Abstract

This document is a guideline for developing Conexxus XML schemas. This guideline will help to ensure that all schemas and the resulting XML will conform to a standard layout and presentation. This guideline will apply to all schemas developed by Conexxus and its committees.
# Revision History

<table>
<thead>
<tr>
<th>Revision Date</th>
<th>Revision Number</th>
<th>Revision Editor(s)</th>
<th>Revision Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 23, 2015</td>
<td>1.3</td>
<td>Linda Toth, Conexxus</td>
<td>Added rule on repetition Updated copyright and disclaimers</td>
</tr>
<tr>
<td>July 29, 2014</td>
<td>1.2</td>
<td>Linda Toth, Conexxus</td>
<td>Approved Release</td>
</tr>
<tr>
<td>June 9, 2014</td>
<td>Draft 1.2</td>
<td>Linda Toth, Conexxus</td>
<td>Changes for merging core schemas into one common namespace</td>
</tr>
<tr>
<td>April 14, 2014</td>
<td>1.1.1</td>
<td>Linda Toth, Conexxus</td>
<td>Updated for Conexxus</td>
</tr>
<tr>
<td>June 7, 2013</td>
<td>1.1</td>
<td>Linda Toth, PCATS David Ezell, VeriFone</td>
<td>Clarified name space for RFT committees, adding P2P (point-to-point encryption). - Updated proprietary extensions section and added an example</td>
</tr>
<tr>
<td>November 15, 2011</td>
<td>1.0</td>
<td>Linda Toth, PCATS</td>
<td>Corrected spelling, grammar and formatting issues once all changes were accepted. - Finalized document for release</td>
</tr>
<tr>
<td>November 7, 2011</td>
<td>Draft 0.7</td>
<td>David Ezell, VeriFone Raymond Lugo, SSSC Gregg Peele, CMI Solutions Donna Perkins, E-Z Stop Foodmarts, Inc. Brian Reynolds, The Pinnacle Corporation Ken Ringeman, Gilbarco Veeder-Root Ann Seki, Chevron Jeff Shein, Radiant Systems Linda Toth, PCATS Bill Wade, PDI Scott Wood, PCATS</td>
<td>- To set off xml from prose, changed the font for xml references and snippets. - To set off file names from prose, changed the font of file names. - Made MUST, MUST NOT, SHALL, SHALL NOT, SHOULD, and SHOULD NOT more uniform with respect to capitalization. - Modified section 1.2 (Guide to the Document) to better reflect what the document provides. - Deleted last paragraph in section 2.1.1 (Overall Schema Design). - Modified section 2.4.4 to use element form default = qualified - Rewrote section 2.4.5 on code list management. - Modified section 2.5.1.1 on URI guidelines to clarify the 4th</td>
</tr>
</tbody>
</table>
- Incorporated the “PCATS Schema Namespace” document into section 2.5.1 (Namespace specification), resulting in additional rules.
- Rewrote section 2.5.3.2 on namespaces to add clarity.
- Modified section 2.5.4.1 on naming conventions to incorporate a three character code for the functional area in the file name.
- Removed quotes around UTF-8 in section 2.5.6.1 (character set encoding).
- Added what to do when a conflict arises to rule 2.5.13.1 (acronyms).
- Rewrote section 2.7.17 (nillability) to provide use cases for when nilability is appropriate.
- Rewrote section 2.5.18 (referencing schemas) to reflect current usage and provide clarity.
- Added clarification of naming and examples of grand vs. net to section 2.5.21 (elements) and 2.5.22 (attributes).
- Modified section 2.5.23 to indicate that xs:boolean should be used for Booleans.
- Added rule 2.5.24.2 on unbounded numeric data types.
- Added rule 2.5.25.1 on unconstrained string values.

