IMS Application Profile Guidelines Technical Manual

Part 2 – Technical Manual
Version 1.0

Date Issued: 10 October 2005

Latest version: http://www.imsglobal.org/ap/

IPR and Distribution Notices

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

IMS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on IMS's procedures with respect to rights in IMS specifications can be found at the IMS Intellectual Property Rights web page: http://www.imsglobal.org/ipr/imsipr_policyFinal.pdf.

If you wish to copy or distribute this document, you must complete a valid Registered User license registration with IMS and receive an email from IMS granting the license to distribute the specification. To register, follow the instructions on the IMS website: http://www.imsglobal.org/specificationdownload.cfm.

This document may be copied and furnished to others by Registered Users who have registered on the IMS website provided that the above copyright notice and this paragraph are included on all such copies. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to IMS, except as needed for the purpose of developing IMS specifications, under the auspices of a chartered IMS project group.

Use of this specification to develop products or services is governed by the license with IMS found on the IMS website: http://www.imsglobal.org/license.html.

The limited permissions granted above are perpetual and will not be revoked by IMS or its successors or assigns.

THIS SPECIFICATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE CONSORTIUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

Document Name: IMS Application Profile Guidelines Technical Manual

Date: 10 October 2005

Executive Summary

This document is the second of two parts which together constitute the IMS Application Profile Guidelines:

- Part 1 – Management Overview;
- Part 2 – Technical Manual.

This Technical Manual describes, from a technical point of view, the profiling of specifications, primarily those developed by the IMS Global Learning Consortium. However, the approach defined in this manual could also be applied to specifications developed by other organizations. In addition to evaluating the methods for defining an application profile, this document provides a means to represent changes to a specification in an XML form. This document also describes a way in which profiles could be tested for conformance purposes, but this is only provided as a suggestion rather than as a normative specification for conformance testing.

The document describes in detail the operations that are available to adapt a specification given by a data model and an XML schema. All these operations may affect interoperability. It is acknowledged that there are situations where nevertheless, the need to deliver community-specific data and services in conjunction with data and services of general relevance requires profiling of a specification. In these situations, the current paper will provide information about the consequences of particular profiling decisions with respect to interoperability and it may contain alternative solutions.

The development of an application profile begins with the establishment of a **community** of shared interests that can identify the need for an application profile and support its development and maintenance. This community, once established, looks at the **requirements** of its members, both in terms of existing systems and processes and its future needs. It also needs to take account of the specifications (and existing application profiles) available to it. Once requirements have been gathered, there needs to be a process of **analysis and synthesis**, where the community produces models of its domain, and identifies gaps in available specifications. It may also develop a reference architecture model to place the specifications within a system design context. At this point, the community has the information with which to make the **decision** whether or not it is appropriate to create a profile. If the community decides it is appropriate to create a profile, it can then make a **choice** of specifications and/or existing profiles, and begin **development** of the application profile. Finally, an initial application profile is created. Profiles need to be published, ideally in a registry of application profiles, so that other organizations wishing to create profiles can locate and reuse existing work. Where this is not applicable or practical, then the profile should at least be published formally with an official identifier.

For each specification utilized that involves a data model of some description, you need to define precisely how that model needs to be adapted to meet your requirements. There are a number of operations that can be used for adapting an information model, each with implications for interoperability within your community or application, and for retaining compatibility with the wider world.

For each operation mentioned below, guidelines are given on how to perform the operation, what the benefits and implications are for your profile and how the intent of the operation can be expressed. The modifications identified below are split into the following categories:

- Restrictive Modifications – instances of the profile can be used by tools and products that correctly implement the base schema in their read profile. The base schema can further be used to export data outside the immediate target community. It is recommended to use only these modifications in write profiles of applications;
- Extensive Modifications – instances of the profile schema may well no longer validate against the base schema and hence tools and products may require some customization to use instances to the resulting profile. Data will also need additional handling to achieve exchange with external communities. These operations are strongly discouraged for use by write profiles and should be adopted only if the communities' needs cannot be addressed by any other means. They are, however, of no harm in read profiles of applications. Applications using extensive modifications in their read profile may well use all instances of the base schema;
- Incompatible Modifications – these operations break interoperability with the base specification completely and should not be undertaken in any circumstances. Note that also the simultaneous usage of restrictive and extensive modifications will result in an incompatible profile.

Table of Contents

1. Introduction

1.1 Scope and Context

This document is the second of two parts which together constitute the IMS Application Profile Guidelines:

- Part 1 – Management Overview;
- Part 2 – Technical Manual.

This Technical Manual describes, from a technical point of view, the profiling of specifications, primarily those developed by the IMS Global Learning Consortium. However, the approach defined in this manual could also be applied to specifications developed by other organizations. In addition to evaluating the methods for defining an application profile, this document provides a means to represent changes to a specification in an XML form. This document also describes a way in which profiles could be tested for conformance purposes, but this is only provided as a suggestion rather than as a normative specification for conformance testing.

The accompanying Management Overview describes what an application profile is in the context of the IMS specifications and the benefits to be gained from undertaking such an exercise – namely more closely meeting the needs of the target user community whilst harnessing the specifications to aid integration and enhance interoperability between tools, products, and services which vendors would supply to that community. Guidance is offered on the key factors for deciding whether or not to embark upon a profiling exercise and a process outlined for how to proceed with such an activity. In sub-section 2.2.2 ‘Lessons Learned from Adoption’ of the Management Overview in particular, highlights the fact that adoption of the specifications often entails a selection of the specifications to adopt, some changes to the information model for these specifications and some adaptation (e.g., language, vocabularies) to serve a particular community. The available documentation for these profiles is highly variable and rarely captures the process by which they were derived. The Application Profile Guidelines makes explicit such a process in this Management Overview and offers further guidance in this, the Technical Manual, on how an application profile should be developed and documented. Conformance issues around an application profile are briefly discussed, as are technology and implementation issues beyond the scope covered by the specifications.

The document is offered as a guideline, based upon the experience of a number of user communities in adopting and implementing the specifications in the hope that their experience will be useful to others facing the same issues which they have had to work through with their users and suppliers. Nothing in this document is mandatory – ultimately the choices are made by implementers and the users of their offerings. However, the document does capture a viable process for helping vendors more closely meet the needs of a community, without necessarily breaking broader interoperability and maximizing the use of implementations against one or more base specifications.

Having opted to create an application profile in the manner prescribed, achieving interoperability across implementations of that profile is still dependent upon a number of independent, ongoing factors, not least:

- *Consistent interpretation by implementers of the application profile;*
- *Consistent use of vocabularies by the information sources;*
- *Consistent use of the information by users of the information.*

The paper describes in detail the operations that are available to adapt a specification given by a data model and an XML schema. All these operations may affect interoperability. It is acknowledged that there are situations where nevertheless the need to deliver community-specific data and services in conjunction with data and services of general relevance requires profiling of a specification. In these situations, the current paper will provide information about the consequences of particular profiling decisions with respect to interoperability and it may contain alternative solutions.

The paper does not handle profiling of specifications of dynamic behavior of systems As far as formal specifications are concerned, only the profiling of XML data model bindings is within the scope of this paper. It is intended to handle specifications of services, of runtime behavior, or of specifications given in UML in a later version of this document when sufficient experience has been gained. The paper also does not consider domain profiling, i.e., the simultaneous adaptation of several specifications and their interconnected use. Moreover, this paper does not discuss secondary profiling, i.e., the further adaptation of an application profile, though many of the issues discussed below are also relevant in this case.

1.2 Definitions

Acceptance Test Criteria	Criteria (e.g., user requirements) guiding the final testing of a system (generally in its operational environment) to enable the customer to determine whether it can be accepted.
ADL	Advance Distributed Learning Programme
AICC	Aviation Industry CBT Committee
ALIC	Advanced Learning Infrastructure Consortium
API*	Application Program Interface. An application program interface is an implementation of a *Service Access Point* (SAP) or collection of SAPs. A set of standard software interrupts, calls, functions, and data formats that can be used by an application program to access network services, devices, or operating systems.
Application Profile	A description of the use of a single technical specification to meet the needs of a particular community.
Base Specification	The specification that is modified by an application profile.
Bound Data Profile	A bounded data profile consists of a profile of both an information model and a binding for a static structure. The binding may be to XML, RDF, or some other technology. It is also possible that a Bound Data Profile will contain a single information model profile but more than one binding profile.
BSI IST/43	UK Learning Technology Standardization group.
CEN/ISSS LT Workshop	Centre for European Normalization, Information Society Standardization Service Learning Technology Workshop.
CWA	CEN Workshop Agreement
Certification*	Certification is the process undertaken to determine whether or not an implementation of an IMS specification conforms to that specification as stated by the associated conformance statement.
Conformance*	This is the statement of the properties that an implementation of a specification must possess in order to be defined as providing the functionality defined within the specification. The implementation may provide other functionality beyond the scope of the defined conformance.
Conformance Testing	Testing to evaluate the adherence or non-adherence of an implementation to a specification.
Content Packaging*	A unit of usable (and reusable) content as defined within the *IMS Content Package Specification*. An IMS Content Package consists of a logical description of the package (the *Manifest*) and the physincal resources.
Content Re-Engineering Tool	Tool to modify content resources or their logical descriptions.
Data Profile	A data profile consists only of a profile of an information model of a static structure. Sometime such a model is called a “Conceptual Data Schema”.
DOM	The Document Object Model is a platform and language-neutral interface that allow programs and scripts to dynamically access and update the content, structure and style of documents.
Domain Profile*	Customizing parts of one or more standards and/or specifications to meet the needs of a particular market or community i.e. a domain. A set of one or more base standards and/or specifications, and where applicable the identification of chosen classes, subsets, options, vocabularies and parameters of those standards/specifications necessary for accomplishing a particular function. In this context, the SCORM is a Domain Profile. In general, a Domain Profile will not consist solely of IMS specifications.

EIfEL	European Institute for e-Learning
ELIG	European e-Learning Industry Group
European SchoolNet	Membership-based consortium of the ministries of education of many of the European member-state and Eastern European countries.
Extensive Profile	An application Profile that permits all data that are permitted by the profiles base schema.
HTTP	Hyper-Text Transfer Protocol. An Internet protocol i.e. a part of the Internet Protocol Suite, which defines message format and transmission for media objects in a TCP/IP network. HTTP is typically used to transmit HTML documents between a web server and a web client e.g. a browser.
ICP	International Conformance Program
IEEE LTSC	Learning Technology Standardization Committee of the IEEE (see *IMS Abstract Framework Glossary* for a more complete definition).
IMS	IMS Global Learning Consortium
ISO/IEC JTC1/SC36*	Learning Technology Committee to Joint Technical Committee 1 (JTC 1) - The International Organization for Standardization and the International Electro technical Commission has formed a Joint Technical Committee (JTC1) that is focused on the area of Information Technology standardization. ISO/IEC JTC1/SC36 (Sub Committee 36) is intended to address standardization in the area of information technologies that support learning, education and training.
Learning Technology Specification	A number of these (by way of example) are available for download at no charge from the IMS Global Learning Consortium website at http://www.imsglobal.org Each Learning Technology Specification is generally comprised of three documents: • Information Model – covering some semantics, conceptual schema and data elements and the requirements expressed as UML use cases; • Binding Document – offering an explicit XML binding for the Information Model; • Best Practice Guide – offering examples of implementations, how to create valid extensions and general guidance on implementing tools/applications which exploit the Learning Technology Specification.
LIP	Learner Information Package
LOM	Learning Object Metadata
MIT	Massachusetts Institute of Technology
Model-Based Testing	An approach to software testing that bases common testing tasks such as test case generation and test result evaluation on a model of the application under test.
OKI*	Open Knowledge Initiative. OKI is defining a service-based architecture, consisting of service and Application Programming Interface (API) specifications, designed to support educational software, e-learning applications, and learning management systems. OKI also provides support services to its developer and architectural specification communities, though on-line forums, documentation, training, and community events. OKI is led by the Massachusetts Institute of Technology.
OMG	Object Management Group
Restrictive Profile	An application profile which permits only data which are also permitted by the profiled base specification.
ROI	Return On Investment
SCORM*	Sharable Content Object Reference Model (see *IMS Abstract Framework Glossary* for a more complete definition).

