<!-- Note this schema is NOT the normative structures schema. -->
<!-- The prose copy in the structures REC is the normative -->
<!-- version (which shouldn't differ from this one except for -->
<!-- this comment and entity expansions, but just in case -->
<xs:schema targetNamespace="http://www.w3.org/2001/XMLSchema" blockDefault="#all"
elementFormDefault="qualified" version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xml:lang="EN" xmlns:hfp="http://www.w3.org/2001/XMLSchema-hasFacetAndProperty">
  <xs:annotation>
    <xs:documentation>
      Part 1 version: Id: structures.xsd,v 1.2 2004/01/15 11:34:25 ht Exp
      Part 2 version: Id: datatypes.xsd,v 1.3 2004/01/23 18:11:13 ht Exp
  </xs:documentation>
  </xs:annotation>

  <xs:annotation>
```

```
    <xs:documentation source="http://www.w3.org/TR/2004/PER-xmlschema-1-
20040318/structures.html">
      The schema corresponding to this document is normative,
      with respect to the syntactic constraints it expresses in the
      XML Schema language. The documentation (within &lt;documentation> elements)
      below, is not normative, but rather highlights important aspects of
      the W3C Recommendation of which this is a part</xs:documentation>
  </xs:annotation>

  <xs:annotation>
    <xs:documentation>
      The simpleType element and all of its members are defined
      towards the end of this schema document</xs:documentation>
  </xs:annotation>

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="xml.xsd">
    <xs:annotation>
      <xs:documentation>
        Get access to the xml: attribute groups for xml:lang
        as declared on 'schema' and 'documentation' below
     </xs:documentation>
    </xs:annotation>
  </xs:import>

  <xs:complexType name="openAttrs">
    <xs:annotation>
      <xs:documentation>
        This type is extended by almost all schema types
        to allow attributes from other namespaces to be
        added to user schemas.
     </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="annotated">
    <xs:annotation>
      <xs:documentation>
        This type is extended by all types which allow annotation
        other than &lt;schema&gt; itself
     </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="xs:openAttrs">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:group name="schemaTop">
    <xs:annotation>
```

```
      <xs:documentation>
        This group is for the
        elements which occur freely at the top level of schemas.
        All of their types are based on the "annotated" type by
extension.</xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:group ref="xs:redefinable" />
      <xs:element ref="xs:element" />
      <xs:element ref="xs:attribute" />
      <xs:element ref="xs:notation" />
    </xs:choice>
  </xs:group>

  <xs:group name="redefinable">
    <xs:annotation>
      <xs:documentation>
        This group is for the
        elements which can self-redefine (see &lt;redefine> below).</xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:element ref="xs:simpleType" />
      <xs:element ref="xs:complexType" />
      <xs:element ref="xs:group" />
      <xs:element ref="xs:attributeGroup" />
    </xs:choice>
  </xs:group>

  <xs:simpleType name="formChoice">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="qualified" />
      <xs:enumeration value="unqualified" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="reducedDerivationControl">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:derivationControl">
      <xs:enumeration value="extension" />
      <xs:enumeration value="restriction" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="derivationSet">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
      <xs:documentation>
        #all or (possibly empty) subset of {extension, restriction}</xs:documentation>
    </xs:annotation>
    <xs:union>
      <xs:simpleType>
```

```
        <xs:restriction base="xs:token">
          <xs:enumeration value="#all" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:list itemType="xs:reducedDerivationControl" />
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>

  <xs:simpleType name="typeDerivationControl">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:derivationControl">
      <xs:enumeration value="extension" />
      <xs:enumeration value="restriction" />
      <xs:enumeration value="list" />
      <xs:enumeration value="union" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="fullDerivationSet">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
      <xs:documentation>
        #all or (possibly empty) subset of {extension, restriction, list,
union}</xs:documentation>
    </xs:annotation>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="#all" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:list itemType="xs:typeDerivationControl" />
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>

  <xs:element name="schema" id="schema">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-schema" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:openAttrs">
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="xs:include" />
              <xs:element ref="xs:import" />
              <xs:element ref="xs:redefine" />
              <xs:element ref="xs:annotation" />
            </xs:choice>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:group ref="xs:schemaTop" />
```

```
            <xs:element ref="xs:annotation" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:sequence>
        <xs:attribute name="targetNamespace" type="xs:anyURI" />
        <xs:attribute name="version" type="xs:token" />
        <xs:attribute name="finalDefault" type="xs:fullDerivationSet" use="optional"
default="" />
        <xs:attribute name="blockDefault" type="xs:blockSet" use="optional"
default="" />
        <xs:attribute name="attributeFormDefault" type="xs:formChoice"
use="optional" default="unqualified" />
        <xs:attribute name="elementFormDefault" type="xs:formChoice" use="optional"
default="unqualified" />
        <xs:attribute name="id" type="xs:ID" />
        <xs:attribute ref="xml:lang" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:key name="element">
    <xs:selector xpath="xs:element" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="attribute">
    <xs:selector xpath="xs:attribute" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="type">
    <xs:selector xpath="xs:complexType|xs:simpleType" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="group">
    <xs:selector xpath="xs:group" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="attributeGroup">
    <xs:selector xpath="xs:attributeGroup" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="notation">
    <xs:selector xpath="xs:notation" />
    <xs:field xpath="@name" />
  </xs:key>

  <xs:key name="identityConstraint">
    <xs:selector xpath=".//xs:key|.//xs:unique|.//xs:keyref" />
    <xs:field xpath="@name" />
  </xs:key>

</xs:element>

<xs:simpleType name="allNNI">
  <xs:annotation>
    <xs:documentation>
```

```
        for maxOccurs</xs:documentation>
    </xs:annotation>
    <xs:union memberTypes="xs:nonNegativeInteger">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="unbounded" />
        </xs:restriction>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>

  <xs:attributeGroup name="occurs">
    <xs:annotation>
      <xs:documentation>
        for all particles</xs:documentation>
    </xs:annotation>
    <xs:attribute name="minOccurs" type="xs:nonNegativeInteger" use="optional"
default="1" />
    <xs:attribute name="maxOccurs" type="xs:allNNI" use="optional" default="1" />
  </xs:attributeGroup>

  <xs:attributeGroup name="defRef">
    <xs:annotation>
      <xs:documentation>
        for element, group and attributeGroup,
        which both define and reference</xs:documentation>
    </xs:annotation>
    <xs:attribute name="name" type="xs:NCName" />
    <xs:attribute name="ref" type="xs:QName" />
  </xs:attributeGroup>

  <xs:group name="typeDefParticle">
    <xs:annotation>
      <xs:documentation>
        'complexType' uses this</xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:element name="group" type="xs:groupRef" />
      <xs:element ref="xs:all" />
      <xs:element ref="xs:choice" />
      <xs:element ref="xs:sequence" />
    </xs:choice>
  </xs:group>



  <xs:group name="nestedParticle">
    <xs:choice>
      <xs:element name="element" type="xs:localElement" />
      <xs:element name="group" type="xs:groupRef" />
      <xs:element ref="xs:choice" />
      <xs:element ref="xs:sequence" />
      <xs:element ref="xs:any" />
    </xs:choice>
  </xs:group>