| October 7, 2011 | Draft 0.6 | David Ezell, VeriFone
David Godwin, VeriFone.
Raymond Lugo, SSCS
Gregg Peele, CMI Solutions
Jeff Shein, Radiant Systems
Linda Toth, PCATS | - Rewrote part of the guide to the document (section 1.2) to reflect what the document is and how it is laid out.
- Rewrote the design objectives (section 2.1)
- Added section on Commercial Messages (2.1.2) |
<table>
<thead>
<tr>
<th>Bill Wade, PDI</th>
<th>Scott Wood, PCATS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminated section on</td>
<td></td>
</tr>
<tr>
<td>proposed versioning</td>
<td></td>
</tr>
<tr>
<td>mechanisms (2.2.1) due</td>
<td></td>
</tr>
<tr>
<td>being redundant and no</td>
<td></td>
</tr>
<tr>
<td>longer a proposal.</td>
<td></td>
</tr>
<tr>
<td>Cleaned up language in</td>
<td></td>
</tr>
<tr>
<td>section 2.3. No change</td>
<td></td>
</tr>
<tr>
<td>to meaning of</td>
<td></td>
</tr>
<tr>
<td>requirements.</td>
<td></td>
</tr>
<tr>
<td>Eliminated rule 2.3.4.3</td>
<td></td>
</tr>
<tr>
<td>Use internal schema</td>
<td></td>
</tr>
<tr>
<td>version attribute for</td>
<td></td>
</tr>
<tr>
<td>version of schema</td>
<td></td>
</tr>
<tr>
<td>Changed rule 2.3.5.1</td>
<td></td>
</tr>
<tr>
<td>to be more clear</td>
<td></td>
</tr>
<tr>
<td>(Designation in</td>
<td></td>
</tr>
<tr>
<td>Business Documents)</td>
<td></td>
</tr>
<tr>
<td>Rewrote common library</td>
<td></td>
</tr>
<tr>
<td>rules (2.4.1.1 and</td>
<td></td>
</tr>
<tr>
<td>2.4.1.2) to be</td>
<td></td>
</tr>
<tr>
<td>clearer and more</td>
<td></td>
</tr>
<tr>
<td>concise. Deleted rule</td>
<td></td>
</tr>
<tr>
<td>2.4.1.3.</td>
<td></td>
</tr>
<tr>
<td>Modified language in</td>
<td></td>
</tr>
<tr>
<td>section on unqualified</td>
<td></td>
</tr>
<tr>
<td>local elements (2.4.4)</td>
<td></td>
</tr>
<tr>
<td>and type names (2.5.11)</td>
<td></td>
</tr>
<tr>
<td>from “will” to “should”</td>
<td></td>
</tr>
<tr>
<td>to be compatible with</td>
<td></td>
</tr>
<tr>
<td>how some committees</td>
<td></td>
</tr>
<tr>
<td>have designed schema.</td>
<td></td>
</tr>
<tr>
<td>Eliminated section 2.4.7</td>
<td></td>
</tr>
<tr>
<td>on naxml common library</td>
<td></td>
</tr>
<tr>
<td>schemas.</td>
<td></td>
</tr>
<tr>
<td>Rewrote section on</td>
<td></td>
</tr>
<tr>
<td>default namespaces (2.5.3.1)</td>
<td></td>
</tr>
<tr>
<td>Eliminated section on</td>
<td></td>
</tr>
<tr>
<td>distribution of source</td>
<td></td>
</tr>
<tr>
<td>files (2.5.4.2)</td>
<td></td>
</tr>
<tr>
<td>Rewrote rule on</td>
<td></td>
</tr>
<tr>
<td>annotation requirements (2.5.5.1).</td>
<td></td>
</tr>
<tr>
<td>Rewrote section 2.5.6 to</td>
<td></td>
</tr>
<tr>
<td>address encoding, BOM and</td>
<td></td>
</tr>
<tr>
<td>declaration.</td>
<td></td>
</tr>
<tr>
<td>Added section on</td>
<td></td>
</tr>
<tr>
<td>proprietary extensions (2.5.14)</td>
<td></td>
</tr>
<tr>
<td>Eliminated section on</td>
<td></td>
</tr>
<tr>
<td>xs:redefine (2.5.15.3)</td>
<td></td>
</tr>
<tr>
<td>Rewrote section on</td>
<td></td>
</tr>
<tr>
<td>use of nillability (now section 2.5.17)</td>
<td></td>
</tr>
<tr>
<td>Reformatted with new PCATS styles for documents,</td>
<td></td>
</tr>
<tr>
<td>Deleted part of section on absolute path attribute (now section 2.5.18.2)</td>
<td></td>
</tr>
<tr>
<td>Modified section on Boolean values (2.5.23) to allow for other meaningful values such as yes/no</td>
<td></td>
</tr>
<tr>
<td>Added the character encoding reference to Appendix A.</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Draft</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
</tr>
<tr>
<td>November 17, 2008</td>
<td>Draft 0.5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>July 30, 2007</td>
<td>Draft 0.4</td>
</tr>
<tr>
<td>January 3, 2007</td>
<td>Draft 0.3</td>
</tr>
<tr>
<td>August 18, 2006</td>
<td>Draft 0.2</td>
</tr>
<tr>
<td>August 18, 2006</td>
<td>Draft 0.1</td>
</tr>
</tbody>
</table>
Copyright Statement

Copyright © CONEXXUS, INC. 2006-2015, All Rights Reserved.

Conexxus members may use this document for purposes consistent with the adoption of the Conexxus Standard; however, Conexxus must pre-approve any inconsistent uses in writing.

Conexxus recognizes that a Member may wish to create a derivative work that comments on, or otherwise explains or assists in implementation, including citing or referring to the standard, specification, protocol, schema, or guideline, in whole or in part. The member may do so, but may share such derivative work ONLY with another Conexxus Member who possesses appropriate document rights (i.e., Gold or Silver Members) or with a direct contractor who is responsible for implementing the standard for the Member. In so doing, a Conexxus Member should require its development partners to download Conexxus documents and schemas directly from the Conexxus website. A Conexxus Member may not furnish this document in any form, along with any derivative works, to non-members of Conexxus or to Conexxus Members who do not possess document rights (i.e., Bronze Members) or who are not direct contractors of the Member. A Member may demonstrate its Conexxus membership at a level that includes document rights by presenting an unexpired digitally signed Conexxus membership certificate.

This document may not be modified in any way, including removal of the copyright notice or references to Conexxus. However, a Member has the right to make draft changes to schema for trial use, before submission to Conexxus for consideration to be included in the existing standard. Translations of this document into languages other than English shall continue to reflect the Conexxus copyright notice.

The limited permissions granted above are perpetual and will not be revoked by Conexxus, Inc. or its successors or assigns, except in the circumstance where an entity, who is no longer a member in good standing but who rightfully obtained Conexxus Standards as a former member, is acquired by a non-member entity. In such circumstances, Conexxus may revoke the grant of limited permissions or require the acquiring entity to establish rightful access to Conexxus Standards through membership.
Disclaimers

Conexxus makes no warranty, express or implied, about, nor does it assume any legal liability or responsibility for, the accuracy, completeness, or usefulness of any information, product, or process described in these materials. Although Conexxus uses reasonable best efforts to ensure this work product is free of any third party intellectual property rights (IPR) encumbrances, it cannot guarantee that such IPR does not exist now or in the future. Conexxus further notifies all users of this standard that their individual method of implementation may result in infringement of the IPR of others. Accordingly, each user is encouraged to carefully review its implementation of this standard and obtain appropriate licenses where needed.
# Table of Contents