SIF*	Schools Interoperability Framework (see *IMS Abstract Framework Glossary* for a more complete definition).
SOAP*	Simple Object Access Protocol. SOAP provides the definition of an XML document which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment.
Stub	A dummy or skeletal implementation of a piece of code temporarily used to develop or test another piece of code that depends on it.
Test Suite	Software tools for testing the degree to which software or hardware conform to the requirements of a standard. Used in software development to assure quality on completion and post completion to demonstrate conformance and achieve certification for customer purposes.
Test System	The combination of test software, test documentation, and test procedures that check an implementation for conformance to a standard.
UI*	User Interface. The visual presentation and its underlying software through which a user interacts with an application.
UML	Unified Modeling Language. A language proposed by the OMG for specifying, visualizing, constructing and documenting the artifacts of a software system as well as for business modeling; it is the de-facto standard diagramming notation for object-oriented modeling.
VDEX	Vocabulary Definition Exchange
VP	Vice-President
Write Profile	An application profile describing the data which can be written by a system.
WSDL*	Web Services Description Language (see *IMS Abstract Framework Glossary* for a more complete definition).
XMI	XML Metadata Interchange. Codification to enable easy interchange of meta-data between modeling tools and repositories in distributed heterogeneous environments, for sharing object models and other meta-data over the Internet.
XML*	Extensible Mark-up Language. XML is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere.

* The entries denoted by '*' are taken from the IMS Abstract Framework Glossary [IAF, 03].

1.3 References

[GWS, 05a]	*General Web Services Base Profiles Public Draft v1.0,* C.Schroeder, S.Raju and C.Smythe, IMS/GLC, January 2005.
[GWS, 05b]	*IMS General Web Services UML to XML Binding Auto-generation Public Draft v1.0,* C.Schroeder, S.Raju and C.Smythe, IMS/GLC, January 2005.
[IAF, 03]	*IMS Abstract Framework: Glossary v1.0,* C.Smythe, IMS/GLC, July 2003.
[VDEX, 04]	*IMS Vocabulary Definition Exchange v1.0,* A.Cooper, IMS/GLC, February 2004.

2. Developing an Application Profile

The development of an application profile begins with the establishment of a **community** of shared interests that can identify the need for an application profile and support its development and maintenance. The community needs to define itself as an entity and create an agreement between its constituents.

This community, once established, looks at the **requirements** of its members, both in terms of existing systems and processes and its future needs. It also needs to take account of the specifications (and existing application profiles) available to it.

Once requirements have been gathered, there needs to be a process of **analysis and synthesis**, where the community produces models of its domain and identifies gaps in available specifications. It may also develop a reference architecture model to place the specifications within a system design context.

At this point, the community has the information with which to make the **decision** whether or not it is appropriate to create a profile. It also needs to consider other practical issues, such as the estimated costs and predicted benefits of creating the profile, and also the opportunities, risks, and strategies associated with the take up of a profile by the implementation community.

If the community decides it is appropriate to create a profile, it can then make a **choice** of specifications and/or existing profiles, and begin **development** of the application profile. During the development process it may be necessary to revisit the requirements and analysis and synthesis phases, to qualify, re-examine, and review their outputs in the light of decision points within the development process.

Finally, an initial application profile is created. Profiles need to be published, ideally in a registry of application profiles, so that other organizations wishing to create profiles can locate and reuse existing work. Where this is not applicable or practical, then the profile should at least be published formally with an official identifier.

This is not the end of the overall process, as this profile still needs to be maintained, and the community must support its implementation community.

This whole process is encapsulated in the diagram in Figure 2.1.

2.1 Types of Application Profile

For the purposes of these guidelines, we define two types of application profile.

A **Data Profile** consists only of a profile of an information model of a static structure. Sometimes such a model is called a “Conceptual Data Schema”.

A **Bound Data Profile** consists of a profile of both an information model and a binding for a static structure. The binding may be to XML, RDF, or some other technology. It is also possible that a Bound Data Profile will contain a single information model profile but more than one binding profile. In the current paper only profiling of a single XML binding is under consideration.

Section 3 of this document describes how to create both a Data Profile and a Bound Data Profile.

Each information model describes a set of documents that are compliant with this model. When two applications interoperate, they exchange documents which have to be compliant with their respective information models. In the elementary case of a single information exchange one of the systems acts as a *sender* (or writer) of the document and the other acts as a *receiver* (reader) of the document. In the current context it does not matter by which methods the document is transmitted – it may be by file transfer with or without human intervention or as payload of a service message exchange.

We assume, subsequently, that the set of documents which an application can read or write is described by a bound or unbound data profile. We note that a system may use different data profiles for reading and writing. For example, it may read a larger variety of data than what it writes itself. Therefore, we shall distinguish between the *read profile* and the *write profile* of an application. In fact, an application may use several read and write profiles in order to interoperate with a variety of other applications but we shall abstract from this since it does not affect the following considerations, i.e., we consider for each application *A* at most one read profile *A.ReadProfile* or write profile *A.WriteProfile*.

For each data profile *DP* let *DP.Data* denote the set of data that are compliant with the profile *DP.* In order to interoperate, each document that can be produced by the sender must be digestible by the receiver. With this notation for guaranteeing interoperability when sending data from a sender *Sender* to a receiver *Receiver* it is necessary that *Sender.WriteProfile.Data* is a subset of *Receiver.ReadProfile.Data*:

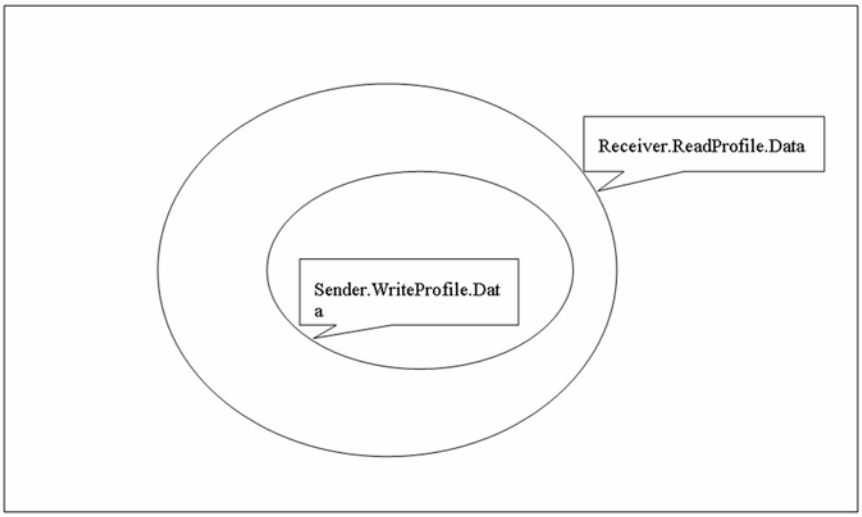


Figure 2.1 Data sets in an interoperable setting.

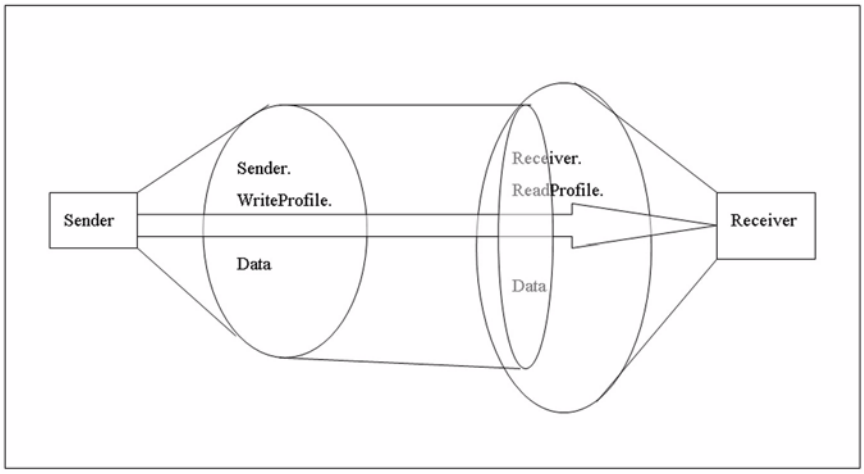


Figure 2.2 Data communication in an interoperable setting.

Different data profiles describe different sets of data. A data profile DP_1 is said to be **restrictive** with respect to another data model DP_2 if all data which are compliant with DP_1 are also compliant with respect to DP_2 , i.e., DP_1***.Data*** is a subset of DP_2*.Data*. In this case we also call DP_2 **extensive** with respect to DP_1. In the interoperable case described above, the profile *Sender.WriteProfile* is restrictive with respect to the profile *Receiver.ReadProfile* and *Receiver.ReadPofile* is extensive with respect to *Sender.WriteProfile*. If two data profiles are not restrictive or extensive with respect to each other we call them **incompatible**.

Now let us consider what may happen when data profiles are derived from a base specification. Each specification has its data model which is considered as a data profile of itself with no changes made.

First of all, we observe that using only data profiles that restrict the base specification data model does not ensure interoperability:

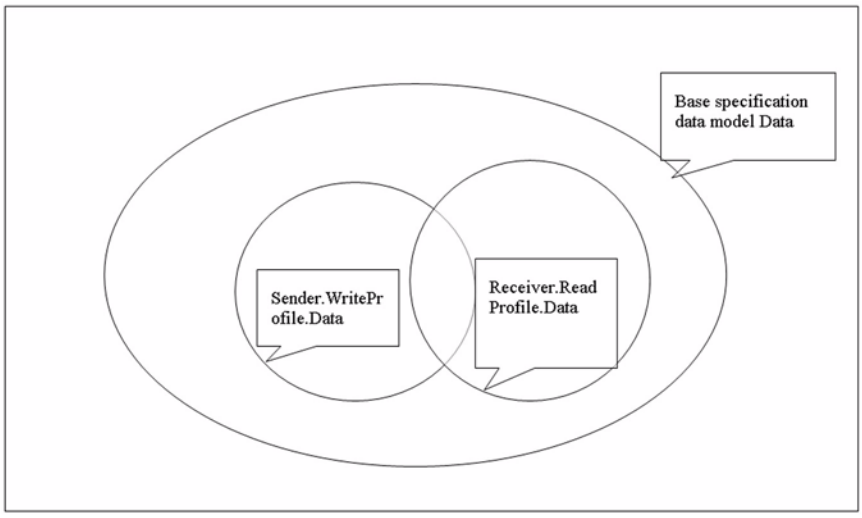


Figure 2.3 Two restrictive data profiles of a base specification data model resulting in non-interoperable systems.

However if the read profile of the receiver is extensive with respect to the base specification data model but the write profile still restrictive, data can be sent and received successfully:

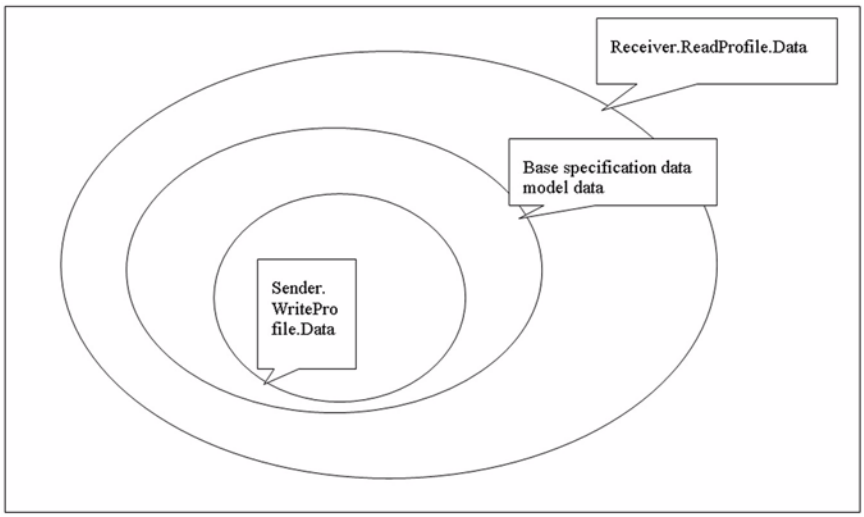


Figure 2.4 An extensive and a restrictive profile of a base specification data model resulting in interoperable applications.

This is the recommended practice, i.e., applications should be capable of reading at least all data which are compliant with the base specification data model but they should write, at most, a subset of the data permitted by the base specification.

In this case, each application which is write-compliant with respect to IM_1 is also write-compliant with respect to IM_2 and each application which is read-compliant with respect to IM_2 is also read compliant with respect to IM_1. Note that an application which is compliant with one of these information models will not necessarily be compliant with the other model.

An application is said to be **write compliant** with a data profile if it produces only data that are compliant with this profile, i.e., its write profile is restrictive with respect to the data profile. The application is called **read compliant** with a data profile if it can read all data which are compliant with this profile, i.e., its read profile is extensive with respect to the data profile. Sender and receiver will only interoperate if there is a data profile such that the sender is write compliant and the receiver is read compliant with this data profile. We call an application **compliant** with a data profile if it is read compliant as well as write compliant with respect to this data profile.

2.2 Bound Data Profiles and Categories of Modifications

A bound data profile has a data model and a binding for a static data structure. Bound data models are usually derived from specifications which have an information model as well as a binding which defines the structure of the compliant data. In this document we confine ourselves to the case where the binding is given in the base specification by an XML schema, referred to subsequently as the **base schema**. An application profile must clearly state which base specification it is based upon, including all applicable version references.

