Introduction
The chair, Mark Crawford, opened the meeting and welcomed the participants. The following participants were present (P) or participated via conference lines (C) during the week:

<table>
<thead>
<tr>
<th>Participant</th>
<th>Membership Status</th>
<th>Present/Conf Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gunther Stuhec</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Hisanao Sugmata</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Jostein Fromyr (Secretary)</td>
<td>Member</td>
<td>C</td>
</tr>
<tr>
<td>Kim Lambert</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Kirs Ketels</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Luc Mouchot</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Marcel Jemio</td>
<td>Member</td>
<td>C</td>
</tr>
<tr>
<td>Mark Crawford (Chair)</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Melanie Kudela</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Paula Heilig</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Thomas Bikeev</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Sue Probert</td>
<td>Candidate Member</td>
<td>C</td>
</tr>
<tr>
<td>John Duker</td>
<td>Member</td>
<td>P</td>
</tr>
<tr>
<td>Greg Wilson</td>
<td>Candidate Member</td>
<td>C</td>
</tr>
<tr>
<td>James Whittle</td>
<td>Candidate Member</td>
<td>C</td>
</tr>
</tbody>
</table>

It was determined that we have a quorum – we have 13 official members and 10 present. Paula agreed to take minutes in the absence of the secretary.

Agenda
1. Priorities
2. Work methodology – how should we work at this meeting and between meetings
3. Review/discussion of Gunther Struhec's paper on XML definition of CCT’s
4. Standard business document header review
5. Input format

Priorities
It was determined at the meeting in March that the five priorities to complete first are those listed below. At that meeting a champion was assigned to each priority with the intention that work would be done via conference calls between meetings. However, due to a small participation on conference calls, not as
much progress as hoped has been made. It was agreed that the top priority for this meeting is to make a decision on local vs. global and to agree to as many other rules as possible during the week.

Top 5 priorities
1. Structuring (to include containership) –
   a. Local vs. global
      i. Qualified vs. unqualified
   b. Schema Construction
      i. Input Form
      ii. Schema structure assembly and containership
2. Type Declarations [simple types, complex types, data types]
3. Modularity, Namespaces, and Versioning
4. Extension, Restriction, and Reuse
5. Dependency (such as XPath, X Pointer, XLink, XSLT) on other stuff

Global vs. Local
At the March meeting the pros and cons of the Venetian Blind and the Garden of Eden approaches were listed. It was decided at this meeting to use those lists and from them decide which items were only a disadvantage or advantage to either global or local and not to list those which they both have in common.

Prior to the advantage/disadvantage exercise we listed assumptions on which we were basing our approach. These assumptions are:

- Data modeling is part of the TBG process
- All complex types will be based on fully harmonized UN/CEFACT Business Information Entities
- One and only one complex type will be defined for an approved business information entity or core component
- The complex type names must be the BIE dictionary entry name
- Each element name within a complex type will be based on the property term and qualifiers and the representation term of the BBIE or be based on the property term and object class of the child ABIE.

It should also be noted that some of the advantages or disadvantages from the March meeting may also have been dropped as they no longer applied because of the assumptions listed above.

Local
The advantages of local are:

- Binding reflects pure OO serialization without the added complexity of dealing with global elements (.Net, JAVA)
  o Structure aligns to other approaches without any modifications (UML, relational database design, interface designs in different programming languages)
This approach facilitates a more robust use of W3C XML standards

- Highly reusable in different applications - Types can be used for defining and processing in different modeling methodologies, interfaces, development environments, applications without changing the structure or names.