1 Introduction .......................................................................................................................... 10
  1.1 Audience ......................................................................................................................... 10
  1.2 Guide to the Document ................................................................................................. 10
2 Architectural Principles ...................................................................................................... 10
  2.1 Design Objectives ........................................................................................................... 10
    2.1.1 Overall Schema Design ......................................................................................... 11
    2.1.2 Commercial Messages ......................................................................................... 11
  2.2 Versioning ....................................................................................................................... 11
    2.2.1 Backward Compatibility ....................................................................................... 11
    2.2.2 Forward Compatibility ......................................................................................... 12
  2.3 Version Numbering ......................................................................................................... 12
    2.3.2 Examples of Changes that can be incorporated in a Minor Version ................. 13
    2.3.3 Examples of Changes that Necessitate a Major Version ..................................... 13
    2.3.4 Reflecting the Version Numbers in the Schema ................................................. 13
    2.3.5 Reflecting the Version Numbers in the Business Document ............................ 14
  2.4 The Common Library .................................................................................................... 15
    2.4.1 Designing the Common Library ............................................................................. 15
    2.4.2 Guidelines for Structuring Schema Documents .................................................. 16
    2.4.3 Namespaces and Versioning ................................................................................. 16
    2.4.4 Qualified Local Elements ..................................................................................... 16
    2.4.5 Code list Management ......................................................................................... 16
    2.4.6 Proposed Code list Management Approach ....................................................... 17
  2.5 Schema Rules ................................................................................................................ 18
    2.5.1 Namespace Specification ....................................................................................... 18
    2.5.2 Prefix for XML Schema namespace ..................................................................... 21
    2.5.3 Default Namespace ............................................................................................... 21
    2.5.4 File Naming Convention ...................................................................................... 21
    2.5.5 Documentation ...................................................................................................... 23
    2.5.6 Prologue and Encoding Declaration ..................................................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.7</td>
<td>Use of Legal Characters in XML</td>
<td>23</td>
</tr>
<tr>
<td>2.5.8</td>
<td>Naming Conventions</td>
<td>24</td>
</tr>
<tr>
<td>2.5.9</td>
<td>Element Tag Names</td>
<td>24</td>
</tr>
<tr>
<td>2.5.10</td>
<td>Attribute Names</td>
<td>24</td>
</tr>
<tr>
<td>2.5.11</td>
<td>Type Names</td>
<td>24</td>
</tr>
<tr>
<td>2.5.12</td>
<td>Enumeration</td>
<td>25</td>
</tr>
<tr>
<td>2.5.13</td>
<td>Acronyms</td>
<td>25</td>
</tr>
<tr>
<td>2.5.14</td>
<td>Proprietary Extensions</td>
<td>25</td>
</tr>
<tr>
<td>2.5.15</td>
<td>Use of Content</td>
<td>28</td>
</tr>
<tr>
<td>2.5.16</td>
<td>Reusing Schemas</td>
<td>28</td>
</tr>
<tr>
<td>2.5.17</td>
<td>Use of Nillability</td>
<td>28</td>
</tr>
<tr>
<td>2.5.18</td>
<td>Referencing Schemas from XML Instance Documents</td>
<td>29</td>
</tr>
<tr>
<td>2.5.19</td>
<td>Compositors: sequence, choice, all</td>
<td>29</td>
</tr>
<tr>
<td>2.5.20</td>
<td>Named Model Groups</td>
<td>30</td>
</tr>
<tr>
<td>2.5.21</td>
<td>Elements</td>
<td>30</td>
</tr>
<tr>
<td>2.5.22</td>
<td>Attributes</td>
<td>30</td>
</tr>
<tr>
<td>2.5.23</td>
<td>Boolean values</td>
<td>30</td>
</tr>
<tr>
<td>2.5.24</td>
<td>Numeric values</td>
<td>31</td>
</tr>
<tr>
<td>2.5.25</td>
<td>String values</td>
<td>31</td>
</tr>
<tr>
<td>A.</td>
<td>References</td>
<td>32</td>
</tr>
<tr>
<td>B.</td>
<td>Glossary</td>
<td>33</td>
</tr>
</tbody>
</table>
1  Introduction

1.1  Audience
The intended audiences of this document include, non-exhaustively:

- Architects and developers designing, developing, or documenting Conexxus schemas.
- Standards architects and analysts developing specifications that make use of Conexxus schemas.

1.2  Guide to the Document
New readers are encouraged to read this guideline in its entirety. The guideline is intended to provide prescriptive guidance that all designers and implementers should follow.

Section 2 describes the architectural principles of the guideline. It provides the following:

- An Introduction to the design objectives and development methodology underlying the Conexxus-NAXML schemas.
- The actual design details, including sub-sections on reusable components, handling of code lists, handling of contexts and their namespaces and versioning mechanisms.

There are two appendices in this guideline:

- A list of references, both normative and non-normative, that are mentioned or used by this guideline.
- A glossary providing definitions of terms that are used throughout the guideline. Terms that are defined in the glossary are marked in bold at their first occurrence in the document.

2  Architectural Principles

2.1  Design Objectives
Design objectives of the Conexxus Schema Library include:

- Maximizing schema component reuse
- Facilitating the use of schema code generators for building XML data binding classes
- Providing consistent naming conventions for elements of a common nature (currency, counts, volumes, etc.)
2.1.1 Overall Schema Design

All schema and instance documents MUST be based on the W3C suite of technical specifications holding recommendation status.

2.1.2 Commercial Messages
All commercial messages in XML and XSD documents SHALL be removed. For example remove any messages similar to:

“Edited by <owner> with <xml editor> V2.0”.

2.2 Versioning
Versioning of Conexxus-NAXML schemas SHALL NOT be tightly coupled with the publication of Conexxus-NAXML Business Document Exchange Standards. This means that all schemas including business document schemas and common library schemas SHALL NOT be mandated to hold the same version number.

An analysis of UBL, RosettaNet and GS1 XML Naming and Design Rules discloses that an optimal approach is the separation of namespace versions from release naming. In addition, the RosettaNet practice of using namespaces to reflect only the major versions should be followed. In the next section, we will resolve the following issues with versioning of Conexxus-NAXML:

- What constitutes a major and a minor version?
- What do we mean by compatibility?
- Do we need to provide for backward/forward compatibility between versions?

2.2.1 Backward Compatibility
Definition: A given schema is backwardly compatible with a prior schema if no document valid under the prior schema is invalid under the later schema.

2.2.1.1 Rule – Backward Compatibility
Conexxus-NAXML schemas SHALL support backward compatibility as specified in section 2.3.
2.2.2 Forward Compatibility
Definition: The ability to design schemas such that even the older schema can validate
the instance documents created according to the newer version is called forward
compatibility.

2.2.2.1 Rule – No Forward Compatibility
Conexxus-NAXML schemas SHALL NOT support forward compatibility.

2.3 Version Numbering
Conexxus standards SHALL be version numbered as follows:

- M.m.r
- Where M indicates the major release version, m indicates the minor release
  version, and r indicates a point release version.
- Major versions contain substantial changes to architectural and/or core
  components where backward compatibility is not a constraint.
- Minor versions contain updates where backward compatibility must be
  preserved.
- Revisions correct errata and annotations and must maintain backward
  compatibility.