Depending on the particular base specification and the intended use of the profile, the profile created may need to be based either on the information model or the binding supplied for the base specification. For example, when creating a profile of the IEEE LOM standard for use internally by a system, it may be necessary to only profile the information model of LOM, and create a new binding. In other cases it may be more advisable to create a profile of both the information model and the binding, especially where the profile is being used for interoperability between parties.

Where base specifications contain variants, it should be noted in the profile which variant has been used. For example, the IMS Learning Design specification has three variants: “Level A”, “Level B” and “Level C”. When creating an application profile of Learning Design, it should clearly be stated which level the profile is based upon.

A profile is derived from a base specification by applying a set of modifications. Subsequently, this paper will discuss a number of possible modifications in detail. For now we observe that we can distinguish three categories of modifications, depending on their effect and on the way they can be encoded in the application profile.

a) **XML Schema modifications** concern the structure of XML documents and can be expressed in a modified XML schema. Making an optional element mandatory is a schema modification;

b) **XML non-Schema modifications** concern the structure of XML documents but cannot be expressed in a modified XML schema. All modifications that depend on the concrete situation in an XML instance document are of this category;

c) **Additional constraints**. These modifications do not concern the structure of XML documents. For example the request that a certain file must exist at a certain place in a data package is of this category.

More examples will be given below. The partition of the modifications into these categories has consequences for the possibilities to ensure compliance and thus to support interoperability.

For XML schema modifications, which can be manifested in a derived XML schema, there is a number of validating XML parsers available which can check automatically that a given XML document has a structure as requested by modifications of this category. It is recommended that application profiles should use XML schema modifications if this is possible. The analysis of a number of application profiles suggests that most modifications can be described as XML schema modifications.

XML non-schema modifications require the development of particular test tools. XML technology supports such developments well. We mention XSLT and Schematron here. The analysis of a number of application profiles suggests that only very special cases require XML non-schema modifications.

Additional constraints often arise from the modification of conditions which are stated in the information model of the base specification but not in the base schema. Checking the conditions set up by these modifications requires ad hoc software development. The analysis of a number of application profiles suggests that such modifications occur frequently but with few variations only.

2.3 The Smallest Permitted Length and Related Restrictions

For some specifications, the information model requires that for specific elements of string type compliant systems should permit values of a specified length *N* at least. This is not a condition on the structure of compliant data but a condition on compliant systems. In fact, this minimum permitted length restriction defines two different conditions for a data profile being extensive or restrictive, depending on whether it is used as a read or write profile.

Restrictive read profiles must have a maximum length restriction for these elements which is either *unbounded* or has a value greater than or equal to *N*. However, for a restrictive write profile the maximum length restriction for these elements should be less than or equal to *N* in order to make sure that systems compliant with the base specification use the written data correctly.

Similar considerations apply to the smallest permitted number of elements in a list when specified in the base specification.

2.4 Extending Specifications in Application Profiles

A major motivation for profiling a specification is the lack of particular data fields which are needed for keeping information which is relevant for a specific community.

Uncontrolled extension of an information model, e.g., by adding at arbitrary locations items to the information model or including structures from other base specifications, is strongly discouraged as this will almost certainly break compliance with the base specification. Such extensions greatly reduce the chance of implementations against the base specification from being able to support the profile. Many base specifications, however, define permitted means of extension, and wherever possible these should be used, as this reduces the risk of people adopting your profile creating instances that are invalid against the base specification.

Permitted means of extension have in most cases the form of wildcards, i.e., these are specific places in the information model where anything can be inserted and the document is still considered to be compliant with the base specification. Therefore any profile being more specific, i.e., specifying what is allowed at these places, will lead to a restrictive profile, even though it introduces new elements. The recommended behavior of systems is that they should tolerate all possible uses of the extension points in their read profile but they may confine to a particular usage of extensions in the data they write, i.e., they may use a restrictive write profile.

It is also possible to create a profile which implicitly extends a base specification - for example, by adding items to a controlled vocabulary or by extending the permitted length of a text item or by casting a restricted type into a less restricted one (e.g., from an Integer to a String, or from an Integer to a Real). In contrast with the use of extension points just described, these are indeed extensive profiling operations. Such operations are damaging with respect to read compliance with the base specification and hence are strongly discouraged for write profiles.

By way of comparison, operations which further constrain the permitted values are generally acceptable for write profiles - such as imposing a vocabulary on a text string or further restricting the permitted length of a text item or casting a restricted type into a more restricted one (e.g., from a Real to an Integer) or applying a restricted range of values of a particular type (e.g., applying an integer range such as [1, .., 10] to an Integer value). In all of these cases, an instance of the information model would still validate against the base specification and hence could be read by tools that implement the base specification.

2.5 Excursion: Translations, Vocabularies, Localizations, and Mappings

Sometimes it may seem desirable to use data structures or names which differ from those described in the specification. Inevitably, systems which implement the base specification will not be able to read or write data which use these new items. Therefore, such modifications should not be made lightly.

When considering an application profile for a community using a language other than US English, it is generally preferable to do a language translation of the information model of the base specification, thereby creating a new information model to use as the control document against any subsequent application profile. The translated information model should introduce no changes to the structure of the information model, the mandatory/optional

status of the elements, nor should it introduce new elements or eliminate existing ones. Translations should be confined to translations of the documentation and to translations of controlled vocabularies. It is recommended to add parts to the documentation which describe the meaning of the translated phrases, provide recommendations for their usage. Translations should provide mappings which relate translated phrases with their US English versions.

Typing and translations of controlled vocabularies should be equivalent to those in the original information model. If a straightforward translation is possible it may be worth investigating whether it is possible to use the base specification and to extend instead the tools which process the data by some wrapper which does the required translations when data are read or written.

Controlled vocabularies occur in specifications either in line (as part of the specification in the form of an enumeration of admitted phrases) or referenced (as a reference to an external vocabulary). In the first case, translations must replace the phrases in the specification with their translations. Systems which are read compliant with the base specification will not be able to process data which are built according to the translated specification. However, the provided translation mappings may be used to implement a conversion of data to their US English variant and conversely.

If controlled vocabularies are referenced only in the base specification, any language-specific community may agree on the usage of a specific vocabulary. Specifying such a vocabulary can be done in a restrictive profile. Therefore systems which implement the full (i.e. vocabulary-independent) base specification in their read profile will be able to process data using the specific vocabulary. It is recommended to provide mappings between phrases in different vocabularies in order to facilitate the reuse of data. Note that the use of referenced vocabularies has – in comparison with using inline vocabularies – the advantage that systems can be adapted to different languages without requiring translations of their input data. The IMS VDEX specification provides more information on the use of referenced vocabularies.

Having derived an equivalent information model in the target language, any application profile should strive for compliance with this language-specific information model in order that it can exploit language-specific implementations against the localized information model.

By their nature, translations, localizations, and mappings differ from the profiling modifications since they not only modify a specification but give addition information that can be used to convert data so that they become compliant with either the base specification or the application profile.

This conversion need not be determined uniquely if the mapping defines only a relation between values but not a function. For example, assume that in the base specification a scale of grades A,B,C,D is used where, in a certain domain, grade A entitles for enrollment in a course and C and D are considered as insufficient for the continuation of a study. In another community, grades 1,2,3,4 are used and grades 1 and 2 entitle enrollment while grade 4 is insufficient for the continuation of the study. Then it makes sense to map A to 1 and 2, B to 3 and C and D to 4. Note that a combination of translation and re-translation will usually not result in the original data if the mapping is not one-to-one. In order to care for situations where some values cannot be mapped at all, translated profiles should allow an indeterminate value like *undefined*.

Except for the most straightforward cases, best practice documents for application profiles should explain the intended usage of the translated vocabulary and the rationale of the mapping provided (if any). Then a correct usage of the vocabulary will be crucial for the semantic interoperability of systems implementing the application profile.

Adding to or removing items from a vocabulary needs to be done with care. If the vocabulary is defined inline in the specification as a list of permitted items, adding or removing items is an extensive respectively restrictive modification. But the situation is more complicated if the vocabulary is given in the base specification by reference only. These referenced vocabularies cannot be changed by a modification of the base specification since they are not part of it. Instead, a new vocabulary, identified by a new URI, should be provided and referenced by the application profile. In order to be restrictive, the new vocabulary must not add new items and must use only identifiers which have been used in the vocabulary referenced by the base schema. Also, the type of the vocabulary should not be changed.

In order to avoid any danger of confusion it is recommended to reference a new vocabulary whenever any vocabulary change is made and to provide vocabulary mappings to support the development of conversion utilities.

3. Creating Application Profiles of Data Specifications

3.1 Defining an Information Model

For each specification utilized that involves a data model of some description, you need to define precisely how that model needs to be adapted to meet your requirements.

There are a number of operations that can be used for adapting an information model, each with implications for interoperability within your community or application, and for retaining compatibility with the wider world.

For each operation mentioned below, guidelines are given on how to perform the operation, what the benefits and implications are for your profile, and how the intent of the operation can be expressed. Ways to formally encode the intended modifications in XML will be explained in the next section.

Although not many people understand formal notation, the precision of meaning provided by a formal expression language is highly beneficial when developing test harnesses, where ambiguities in English and other natural languages can prove problematic. It can also be useful for automatically generating bindings for various technologies, such as XML Schema, Java, or SQL.

Because not everyone understands XML, it is important to always provide a less formal description of each profiling statement using natural language. The XML binding described in the next section provides fields for the application profile and for the individual modifications to add explanations or a documentation of the rationale in various languages. It is recommended to fill these fields and to generate a human readable documentation of the application profile from these annotations.

The modifications identified below are split into the categories explained in the last section:

- Restrictive Modifications – instances of the profile can be used by tools and products that correctly implement the base schema in their read profile. The base schema can further be used to export data outside the immediate target community. It is recommended to use only these modifications in write profiles of applications;
- Extensive Modifications – instances of the profile schema may well no longer validate against the base schema and hence tools and products may require some customization to use instances to the resulting profile. Data will also need additional handling to achieve exchange with external communities. These operations are strongly discouraged for use by write profiles and should be adopted only if the communities need cannot be addressed by any other means. They are, however, of no harm in read profiles of applications. Applications using extensive modifications in their read profile may well use all instances of the base schema;
- Incompatible Modifications – these operations break interoperability with the base specification completely and should not be undertaken in any circumstances. Note that also the simultaneous usage of restrictive and extensive modifications will result in an incompatible profile.

3.1.1 Restrictive Modifications

3.1.1.1 Modifying the Size of a Text Item

Method: Specify a size constraint for an item in the specification, either:

- Lowering an existing maximum text length constraint; or
- Increasing an existing minimum text length constraint; or
- Adding a constraint to an unconstrained String-type item.

Benefits: Constraining text lengths prevents problems with relational databases storing data and allows relational schema to be developed with less ambiguity. Specifying minimum text lengths may also be desirable in some circumstances.

Implications: By constraining allowable sizes you run the risk of values of items created to the original specification containing illegal String lengths for your application profile, resulting in either truncation or errors when read into a system conforming to your profile.

Example:

Item	Type	Original Description	Application Profile Guidelines
Title	String	The name of the resource.	The content of this item must not be longer than 255 characters.
short	String	Short description of the resource, must be no greater than 1024 characters.	May be up to 512 characters.
password	String	The password for this user. May be up to 32 characters long.	The password must be a minimum of 8 characters.

3.1.1.2 Modifying an Item by Prescribing the Use of a Controlled Vocabulary

Method: For an item in the specification:

- If no vocabulary is defined in the specification but the usage of a vocabulary is foreseen specify a particular vocabulary for your domain;
- If a vocabulary is defined in the specification a more restricted vocabulary may be used;
- If an item is defined in the specification only to have a string value a list of admitted values may be prescribed.

Benefits: Maintain the benefits offered by the use of vocabularies, but with values that are relevant to your community.

Implications: Restricting a vocabulary by removing items risks non-interoperability with existing applications and data that conforms to the specification. However, it will not prevent data created with your profile to conform to the original specification. Using references into external vocabularies instead of specifying a fixed list of phrases makes it easier to adapt existing systems. It is also to be preferred if a multilingual community is to be served. The IMS VDEX [VDEX, 04] specification provides more information on the recommended use of controlled vocabularies.

In any case, items from a controlled vocabulary should only be used if their meaning is sufficiently clear in the community addressed. They should not be used with a different meaning. Abusing an item from a controlled vocabulary to describe “something similar” may result in semantic non-interoperability, misinterpretation, and misbehavior which cannot be detected by automated tests and which may therefore remain uncovered for a long time.

Example:

Item	Type	Original Description	Application Profile Guidelines
GroupType	String	The type of group; must be one of “Course”, “Module”, “Section”.	The type of group; must be one of “Course”, “Module”.
gender	String	The gender of the participant; must be one of “Male”, “Female”, “Other”, “Unknown”, “Unspecified”.	The gender of the participant; must be one of “Male”, “Female”, “Other”.