  <xs:group name="particle">
    <xs:choice>
      <xs:element name="element" type="xs:localElement" />
```

```
    <xs:element name="group" type="xs:groupRef" />
    <xs:element ref="xs:all" />
    <xs:element ref="xs:choice" />
    <xs:element ref="xs:sequence" />
    <xs:element ref="xs:any" />
  </xs:choice>
</xs:group>

<xs:complexType name="attribute">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:sequence>
        <xs:element name="simpleType" minOccurs="0" type="xs:localSimpleType" />
      </xs:sequence>
      <xs:attributeGroup ref="xs:defRef" />
      <xs:attribute name="type" type="xs:QName" />
      <xs:attribute name="use" use="optional" default="optional">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="prohibited" />
            <xs:enumeration value="optional" />
            <xs:enumeration value="required" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="default" type="xs:string" />
      <xs:attribute name="fixed" type="xs:string" />
      <xs:attribute name="form" type="xs:formChoice" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="topLevelAttribute">
  <xs:complexContent>
    <xs:restriction base="xs:attribute">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:element name="simpleType" minOccurs="0" type="xs:localSimpleType" />
      </xs:sequence>
      <xs:attribute name="ref" use="prohibited" />
      <xs:attribute name="form" use="prohibited" />
      <xs:attribute name="use" use="prohibited" />
      <xs:attribute name="name" use="required" type="xs:NCName" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:group name="attrDecls">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="attribute" type="xs:attribute" />
      <xs:element name="attributeGroup" type="xs:attributeGroupRef" />
    </xs:choice>
    <xs:element ref="xs:anyAttribute" minOccurs="0" />
  </xs:sequence>
</xs:group>

<xs:element name="anyAttribute" type="xs:wildcard" id="anyAttribute">
```

```
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-
anyAttribute" />
    </xs:annotation>
  </xs:element>

  <xs:group name="complexTypeModel">
    <xs:choice>
      <xs:element ref="xs:simpleContent" />
      <xs:element ref="xs:complexContent" />
      <xs:sequence>
        <xs:annotation>
          <xs:documentation>
            This branch is short for
            &lt;complexContent>
            &lt;restriction base="xs:anyType">
            ...
            &lt;/restriction>
            &lt;/complexContent></xs:documentation>
        </xs:annotation>
        <xs:group ref="xs:typeDefParticle" minOccurs="0" />
        <xs:group ref="xs:attrDecls" />
      </xs:sequence>
    </xs:choice>
  </xs:group>

  <xs:complexType name="complexType" abstract="true">
    <xs:complexContent>
      <xs:extension base="xs:annotated">
        <xs:group ref="xs:complexTypeModel" />
        <xs:attribute name="name" type="xs:NCName">
          <xs:annotation>
            <xs:documentation>
              Will be restricted to required or forbidden</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="mixed" type="xs:boolean" use="optional" default="false">
          <xs:annotation>
            <xs:documentation>
              Not allowed if simpleContent child is chosen.
              May be overriden by setting on complexContent child.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="abstract" type="xs:boolean" use="optional" default="false"
/>
        <xs:attribute name="final" type="xs:derivationSet" />
        <xs:attribute name="block" type="xs:derivationSet" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="topLevelComplexType">
    <xs:complexContent>
      <xs:restriction base="xs:complexType">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
          <xs:group ref="xs:complexTypeModel" />
        </xs:sequence>
        <xs:attribute name="name" type="xs:NCName" use="required" />
```

```
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="localComplexType">
  <xs:complexContent>
    <xs:restriction base="xs:complexType">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:group ref="xs:complexTypeModel" />
      </xs:sequence>
      <xs:attribute name="name" use="prohibited" />
      <xs:attribute name="abstract" use="prohibited" />
      <xs:attribute name="final" use="prohibited" />
      <xs:attribute name="block" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="restrictionType">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:sequence>
        <xs:choice minOccurs="0">
          <xs:group ref="xs:typeDefParticle" />
          <xs:group ref="xs:simpleRestrictionModel" />
        </xs:choice>
        <xs:group ref="xs:attrDecls" />
      </xs:sequence>
      <xs:attribute name="base" type="xs:QName" use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="complexRestrictionType">
  <xs:complexContent>
    <xs:restriction base="xs:restrictionType">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:choice minOccurs="0">
          <xs:annotation>
            <xs:documentation>This choice is added simply to
              make this a valid restriction per the REC</xs:documentation>
          </xs:annotation>
          <xs:group ref="xs:typeDefParticle" />
        </xs:choice>
        <xs:group ref="xs:attrDecls" />
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="extensionType">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:sequence>
```

```
          <xs:group ref="xs:typeDefParticle" minOccurs="0" />
          <xs:group ref="xs:attrDecls" />
        </xs:sequence>
        <xs:attribute name="base" type="xs:QName" use="required" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="complexContent" id="complexContent">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-
complexContent" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:choice>
            <xs:element name="restriction" type="xs:complexRestrictionType" />
            <xs:element name="extension" type="xs:extensionType" />
          </xs:choice>
          <xs:attribute name="mixed" type="xs:boolean">
            <xs:annotation>
              <xs:documentation>
                Overrides any setting on complexType parent.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="simpleRestrictionType">
    <xs:complexContent>
      <xs:restriction base="xs:restrictionType">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
          <xs:choice minOccurs="0">
            <xs:annotation>
              <xs:documentation>This choice is added simply to
                make this a valid restriction per the REC</xs:documentation>
            </xs:annotation>
            <xs:group ref="xs:simpleRestrictionModel" />
          </xs:choice>
          <xs:group ref="xs:attrDecls" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="simpleExtensionType">
    <xs:complexContent>
      <xs:restriction base="xs:extensionType">
        <xs:sequence>
          <xs:annotation>
            <xs:documentation>
              No typeDefParticle group reference</xs:documentation>
          </xs:annotation>
          <xs:element ref="xs:annotation" minOccurs="0" />
```

```
          <xs:group ref="xs:attrDecls" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="simpleContent" id="simpleContent">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-
simpleContent" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:choice>
            <xs:element name="restriction" type="xs:simpleRestrictionType" />
            <xs:element name="extension" type="xs:simpleExtensionType" />
          </xs:choice>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="complexType" type="xs:topLevelComplexType" id="complexType">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-complexType"
/>
    </xs:annotation>
  </xs:element>


  <xs:simpleType name="blockSet">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
      <xs:documentation>
        #all or (possibly empty) subset of {substitution, extension,
        restriction}</xs:documentation>
    </xs:annotation>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="#all" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:list>
          <xs:simpleType>
            <xs:restriction base="xs:derivationControl">
              <xs:enumeration value="extension" />
              <xs:enumeration value="restriction" />
              <xs:enumeration value="substitution" />
            </xs:restriction>
          </xs:simpleType>
        </xs:list>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
```

```
  <xs:complexType name="element" abstract="true">
    <xs:annotation>
      <xs:documentation>
        The element element can be used either
        at the top level to define an element-type binding globally,
        or within a content model to either reference a globally-defined
        element or type or declare an element-type binding locally.
        The ref form is not allowed at the top level.</xs:documentation>
    </xs:annotation>