2.3.1.1 Rule – Minor Versions are Backwardly Compatible
All minor versions of a schema within a major version MUST be backwardly compatible
with the preceding minor versions for same major version, and with the major version
itself.

2.3.1.2 Rule – Revisions are Backwardly Compatible
All revisions of a schema within a minor version MUST be backwardly compatible.

2.3.1.3 Rule – All Schemas within a Business Process have same
version
To ease the ongoing maintenance of Conexxus-NAXML schema versioning, all schemas
within a Business Process (e.g. the POS-Back Office specification) MUST have the same
version.

This means that if one schema within a suite of XML schemas that come under a
particular business process needs to be upgraded to the next version number, all the
schemas within that business process MUST be upgraded to that version number.
2.3.2 Examples of Changes that can be incorporated in a Minor Version

- Adding new optional elements or optional attributes.
- Changing attributes from mandatory to optional.
- Changing element cardinality from old schema [0..1] to new schema [0..*].
- Changing element cardinality from old schema [1..1] to new schema [1..*].

Note – changes to element cardinality carry a risk of allowing content changes that will break a consuming application, even though the documents conform to the rule regarding backward compatibility.

- Adding a term to an enumerated list.
- Changing a fixed value to an unfixed value.

2.3.3 Examples of Changes that Necessitate a Major Version

- Changing an attribute from optional to mandatory.
- Adding a mandatory element.
- Eliminating an optional element.
- Eliminating a mandatory element.
- Eliminating an optional attribute.
- Eliminating a mandatory attribute.
- Changing an attribute or element tag name.
- Changing element cardinality from old schema [0..*] to new schema [0..1].
- Changing element cardinality from old schema [1..*] to new schema [1..1].
- Changing the sequence of elements in a <xs:sequence> tag.
- Removing a term from an enumerated list.
- Changing an unfixed value to fixed.
- Changing a fixed value.

2.3.4 Reflecting the Version Numbers in the Schema

2.3.4.1 Rule – Versions will be Represented Using Numeric Characters

- Major, minor and revision numbers will be represented using numeric characters only. The complete representation of the version will be of the format

  Majorversion.Minorversion.Revision (1.5.1)

- The first release of a major version will be numbered M.0.
- The first minor version of a given major version will be numbered M.1
- The first release of a minor version will be numbered M.m, instead of M.m.0.
- The first revision of a minor version will be numbered M.m.1.
2.3.4.2 Rule – Major Version Number Reflected in Namespace

Only the major version number of the schema will be reflected in the namespace.

The rationale behind this decision stems from the fact that minor versions need to be backwardly compatible. If the minor versions were reflected in the namespace of a schema, instance documents would not validate unless they were changed to designate the new target namespace. In addition, any schemas that ‘include’ this schema would have to change because the target namespace of the included components must be the same as the target namespace of the including schema.

Example: urn:conexxus.naxml:lottery:activity1

The chosen approach to indicating the major version number is to simply change the number in the version attribute at the start of the XML schema. There are many advantages to this approach. It’s easy to use since it’s a part of the schema specification. The Instance documents would not have to change if they remain valid with the new version of the schema, which is our backward compatibility requirement. The only disadvantage is that the validator ignores the version attribute, which means that it is not an enforceable constraint. This lack of enforcement is partly mitigated by the fact that the major version is reflected in the namespace.

2.3.5 Reflecting the Version Numbers in the Business Document

The schema document uses the internal version attribute to reflect the complete (major & minor) version of the schema. Since, the major version is reflected in the namespace, the Business Document (xml instance document) will contain the major version by default. However, we also need to ensure that the complete version is specified in the instance document. This is essential because if the Business Document does not contain the minor version information and we do not support for forward compatibility between two major versions then it would be left up to the application systems that receive the Business Documents to know what version all the trading partners are on.

2.3.5.1 Rule – Complete Version Designation in Business Documents

Conexxus-NAXML Business Documents SHALL use the abstract complex type “DocumentType” to reflect the complete version.
This class is defined as:

```xml
<xs:complexType name="DocumentType" abstract="true">
  <xs:attribute name="version" use="required"/>
  <xs:attribute name="release" type="nax:pointRelease"/>
</xs:complexType>
```

### 2.4 The Common Library

The common library consists of schema components that are used in two or more Business Documents. Placing shared components in a common library increases interoperability and simplifies schema maintenance. However, it can also result in some additional complexities, which are addressed in this chapter.

#### 2.4.1 Designing the Common Library

Specifically, these areas need to be addressed:

1. Structuring the schema documents: breaking down the schema documents into smaller units to avoid the inclusion of document structures not required for a given specification.
2. Namespaces and versioning: creating one or more separate namespaces, which will address the lack of a separate life cycle.
3. Configuration management: determining a mechanism for storing, managing and distributing the schemas.
4. Structuring the schema documents involves deciding how large each schema document should be, and which components should be included together in a single schema document.
5. The approach chosen for Conexxus-NAXML documents is to include element declarations for those elements that are shared across multiple Conexxus specifications in shared schemas, commonly called "dictionaries". Code list enumerations and other shared data may also be defined in separate shared schema documents.

#### 2.4.1.1 Rule – Elements shared by two or more specifications MUST be defined in a shared common schema component

#### 2.4.1.2 Rule – Elements shared by two or more components within a specification MUST be defined in a shared schema component
2.4.2 Guidelines for Structuring Schema Documents
Some components are more likely to change than others. Specifically, code list types tend to change frequently and depend on context. For this reason, code list types SHOULD be separated from complex types that validate the structure of the document.

2.4.3 Namespaces and Versioning
Several different namespaces are used in the common library. First, one namespace is assigned for the context-less components, and then common components that are related to a specific business process have that context in their namespaces. Refer to the section on context for more details.

2.4.3.1 Rule – Common Library Version Changes Require Version Changes to Business Documents
The individual files that constitute the common library can have minor versions, with backward compatible changes. However, when the common library has a major version change, all business documents that use the library MUST be upgraded.