3.1.1.3 Modifying an Item’s Data Type

Method: Specify a new, more restricted data type for an item in the specification, such as stating that an item of type Real in the specification should instead be an Integer in your profile.

Benefits: In some cases, greater precision may be necessary for specifying an item than is possible using the default type given in the specification, for example if you require an item to only contain whole numbers and no fractions, but the default type is a Real or String. In such cases, recasting the type may be more efficient than applying a range of complex restrictions to achieve the same effect.

Implications: In cases where a type is being changed in a way which restricts the range of values, such as from a Real to an Integer, this will have the effect that instances conforming to the profile can still be cast to conform to the specification. Note that it is only important that the modification restricts the lexical space of the item's original data space. Their value spaces, i.e. their interpretation according to the data type may well be incompatible without interfering with interoperability.

Example:

Item	Original Type	Original Description	Profile Type	Application Profile Guidelines
grade	String	The student's grade for the work (e.g., "A", "55%" etc.).	Integer	The student's grade for the work as an integer representing a whole-number percentile.

3.1.1.4 Refining an Item by Clarifying its Meaning

Method: Provide additional narrative for an item explaining its specific interpretation for the application profile. This also can include providing localization of descriptions and providing assistive text.

Benefits: By augmenting usage descriptions provided in the specification it makes the use of the item clearer and less ambiguous and can reduce errors in interpreting the specification for the local context.

Implications: The refinement you provide may actually change the meaning of the field if it strays too far from the given description. This could cause existing tools to use the data incorrectly. If a refinement does this then it becomes a redefinition of the item, and it may be worth considering using an *extension* instead, as redefining existing items is strongly discouraged.

Example:

Item	Type	Original Description	Application Profile Guidelines
Course	String	The name of the Course the student is attending.	This MUST be the name of the course offering the student is enrolled upon, exactly as it appears within the student records system, and NOT the course name or identifier used in the prospectus.

3.1.1.5 Changing the Scale or Precision of a Numeric Item

Method: For an item in the specification that is a Real number, alter the allowed scale and precision to suit your profile; either alter the precision alone, or as a factor of scale.

Benefits: Restricting the precision of a Real number to an agreed number of places reduces the likelihood of rounding errors. Specifying both scale and precision for a Real number also makes it possible to directly relate the possible values of instances to fit within database structures defined in terms of both scale and precision.

Implications: Instances conforming to the profile may be subject to rounding and possible error if used in processes that conform to the original specification.

Example:

Item	Type	Original Description	Application Profile Guidelines
gradeAverage	Real	The average of the percentile grades.	Must be given with precision to two decimal places only (e.g., "83.25").
length	Real	The length of the item in millimeters.	Values must be provided to a scale of 6, precision 2 (e.g., "1234.56").

3.1.1.6 Restricting the Range of a Numeric Item

Method: For an item in the specification that is an Integer or Real number, restrict the range of valid number, either by creating a new range constraint or by altering an existing range constraint.

Benefits: Restricting the range of a number ensures it falls within values which are meaningful in context. For example, if numbered grades in your educational system range from 25-120, then requiring grade values to fall within this range helps to ensure only valid grades are entered.

Implications: While all instances of this item within your profile will also conform to the specification, the reverse is not true.

Example:

Item	Type	Original Description	Application Profile Guidelines
grade	Integer	The student's grade.	Must be between 25 and 120, inclusive.
age	Integer	The age of the Person.	Must not be negative, and may not exceed 125.

3.1.1.7 Mandating an Optional Item

Method: For an item in the specification that is indicated to be optional, require that for your application profile this item is mandatory.

Benefits: Allowing optional items hampers interoperability, as different systems may end up choosing different sets of optional items to support. By mandating an item you can guarantee that systems will support an item, and that data will always contain that item, reducing the number of possible data combinations that need to be supported.

Implications: Tools that are based on the profile may not be able to read instances constructed to the original specification as they may omit the item that the profile has mandated. In case of attributes you should consider to specify a default value in the application profile so that systems compliant with your application profile may still read instances that omit the mandated attribute.

Example:

Item	Type	Original Description	Application Profile Guidelines
Title	String	Optional title for the resource.	All Resources MUST contain a valid title.

3.1.1.8 Requiring an Optional Item to be Mandatory Under Certain Conditions

Method: Require that an item defined as optional in the specification must be used depending on the values of other items.

Benefits: Often the value of one item affects another, so it is useful to constrain an application profile to take account of these effects.

Implications: Tools that are based on the profile may not be able to work with instances constructed to the original specification as they may omit the item that the profile has mandated.

Note that this modification is fundamentally different from the modifications described previously since it is dependent on the situation in the particular instance document which is not known at the time the application profile is written. These conditional modifications cannot be captured in a modified XML schema. However, they can be expressed in XML in the XML binding of an application profile as described in the next section. Other tools like Schematron rules can be used as well.

Example:

Item	Type	Original Description	Application Profile Guidelines
gradeType	String	The type of grade used.	Must be either “percentage” or “narrative”.
numberGrade	Int	Actual grade as a number (optional).	If the gradeType is “percentage” then this item must contain a number, otherwise it must be left empty.
Comment	String	Comment on the grade given (optional).	If the gradeType is “narrative” then this item should contain a description of the assignment result, otherwise it must be left empty.

3.1.1.9 Excluding an Item

Method: Require that an optional item in the specification should not be used for your profile. You can require it to be not present at all or require that it must always be set to a null or empty value.

Benefits: If an item has no meaning in the context of your profile, then excluding it simplifies the overall model and prevents the item being used inappropriately.

Implications: If the item is mandatory in the specification, than interoperability is no longer possible between profile and specification-conformant systems – the modification is incompatible.

If the item is optional in the specification, then instances conforming to the profile are still conformant to the specification; however, tools that are based on the profile may not be able to work with instances constructed to the original specification as they may include the item that the profile has excluded. Instead of excluding a required item, it is recommended to require the usage of a default value which may be empty.

When a mandatory item is excluded, the profile is incompatible.

Example:

Item	Type	Multiplicity	Original Description	Application Profile Multiplicity	Application Profile Guidelines
price	Real	Optional [0..1]	Optional price for the resource.	Excluded [0]	All Resources MUST NOT contain a Price.

3.1.1.10 Restricting an Item by Mandating a Controlled Vocabulary

Method: Mandate for an item a specific controlled vocabulary of acceptable values within the application profile.

Benefits: Restricting the number of possible values improves interoperability by reducing the number of interpretations required to be supported for the value of the item.

Implications: Existing data or systems not specific to your community can use values other than the ones specified by your vocabulary, so this will affect interoperability; however, tools and data constructed for your profile will still meet the requirements of the broader community, provided that you are providing a restricted vocabulary for an existing open item, such as a String and not changing a vocabulary (see below).

Example:

Item	Type	Original Description	Application Profile Guidelines
groupType	String	The type of the Group.	The content of this item must be one of the following values: “Course”: A course offering “Module”: A module within a course “Section”: A section within a module
Country	String	Name or identifier for country.	The content of this item must be a value defined by the ISO 3166 Country Codes list, using two-letter codes only.

3.1.1.11 Restricting a Text Item by Specifying a Pattern

Method: Specify that a text item must conform to a pattern, such as defined using a regular expression.

Benefits: This ensures that all such entries are in a format that can be readily used without being re-interpreted by conformant systems.

Implications: While all instances of this item within your profile will also conform to the specification, the reverse is not true.

Example:

Item	Type	Original Description	Application Profile Guidelines
postalCode	String	Code used to identify postal destination.	This should be a UK PostCode in the format ‘AA9[9]+” “+9AA’ For example: LL33 0AT The space between the two parts of the postcode must be included. All letters must be uppercase.

3.1.1.12 Requiring a Set of Items to Appear in a Particular Order

Method: For a set of items, specify the order that they must appear in within instances.

Benefits: By specifying an order for items, it may be possible to make implementation easier, or better optimized for performance.

Implications: In some binding technologies it is not always a simple matter to ensure the ordering of items, or that ordering will be preserved during the import and export process, making conformance more difficult. Notably the XML binding described in the next section does not permit modifying a choice of elements to become a sequence. In some cases the intended effect may be achieved nevertheless using conditional modifications.

Example: Where a base specification defines “groups” and “members” with associations between them, requiring a “group” to occur before any of its members means that the structure can be parsed line-by-line, rather than having to hold “members” in memory until their associated “group” is reached. A condition in the profile may test whether each “member” element is preceded by a “group” element in an instance document and may impose a type conflict otherwise.

3.1.1.13 Requiring an Item to Have a Particular Default Value

Method: Specify that, where an item in an instance does not have a value, that a particular value defined within the profile should be used.

Benefits: You may need to ensure that an item always holds a valid value, even if it is the default, so that systems operate in a known fashion when encountering blank items.

Implications: If the item already has a default value in the specification, and you are requiring a different default, then this will have interoperability implications for any instances that do not supply the value, as they will be treated differently by systems conforming to the profile than from systems conforming to the specification.

Example:

Item	Type	Original Description	Application Profile Guidelines
TypeValue	String	Type for this Group.	Where no type is given, the default value of "Undefined" must be assumed.

3.1.1.14 Requiring an Item to be Expressed in a Particular Human Language

Method: For an item in the base specification, require that the content of that item be expressed in a particular human language.

Benefits: You may want to require that human-readable information held in an instance is available in languages specified by your community.

Implications: There are no specific interoperability implications, although instances conforming to the specification may require translation before they would interoperate properly with profile-conformant systems.

Example:

Item	Type	Original Description	Application Profile Guidelines
title	String	Title of the resource.	The title must be expressed in International English, with the original language and title in parentheses where translated, e.g.: The Big Cheese (fr: Le Grand Fromage).
description.short	LangString [0..*]	Short description of the resource in multiple language equivalents.	A description must be provided in both English and French as a minimum; other languages are optional.

3.1.1.15 Requiring Everything within a Profile to be in a Particular Human Language

Method: Using some means of identifying exceptions, require all text items within the profile to have an expression in the specified human language.

Benefits: You may want to express language constraints globally.

Implications: There are no specific interoperability implications, although instances conforming to the specification may require translation before they would interoperate properly with profile-conformant systems.

Example: This is typically expressed using a statement such as "The default language for all items is English unless stated otherwise".

3.1.1.16 Referencing or Including Items or Structures from Other Data Models

Method: Within the information model for the base specification, include (or reference) parts of another information model; for example, from another specification. Many specifications contain extension points where you can easily add additional items from other specifications or from your own information model. The simultaneous modification of several specifications is called domain profiling. It goes beyond the scope of this document.

Benefits: In some cases, information you need to represent is not present in the specification, but is part of another specification. By including this information you can meet the requirements for your community while also adhering to standards and specifications rather than constructing your own extensions. A typical example is the addition of meta-data structures.

Implications: The same warnings apply to inclusion of other models as to the construction and placement of extensions – if the base specification does not provide a mechanism for including additional data types, or you decide to incorporate the structures in a different way than is advised by the base specifications, then this will result in interoperability problems. When extension points are used, instance documents will have to include the location of the definitions of the additional elements. The recommended behavior for applications is to store extensions which they find in imported instance documents at extension points. If the same document is re-exported, the extension information should be preserved.

Example: In IMS Content Packaging, there are meta-data items which typically are used as containers for structures included from either the IMS Meta-Data or Dublin Core meta-data specifications.

3.1.1.17 Restricting the Levels of Nesting of a Recursive Item

Method: For an item that has a recursive structure (that is, it contains instances of itself, like a self-join in a database), restrict the levels of nesting allowed. This can be expressed in an application profile by making the usage of the element in question dependent on a condition which tests that the level of nesting of the item in question in instance documents does not exceed the specified limit.

Benefits: At its most extreme, this allows the profile to insist on a certain degree of structural homogeneity amongst instances, such as requiring instances to be constructed in an ontology-like fashion (as flat lists of items joined by explicitly typed relationships) rather than in a hierarchic fashion with implied relationships. In other cases, restricting nesting levels provides developers with a fixed boundary against which to code, and may reduce the overall memory footprint and threading requirement for applications supporting the profile.

Implications: Restricting nesting levels will result in instances conforming to the profile also conforming to the specification, but not necessarily the reverse.

Example:

Item	Type	Multiplicity	Original Description	Application Profile Multiplicity	Application Profile Guidelines
goal	Goal	Optional [0..*]	A sub-goal of the parent goal.	Excluded [0]	Goals may not contain sub-goals; instead, all goals must be related explicitly using the Relationship structure.
resource	Resource	Optional [0..*]	A sub-resource of the parent resource.	Optional [0..*]	A maximum of three levels of Resource are permitted.

3.1.1.18 Restricting the Target of an Association by Type

Method: For an item that can contain a reference to another item, such as a pointer or index reference, restrict the type of item that can be the target of the reference. This is an additional constraint modification.