    <xs:complexContent>
      <xs:extension base="xs:annotated">
        <xs:sequence>
          <xs:choice minOccurs="0">
            <xs:element name="simpleType" type="xs:localSimpleType" />
            <xs:element name="complexType" type="xs:localComplexType" />
          </xs:choice>
          <xs:group ref="xs:identityConstraint" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attributeGroup ref="xs:defRef" />
        <xs:attribute name="type" type="xs:QName" />
        <xs:attribute name="substitutionGroup" type="xs:QName" />
        <xs:attributeGroup ref="xs:occurs" />
        <xs:attribute name="default" type="xs:string" />
        <xs:attribute name="fixed" type="xs:string" />
        <xs:attribute name="nillable" type="xs:boolean" use="optional" default="false"
/>
        <xs:attribute name="abstract" type="xs:boolean" use="optional" default="false"
/>
        <xs:attribute name="final" type="xs:derivationSet" />
        <xs:attribute name="block" type="xs:blockSet" />
        <xs:attribute name="form" type="xs:formChoice" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="topLevelElement">
    <xs:complexContent>
      <xs:restriction base="xs:element">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
          <xs:choice minOccurs="0">
            <xs:element name="simpleType" type="xs:localSimpleType" />
            <xs:element name="complexType" type="xs:localComplexType" />
          </xs:choice>
          <xs:group ref="xs:identityConstraint" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="ref" use="prohibited" />
        <xs:attribute name="form" use="prohibited" />
        <xs:attribute name="minOccurs" use="prohibited" />
        <xs:attribute name="maxOccurs" use="prohibited" />
        <xs:attribute name="name" use="required" type="xs:NCName" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="localElement">
```

```
  <xs:complexContent>
    <xs:restriction base="xs:element">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:choice minOccurs="0">
          <xs:element name="simpleType" type="xs:localSimpleType" />
          <xs:element name="complexType" type="xs:localComplexType" />
        </xs:choice>
        <xs:group ref="xs:identityConstraint" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="substitutionGroup" use="prohibited" />
      <xs:attribute name="final" use="prohibited" />
      <xs:attribute name="abstract" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:element name="element" type="xs:topLevelElement" id="element">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-element" />
  </xs:annotation>
</xs:element>

<xs:complexType name="group" abstract="true">
  <xs:annotation>
    <xs:documentation>
      group type for explicit groups, named top-level groups and
      group references</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:group ref="xs:particle" minOccurs="0" maxOccurs="unbounded" />
      <xs:attributeGroup ref="xs:defRef" />
      <xs:attributeGroup ref="xs:occurs" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="realGroup">
  <xs:complexContent>
    <xs:restriction base="xs:group">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:choice minOccurs="0" maxOccurs="1">
          <xs:element ref="xs:all" />
          <xs:element ref="xs:choice" />
          <xs:element ref="xs:sequence" />
        </xs:choice>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="namedGroup">
  <xs:complexContent>
    <xs:restriction base="xs:realGroup">
      <xs:sequence>
```

```
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:choice minOccurs="1" maxOccurs="1">
          <xs:element name="all">
            <xs:complexType>
              <xs:complexContent>
                <xs:restriction base="xs:all">
                  <xs:group ref="xs:allModel" />
                  <xs:attribute name="minOccurs" use="prohibited" />
                  <xs:attribute name="maxOccurs" use="prohibited" />
                  <xs:anyAttribute namespace="##other" processContents="lax" />
                </xs:restriction>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="choice" type="xs:simpleExplicitGroup" />
          <xs:element name="sequence" type="xs:simpleExplicitGroup" />
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="name" use="required" type="xs:NCName" />
      <xs:attribute name="ref" use="prohibited" />
      <xs:attribute name="minOccurs" use="prohibited" />
      <xs:attribute name="maxOccurs" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="groupRef">
  <xs:complexContent>
    <xs:restriction base="xs:realGroup">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="ref" use="required" type="xs:QName" />
      <xs:attribute name="name" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="explicitGroup">
  <xs:annotation>
    <xs:documentation>
      group type for the three kinds of group</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="xs:group">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:group ref="xs:nestedParticle" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:NCName" use="prohibited" />
      <xs:attribute name="ref" type="xs:QName" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="simpleExplicitGroup">
```

```
  <xs:complexContent>
    <xs:restriction base="xs:explicitGroup">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:group ref="xs:nestedParticle" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="minOccurs" use="prohibited" />
      <xs:attribute name="maxOccurs" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:group name="allModel">
  <xs:sequence>
    <xs:element ref="xs:annotation" minOccurs="0" />
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>This choice with min/max is here to
          avoid a pblm with the Elt:All/Choice/Seq
          Particle derivation constraint</xs:documentation>
      </xs:annotation>
      <xs:element name="element" type="xs:narrowMaxMin" />
    </xs:choice>
  </xs:sequence>
</xs:group>


<xs:complexType name="narrowMaxMin">
  <xs:annotation>
    <xs:documentation>restricted max/min</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="xs:localElement">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:choice minOccurs="0">
          <xs:element name="simpleType" type="xs:localSimpleType" />
          <xs:element name="complexType" type="xs:localComplexType" />
        </xs:choice>
        <xs:group ref="xs:identityConstraint" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="minOccurs" use="optional" default="1">
        <xs:simpleType>
          <xs:restriction base="xs:nonNegativeInteger">
            <xs:enumeration value="0" />
            <xs:enumeration value="1" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="maxOccurs" use="optional" default="1">
        <xs:simpleType>
          <xs:restriction base="xs:allNNI">
            <xs:enumeration value="0" />
            <xs:enumeration value="1" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax" />
```

```
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="all">
  <xs:annotation>
    <xs:documentation>
      Only elements allowed inside</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="xs:explicitGroup">
      <xs:group ref="xs:allModel" />
      <xs:attribute name="minOccurs" use="optional" default="1">
        <xs:simpleType>
          <xs:restriction base="xs:nonNegativeInteger">
            <xs:enumeration value="0" />
            <xs:enumeration value="1" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="maxOccurs" use="optional" default="1">
        <xs:simpleType>
          <xs:restriction base="xs:allNNI">
            <xs:enumeration value="1" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:element name="all" id="all" type="xs:all">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-all" />
  </xs:annotation>
</xs:element>

<xs:element name="choice" type="xs:explicitGroup" id="choice">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-choice" />
  </xs:annotation>
</xs:element>

<xs:element name="sequence" type="xs:explicitGroup" id="sequence">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-sequence" />
  </xs:annotation>
</xs:element>

<xs:element name="group" type="xs:namedGroup" id="group">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-group" />
  </xs:annotation>
</xs:element>

<xs:complexType name="wildcard">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
```

```
        <xs:attribute name="namespace" type="xs:namespaceList" use="optional"
default="##any" />
        <xs:attribute name="processContents" use="optional" default="strict">
          <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="skip" />
              <xs:enumeration value="lax" />
              <xs:enumeration value="strict" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="any" id="any">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-any" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:wildcard">
          <xs:attributeGroup ref="xs:occurs" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:annotation>
    <xs:documentation>
      simple type for the value of the 'namespace' attr of
      'any' and 'anyAttribute'</xs:documentation>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      Value is
      ##any - - any non-conflicting WFXML/attribute at all