2.4.4 Qualified Local Elements

2.4.4.1 Rule – Conexxus XML schema MUST use elementFormDefault= “qualified”

2.4.5 Code list Management
Third-party code lists used within the Conexxus-NAXML schemas SHOULD be defined as enumerated types in individual schema files, and assigned to a namespace other than the NAXML specification namespace. Where the third party in question has defined an XML namespace URI for use with their code lists, that namespace should be used. Otherwise a namespace URI based on the third party’s Web URL should be used.

2.4.5.1 Rule – New Code List Enumerations May Be Added In a Minor Version
Since the addition of a new enumerated value to an existing enumeration is backward-compatible with documents valid under the previous version of the code list, the addition of new code list values MAY be included in a minor version of a given Conexxus schema.

2.4.5.2 Rule – Code List Enumerations May Only Be Removed In a Major Version
The removal of an enumerated value from a code list enumeration breaks backward compatibility, and MUST therefore occur in major versions only.
2.4.6 Proposed Code list Management Approach

As with current Conexxus-NAXML XML schemas, we continue to use xs:enumerations or the enumerated list approach to defining code lists in XML.

Some code lists, particularly those defined by third parties, are defined in individual schema files. The code list schema document is then imported or included in a parent schema document.

2.4.6.1 Rule – Addition of Code Values to a Code List Constitutes a Minor Version Change

Addition of code values constitutes a minor version change.

Any changes to or deletions of existing code values constitutes a major version change.

For Conexxus-NAXML the semantics of the code list content will be defined in the Conexxus Data Dictionary. The context in which the code list is used in Conexxus-NAXML schemas is identified by the namespace of the schema document that uses the code list schema.

As far as upgradeability is concerned, the versioning mechanism discussed in the earlier section lets us add values to a code list without having to change the major version of a schema.

Example Depicting Code list Management

The schema below shows an example of code list management that is extensible:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:conexxus.naxml:BP:v3"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:nax="urn:conexxus.naxml:BP:v3"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<!-- test element for test target document -->
<xs:element name="testDocument">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="testElement" maxOccurs="unbounded"
type="nax:itemTypeCode"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- code4 element from the guidelines -->
<xs:simpleType name="code4">
  <xs:restriction base="xs:string">
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>
```
New code values could be added to the enumeration. Since this is a minor change, the
namespace will remain the same.