Benefits: Sometimes a specification will support a wide range of associations, only some of which are applicable to your community or usage context. By restricting the types of associations you reduce the range of possible behaviors that systems will have to interpret, making implementation easier.

Implications: Restricting associations means that instances that conform to your profile will also conform to the base specification in this regard, however the reverse is not necessarily true, and valid instances of the base specification may not conform to your profile or be usable in systems that support it.

Example: In IMS Content Packaging, an Item contains a reference that can be made to point to either a Resource or a Manifest (a "submanifest" in IMS CP parlance). An application profile may be created that prohibits the use of submanifests, and so restrict an Item reference to only point to a Resource.

3.1.1.19 Restricting the Structure of a Data Package

Method: If data are exchanged not as a single file but as a set of files, a profile may impose additional constraints on the structure, content, or packaging of this data package.

Benefits: Being stricter with the structure, data formats or packaging formats of data packages simplifies the implementation of software that can read these packages. Also, data packages that conform to the profile can be read by systems which fully implement the base specification.

Implications: Valid instances of the base specification may not conform to your profile or be usable in systems that support it. However, it may be possible to provide filtering software which converts data packages from other formats into the format requested by the application profile. The application profile should document how the information found in other base specification conformant data packages should be encoded in the more restricted way.

Example: A profile may request an IMS Content Package to be delivered as a zip file and that each data file in the package must be referenced by a *resource* item.

3.1.2 Extensive Modifications

Extensive modifications extend the set of admitted data. Consequently, applications which conform to the base specification in their read profile may not be able to read data which conform to the application profile. On the other hand, the use of extensive modifications is not critical in read profiles of applications. Note that the use of extension points of the base specification (mild extension) is a restrictive modification since it restricts the arbitrariness of items which can be used at these points. The introduction of new elements at other points (wild extension) will be in most cases neither a restrictive nor an extensive but an incompatible modification. If there is no extension point available where needed, it should be explored whether instead an available extension point can be used. If even this is not possible, new items should be added at the end of existing item sequences, hoping that future versions of the base specification may add the required extension point later.

3.1.2.1 Making a Mandatory Item Optional

Method: For an item in the specification that is defined as mandatory, allow anyone implementing your profile to treat that item as optional.

Benefits: If an item mandated by the specification has no meaning (or an ambiguous meaning) in your application context, then it may be beneficial to allow it to be optional.

Implications: Allowing mandated items to be optional means that instances conforming to your profile are unlikely to be interoperable with systems and instances that conform to the original specification, although interoperability in the other direction (consuming instances or interoperating with systems conforming to the specification) is not likely to be affected. It is worth considering whether the specification you have chosen is the one appropriate to your needs if you have to utilize this strategy, as it is generally only those items that are critical to its function that are mandated in a specification.

Example:

Item	Type	Multiplicity	Original Description	Application Profile Multiplicity	Application Profile Guidelines
title	String	Mandatory [1]	Title of the resource.	Optional [0..1]	Resources may contain a title.

3.1.3 Incompatible Modifications

3.1.3.1 Altering the Relative Location of an Existing Item

Method: For an item in the specification, define in your profile a new location within the model for this item.

Benefits: There is no real benefit to this; perhaps all except one attribute of a larger structure is being excluded and rather than have a sparse structure you have decided to aggregate the remaining item into another part of the data model for ease of readability.

Implications: Altering the data model in this fashion will almost certainly break interoperability between profile and base specification.

Example: There are no obvious examples of this strategy.

3.1.3.2 Creating a New Element that Mimics the Semantic Intent of an Existing Element

Method: Create an item in the profile for something that already exists in the specification.

Benefits: There is no real benefit to this – possibly the intent may be actually to modify the multiplicity or length of an existing item, or otherwise avoid some of its restrictions, but for some reason this is not being done explicitly.

Implications: Having multiple items with the same meaning, but in different locations with differing names or other syntax, is problematic in terms of interpreting what is correct conformant behavior. If the item in the specification is unsuitable, you should consider profiling that item rather than duplicating it.

Example: An example may be to add an additional “ExtTitle” item, with a greater allowed length than the “Title” item defined in the specification.

3.1.3.3 Changing the Meaning of an Existing Item

Method: Change the meaning of an item in the specification to suit your profile.

Benefits: The intention of this action is to effectively extend the specification without using an extension.

Implications: Changing the meaning of specification items – such as using an Address line to contain a cost code – can cause all manner of problems for interoperability, as while the syntax of both profile and base specification remain the same in this regard, the expected behaviors of conformant systems now differ. You should in all circumstances define an extension if existing items in the specification are unsuitable.

Example: A typical example would be to use an “unused” item to contain information that has no other logical location, such as detailed parts of an address structure to house proprietary coding information.

3.2 Creating a Binding for the Information Model

Profiling of a base schema is restricted to the following modifications which will be explained in detail in subsequent parts.

- Changing the cardinality of elements in the base schema, i.e., changing the *maxOccurs* and *minOccurs* attributes. This includes making elements required, optional or prohibited.
- Making attributes required, optional, or prohibited or supplying a default value.

- Replacing simple value types of attributes or elements by newly defined types or replacing them by fixed values.
- Adding elements or attributes at locations where the base schema provides appropriate wildcards.

For defining new types a specific set of operations will be admitted. Other modifications of the base schema are discouraged. The base schema modifications can be either unconditional, thus leading to the definition of a new schema, or they can depend on conditions which are to be evaluated when a concrete instance document is checked for compliance.

3.2.1 Global Structure of an Application Profile

An application profile is encoded in an XML document with a root element <schema_mod>. This element contains modifications in <modifications> element and definitions in a <definitions> element. The <schema_mod> element defines in its *baseSchema* attribute the base schema.

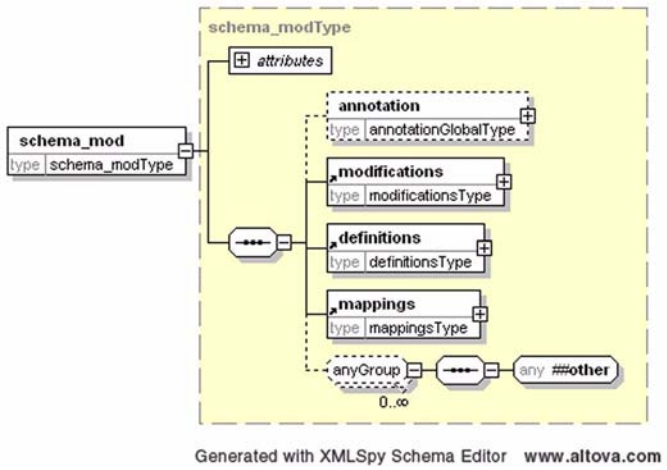
Example:

```
<?xml version="1.0"?>
<schema_mod xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://iwm.uni-koblenz.de/xsd/ims_apv1p3 ims_apv1p3.xsd"
     baseSchema="ims_qtilitev1p2.xml">

     <modifications>
     …
     </modifications>

     <definitions>
     …
     </definitions>

</schema_mod>
```



3.2.2 Modifications

The <modifications> element encodes a sequence of <modification> elements describing the modification that the profile applies.

Each modification modifies a specific element of the base schema. The modified element is determined by an XPath expression in the *element* attribute of the <modification> tag.

The *element* attribute sets the context for the components of the modification as follows. If it points to a direct element or attribute definition, then the location of this definition is modified. However, if it points to a definition by reference using the ref attribute, then the target of this reference becomes the context for the individual components of the <modification>. We will refer to this way of calculating the context for modifications of references as the *Principle of Reference Tracing*.

The components of a <modification> element are:

- Cardinality modifications <cardinality>; or
- Value type modifications <attribute_properties>; or
- Extensions defined by <element_extension> or <attribute_extension>.

Moreover a <modification> can have other <modification> elements as components. The role of a <modification> as a component of another <modification> will be explained in sub-section 3.2.7.

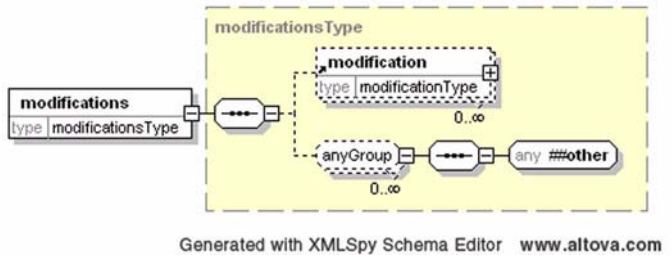
Each of these elements can have their own XPath expression as values of the *subelement* attribute respectively in order to determine which item they modify, pointing to items in the base schema. If any of these XPath expressions is a relative path, it is taken relative to the XPath expression of its parent element's *element* attribute.

In case of a cardinality modification, the XPath expression must refer to an element of the base schema inside a complex type, while it must refer to an attribute or to an element with simple content in the base schema in case of an attribute properties modification. Otherwise the concerned modification component is ignored. It is recommended that evaluating systems issue a warning in this case.

The XPath expression must denote exactly one element of the base schema.

Example:

```
<modifications>
   <modification element="/xs:schema/xs:complexType[@name='altmaterialType']">
      <cardinality subelement="./xs:choice[1]" maxOccurs="1" cnd="cnd1"/>
      <attribute_properties subelement="./xs:attribute[@name='lang']" type="sid001"
         use="optional" default="de"/>
   </modification>
</modifications>
```



3.2.3 Cardinality Modifications

Cardinality modifications, designated by *cardinality*, can only declare modifications of the minOccurs or maxOccurs attribute of the element to which the XPath expression in their *subelement* attribute refers.

A cardinality modification may depend on a condition referred to in its *cnd* attribute (see below for the discussion of conditions).

Note that increasing the *minOccurs* attribute and decreasing the *maxOccurs* attribute will restrict the set of documents that are compliant with the newly defined application profile. Consequently, systems which are read compliant with the new profile may not be read compliant with the previous profile and systems which are write compliant with the previous profile may not be write compliant with the new profile.

3.2.4 Attribute Properties Modifications

Attribute properties modifications, designated by *attribute_properties*, can change for the attribute or text node to which the XPath expression in their *subelement* attribute refers:

- the datatype (by giving in their *type* attribute a type already defined in the base schema or a reference to a type definition in the definitions section (see below));
- the usage (by giving their *use* attribute one of the values *prohibited*, *optional* or *required*);
- the default value (by defining their *default* attribute) or fix a specific value (by defining their *fixed* attribute).

An attribute properties modification may depend on a condition referred to in its *cnd* attribute (see below for the discussion of conditions).

3.2.5 Element Extensions

Element extensions allow the introduction of new elements at locations where this has been foreseen in the base specification. <element_extension> is allowed only if the XPATH expression in its *subelement* attribute points to an element with name *xs:any.* It must give a list of namespaces in its *namespace* attribute in order to specify which elements are expected. Moreover it may provide in its *processContents* attribute one of the tokens *skip*, *lax*, or *strict*, the default being *skip*. These values are used as in the W3C documentation of the *xs:any* wildcard.

3.2.6 Attribute Extensions

Attribute extensions allow adding new attributes for existing elements. <attribute_extension> is allowed only if the XPATH expression in its *subelement* attribute points to an element with name *xs:anyAttribute*. It must give a list of namespaces in its *namespace* attribute in order to define where the attributes to be inserted at this place can be taken from. Moreover it may provide in its *processContents* attribute one of the tokens *skip*, *lax*, or *strict*, the default being *skip*. These values are used as in the W3C documentation of the *xs:anyAttribute* wildcard.

3.2.7 Restricting the Effect of a Modification - Handling References

Specification schemas may reuse their definitions by referencing one definition from several places within the schema. When an element in such a referenced definition is modified without precaution, this modification will affect all places that reference this element. Sometimes the consequences of such a modification may not be easy to detect since references may be nested; for example, the definition of an element will be affected if its type references a complex type that has inside it a reference to an element that has been modified.

In order to prevent this effect where it is not intended, a reference to a definition must be conceptually replaced by a copy of the definition. Then modifications of this copy can be made without affecting the reuse of the original definition at other places.

The need to make a modification to a local copy of the referenced definition instead of the global definition is satisfied by setting the *element* attribute of the modification to point to the definition by reference and not to the location of the global definition. Nevertheless, due to the Principle of Reference Tracing, the XPATH values of the *subelement* attributes of the modification components can be calculated in the same way as for modifications of the global definition, however they will take effect only for the use of the global definition at the referenced place.

Note that references are always references to global definitions in the base schema and that therefore modifications with an element attribute pointing to a definition by reference are interpreted as working on a copy of the global definition from the *base schema*.

Hence, if modifications are made to a global definition as well as to a place where this global definition is referenced, the modifications made to the global definition will no longer apply at the point where this definition is referenced.

So a definition by reference:

- Either has no specific modification assigned and all modifications of the reference target take effect for this definition; or
- It has a modification assigned and only the modifications assigned to this definition take effect for it.