- It allows for shorter tag names through name scoping and establishment and application of easier truncation rules
  - No redundancy of the object class term in the sub-classes (attributes)

- Allows more precise relationships between elements by enforcing hierachical structure
  - Promotes tag reusability by allowing to use the same tag name for different semantic meanings and in different contexts

The disadvantages of local are:

- Element reuse (even if namespace qualified) only occurs through binding the whole type in which it is declared
- It requires all of the assumptions that we’ve made in order to ensure the development of a consistent dictionary

**Note:** The disadvantage that ‘tools must be type aware’ was removed because no one could support that this was true.

**Global**

The advantages of global are:

- Provides for OO-serialization for types
- Elements are semantically and contextually unique without the need to refer to the hierarchical structure
- Elements and types are reusable
  - Elements can be reused directly
- Does not require type aware processors
  - An element in the instance is always of the same type
- Does not require XPath to be able to access the content of the elements
- Makes 1:1 mapping between EDIFACT and XML dictionary at the element level more straightforward
- Fragment processing is uncomplicated

The disadvantages of global are:

- Two semantically identical constructs that have different structures or different characteristics **must** have different tag names
  - Requires more tag names
• Truncation rules, although possible, are more difficult to derive and automate than with local
• Results in larger instance documents that are on average 1.5 times greater than local based instance documents
• Pseudo classes in the OO design had to be used for declared elements in every application – Although a trivial exercise, this leads to a solution that is more complex than that provided by the local element approach.
• Requires a larger DOM-tree which impacts performance

Following this discussion, a motion was made by Paula Heilig and seconded by Kris Ketels that elements be declared locally. Mark Crawford made a motion to amend the original motion, to say that ABIE’s be declared globally and BBIE’s be declared locally. There was no second for this motion.

During discussion of the original motion, Mark Crawford stated that for the record he is opposed because he doesn’t think it allows us to do all that we want to do with XML and that he has a business case to manage both the element and the type and that locally declared elements do not support his business requirement.

Hisanao Sugamata then made a motion to amend the original motion, to say that ABIE’s be declared globally and BBIE’s be declared locally. Mark Crawford seconded this motion. The motion failed with a 3 in favor, 6 opposed votes.

Paula then amended her original motion to add ‘except for the root element and Kris seconded. This motion carried with a 6-2 vote with 1 abstention.

Mark then made the motion that all types be defined globally, Marcel Jemio seconded, and the motion carried unanimously.

Rules
Following the decision on locally defined elements, the group agreed to go through the draft of the UBL Schema Naming and Design Rules Checklist and see if we agreed/disagreed with each rule and/or if changes needed to be made. The list of agreed upon naming and design rules from the March minutes was also referenced during this work. The result of this work can be found at the end of the minutes in Attachment A. Each rule is highlighted with a color and an explanation of the colors can be found at the beginning of the document. The group will continue to work on the rules during the bi-weekly conference calls.

A number of issues still need to be resolved before some of the rules that were deferred can be finalized. These issues include:

• Code lists – do we use the UBL approach to code lists, do we allow enumerations in a schema (rule 29)
• Qualifiers in containers
• Need to address the issue of when the supplemental component is expressed by an existing xsd data type of the content component then
whether the supplementary component should or should not be expressed as an attribute (e.g. date) (rule 53)

- Do we have sufficient rules covering the names of ASBIE elements
- Namespaces – do the context drivers get expressed in the namespaces. Thomas Bikeev agreed to be the champion for namespaces

While working on the rules, it was noted that some of the columns in table 8.1 in the CCTS need to be corrected and also that there does not seem to be any rules for supplementary components in the CCTS. Mark will pass these comments to Alan Stitzer.

**XML Definition of CCT's**
Gunther Struhec reviewed his paper explaining the layout and his recommendation for two new core component types. Due to time constraints the paper was not fully reviewed and further review/discussion will be required.

**Input Formats**
Paula Heilig had distributed the ICG steps and the Business Requirements Specification (BRS) and the Requirements Specification Mapping (RSM) documents and the TBG17 ‘Easy Steps from Requirements to Submission for Harmonisation’ document. The RSM requires only minimal information for submitting BIE’s while the TBG17 document requires that all information for a BIE be submitted. The group agreed that a submitter should not be submitting different information depending on whom they are submitting to and that ATG2 needs more information than is included in the RSM. We will send comments to ICG recommending that the RSM include all the information required in the TBG17 document. Also, will ask the question of how the BRS is linked to the RSM.