```
<xs:simpleType name="itemTypeCodeDefined">
  <xs:restriction base="nax:code4">
    <xs:enumeration value="carwash"/>
    <xs:enumeration value="fees"/>  
    <xs:enumeration value="fuel"/>
    <xs:enumeration value="gift"/>
    <xs:enumeration value="list"/>
    <xs:enumeration value="lottery"/>
    <xs:enumeration value="lotto"/>
    <xs:enumeration value="misc"/>
    <xs:enumeration value="mord"/>
    <xs:enumeration value="mrh"/>
    <xs:enumeration value="phon"/>
  </xs:restriction>
</xs:simpleType>`
```

### 2.5 Schema Rules

Conexxus schemas are created using a specific set of rules to ensure uniformity of
definition and usage.

#### 2.5.1 Namespace Specification

Namespaces provide context for schema documents. This allows the same name to be
used in schema documents without a name conflict.
Conexxus-NAXML namespaces MUST be formally specified so that they can be used unambiguously. Any conflict with external namespaces or URIs must be addressed to ensure unambiguous and reliable resolution of resources.

2.5.1.1 Rule - Conexxus-NAXML namespaces MUST be Uniform Resource Identifiers (URI)

Design Guideline:

- Every reusable XML resource that can have an independent lifecycle must have a namespace declaration URN (URI).
- Every XML Schema must have a target namespace.
- The XML Schema namespace URI should include its major version number.
- Every element and attribute from another organization’s XML namespace used in a Conexxus schema shall be explicitly associated with that third-party namespace.

2.5.1.2 Rule - All Conexxus namespace URIs SHALL employ lowercase letters only, and SHALL otherwise conform to the generic syntax for URIs as published in [IETF STD 66]


Rule – All Conexxus schemas must extend from a base URI

All Conexxus defined schemas MUST be defined within a namespace. Each namespace has a base URI for its root. The Conexxus base URI is [http://www.conexxus.org/schema](http://www.conexxus.org/schema). Schemas within other namespaces will extend from this base URI. For example, the base URI for the naxml related schemas is [http://www.conexxus.org/schema/naxml](http://www.conexxus.org/schema/naxml).

In order to facilitate a library of schemas, the schema designer SHOULD define a hierarchy of nested namespaces. This hierarchy will occur within the appropriate Conexxus root namespace. For example, the root URI for the naxml library will have an additional hierarchy of related namespaces. As an example, the POS/Back Office Interface (PBI) specification defined as a subset of naxml would have a namespace of [http://www.conexxus.org/schema/naxml/pbi](http://www.conexxus.org/schema/naxml/pbi).
2.5.1.3 Rule – Namespaces Reflect Context and Major Version of Schema Components

2.5.1.4 Common Namespace
In order to share common types across all schemas, the namespace hierarchy defines a common schema namespace that can be imported by other schemas. These schemas define core data types and other elements that are common across many Conexxus schemas. The namespace is

http://www.conexxus.org/schema/common

Examples of schemas defined within the common namespace are

- conexxusDictionary.xsd
- conexxusBaseTypes.xsd

2.5.1.5 Open Site Architecture (OSA) Namespace
Schemas that are defined to work within the Conexxus Open Site Architecture reside within the OSA namespace:

http://www.conexxus.org/schema/osa

The OSA namespace has individual working group namespaces. Each working group namespace might have one or more separate namespaces depending on the complexity and type of services defined. The base hierarchy is as follows:

http://www.conexxus.org/schema/osa
http://www.conexxus.org/schema/osa/fp/vXX
http://www.conexxus.org/schema/osa/sign/vXX
http://www.conexxus.org/schema/osa/atg/vXX
http://www.conexxus.org/schema/osa/safe/vXX

2.5.1.6 NAXML Namespace
The NAXML namespace covers all of the schemas that Conexxus defines for EB2B, Retail Financial Transactions (other than EPS), and POS/Back Office data sharing. This URI is

http://www.conexxus.org/schema/naxml

The current hierarchy is as follows:

http://www.conexxus.org/schema/naxml
2.5.1.7 Electronic Payment Server (EPS) Namespace
Because the EPS specification is shared with the International Forecourt Standards Forum (IFSF), the EPS namespace is defined by IFSF and is as follows:

http://www.nrf-arts.org/IXRetail/namespca

2.5.2 Prefix for XML Schema namespace
The World Wide Web Consortium (W3C) uses the URL http://www.w3.org/2001/XMLSchema as the namespace for the XML Schema Language. All the existing schema modules in Conexxus-NAXML suite of standards use xs: as the prefix for this namespace.

2.5.2.1 Rule – Schemas Use xs: as Prefix for XML Schema Namespace
Conexxus-NAXML schemas MUST use xs: as the prefix for the XML Schema namespace.

2.5.3 Default Namespace

2.5.3.1 Rule – No Default Namespace in Schema Documents
Conexxus-NAXML XML schemas MUST NOT use the default namespace. This gives schemas a consistent layout throughout libraries. In addition, it provides for the use of component libraries that have multiple namespaces.

2.5.3.2 Rule – Namespace Names SHOULD Reflect the Intended Use of the Schema
It is essential that Conexxus-NAXML schemas use namespace URIs that reflect the intended use of the schema. For example, http://www.naxml.org/schema/POSBO/V3 to indicate the intended use of the schema is for POS/Back Office interfaces.

2.5.4 File Naming Convention
Conexxus-NAXML will apply the following naming convention to all Conexxus-NAXML XML schemas.
2.5.4.1 Rule – Schemas Named According to Functional Purpose of Schema

Conexxus -NAXML schema files SHALL be named according to the functional aspect of the schema itself. A top level functional area may be chosen from:

- NAXML
- OSA – Open Site Architecture

A three character code will identify the functional areas as follows:

- LOT – Lottery
- LOY – Loyalty
- MTF – Motorfuel
- PBI – POS/Back Office
- RFT – Retail Financial Transactions
- B2B – Electronic Business to Business
- FPS – Fueling Point
- SGN – Electronic Sign
- ATG – Automatic Tank Gauge
- SAF – Electronic Safe
- EPS – Electronic Payment Server

Conexxus-NAXML schema documents will be given a name reflecting the business nomenclature of that particular message. For example, a Purchase Order schema document will be named “NAXML_B2B_PurchaseOrder.xsd”. Schemas serving as a library of information components will be given a name that reflects the functional aspect of the library. A library of components used to describe base type information may be given the name of NAXML-BaseTypes.xsd. Code list definition schemas will be given the exact name of the code list defined within the schema itself (i.e. a country code list schema will be named “NAXML-CountryCodes.xsd”).

To make Conexxus -NAXML schema documents easier to find and to use, to avoid naming conflicts between contexts, and to control access, they shall be bundled into logical groups of related schemas that share the same context and thus the same namespace. These logical groups of schemas will be identified with their corresponding context identifier/namespace and shall follow the rules of the context category hierarchy. This convention will have a direct impact on how all Conexxus -NAXML source files will be distributed and represented on a file system. That is, the path qualifier for the source files (schemas) in a file system will be directly correlated to the context category hierarchy.
2.5.5 Documentation

2.5.5.1 Rule – Annotation Requirements

- Every enumeration SHOULD have an annotation.
- Every simple or complex type defined in the Conexxus schema documents SHOULD have an annotation.
- Every element and attribute, including the root element, defined in the Conexxus schema documents SHOULD have an annotation.

2.5.5.2 Rule – Schema Annotations in English

All Schema annotations shall be in English language text.

2.5.6 Prologue and Encoding Declaration

All Conexxus schema documents and Conexxus sample XML instance documents SHALL conform to the W3C XML 1.0 Recommendation with respect to character encoding and the use of Byte Order Marks (BOM).

2.5.6.1 Rule – Character Set Encoding to be UTF-8

All Conexxus schema MUST employ UTF-8 encoding. Conexxus instance documents SHOULD employ UTF-8 encoding, unless business requirements dictate otherwise. If UTF-8 encoding is not used, interoperability between trading partners may be compromised and must be independently evaluated by the trading partners involved.

2.5.6.2 Rule - Byte Order Mark

All Conexxus schema MUST NOT use a byte order mark (BOM). Conexxus instance documents SHOULD NOT use a byte order mark. However, if business requirements dictate the use of an encoding method other than UTF-8, the instance document SHALL conform to the W3C XML 1.0 Recommendation governing the use of BOM with the encoding method specified for the instance document. See http://www.w3.org/TR/Rec-xml/#charencoding for more details.

2.5.6.3 Rule – Encoding Declaration

All Conexxus schema and instance documents MUST begin with a text declaration containing an encoding declaration, e.g.