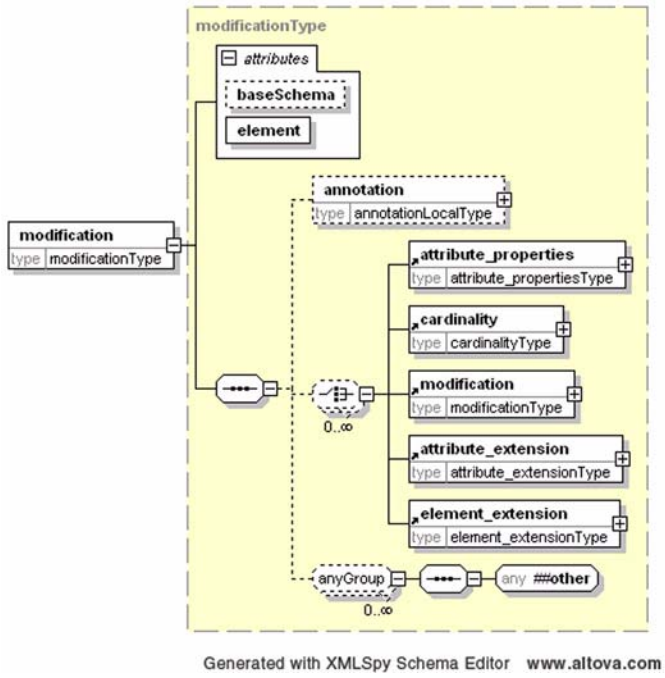
As a special application of this, the location of a reference can be protected from all modifications by assigning an empty modification.

Referenced definitions may use again references to other definitions. For example, let us suppose that in the base schema inside the definition of an element A at XPathReferencingB there is a reference to a definition B which is located at XPathB. Inside B, at the location XPathReferencingC relative to the position XPathB there can be a reference to another definition C. In order for a modification of C, say a cardinality modification <cardinality …>…</cardinality>, to take effect only when used inside the local copy of B referenced within A, this modification must be defined inside the modification assigned to XPathReferencingB as:

```
<modification element=XPathReferencingB>
   <modification element=XPathReferencingC>
      <cardinality…>…</cardinality>
   </modification>
</modification>
```

The semantics of this construct is that in the base schema:

- Definition B is copied into a new definition B1 at a new location XPathB1;
- The reference at XPathReferencingB is redirected to point to B1;
- Definition C is copied into a new definition C1 at a new location XPathC1;
- The reference inside B1 at the relative path XPathReferencingC is redirected to point to C1;
- C1 is modified as described in <cardinality…>…</cardinality>.



Example of a local modification:

```
<modification element="/xs:schema/xs:complexType[@name='itemType']">
    <modification element="./xs:sequence/xs:element[@name='objectives']">
        <cardinality maxOccurs="1" minOccurs="1"
            subelement="./xs:sequence/xs:element[@name='material']">
            <annotation>
                <documentation category="explanation" xml:lang="en">
                    Allow only one material-element in the objectives, but only if objectives is a
                    child of the item-element.
                </documentation>
                    </annotation>
        </cardinality>
    </modification>
</modification>
```

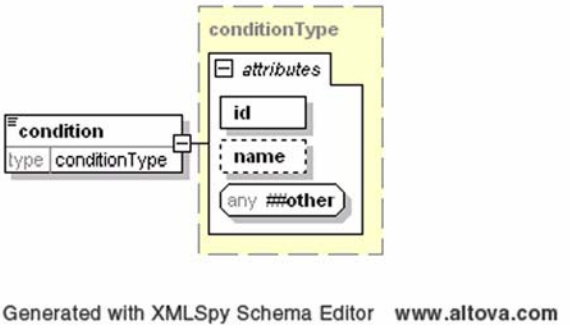
3.2.8 Conditions

Modifications which are bound to conditions may or may not take effect, depending on the situation during the validation of a particular document. Especially, the conditions are evaluated only during the validation of such a document. The content of a condition is an XPATH expression which evaluates to *true* or *false* as it could be written in XSLT as the value of the *test* attribute of an *xsl:if* element. When an occurrence of an element or attribute at a

concrete place in a concrete document is validated and for this element there is a modification depending on a condition, the XPATH expression is evaluated. If this gives *true* (see any XSLT documentation for details) it is checked whether the cardinality or value type of the respective element or attribute is as required by the modification. If there are several conditions bound to this modification, only the first whose content evaluates to *true* takes effect.

It is important to note that the XPATH expressions in conditions refer to items in the concrete instance documents while the XPATH expressions we met in the modification elements refer to items in the base schema. Accordingly, relative XPATH expressions from conditions will be evaluated relative to the position of the item in the document whose definition is modified in the profile.

Conditions must have an identifier attribute *id* by which they can be referred. Note that a condition can be used by several modifications. A condition may well evaluate to *true* in one context and to *false* in another context in the same document. Also, a modification element may contain references to several conditions. In this case all referenced conditions are checked in the sequence in which they occur. Only the *cardinality* modification and the *attribute_properties* modification with the first condition evaluating to *true* will be considered. It is recommended that the evaluating system issues a warning if it finds several referenced conditions to be true at the same place.



Example:

```
<condition id="cnd001">count(objectives)=0</condition>
```

3.2.9 Type Definitions

Only simple types can be redefined. For these redefinitions XML Schema provides a number of basic tools. These are:

- Restrictions
- Unions
- Lists

Restrictions of an existing simple type restrict the set of compliant documents and can render applications write incompliant which had been write compliant before. Unions using the previously assigned type as one of their arguments extend the set of compliant documents and can therefore render applications read incompliant which had been read compliant before. Lists introduce completely incompatible types and their usage is discouraged.

The basic type operations can be modified by a variety of facets. The concrete set of facets available depends also on the basic type to be modified. Enumeration of a set of standard values for numbers or strings as well as the declaration of minimal or maximal values for ranges of numbers is frequently used as a facet. More on facets can be found in the documentation of XML Schemas.

Type definitions are enclosed in *simpleType* tags and are defined as usual in XML Schema. Type definitions must have an attribute *name* by which they can be referred from the *attribute_properties* modifications.

Example:

```
<xs:simpleType name="sid001">
   <xs:restriction base="xml:lang">
      <xs:enumeration value="de_DE"/>
   </xs:restriction>
</xs:simpleType>
```

3.2.10 Mappings

Mappings provide means to relate values of a simple type in the base schema to values in possibly another simple type in the profile. This can be used for example to translate a standard vocabulary of the base schema into a standard vocabulary of the community for which the profile was created. However the mapping does not prescribe any particular usage.

A mapping is given by:

a) An XPath expression pointing to the location of the enumeration type to be mapped in the base schema. This type is subsequently called the *domain type* of the mapping;

b) A reference to the type to which the members of the domain type are mapped. This type, called the *range type* of the mapping. It must be defined in the base schema or in the *definitions* section of the profile;

c) A sequence of pairs of items, consisting of a member of the domain type and a member of the range type. The pair is called an argument-value pair (*avpair*). The first component of the pair, taken from the domain type, is called the *argument* while the second, taken from the range type, is called the *value*.

Mappings are referred to only from value type modifications. The value type modification itself determines already the domain type and the range type. Therefore, it is sufficient to give the sequence of argument-value pairs to determine a mapping sufficiently.

Note that not necessarily each member of the domain type or of the range type must occur in an argument-value pair of a mapping. Also, note that there can be multiple argument-value pairs in a mapping which share the same argument or value. However, some types of mappings deserve special attention.

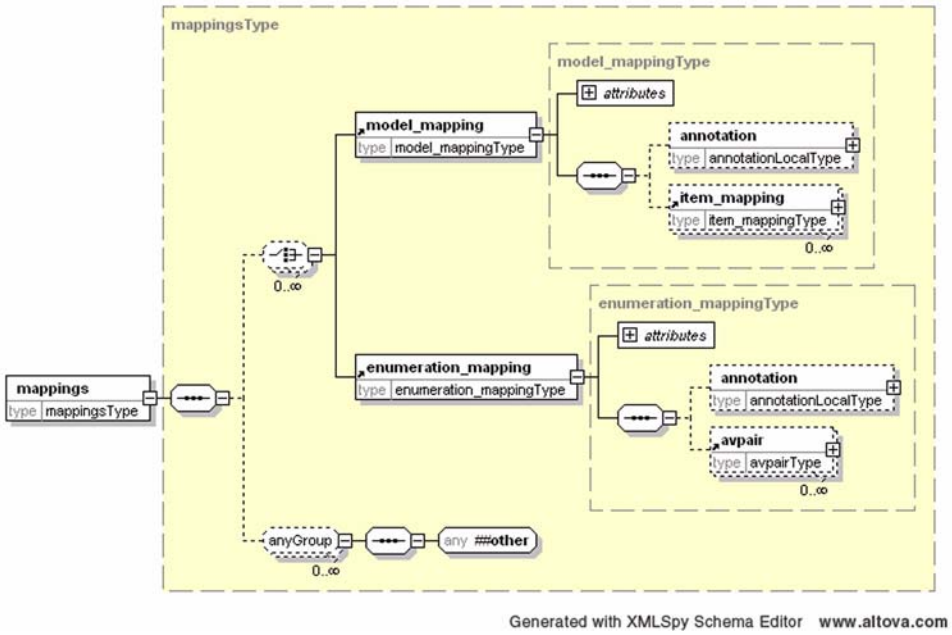
A mapping is said to be *from* the domain type if each member of the domain type occurs in at least one argument-value pair. It is said to be *onto* the range type if each member of the range type occurs in at least one argument-value pair. A mapping is called *functional* if each member of the domain type occurs in at most one argument-value pair. Note that in a functional mapping from the domain type each member of the domain type occurs in exactly one argument-value pair. This can be used to provide a translation of items in the domain type in a document which is compliant with the base schema into items as required by the application profile.

A mapping is called *inverse functional* if each member of the range type occurs in at most one argument-value pair in the mapping. Note that in an inverse functional mapping onto the range type each member of the range type occurs in exactly one argument-value pair. This can be used to provide a translation of items in the range type in a document which is compliant with the application profile into items as required by the base schema.

A mapping is called *bijective* if each member of the domain type and each member of the range type occur in exactly one argument-value pair. Bijective mappings can be used to provide back and forth translations between the domain and range type. In this case the domain and range type must have the same number of members.

An important point to note is the restricted effect of mappings for compliance testing. They are applied only at places where concrete members of the domain type occur in the base schema, for example in setting default elements.

The concept of mapping defined here is restricted to enumerations. Therefore, it does not support mappings between infinite types, say from Integers to Reals.



Example:

```
<modifications>
   <modification element="/xs:schema/xs:complexType[@name='decvarType']">
      <attribute_properties name="./xsattribute[@name='grade']" type="de_grades"
      mapping="grademap"/>
   </modification>
</modifications>

<definitions>
   <xs:simpleType name="de_grades">
      <xs:restriction base="xs:PositiveInteger">
         <xs:enumeration value="1"/>
         <xs:enumeration value="2"/>
         <xs:enumeration value="3"/>
      …
      </xs:restriction>
   </xs:simpleType>
</definitions>

<mappings>
   <enumeration_mapping id="grademap">
      <avpair argument="A" value="1"/>
      <avpair argument="B" value="2"/>
   </enumeration_mapping>
</mappings>
```

3.2.11 Expressing Human-readable Information in a Profile

A distinction can be made between documenting the application profile as a whole and documenting parts of the application profile. For documenting the whole application profile there exist five different categories which you can choose by setting the *category* attribute to one of the following values:

- *name*: The name of the application profile;
- *scope*: The scope and the responsible organization for the application profile;
- *policy*: The policy for using the application profile;

- *conformance*: Conformance testing procedures for the application profile and statement of conformance to the base schema;
- *general*: General information about the application profile which couldn't be assigned to the previous categories.

For documenting a certain part of the application profile there exist two different categories:

- *explanation*: Clarifying the meaning of an item;
- *rationale*: Providing further explanations for an item.

It is also possible to provide different translations of the documentation. The documentation-elements are encapsulated in an annotation-element.

Example commenting the application profile:

```
<schema_mod baseSchema="ims_qtilitev1p2.xsd" … >
   <annotation>
      <documentation category="name" xml:lang="en">
         QTI Lite example profile
      </documentation>
      <documentation category="general" xml:lang="en">
         This profile should demonstrate the usage of application profiles.
      </documentation>
   </annotation>
   …
</schema_mod>
```

Example commenting a modification:

```
<cardinality maxOccurs="1" minOccurs="1"
   subelement="./xs:sequence/xs:element[@name='objectives']">
   <annotation>
      <documentation category="explanation" xml:lang="en">Make the objectives mandatory.
      </documentation>
   </annotation>
</cardinality>
```

3.2.12 Things That Should Not be Changed

There are a number of possible schema modifications which are very likely to destroy interoperability and which are not supported by the XML binding described here. Thus it is not possible to redefine complex types. Especially it is not possible to add elements or attributes, except at points where this has been foreseen by the base schema providing appropriate wildcards or to change the order of elements which are already in canonical form.