      ##other - - any non-conflicting WFXML/attribute from
      namespace other than targetNS

      ##local - - any unqualified non-conflicting WFXML/attribute

      one or - - any non-conflicting WFXML/attribute from
      more URI the listed namespaces
      references
      (space separated)

      ##targetNamespace or ##local may appear in the above list, to
      refer to the targetNamespace of the enclosing
      schema or an absent targetNamespace respectively</xs:documentation>
  </xs:annotation>

  <xs:simpleType name="namespaceList">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
    </xs:annotation>
    <xs:union>
```

```
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="##any" />
          <xs:enumeration value="##other" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:list>
          <xs:simpleType>
            <xs:union memberTypes="xs:anyURI">
              <xs:simpleType>
                <xs:restriction base="xs:token">
                  <xs:enumeration value="##targetNamespace" />
                  <xs:enumeration value="##local" />
                </xs:restriction>
              </xs:simpleType>
            </xs:union>
          </xs:simpleType>
        </xs:list>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>

  <xs:element name="attribute" type="xs:topLevelAttribute" id="attribute">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-attribute"
/>
    </xs:annotation>
  </xs:element>

  <xs:complexType name="attributeGroup" abstract="true">
    <xs:complexContent>
      <xs:extension base="xs:annotated">
        <xs:group ref="xs:attrDecls" />
        <xs:attributeGroup ref="xs:defRef" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="namedAttributeGroup">
    <xs:complexContent>
      <xs:restriction base="xs:attributeGroup">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
          <xs:group ref="xs:attrDecls" />
        </xs:sequence>
        <xs:attribute name="name" use="required" type="xs:NCName" />
        <xs:attribute name="ref" use="prohibited" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="attributeGroupRef">
    <xs:complexContent>
      <xs:restriction base="xs:attributeGroup">
        <xs:sequence>
          <xs:element ref="xs:annotation" minOccurs="0" />
        </xs:sequence>
```

```
        <xs:attribute name="ref" use="required" type="xs:QName" />
        <xs:attribute name="name" use="prohibited" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="attributeGroup" type="xs:namedAttributeGroup" id="attributeGroup">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-
attributeGroup" />
    </xs:annotation>
  </xs:element>

  <xs:element name="include" id="include">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-include" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:attribute name="schemaLocation" type="xs:anyURI" use="required" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="redefine" id="redefine">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-redefine" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:openAttrs">
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xs:annotation" />
            <xs:group ref="xs:redefinable" />
          </xs:choice>
          <xs:attribute name="schemaLocation" type="xs:anyURI" use="required" />
          <xs:attribute name="id" type="xs:ID" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="import" id="import">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-import" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:attribute name="namespace" type="xs:anyURI" />
          <xs:attribute name="schemaLocation" type="xs:anyURI" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="selector" id="selector">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-selector" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:attribute name="xpath" use="required">
            <xs:simpleType>
              <xs:annotation>
                <xs:documentation>A subset of XPath expressions for use
                  in selectors</xs:documentation>
                <xs:documentation>A utility type, not for public
                  use</xs:documentation>
              </xs:annotation>
              <xs:restriction base="xs:token">
                <xs:annotation>
                  <xs:documentation>The following pattern is intended to allow XPath
                    expressions per the following EBNF:
                    Selector ::= Path ( '|' Path )*
                    Path ::= ('.//')? Step ( '/' Step )*
                    Step ::= '.' | NameTest
                    NameTest ::= QName | '*' | NCName ':' '*'
                    child:: is also allowed
         </xs:documentation>
                </xs:annotation>
                <xs:pattern

value="(\.//)?(((child::)?((\i\c*:)?(\i\c*|\*))|\.)(((((child::)?((\i\c*:)?(\i\c*|\*)
))|\.))*(\|(\.//)?(((child::)?((\i\c*:)?(\i\c*|\*))|\.)(((((child::)?((\i\c*:)?(\i\c*
|\*))|\.))*)*">
                </xs:pattern>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="field" id="field">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-field" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:attribute name="xpath" use="required">
            <xs:simpleType>
              <xs:annotation>
                <xs:documentation>A subset of XPath expressions for use
                  in fields</xs:documentation>
                <xs:documentation>A utility type, not for public
                  use</xs:documentation>
              </xs:annotation>
              <xs:restriction base="xs:token">
                <xs:annotation>
                  <xs:documentation>The following pattern is intended to allow XPath
                    expressions per the same EBNF as for selector,
```

```
                    with the following change:
                    Path ::= ('.//')? ( Step '/' )* ( Step | '@' NameTest )
          </xs:documentation>
                </xs:annotation>
                <xs:pattern

value="(\.//)?((((child::)?((\i\c*:)?(\i\c*|\*)))|\.)/)*((((child::)?((\i\c*:)?(\i\c*|
\*)))|\.)|((attribute::|@)((\i\c*:)?(\i\c*|\*))))(\|(\.//)?((((child::)?((\i\c*:)?(\i\
c*|\*)))|\.)/)*((((child::)?((\i\c*:)?(\i\c*|\*)))|\.)|((attribute::|@)((\i\c*:)?(\i\c
*|\*))))*)*">
                </xs:pattern>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="keybase">
    <xs:complexContent>
      <xs:extension base="xs:annotated">
        <xs:sequence>
          <xs:element ref="xs:selector" />
          <xs:element ref="xs:field" minOccurs="1" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="name" type="xs:NCName" use="required" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:group name="identityConstraint">
    <xs:annotation>
      <xs:documentation>The three kinds of identity constraints, all with
        type of or derived from 'keybase'.
   </xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:element ref="xs:unique" />
      <xs:element ref="xs:key" />
      <xs:element ref="xs:keyref" />
    </xs:choice>
  </xs:group>

  <xs:element name="unique" type="xs:keybase" id="unique">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-unique" />
    </xs:annotation>
  </xs:element>
  <xs:element name="key" type="xs:keybase" id="key">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-key" />
    </xs:annotation>
  </xs:element>
  <xs:element name="keyref" id="keyref">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-keyref" />
    </xs:annotation>
    <xs:complexType>
```

```
      <xs:complexContent>
        <xs:extension base="xs:keybase">
          <xs:attribute name="refer" type="xs:QName" use="required" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="notation" id="notation">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-notation" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:attribute name="name" type="xs:NCName" use="required" />
          <xs:attribute name="public" type="xs:public" />
          <xs:attribute name="system" type="xs:anyURI" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="public">
    <xs:annotation>
      <xs:documentation>
        A utility type, not for public use</xs:documentation>
      <xs:documentation>
        A public identifier, per ISO 8879</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token" />
  </xs:simpleType>

  <xs:element name="appinfo" id="appinfo">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-appinfo" />
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:any processContents="lax" />
      </xs:sequence>
      <xs:attribute name="source" type="xs:anyURI" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
  </xs:element>

  <xs:element name="documentation" id="documentation">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-
documentation" />
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:any processContents="lax" />
      </xs:sequence>
      <xs:attribute name="source" type="xs:anyURI" />
      <xs:attribute ref="xml:lang" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
```