```xml
<?xml encoding='UTF-8'>.
```

2.5.7 Use of Legal Characters in XML

Conexxus schema documents SHALL use the character set specified by the XML recommendation for the version of XML indicated in the schema document XML processing instruction.
2.5.8 Naming Conventions
Element, attribute and type names MUST be in the English language.

2.5.9 Element Tag Names

2.5.9.1 Rule – Element Tag Names Use Upper Camel Case
Element Tag names are based on the Business Terms defined in the Conexxus Data Dictionary. For element tag names, the Upper Camel Case (‘UCC’) convention MUST be used, i.e. the leading character in the first word should be in upper case and the leading characters of each subsequent word in upper case. The remainder of each word is lower case, without using hyphens or underscores between words.

2.5.10 Attribute Names

2.5.10.1 Rule – Attribute Names Use Lower Camel Case
For element attribute names, the lower Camel Case (‘LCC’) convention MUST be used. Attribute Tag names are based on Business Terms defined in the Conexxus Data Dictionary. The first word in an element name will begin with a lower case letter with subsequent words beginning with a capital letter without using hyphens or underscores between words.

2.5.11 Type Names

2.5.11.1 Rule - Type Names Use Upper Camel Case
For type names, the Upper Camel Case (‘UCC’) convention SHOULD be used, i.e. the leading character of each word is in upper case while the remainder of each word is in lower case without using hyphens or underscores between words.

2.5.11.2 Rule - Type Names have ‘Type’ or ‘Abstract’ Suffix
All type names SHOULD have ‘Type’ or ‘Abstract’ appended to the name. ‘Type’ is used for those complex or simple types that can be used directly. ‘Abstract’ is used for those complex or simple types that are considered abstract and must be extended.

2.5.11.3 Rule - In XML instance files and in XML schemas LCC or UCC convention SHOULD be applied to the Dictionary Entry Name and any white space should be removed
<xs:simpleType name="ShipPartyIdentificationListType">
<xs:complexType name="ShipPartyIdentificationType">
2.5.11.4 Rule - Attribute Groups MAY be used.

2.5.12 Enumeration

2.5.12.1 Rule - For enumeration values the Lower Camel Case (‘LCC’) convention MUST be used.

2.5.12.2 Rule – Enumerations imported from other dictionaries (i.e. states) MAY be used without modification.

2.5.12.3 Rule - Enumerations Names have ‘Enum’
All enumeration element names MUST have ‘Enum’ or ‘Abstract’ appended to the name.

2.5.13 Acronyms

2.5.13.1 Rule - Acronyms are defined in the Conexxus Data Dictionary. Acronyms SHOULD be written using uppercase. Word abbreviations should be avoided.
When this rule conflicts with another rule that specifically calls for LCC or UCC, that rule requiring LCC or UCC SHALL override.

2.5.14 Proprietary Extensions
Extension points are provided in the schema so that vendors can include additional proprietary data in their documents, and those documents can then be validated and also used with other XML tools. In order to use an extension, the following preferred method should be used:

1. Create a new, proprietary schema document that targets a schema document in the specification. This document SHOULD include all the extensions defined for various elements in the target document. It SHOULD NOT include extensions destined for other targets – one target document at a time is the preferred method.

2. The proprietary schema document SHOULD use an <xs:import> statement to reference the target schema document. The proprietary schema document SHOULD also have a different “targetNamespace” from the target of the extension. This arrangement will help prevent naming conflicts with the specification.

3. The proprietary schema MAY define new elements (in the existing namespace) to be used in the extensions. These SHOULD be defined within the context of the
extension element (to avoid element name conflicts) but they MAY be defined in the namespace, provided care is given to define the names such that they are unlikely to conflict with future versions of the specification.

(Note: if the schema document uses elementFormDefault="qualified", the namespace qualifier must be used. The scope of the element will prevent a conflict.)

4. In the proprietary schema, for each extension point you want to extend:
   a. Create an appropriately named proprietary element declaration that will appear in the instance document at the extension point. This element MUST be in the substitution group of the desired extension point. This element SHOULD be named so that the extension point it addresses is easily recognized.
   b. Create a complex type definition derived (by extension) from the “nax:AbstractExtensionType”. This extension MAY be locally defined within the extension element from 4a (as in the accompanying example).
   c. Associate this complex type with the element declared in 4a, either by reference (type="xxx") or by declaring it locally within the element declaration.

5. To validate an XML document containing extensions:
   a. Make sure that the elements created in 4 are in scope during validation.
   b. The namespace(s) for both the specification and for the proprietary schema must be included in the XML document at the appropriate point, normally in the root element, but optionally in the extension element(s) themselves.

2.5.14.1 Extension Example
The following extension example uses the MCTDetail element from the POS/BO schema PBIMaintenance. Shown below is the proprietary schema document and a corresponding XML document that it validates. The example follows the rules explained in the previous section.

Proprietary Schema Document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:nax="http://www.naxml.org/POSBO/Vocabulary/2003-10-16"
  xmlns:prv="http://MyPrivateNamespace.org"
  xmlns:html="http://www.w3.org/1999/xhtml"
  targetNamespace="http://MyPrivateNamespace.org"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- target for extension -->
  <xs:import namespace="http://www.naxml.org/POSBO/Vocabulary/2003-10-16">
```
schemaLocation="NAXML-PBIMaintenance34.xsd"/>
<xs:element name="PROP-CouponRestriction" type="xs:positiveInteger"/>
<xs:element name="PROP-MCTDetailExtension"
    substitutionGroup="nax:MCTDetailExtension">
<xs:complexType>
<xs:complexContent>
<xs:extension base="nax:AbstractExtensionType">
<xs:sequence>
<xs:element ref="nax:JobCode"/>
<!-- reuse from the NAXML-PBIDictionary -->
<xs:element ref="nax:JobID"/>
<!-- reuse from the NAXML-PBIDictionary -->
<xs:element ref="prv:PROP-CouponRestriction"/>
<!-- new element declared at global scope -->
<xs:element name="GUID" type="xs:NMTOKEN"/>
<!-- new element declared at local scope -->
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
</xs:schema>