Mappings, replacing specific values of an enumeration by others, are also not covered by this binding. It is, however, possible to replace a given enumeration by another enumeration.

3.2.13 Wild Extensions

In some cases there is a desire to introduce new elements or attributes at places where this has not been foreseen in the profiled base schema. This is likely to seriously damage interoperability and it is STRONGLY DISCOURAGED and it is therefore not supported by the Application Profile Binding XML Schema. However, realizing that such extensions have been applied in real world application profiles, we define a way in which this can be described in XML using a profile of the Application Profile Binding XML Schema. Readers who confine extensions to pre-defined extension points in the base schema may skip the rest of this subsection.

The first step towards introducing a new element or attribute is to extend the definition of application profiles by allowing additional elements *new_element_extension* and *new_attribute_extension* within the *modification* element of an application profile. These additional elements are defined in the additional schema **telcert_extensions_schemav1p0**. These new elements are equipped with an optional attribute *next_sibling*. The use of this attribute will be explained in the example given below.

Schema telcert_extensions_schemav1p0:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- filename=telcert_extensions_schemav1p0.xsd -->
<xs:schema targetNamespace="http://iwm.uni-koblenz.de/xsd/telcert_extensions_schemav1p0"
   xmlns="http://iwm.uni-koblenz.de/xsd/telcert_extensions_schemav1p0"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:ap="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
   elementFormDefault="qualified"
   version="TELCERT EXTENSIONS 1.0">

   <xs:annotation>
      <xs:documentation>
         This schema is used for extending ims_apv1p3.xsd.
      </xs:documentation>
   </xs:annotation>
   <!-- ************************** -->
   <!-- ** Inclusions and Imports ** -->
   <!-- ************************** -->
   <xs:import namespace="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
            schemaLocation="ims_apv1p3.xsd"/>
   <!-- ************************* -->
   <!-- ** Element Declarations ** -->
   <!-- ************************* -->
   <xs:element name="new_element_extension" type="new_element_extensionType"/>
   <xs:element name="new_attribute_extension" type="new_attribute_extensionType"/>
   <!-- ********************************* -->
   <!-- ** Attribute Group Declarations ** -->
   <!-- ********************************* -->
   <xs:attributeGroup name="newExtensionAttrGroup">
      <xs:attribute name="nextSibling" type="xs:token"/>
   </xs:attributeGroup>
   <!-- ************************** -->
   <!-- ** new_element_extension ** -->
   <!-- ************************** -->
   <xs:complexType name="new_element_extensionType">
      <xs:complexContent>
         <xs:extension base="ap:element_extensionType">
            <xs:attributeGroup ref="newExtensionAttrGroup"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <!-- **************************** -->
   <!-- ** new_attribute_extension ** -->
   <!-- **************************** -->
   <xs:complexType name="new_attribute_extensionType">
      <xs:complexContent>
         <xs:extension base="ap:attribute_extensionType">
            <xs:attributeGroup ref="newExtensionAttrGroup"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

The application profile **telcert_extensions_profilev1p0.xml** enables the addition of the elements *new_element_extension* and *new_attribute_extension,* which have been defined in the previous schema, within the *modification* elements of application profiles. This profile makes use of the *element_extension* extension point provided within the *modificationType* definition element of the schema **ims_apv1p3**.

Application Profile telcert_extensions_profilev1p0:

```
<schema_mod baseSchema="ims_apv1p3.xsd"
   xmlns="http://iwm.uni-koblenz.de/xsd/ims_apv1p2"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://iwm.uni-koblenz.de/xsd/ims_apv1p3 ims_apv1p3.xsd">
   <annotation>
      <documentation category="general" xml:lang="en">
         This profile extends the schema for application profiles so that new types of
         Modifications are allowed. These modifications add new extension points for elements
```

```
         Or attributes to a base schema.
       </documentation>
       <documentation category="name" xml:lang="en">TELCERT EXTENSIONS 1.0</documentation>
   </annotation>
   <modifications>
      <modification element="/xs:schema/xs:complexType[@name='modificationType']">
         <modification element="./xs:sequence[1]/xs:group[1]">
            <element_extension maxOccurs="1" minOccurs="1"
             namespaceList="http://iwm.uni-koblenz.de/xsd/telcert_extensions_schemav1p0"
             processContents="lax" subelement="./xs:sequence[1]/xs:any[1]">
               <annotation>
                  <documentation category="explanation" xml:lang="en">
                     Allow adding new extensions points to the base schema.
                  </documentation>
               </annotation>
            </element_extension>
         </modification>
      </modification>
   </modifications>
   <definitions/>
   <mappings/>
</schema_mod>
```

Now we are ready to define extended application profiles using the elements *new_element_extension* and *new_attribute_extension* allowed by our profiled definition of application profiles. As an example, we define an extended profile **new_element_example_profile** which adds to the IMS Content Packaging Schema definition of the type *itemType* a new wildcard element before the *item* element. To accomplish this, the profile locates in the Content Packaging Schema the *item* element in the *sequence* group of the *itemType* definition and introduces the intended additional element before this *item* element. In order to introduce a new element as the last element of the group, the *nextSibling* attribute should be omitted.

Profile new_element_example_profile:

```
<schema_mod baseSchema="imscp_rootv1p1p3.xsd"
   xmlns="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:te="http://iwm.uni-koblenz.de/xsd/telcert_extensions_schemav1p0"
   xsi:schemaLocation="http://iwm.uni-koblenz.de/xsd/ims_apv1p3 ims_apv1p3.xsd
      http://iwm.uni-koblenz.de/xsd/telcert_extensions_schemav1p0
      telcert_extensions_schemav1p0.xsd">
   <annotation></annotation>
   <modifications>
      <modification element="/xsd:schema/xsd:complexType[@name='itemType']">
         <te:new_element_extension
            subelement="./xsd:sequence" nextSibling="./xsd:element[@ref='item']"
            minOccurs="1" maxOccurs="1" namespace="##any" processContents="lax"/>
      </modification>
   </modifications>
   <definitions></definitions>
   <mappings></mappings>
</schema_mod>
```

Thus the overall hierarchy of definitions is shown in Figure 3.1.

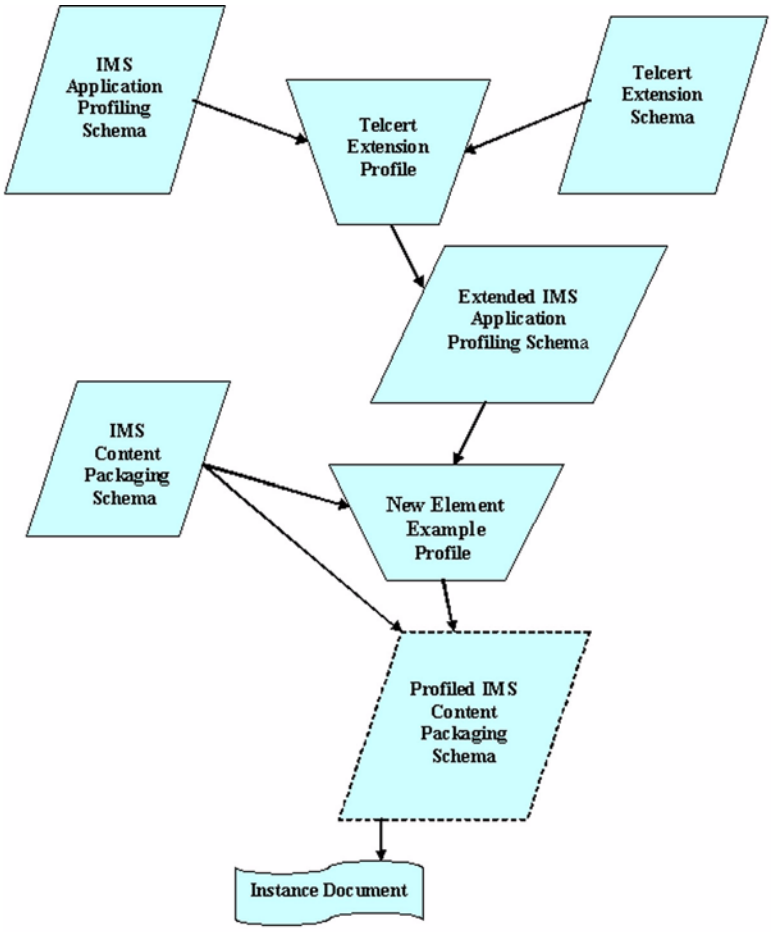


Figure 3.1 Schematic representation of the definitions hierarchy.

In our example, where neither the **Telcert Extension Profile** nor the **New Element Example Profile** makes use of conditions it is possible to derive:

- An intermediate **Extended IMS Application Profiling Schema** which contains additionally the definitions from the **Telcert Extension Schema**.
- A **Profiled IMS Content Packaging Schema** that contains in the definition of the *itemType* type the additional wildcard defined in the **New Element Example Profile**.

3.2.14 Semantics: How to Validate a Document Against an Application Profile

A system that validates a document against an application profile should proceed in two steps. In the first step it generates a modified schema out of the base schema. This step is independent of any concrete document. To this end it first replaces in the base schema all definitions by reference to which modifications are attached with references to copies of the respective definitions and lets the references and the respective modifications in the application profile point to these copies. This has to be iterated until all modified definitions by reference are resolved.

Then it takes all modifications which do not refer to any condition and all type definitions. Within the base schema the required replacements of cardinalities, attribute groups and type values are performed and the new type definitions are copied into the schema. We mention that fixing the value of text nodes is not supported in XML Schemas. Therefore, these modifications will also not be incorporated into the modified schema, even if they are not subject to any condition. These modifications must be checked in the following step as if they were defined with a single condition *true*.

.

The second step validates the concrete document against the modified schema. In addition, it tests for each occurrence of an element of the base schema whether the profile contains for this element a modification which is bound to a condition. If this is the case, the XSLT expressions forming the content of these conditions are evaluated. For the first which evaluates to true it is checked whether the number of occurrences of this element at this place is within the limits defined by the *maxOccurs* and *minOccurs* values of the respective cardinality modification. If the modification is a value type modification it is checked whether the concrete value has the type pointed to by the *name* attribute of the value type modification.

3.2.15 A Sample Binding of an Application Profile

```
<schema_mod baseSchema="ims_qtilitev1p2.xsd"
   xmlns="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://iwm.uni-koblenz.de/xsd/ims_apv1p2 ims_apv1p3.xsd">

   <annotation>
      <documentation category="name" xml:lang="en">QTI Lite example profile</documentation>
      <documentation category="general" xml:lang="en">
         This profile should demonstrate the usage of application profiles.
      </documentation>
   </annotation>
   <modifications>
      <modification element="/xs:schema/xs:complexType[@name='decvarType']">
         <attribute_properties fixed="Float"
            subelement="./xs:simpleContent/xs:extension/xs:attribute[@name='vartype']"
            type="modifiedVartype" mapping="em001">
            <annotation>
               <documentation category="rationale" xml:lang="en">
                  Showing the use of a attribute_properties-modification.
               </documentation>
            </annotation>
         </attribute_properties>
      </modification>
      <modification element="/xs:schema/xs:complexType[@name='itemType']">
         <modification element="./xs:sequence/xs:element[@name='objectives']">
            <cardinality maxOccurs="1" minOccurs="1"
               subelement="./xs:sequence/xs:element[@name='material']">
               <annotation>
                  <documentation category="explanation" xml:lang="en">
                     Allow only one material-element in the objectives
                     but only if objectives is a child of the item-element.
                  </documentation>
               </annotation>
            </cardinality>
         </modification>
         <cardinality maxOccurs="1" minOccurs="1"
            subelement="./xs:sequence/xs:element[@name='objectives']">
            <annotation>
               <documentation category="explanation" xml:lang="en">
                  Make the objectives mandatory.
               </documentation>
            </annotation>
         </cardinality>
      </modification>
   </modifications>
   <definitions>
      <xs:simpleType name="modifiedVartype">
         <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="Integer"/>
            <xs:enumeration value="PositiveInteger"/>
            <xs:enumeration value="Float"/>
         </xs:restriction>
      </xs:simpleType>
```

```
   </definitions>
   <mappings>
      <enumeration_mapping id="em001">
         <avpair argument="Integer" value="Integer"/>
         <avpair argument="Integer" value="PositiveInteger"/>
      </enumeration_mapping>
   </mappings>
</schema_mod>
```