```
  </xs:element>

  <xs:element name="annotation" id="annotation">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-1/#element-annotation"
/>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="xs:openAttrs">
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xs:appinfo" />
            <xs:element ref="xs:documentation" />
          </xs:choice>
          <xs:attribute name="id" type="xs:ID" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:annotation>
    <xs:documentation>
      notations for use within XML Schema schemas</xs:documentation>
  </xs:annotation>

  <xs:notation name="XMLSchemaStructures" public="structures"
system="http://www.w3.org/2000/08/XMLSchema.xsd" />
  <xs:notation name="XML" public="REC-xml-19980210"
system="http://www.w3.org/TR/1998/REC-xml-19980210" />

  <xs:complexType name="anyType" mixed="true">
    <xs:annotation>
      <xs:documentation>
        Not the real urType, but as close an approximation as we can
        get in the XML representation</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
    </xs:sequence>
    <xs:anyAttribute processContents="lax" />
  </xs:complexType>

  <xs:annotation>
    <xs:documentation>
      First the built-in primitive datatypes. These definitions are for
      information only, the real built-in definitions are magic.
    </xs:documentation>

    <xs:documentation>
      For each built-in datatype in this schema (both primitive and
      derived) can be uniquely addressed via a URI constructed
      as follows:
      1) the base URI is the URI of the XML Schema namespace
      2) the fragment identifier is the name of the datatype

      For example, to address the int datatype, the URI is:

      http://www.w3.org/2001/XMLSchema#int
```

```
    Additionally, each facet definition element can be uniquely
    addressed via a URI constructed as follows:
    1) the base URI is the URI of the XML Schema namespace
    2) the fragment identifier is the name of the facet

    For example, to address the maxInclusive facet, the URI is:

    http://www.w3.org/2001/XMLSchema#maxInclusive

    Additionally, each facet usage in a built-in datatype definition
    can be uniquely addressed via a URI constructed as follows:
    1) the base URI is the URI of the XML Schema namespace
    2) the fragment identifier is the name of the datatype, followed
    by a period (".") followed by the name of the facet

    For example, to address the usage of the maxInclusive facet in
    the definition of int, the URI is:

    http://www.w3.org/2001/XMLSchema#int.maxInclusive

  </xs:documentation>
</xs:annotation>

<xs:simpleType name="string" id="string">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#string" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="preserve" id="string.preserve" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="boolean" id="boolean">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="finite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#boolean" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="boolean.whiteSpace" />
  </xs:restriction>
```

```
</xs:simpleType>

<xs:simpleType name="float" id="float">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="total" />
      <hfp:hasProperty name="bounded" value="true" />
      <hfp:hasProperty name="cardinality" value="finite" />
      <hfp:hasProperty name="numeric" value="true" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#float" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="float.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="double" id="double">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="total" />
      <hfp:hasProperty name="bounded" value="true" />
      <hfp:hasProperty name="cardinality" value="finite" />
      <hfp:hasProperty name="numeric" value="true" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#double" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="double.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="decimal" id="decimal">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="totalDigits" />
      <hfp:hasFacet name="fractionDigits" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="total" />
```

```
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="true" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#decimal" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="decimal.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="duration" id="duration">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#duration" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="duration.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="dateTime" id="dateTime">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#dateTime" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="dateTime.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="time" id="time">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
```

```
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#time" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="time.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="date" id="date">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#date" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="date.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="gYearMonth" id="gYearMonth">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gYearMonth" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="gYearMonth.whiteSpace" />
```

```
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="gYear" id="gYear">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gYear" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="gYear.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="gMonthDay" id="gMonthDay">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gMonthDay" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="gMonthDay.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="gDay" id="gDay">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
```

```
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gDay" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="gDay.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="gMonth" id="gMonth">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="maxInclusive" />
      <hfp:hasFacet name="maxExclusive" />
      <hfp:hasFacet name="minInclusive" />
      <hfp:hasFacet name="minExclusive" />
      <hfp:hasProperty name="ordered" value="partial" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#gMonth" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="gMonth.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="hexBinary" id="hexBinary">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#binary" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="hexBinary.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="base64Binary" id="base64Binary">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
```

```
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#base64Binary" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="base64Binary.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="anyURI" id="anyURI">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#anyURI" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="anyURI.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="QName" id="QName">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#QName" />
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="QName.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NOTATION" id="NOTATION">
  <xs:annotation>
```

```
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#NOTATION" />
    <xs:documentation>
      NOTATION cannot be used directly in a schema; rather a type
      must be derived from it by specifying at least one enumeration
      facet whose value is the name of a NOTATION declared in the
      schema.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anySimpleType">
    <xs:whiteSpace value="collapse" fixed="true" id="NOTATION.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:annotation>
  <xs:documentation>
    Now the derived primitive types
  </xs:documentation>
</xs:annotation>

<xs:simpleType name="normalizedString" id="normalizedString">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#normalizedString" />
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="replace" id="normalizedString.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="token" id="token">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#token" />
  </xs:annotation>
  <xs:restriction base="xs:normalizedString">
    <xs:whiteSpace value="collapse" id="token.whiteSpace" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="language" id="language">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#language" />
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:pattern value="[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*" id="language.pattern">
      <xs:annotation>
        <xs:documentation source="http://www.ietf.org/rfc/rfc3066.txt">
          pattern specifies the content of section 2.12 of XML 1.0e2
          and RFC 3066 (Revised version of RFC 1766).
```

```
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="IDREFS" id="IDREFS">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#IDREFS" />
  </xs:annotation>
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:IDREF" />
    </xs:simpleType>
    <xs:minLength value="1" id="IDREFS.minLength" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ENTITIES" id="ENTITIES">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#ENTITIES" />
  </xs:annotation>
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:ENTITY" />
    </xs:simpleType>
    <xs:minLength value="1" id="ENTITIES.minLength" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NMTOKEN" id="NMTOKEN">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#NMTOKEN" />
  </xs:annotation>
  <xs:restriction base="xs:token">
```

```
    <xs:pattern value="\c+" id="NMTOKEN.pattern">
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/REC-xml#NT-Nmtoken">
          pattern matches production 7 from the XML spec
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NMTOKENS" id="NMTOKENS">
  <xs:annotation>
    <xs:appinfo>
      <hfp:hasFacet name="length" />
      <hfp:hasFacet name="minLength" />
      <hfp:hasFacet name="maxLength" />
      <hfp:hasFacet name="enumeration" />
      <hfp:hasFacet name="whiteSpace" />
      <hfp:hasFacet name="pattern" />
      <hfp:hasProperty name="ordered" value="false" />
      <hfp:hasProperty name="bounded" value="false" />
      <hfp:hasProperty name="cardinality" value="countably infinite" />
      <hfp:hasProperty name="numeric" value="false" />
    </xs:appinfo>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#NMTOKENS" />
  </xs:annotation>
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:NMTOKEN" />
    </xs:simpleType>
    <xs:minLength value="1" id="NMTOKENS.minLength" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Name" id="Name">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#Name" />
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:pattern value="\i\c*" id="Name.pattern">
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/REC-xml#NT-Name">
          pattern matches production 5 from the XML spec
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NCName" id="NCName">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#NCName" />
  </xs:annotation>
  <xs:restriction base="xs:Name">
    <xs:pattern value="[\i-[:]][\c-[:]]*" id="NCName.pattern">
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/REC-xml-names/#NT-NCName">
          pattern matches production 4 from the Namespaces in XML spec
        </xs:documentation>
```