Example XML Document Using the Proprietary Extension Element:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nax:NAXML-MaintenanceRequest version="3.4.4"
xmlns:nax="http://www.naxml.org/POSBO/Vocabulary/2003-10-16"
xmlns:prv="http://MyPrivateNamespace.org">
    <nax:TransmissionHeader>
        <nax:StoreLocationID>4422</nax:StoreLocationID>
        <nax:VendorName>POSBO</nax:VendorName>
        <nax:VendorModelVersion>NAXML3.4.4</nax:VendorModelVersion>
    </nax:TransmissionHeader>
    <nax:MerchandiseCodeMaintenance>
        <nax:TableAction type="initialize"/>
        <nax:RecordAction type="create"/>
        <nax:MCTDetail>
            <nax:MerchandiseCode level="1">004.003.002</nax:MerchandiseCode>
            <nax:ActiveFlag value="yes"/>
            <nax:MerchandiseCodeDescription>Aspirin</nax:MerchandiseCodeDescription>
            <nax:SalesRestriction>
                <nax:SalesRestrictFlag type="forceQuantity"/>
            </nax:SalesRestriction>
        </nax:MCTDetail>
        <prv:PROP-MCTDetailExtension>
            <nax:JobCode>44</nax:JobCode>
            <nax:JobID>521</nax:JobID>
            <prv:PROP-CouponRestriction>01110</prv:PROP-CouponRestriction>
            <prv:GUID>1553:2241:2121</prv:GUID>
        </prv:PROP-MCTDetailExtension>
    </nax:MerchandiseCodeMaintenance>
</nax:NAXML-MaintenanceRequest>
```
2.5.15 Use of Content
There are four types of content for complex types: simple, element-only, mixed and empty.

2.5.15.1 Rule – Mixed content SHOULD be used only for documentation
Mixed content is to be used on a very limited basis, as the character data in mixed content is completely unrestricted. An exception to this rule is the "documentation" element, which is a W3C XSD defined element with mixed content.

Mixed content allows character data as well as child elements.

2.5.16 Reusing Schemas
The following sections describe how to reuse schemas in Conexxus-NAXML using the W3C XML Specification.

2.5.16.1 Rule - The xs:import element MUST contain the schemaLocation attribute that points to the imported schema document(s) via relative paths with respect to the location where the current Schema is stored

2.5.16.2 Rule - The xs:include element MUST contain the schemaLocation attribute that points to the included schema document(s) via relative paths with respect to the location where the current Schema is stored

2.5.16.3 Recommendation – Keep all schemas for a specification in the same folder (i.e., relative path).

2.5.17 Use of Nillability

2.5.17.1 Rule - Nillability MAY be used if appropriate.
There are two cases in which nillability may be useful:

- When the sending system cannot provide a value for a required element, the use of nil for that element may be appropriate, as determined by the schema designers.
- When the sending system must indicate that the value of an optional element has changed from a non-null value to null, the use of nil is appropriate.
2.5.18 Referencing Schemas from XML Instance Documents

2.5.18.1 Rule – XML instance documents MAY use xsi:schemaLocation with an unqualified schema file name. For example, xsi:schemaLocation=http://www.naxml.org/POSBO/v04NAXML-PBIMainenance.xsd

2.5.18.2 Rule - XML Instance documents MUST NOT use a relative or absolute path in the xsi:schemaLocation attribute. The xsi:schemaLocation attribute provides a hint to the processor as to where to find a Schema that declares components for that namespace. The use of an absolute path or a relative path other than the unqualified schema file name is not recommended.

2.5.19 Compositors: sequence, choice, all

2.5.19.1 Rule - While composing groups of elements <xs:sequence> is the preferred compositor, while the use of <xs:all> is NOT RECOMMENDED. The biggest disadvantage of <xs:all> is that it cannot be repeated any further. This limits the use of <xs:all> to the first occurrence of its set of elements. If a content model requires an element that occurs more than once then <xs:all> cannot be used.

2.5.19.2 Rule – If repetition is required, repeating elements SHOULD be used and repeating compositors SHOULD be avoided.

The following example shows both ways:
2.5.20 Named Model Groups

XML schemas allow for grouping and ordering. Ordering is implemented using sequence compositor. Grouping is performed using the `<xs:group>` element. Groups represent a set of element declarations so that they can be incorporated as a group into complex type definitions.

`<xs:group>` MUST be defined globally in order to be able to reuse it within the document. This might not be acceptable in terms of overall design.

2.5.20.1 Rule - For model group names, the Upper Camel Case (`'UCC'`) convention MUST be used

The name MUST be written as group name (in UpperCamelCase) + Group, for example, TextualDescriptionGroup.

`<xs:group>` MAY be used when there is a need to reuse a set of elements when application design requires presentation to be structured. `<xs:group>` provides code reuse whereas type definitions provide definition reuse.

`<xs:group>` SHOULD be created only when you need to group logically related content models.

2.5.21 Elements

As a rule of thumb elements SHOULD be used to convey business information entities, i.e. terms that have a distinct meaning when used in a specific business context. Element names and descriptions SHOULD be chosen to accurately reflect the information provided. For example, a “total” may need to include the word “grand” or “net” in the name to accurately identify the total. Clarification on the meaning or the rationale behind the choice of name could be provided in the annotation.

2.5.22 Attributes

The W3C XML Specification 1.0 states that attributes are used to associate name-value pairs with elements or, in other words, add characteristics to a business information object. Attribute names and descriptions SHOULD be chosen to accurately reflect the information provided. For example, a “total” may need to include the word “grand” or “net” in the name to accurately identify the total. Clarification on the meaning or the rationale behind the choice of name could be provided in the annotation.

2.5.23 Boolean values

Boolean elements and attributes SHOULD use the W3C `<xs:boolean>` data type.
2.5.24 Numeric values

2.5.24.1 Rule - Numeric values in schema SHOULD be defined as positive.
The use of xs:nonNegativeInteger is encouraged but not required. The element name itself should imply the type of value contained so that a positive value makes sense. As an example, a bank Amount element should be defined as either Credit or Debit so that the intended type is explicit.

2.5.24.2 Rule – Conexxus schemas SHALL NOT use unbounded numeric data types without proper constraints
Either the minimum and maximum values or the maximum number of digits for elements and attributes of numeric data types should be specified. Shrinking the boundary conditions for an element or attribute may only be done in a major version. Enlarging the boundary conditions for an element or an attribute may be done in minor or major versions.

2.5.25 String values

2.5.25.1 Rule – Conexxus schemas SHALL NOT use elements or attributes of type xs:string without an accompanying constraint on the overall length of the string.
Shrinking the boundary conditions for an element or attribute may only be done in a major version. Enlarging the boundary conditions for an element or an attribute may be done in minor or major versions.
A. References

A.1 Normative References

XML Schema Part 1: Structures


XML Schema Part 2: Data Types

XML Schema Part 2: Data Types. Available at: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

XML 1.0: Character Encoding in Entities

XML 1.0 Character Encoding in Entities. Available at: http://www.w3.org/TR/Rec-xml/#charencoding

Uniform Resource Identifier (URI): Generic Syntax


A.2 Non-Normative References

None
## B. Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IETF</td>
<td>The Internet Engineering Task Force</td>
</tr>
<tr>
<td>IFSF</td>
<td>International Forecourt Standards Forum</td>
</tr>
<tr>
<td>NACS</td>
<td>National Association of Convenience Stores</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>