3.2.16 Application Profiling XML Schema

The following schema defines the structure of the XML encoding of an application profile.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- filename=ims_apv1p3.xsd -->
<xs:schema targetNamespace="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
   xmlns="http://iwm.uni-koblenz.de/xsd/ims_apv1p3"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   version="IMS APPLICATION PROFILING 1.3">
   <!-- ************************** -->
   <!-- ** Inclusions and Imports ** -->
   <!-- ************************** -->
   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
   schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
   <!-- ****************** -->
   <!-- ** Root Element ** -->
   <!-- ****************** -->
   <xs:element name="schema_mod" type="schema_modType"/>
   <!-- ************************* -->
   <!-- ** Element Declarations ** -->
   <!-- ************************* -->
   <xs:element name="attribute_properties" type="attribute_propertiesType"/>
   <xs:element name="attribute_extension" type="attribute_extensionType"/>
   <xs:element name="avpair" type="avpairType"/>
   <xs:element name="cardinality" type="cardinalityType"/>
   <xs:element name="condition" type="conditionType"/>
   <xs:element name="definitions" type="definitionsType"/>
   <xs:element name="element_extension" type="element_extensionType"/>
   <xs:element name="enumeration_mapping" type="enumeration_mappingType"/>
   <xs:element name="item_mapping" type="item_mappingType"/>
   <xs:element name="mappings" type="mappingsType"/>
   <xs:element name="model_mapping" type="model_mappingType"/>
   <xs:element name="modification" type="modificationType"/>
   <xs:element name="modifications" type="modificationsType"/>
   <!-- ***************************** -->
   <!-- ** Simple Type Declarations ** -->
   <!-- ***************************** -->
   <xs:simpleType name="namespaceList">
      <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>
   <!-- ********************************* -->
   <!-- ** Attribute Group Declarations ** -->
   <!-- ********************************* -->
   <xs:attributeGroup name="extensionAttrGroup">
      <xs:attribute name="namespace" type="namespaceList"/>
      <xs:attribute name="processContents" default="skip">
         <xs:simpleType>
            <xs:restriction base="xs:token">
               <xs:enumeration value="skip"/>
               <xs:enumeration value="lax"/>
               <xs:enumeration value="strict"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:attributeGroup>
   <xs:attributeGroup name="anyAttrGroup">
      <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:attributeGroup>
   <!-- ************************ -->
   <!-- ** Group Declarations ** -->
   <!-- ************************ -->
   <xs:group name="anyGroup">
      <xs:sequence>
         <xs:any namespace="##other" processContents="lax"/>
      </xs:sequence>
   </xs:group>
   <!-- **************** -->
   <!-- ** annotation ** -->
   <!-- **************** -->
   <xs:complexType name="annotationGlobalType">
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" name="documentation"
            type="documentationGlobalType"/>
         <xs:group minOccurs="0" maxOccurs="unbounded" ref="anyGroup"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="annotationLocalType">
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" name="documentation"
            type="documentationLocalType"/>
         <xs:group minOccurs="0" maxOccurs="unbounded" ref="anyGroup"/>
      </xs:sequence>
   </xs:complexType>
   <!-- ************************** -->
   <!-- ** attribute_properties ** -->
   <!-- ************************** -->
   <xs:complexType name="attribute_propertiesType">
      <xs:complexContent>
         <xs:extension base="baseModification">
            <xs:annotation>
               <xs:documentation>
                  The type-attribute references a simpleType-definition either from the
                  definitions-section of the profile or from one of the namespaces
                  known in the profile.
               </xs:documentation>
            </xs:annotation>
            <xs:attribute name="use">
                  <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="type" type="xs:QName"/>
            <xs:attribute name="mapping" type="xs:IDREF"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <!-- ************************ -->
   <!-- ** attribute_extension ** -->
   <!-- ************************ -->
   <xs:complexType name="attribute_extensionType">
   <xs:complexContent>
      <xs:extension base="baseModification">
         <xs:attributeGroup ref="extensionAttrGroup"/>
   </xs:extension>
   </xs:complexContent>
   </xs:complexType>
   <!-- ************ -->
   <!-- ** avpair ** -->
   <!-- ************ -->
   <xs:complexType name="avpairType">
      <xs:attribute name="argument" type="xs:string" use="required"/>
      <xs:attribute name="value" type="xs:string" use="required"/>
   </xs:complexType>
   <!-- ******************** -->
   <!-- * baseModification * -->
   <!-- ******************** -->
   <xs:complexType name="baseModification">
      <xs:sequence>
         <xs:element name="annotation" type="annotationLocalType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="subelement" type="xs:token" use="required"/>
      <xs:attribute name="cnd" type="xs:IDREF"/>
      <xs:attributeGroup ref="anyAttrGroup"/>
   </xs:complexType>
   <!-- ***************** -->
   <!-- ** cardinality ** -->
   <!-- ***************** -->
   <xs:complexType name="cardinalityType">
      <xs:complexContent>
         <xs:extension base="baseModification">
            <xs:annotation>
               <xs:documentation>
                  The ref-attribute is a XPath-expression which points to the part of the
                  subelements-definition whose cardinality should be modified. The context
                  node for this XPath-expression is the node which is selected by the ref-
                  attribute of the parent modification-element.
               </xs:documentation>
            </xs:annotation>
            <xs:attribute name="maxOccurs" type="xs:nonNegativeInteger"/>
            <xs:attribute name="minOccurs" type="xs:nonNegativeInteger"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <!-- *************** -->
   <!-- ** condition ** -->
   <!-- *************** -->
   <xs:complexType name="conditionType">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="id" type="xs:ID" use="required"/>
            <xs:attribute name="name" type="xs:string"/>
            <xs:attributeGroup ref="anyAttrGroup"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!-- ***************** -->
```

```
<!-- ** definitions ** -->
<!-- ***************** -->
<xs:complexType name="definitionsType">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="condition"/>
      <xs:any namespace="##other" processContents="lax"/>
   </xs:choice>
</xs:complexType>
<!-- ****************** -->
<!-- ** documentation ** -->
<!-- ****************** -->
<xs:complexType name="documentationType">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute ref="xml:lang"/>
         <xs:attributeGroup ref="anyAttrGroup"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="documentationGlobalType">
   <xs:simpleContent>
      <xs:extension base="documentationType">
         <xs:attribute name="category" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:NMTOKEN">
                  <xs:enumeration value="name"/>
                  <xs:enumeration value="scope"/>
                  <xs:enumeration value="policy"/>
                  <xs:enumeration value="conformance"/>
                  <xs:enumeration value="general"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="documentationLocalType">
   <xs:simpleContent>
      <xs:extension base="documentationType">
         <xs:attribute name="category" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:NMTOKEN">
                  <xs:enumeration value="explanation"/>
                  <xs:enumeration value="rationale"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<!-- ********************** -->
<!-- ** element_extension ** -->
<!-- ********************** -->
<xs:complexType name="element_extensionType">
   <xs:complexContent>
      <xs:extension base="cardinalityType">
         <xs:attributeGroup ref="extensionAttrGroup"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<!-- ************************ -->
<!-- ** enumeration_mapping ** -->
<!-- ************************ -->
<xs:complexType name="enumeration_mappingType">
   <xs:sequence>
      <xs:element name="annotation" type="annotationLocalType" minOccurs="0"/>
      <xs:element ref="avpair" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
```

```
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attributeGroup ref="anyAttrGroup"/>
   </xs:complexType>
   <!-- ****************** -->
   <!-- ** item_mapping ** -->
   <!-- ****************** -->
   <xs:complexType name="item_mappingType">
      <xs:attribute name="schema_item" type="xs:token"/>
      <xs:attribute name="external_item" type="xs:token"/>
   </xs:complexType>
   <!-- ************** -->
   <!-- ** mappings ** -->
   <!-- ************** -->
   <xs:complexType name="mappingsType">
      <xs:sequence>
         <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="model_mapping"/>
            <xs:element ref="enumeration_mapping"/>
         </xs:choice>
         <xs:group ref="anyGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <!-- ******************* -->
   <!-- ** model_mapping ** -->
   <!-- ******************* -->
   <xs:complexType name="model_mappingType">
      <xs:sequence>
         <xs:element name="annotation" type="annotationLocalType" minOccurs="0"/>
         <xs:element ref="item_mapping" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="model" type="xs:anyURI" use="required"/>
      <xs:attributeGroup ref="anyAttrGroup"/>
   </xs:complexType>
   <!-- ****************** -->
   <!-- ** modification ** -->
   <!-- ****************** -->
   <xs:complexType name="modificationType">
      <xs:annotation>
         <xs:documentation>
            The ref-attribute is a XPath-expression which points to the complexType-element
            in the base schema which should be modified.
         </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="annotation" type="annotationLocalType" minOccurs="0"/>
         <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="attribute_properties"/>
            <xs:element ref="cardinality"/>
            <xs:element ref="modification"/>
            <xs:element ref="attribute_extension"/>
            <xs:element ref="element_extension"/>
         </xs:choice>
         <xs:group ref="anyGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="baseSchema" type="xs:anyURI" use="optional"/>
      <xs:attribute name="element" type="xs:token" use="required"/>
   </xs:complexType>
   <!-- ******************* -->
   <!-- ** modifications ** -->
   <!-- ******************* -->
   <xs:complexType name="modificationsType">
      <xs:sequence>
         <xs:element minOccurs="0" maxOccurs="unbounded" ref="modification"/>
         <xs:group ref="anyGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <!-- ***************** -->
   <!-- ** schema_mod ** -->
```

```
   <!-- *************** -->
   <xs:complexType name="schema_modType">
      <xs:sequence>
         <xs:element name="annotation" type="annotationGlobalType" minOccurs="0"/>
         <xs:element ref="modifications"/>
         <xs:element ref="definitions"/>
         <xs:element ref="mappings"/>
         <xs:group ref="anyGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="baseSchema" type="xs:anyURI" use="required"/>
      <xs:attributeGroup ref="anyAttrGroup"/>
   </xs:complexType>
</xs:schema>
```

4. Representing and Documenting an Application Profile

This section outlines a proposed structure for documenting an application profile.

4.1 Scope

Any application profile document set must include some scoping statement, defining what the profile's intended usage is, what base specifications the profile is based upon, and who the community is that are intended users.

4.2 Terms of Reference

Application profiles should clearly indicate their provenance, ownership, and governance.

4.3 Information Model

As described in sub-section 3.1, the profiling statements made against the information model need to be listed in natural language and also preferably include information about what motivated the authors to propose the statement.

4.4 Binding

If the profile is a bound data profile, then it needs to contain at least one binding. Bindings may be expressed using the methods described in this document. If the application profile has a binding as described in the previous section it is recommended to include the documentation of the profile within the binding and to generate the human readable documentation, any derived XML schema or XML Schematron rules from the Bindings.

4.5 Policy and Processes

Profiles may contain documents describing processes for ensuring data quality, policies on collection and distribution of data, training and certification requirements for staff working with the data, and so on.

4.6 Conformance Testing

If a profile is associated with a conformance program, then it should contain information on how to test systems or content for conformance (or how to find an approved test centre), and how to apply for certification. The profile may also describe other requirements for conformance or certification that are not directly related to the information model or binding, such as approved quality processes, or non-functional requirements for systems.

4.7 Tools

It is helpful to implementers of application profiles if there are tools available to assist them, and these can be usefully referenced within the application profile documentation. Such tools include self-testing kits, tools for re-engineering content, and authoring systems that produce conformant content.

Tools for editing application profiles, for producing derived XML schemas and Schematron rules and for editing data in accordance with application profiles can be found on the IMS Website: http://www.imsglobal.org/testing.html

About This Document

Title	IMS Application Profile Guidelines Part 2 – Technical Manual
Editors	Kevin Riley (IMS)
Team Co-Leads	Scott Wilson (JISC)
Version	1.0
Version Date	01 August 2005
Status	**Final**
Summary	This document offers guidance on specifying an Application Profile for a Community of Practice from a technical viewpoint.
Revision Information	10 October 2005
Purpose	This document is not a formal IMS specification. Its purpose is to act as an aid to adopters of IMS specs in drafting up an Application Profile, detailing how they are using the specifications for their implementation. As such, the Application Profile has value simply as an aid to implementation across a community, but can also be used by that community to identify their test acceptance criteria for conformance purposes.
Document Location	http://www.imsglobal.org/ap/apv1p0/imsap_techv1p0.html

To register any comments or questions about this specification please visit:
http://www.imsglobal.org/developers/ims/imsforum/categories.cfm?catid=17

List of Contributors

The following individuals contributed to the development of this document:

Name	Organization
John Bell	Ufi
Kerry Blinco	IMS Australia
Lorna Campbell	JISC
Ingo Dahn	University of Koblenz-Landau
Norm Friesen	Industry Canada
Dick Hill	UK e-Universities
Peter Hope	Canadian National Defence
Jeff Merriman	MIT
Bill Olivier	JISC
Kevin Riley	IMS
Anthony Roberts	Industry Canada
Colin Smythe	IMS
Scott Wilson	JISC

Revision History

Version No.	Release Date	Comments
Final 1.0	10 October 2005	The first formal release of this document.

Index