```
        </xs:annotation>
      </xs:pattern>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ID" id="ID">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#ID" />
    </xs:annotation>
    <xs:restriction base="xs:NCName" />
  </xs:simpleType>

  <xs:simpleType name="IDREF" id="IDREF">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#IDREF" />
    </xs:annotation>
    <xs:restriction base="xs:NCName" />
  </xs:simpleType>

  <xs:simpleType name="ENTITY" id="ENTITY">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#ENTITY" />
    </xs:annotation>
    <xs:restriction base="xs:NCName" />
  </xs:simpleType>

  <xs:simpleType name="integer" id="integer">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#integer" />
    </xs:annotation>
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="0" fixed="true" id="integer.fractionDigits" />
      <xs:pattern value="[\-+]?[0-9]+" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="nonPositiveInteger" id="nonPositiveInteger">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#nonPositiveInteger"
/>
    </xs:annotation>
    <xs:restriction base="xs:integer">
      <xs:maxInclusive value="0" id="nonPositiveInteger.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="negativeInteger" id="negativeInteger">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#negativeInteger" />
    </xs:annotation>
    <xs:restriction base="xs:nonPositiveInteger">
      <xs:maxInclusive value="-1" id="negativeInteger.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="long" id="long">
    <xs:annotation>
      <xs:appinfo>
        <hfp:hasProperty name="bounded" value="true" />
```

```
        <hfp:hasProperty name="cardinality" value="finite" />
      </xs:appinfo>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#long" />
    </xs:annotation>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="-9223372036854775808" id="long.minInclusive" />
      <xs:maxInclusive value="9223372036854775807" id="long.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="int" id="int">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#int" />
    </xs:annotation>
    <xs:restriction base="xs:long">
      <xs:minInclusive value="-2147483648" id="int.minInclusive" />
      <xs:maxInclusive value="2147483647" id="int.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="short" id="short">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#short" />
    </xs:annotation>
    <xs:restriction base="xs:int">
      <xs:minInclusive value="-32768" id="short.minInclusive" />
      <xs:maxInclusive value="32767" id="short.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="byte" id="byte">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#byte" />
    </xs:annotation>
    <xs:restriction base="xs:short">
      <xs:minInclusive value="-128" id="byte.minInclusive" />
      <xs:maxInclusive value="127" id="byte.maxInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="nonNegativeInteger" id="nonNegativeInteger">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#nonNegativeInteger"
/>
    </xs:annotation>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0" id="nonNegativeInteger.minInclusive" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="unsignedLong" id="unsignedLong">
    <xs:annotation>
      <xs:appinfo>
        <hfp:hasProperty name="bounded" value="true" />
        <hfp:hasProperty name="cardinality" value="finite" />
      </xs:appinfo>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#unsignedLong" />
    </xs:annotation>
    <xs:restriction base="xs:nonNegativeInteger">
```

```
    <xs:maxInclusive value="18446744073709551615" id="unsignedLong.maxInclusive" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="unsignedInt" id="unsignedInt">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#unsignedInt" />
  </xs:annotation>
  <xs:restriction base="xs:unsignedLong">
    <xs:maxInclusive value="4294967295" id="unsignedInt.maxInclusive" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="unsignedShort" id="unsignedShort">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#unsignedShort" />
  </xs:annotation>
  <xs:restriction base="xs:unsignedInt">
    <xs:maxInclusive value="65535" id="unsignedShort.maxInclusive" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="unsignedByte" id="unsignedByte">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#unsignedByte" />
  </xs:annotation>
  <xs:restriction base="xs:unsignedShort">
    <xs:maxInclusive value="255" id="unsignedByte.maxInclusive" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="positiveInteger" id="positiveInteger">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#positiveInteger" />
  </xs:annotation>
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:minInclusive value="1" id="positiveInteger.minInclusive" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="derivationControl">
  <xs:annotation>
    <xs:documentation>
      A utility type, not for public use</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="substitution" />
    <xs:enumeration value="extension" />
    <xs:enumeration value="restriction" />
    <xs:enumeration value="list" />
    <xs:enumeration value="union" />
  </xs:restriction>
</xs:simpleType>

<xs:group name="simpleDerivation">
  <xs:choice>
    <xs:element ref="xs:restriction" />
    <xs:element ref="xs:list" />
    <xs:element ref="xs:union" />
```

```
  </xs:choice>
</xs:group>

<xs:simpleType name="simpleDerivationSet">
  <xs:annotation>
    <xs:documentation>
      #all or (possibly empty) subset of {restriction, union, list}
 </xs:documentation>
    <xs:documentation>
      A utility type, not for public use</xs:documentation>
  </xs:annotation>
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="#all" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:list>
        <xs:simpleType>
          <xs:restriction base="xs:derivationControl">
            <xs:enumeration value="list" />
            <xs:enumeration value="union" />
            <xs:enumeration value="restriction" />
          </xs:restriction>
        </xs:simpleType>
      </xs:list>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:complexType name="simpleType" abstract="true">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:group ref="xs:simpleDerivation" />
      <xs:attribute name="final" type="xs:simpleDerivationSet" />
      <xs:attribute name="name" type="xs:NCName">
        <xs:annotation>
          <xs:documentation>
            Can be restricted to required or forbidden
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="topLevelSimpleType">
  <xs:complexContent>
    <xs:restriction base="xs:simpleType">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:group ref="xs:simpleDerivation" />
      </xs:sequence>
      <xs:attribute name="name" use="required" type="xs:NCName">
        <xs:annotation>
          <xs:documentation>
            Required at the top level
          </xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="localSimpleType">
  <xs:complexContent>
    <xs:restriction base="xs:simpleType">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
        <xs:group ref="xs:simpleDerivation" />
      </xs:sequence>
      <xs:attribute name="name" use="prohibited">
        <xs:annotation>
          <xs:documentation>
            Forbidden when nested
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="final" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:element name="simpleType" type="xs:topLevelSimpleType" id="simpleType">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-simpleType"
/>
  </xs:annotation>
</xs:element>

<xs:group name="facets">
  <xs:annotation>
    <xs:documentation>
      We should use a substitution group for facets, but
      that's ruled out because it would allow users to
      add their own, which we're not ready for yet.
  </xs:documentation>
  </xs:annotation>
  <xs:choice>
    <xs:element ref="xs:minExclusive" />
    <xs:element ref="xs:minInclusive" />
    <xs:element ref="xs:maxExclusive" />
    <xs:element ref="xs:maxInclusive" />
    <xs:element ref="xs:totalDigits" />
    <xs:element ref="xs:fractionDigits" />
    <xs:element ref="xs:length" />
    <xs:element ref="xs:minLength" />
    <xs:element ref="xs:maxLength" />
    <xs:element ref="xs:enumeration" />
    <xs:element ref="xs:whiteSpace" />
    <xs:element ref="xs:pattern" />
  </xs:choice>
</xs:group>

<xs:group name="simpleRestrictionModel">
```

```
    <xs:sequence>
      <xs:element name="simpleType" type="xs:localSimpleType" minOccurs="0" />
      <xs:group ref="xs:facets" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:group>

  <xs:element name="restriction" id="restriction">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
restriction">
          base attribute and simpleType child are mutually
          exclusive, but one or other is required
        </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:group ref="xs:simpleRestrictionModel" />
          <xs:attribute name="base" type="xs:QName" use="optional" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="list" id="list">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-list">
          itemType attribute and simpleType child are mutually
          exclusive, but one or other is required
        </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:sequence>
            <xs:element name="simpleType" type="xs:localSimpleType" minOccurs="0" />
          </xs:sequence>
          <xs:attribute name="itemType" type="xs:QName" use="optional" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="union" id="union">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-union">
          memberTypes attribute must be non-empty or there must be
          at least one simpleType child
        </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="xs:annotated">
          <xs:sequence>
            <xs:element name="simpleType" type="xs:localSimpleType" minOccurs="0"
maxOccurs="unbounded" />
          </xs:sequence>
          <xs:attribute name="memberTypes" use="optional">
            <xs:simpleType>
```

```
            <xs:list itemType="xs:QName" />
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:complexType name="facet">
  <xs:complexContent>
    <xs:extension base="xs:annotated">
      <xs:attribute name="value" use="required" />
      <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="noFixedFacet">
  <xs:complexContent>
    <xs:restriction base="xs:facet">
      <xs:sequence>
        <xs:element ref="xs:annotation" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="fixed" use="prohibited" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:element name="minExclusive" id="minExclusive" type="xs:facet">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
minExclusive" />
  </xs:annotation>
</xs:element>
<xs:element name="minInclusive" id="minInclusive" type="xs:facet">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
minInclusive" />
  </xs:annotation>
</xs:element>

<xs:element name="maxExclusive" id="maxExclusive" type="xs:facet">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
maxExclusive" />
  </xs:annotation>
</xs:element>
<xs:element name="maxInclusive" id="maxInclusive" type="xs:facet">
  <xs:annotation>
    <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
maxInclusive" />
  </xs:annotation>
</xs:element>

<xs:complexType name="numFacet">
  <xs:complexContent>
    <xs:restriction base="xs:facet">
      <xs:sequence>
```

```
          <xs:element ref="xs:annotation" minOccurs="0" />
        </xs:sequence>
        <xs:attribute name="value" type="xs:nonNegativeInteger" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="totalDigits" id="totalDigits">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-totalDigits"
/>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="xs:numFacet">
          <xs:sequence>
            <xs:element ref="xs:annotation" minOccurs="0" />
          </xs:sequence>
          <xs:attribute name="value" type="xs:positiveInteger" use="required" />
          <xs:anyAttribute namespace="##other" processContents="lax" />
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="fractionDigits" id="fractionDigits" type="xs:numFacet">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-
fractionDigits" />
    </xs:annotation>
  </xs:element>

  <xs:element name="length" id="length" type="xs:numFacet">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-length" />
    </xs:annotation>
  </xs:element>
  <xs:element name="minLength" id="minLength" type="xs:numFacet">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-minLength"
/>
    </xs:annotation>
  </xs:element>
  <xs:element name="maxLength" id="maxLength" type="xs:numFacet">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-maxLength"
/>
    </xs:annotation>
  </xs:element>

  <xs:element name="enumeration" id="enumeration" type="xs:noFixedFacet">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-enumeration"
/>
    </xs:annotation>
  </xs:element>

  <xs:element name="whiteSpace" id="whiteSpace">
    <xs:annotation>
```

```
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-whiteSpace"
/>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="xs:facet">
          <xs:sequence>
            <xs:element ref="xs:annotation" minOccurs="0" />
          </xs:sequence>
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="preserve" />
                <xs:enumeration value="replace" />
                <xs:enumeration value="collapse" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:anyAttribute namespace="##other" processContents="lax" />
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="pattern" id="pattern">
    <xs:annotation>
      <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#element-pattern" />
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="xs:noFixedFacet">
          <xs:sequence>
            <xs:element ref="xs:annotation" minOccurs="0" />
          </xs:sequence>
          <xs:attribute name="value" type="xs:string" use="required" />
          <xs:anyAttribute namespace="##other" processContents="lax" />
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

7.22 xml.xsd

```
<?xml version='1.0'?>
<?xml-stylesheet href="../2008/09/xsd.xsl" type="text/xsl"?>
<xs:schema targetNamespace="http://www.w3.org/XML/1998/namespace"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns   ="http://www.w3.org/1999/xhtml"
  xml:lang="en">

 <xs:annotation>
  <xs:documentation>
   <div>
    <h1>About the XML namespace</h1>

    <div class="bodytext">
```

```
     <p>
      This schema document describes the XML namespace, in a form
      suitable for import by other schema documents.
     </p>
     <p>
      See <a href="http://www.w3.org/XML/1998/namespace.html">
      http://www.w3.org/XML/1998/namespace.html</a> and
      <a href="http://www.w3.org/TR/REC-xml">
      http://www.w3.org/TR/REC-xml</a> for information
      about this namespace.
     </p>
     <p>
      Note that local names in this namespace are intended to be
      defined only by the World Wide Web Consortium or its subgroups.
      The names currently defined in this namespace are listed below.
      They should not be used with conflicting semantics by any Working
      Group, specification, or document instance.
     </p>
     <p>
      See further below in this document for more information about <a
      href="#usage">how to refer to this schema document from your own
      XSD schema documents</a> and about <a href="#nsversioning">the
      namespace-versioning policy governing this schema document</a>.
     </p>
    </div>
   </div>
  </xs:documentation>
 </xs:annotation>

 <xs:attribute name="lang">
  <xs:annotation>
   <xs:documentation>
    <div>

     <h3>lang (as an attribute name)</h3>
     <p>
      denotes an attribute whose value
      is a language code for the natural language of the content of
      any element; its value is inherited.  This name is reserved
      by virtue of its definition in the XML specification.</p>

    </div>
    <div>
     <h4>Notes</h4>
     <p>
      Attempting to install the relevant ISO 2- and 3-letter
      codes as the enumerated possible values is probably never
      going to be a realistic possibility.
     </p>
     <p>
      See BCP 47 at <a href="http://www.rfc-editor.org/rfc/bcp/bcp47.txt">
       http://www.rfc-editor.org/rfc/bcp/bcp47.txt</a>
      and the IANA language subtag registry at
      <a href="http://www.iana.org/assignments/language-subtag-registry">
       http://www.iana.org/assignments/language-subtag-registry</a>
      for further information.
     </p>
     <p>
      The union allows for the 'un-declaration' of xml:lang with
```

```
     the empty string.
    </p>
   </div>
  </xs:documentation>
 </xs:annotation>
 <xs:simpleType>
  <xs:union memberTypes="xs:language">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value=""/>
    </xs:restriction>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
</xs:attribute>

<xs:attribute name="space">
 <xs:annotation>
  <xs:documentation>
   <div>

    <h3>space (as an attribute name)</h3>
    <p>
     denotes an attribute whose
     value is a keyword indicating what whitespace processing
     discipline is intended for the content of the element; its
     value is inherited.  This name is reserved by virtue of its
     definition in the XML specification.</p>

   </div>
  </xs:documentation>
 </xs:annotation>
 <xs:simpleType>
  <xs:restriction base="xs:NCName">
   <xs:enumeration value="default"/>
   <xs:enumeration value="preserve"/>
  </xs:restriction>
 </xs:simpleType>
</xs:attribute>

<xs:attribute name="base" type="xs:anyURI"> <xs:annotation>
  <xs:documentation>
   <div>

    <h3>base (as an attribute name)</h3>
    <p>
     denotes an attribute whose value
     provides a URI to be used as the base for interpreting any
     relative URIs in the scope of the element on which it
     appears; its value is inherited.  This name is reserved
     by virtue of its definition in the XML Base specification.</p>

    <p>
     See <a
     href="http://www.w3.org/TR/xmlbase/">http://www.w3.org/TR/xmlbase/</a>
     for information about this attribute.
    </p>
   </div>
  </xs:documentation>
```

```
 </xs:annotation>
</xs:attribute>

<xs:attribute name="id" type="xs:ID">
 <xs:annotation>
  <xs:documentation>
   <div>

     <h3>id (as an attribute name)</h3>
     <p>
      denotes an attribute whose value
      should be interpreted as if declared to be of type ID.
      This name is reserved by virtue of its definition in the
      xml:id specification.</p>

    <p>
     See <a
     href="http://www.w3.org/TR/xml-id/">http://www.w3.org/TR/xml-id/</a>
     for information about this attribute.
    </p>
   </div>
  </xs:documentation>
 </xs:annotation>
</xs:attribute>

<xs:attributeGroup name="specialAttrs">
 <xs:attribute ref="xml:base"/>
 <xs:attribute ref="xml:lang"/>
 <xs:attribute ref="xml:space"/>
 <xs:attribute ref="xml:id"/>
</xs:attributeGroup>

<xs:annotation>
 <xs:documentation>
  <div>

   <h3>Father (in any context at all)</h3>

   <div class="bodytext">
    <p>
     denotes Jon Bosak, the chair of
     the original XML Working Group.  This name is reserved by
     the following decision of the W3C XML Plenary and
     XML Coordination groups:
    </p>
    <blockquote>
      <p>
     In appreciation for his vision, leadership and
     dedication the W3C XML Plenary on this 10th day of
     February, 2000, reserves for Jon Bosak in perpetuity
     the XML name "xml:Father".
      </p>
    </blockquote>
   </div>
  </div>
 </xs:documentation>
</xs:annotation>

<xs:annotation>
```

```
 <xs:documentation>
  <div xml:id="usage" id="usage">
   <h2><a name="usage">About this schema document</a></h2>

   <div class="bodytext">
    <p>
     This schema defines attributes and an attribute group suitable
     for use by schemas wishing to allow <code>xml:base</code>,
     <code>xml:lang</code>, <code>xml:space</code> or
     <code>xml:id</code> attributes on elements they define.
    </p>
    <p>
     To enable this, such a schema must import this schema for
     the XML namespace, e.g. as follows:
    </p>
    <pre>
          &lt;schema . . .>
           . . .
           &lt;import namespace="http://www.w3.org/XML/1998/namespace"
                       schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    </pre>
    <p>
     or
    </p>
    <pre>
           &lt;import namespace="http://www.w3.org/XML/1998/namespace"
                       schemaLocation="http://www.w3.org/2009/01/xml.xsd"/>
    </pre>
    <p>
     Subsequently, qualified reference to any of the attributes or the
     group defined below will have the desired effect, e.g.
    </p>
    <pre>
          &lt;type . . .>
           . . .
           &lt;attributeGroup ref="xml:specialAttrs"/>
    </pre>
    <p>
     will define a type which will schema-validate an instance element
     with any of those attributes.
    </p>
   </div>
  </div>
 </xs:documentation>
</xs:annotation>

<xs:annotation>
 <xs:documentation>
  <div id="nsversioning" xml:id="nsversioning">
   <h2><a name="nsversioning">Versioning policy for this schema document</a></h2>
   <div class="bodytext">
    <p>
     In keeping with the XML Schema WG's standard versioning
     policy, this schema document will persist at
     <a href="http://www.w3.org/2009/01/xml.xsd">
      http://www.w3.org/2009/01/xml.xsd</a>.
    </p>
    <p>
     At the date of issue it can also be found at
```

```
      <a href="http://www.w3.org/2001/xml.xsd">
       http://www.w3.org/2001/xml.xsd</a>.
     </p>
     <p>
      The schema document at that URI may however change in the future,
      in order to remain compatible with the latest version of XML
      Schema itself, or with the XML namespace itself.  In other words,
      if the XML Schema or XML namespaces change, the version of this
      document at <a href="http://www.w3.org/2001/xml.xsd">
       http://www.w3.org/2001/xml.xsd
      </a>
      will change accordingly; the version at
      <a href="http://www.w3.org/2009/01/xml.xsd">
       http://www.w3.org/2009/01/xml.xsd
      </a>
      will not change.
     </p>
     <p>
      Previous dated (and unchanging) versions of this schema
      document are at:
     </p>
     <ul>
      <li><a href="http://www.w3.org/2009/01/xml.xsd">
       http://www.w3.org/2009/01/xml.xsd</a></li>
      <li><a href="http://www.w3.org/2007/08/xml.xsd">
       http://www.w3.org/2007/08/xml.xsd</a></li>
      <li><a href="http://www.w3.org/2004/10/xml.xsd">
       http://www.w3.org/2004/10/xml.xsd</a></li>
      <li><a href="http://www.w3.org/2001/03/xml.xsd">
       http://www.w3.org/2001/03/xml.xsd</a></li>
     </ul>
    </div>
   </div>
  </xs:documentation>
 </xs:annotation>

</xs:schema>
```

8 References

Ref.	Document Number	Version	Title
1.	IAWG-ECOA-TR-001	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume I Key Concepts
2.	IAWG-ECOA-TR-002	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume II Developers Guide
3.	IAWG-ECOA-TR-003	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 1: Ada Binding Reference Manual
4.	IAWG-ECOA-TR-004	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 2: C Binding Reference Manual
5.	IAWG-ECOA-TR-005	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 3: C++ Binding Reference Manual
6.	IAWG-ECOA-TR-006	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 4: ELI and Transport Binding Reference Manual
7.	IAWG-ECOA-TR-007	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 5: Mechanisms Reference Manual
8.	IAWG-ECOA-TR-008	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 6: Platform Requirements Reference Manual
9.	IAWG-ECOA-TR-009	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 7: Approach to Safety and Security Reference Manual
10.	IAWG-ECOA-TR-010	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume III Part 8: Software Interface Reference Manual
11.	IAWG-ECOA-TR-012	Issue 2	European Component Oriented Architecture (ECOA) Collaboration Programme: Volume lV Common Terminology

Table 7 - Table of ECOA references