ATG2 met with ATG1 to discuss a document they have started on management procedures. This document also speaks to the input format. Their discussions questioned whether there should be different procedures based on what is being requested and also what changes would not require a change to the BRS and can go directly from a TBG group to ATG. Also, we are still awaiting information from TMG on what should be in a class diagram and how do you get the BIE’s from the class diagram.

ATG1 will post their draft management procedures document on the e-room and also send to TBG and ICG for comments before the September FCT meeting.

**Future Conference Calls**
The group will continue to have bi-weekly conference calls on Monday at either 12:00 or 16:00 European Standard Time. However, if there is no participation to warrant the 12:00 time, all calls may be changed to 16:00. The next call will be Monday, June 30 at 12:00 European Standard Time.

**Meeting Adjourned Friday, 20 June 2003, 14:00.**
Attachment A

Schema Naming and Design Rules Checklist

[Editors Note – The ATG2 Position on these rules is based on the following set of assumptions:

- Data modeling is part of the TBG process
- All complex types will be based on fully harmonized UN/CEFACT Business Information Entities
- One and only one complex type will be defined for an approved business information entity or core component
- The complex type names must be the BIE dictionary entry name
- Each element name within a complex type will be based on the property term and qualifiers and the representation term of the BBIE or be based on the property term and object class of the child ABIE.

[Editors Note – The numbers reflect those from the current UBL XSD schema NDR checklist. In instances where ATG has added rules, they have been given an alphabetic suffix to the existing rule so as to maintain direct relationship between the UBL rules and those of ATG. All rules have been color coded (and in some cases multi-color coded) in accordance with the following legend:

- light blue – rule has been adopted by ATG2 and reflects consensus of a quorum of its membership or the results of a vote. These rules include those approved in San Diego and Waldorf meetings.
- Yellow – rule has been discussed by ATG2, there may or may not be general consensus with the rule, and no final decision has yet been reached.
- Green – rule has not yet been discussed by ATG2
- Red – UBL rule has been modified or rejected by ATG2. Disagreed text is lined through. Reason for disagreement can be due to:
  - Disagreement with UBL position
  - Difference in current decision regarding local versus global element declarations
  - Disagreement on the need for the rule as being duplicative of what is already clearly defined in the W3C XSD.


[R 2] All schemata and messages must be based on the W3C suite of technical specifications holding recommendation status]
[R 3] Each dictionary entry name must have one and only one fully qualified XPath (FQXP) for an element or attribute.

[R 4] Element, attribute and type names must be in the English language, using the primary English spellings provided in the Oxford English Dictionary.

[R 5] XML names constructed from dictionary entry names must not include periods, spaces, or other separators.

[R 6] Names must not use acronyms, abbreviations, or other word truncations, with the following list of exceptions.

[R 7] Names must not contain non-letter characters, unless required by language-specific rules.

[R 8] Names must be in singular form unless the concept itself is plural (example: Goods), NOT YET ACCEPTED

[.R 9] Upper-camel-case (UCC) MUST be used for naming elements and types.

[.R 10] Lower-camel-case (LCC) MUST be used for naming attributes.

[.R 11] Every business message must have a single corresponding top-level element, NOT YET ACCEPTED

[.R 12] Every top-level element must be named according to the portion of the business process that it initiates, NOT YET ACCEPTED

[.R 13] For every object class identified in the syntax-neutral model, a complex type definition must be created, NOT YET ACCEPTED

[.R 14] The name of a complex type based on an object class must be the name of the object class, with the separators removed and with the “Details” suffix replaced with “Type”, NOT YET ACCEPTED

[.R 15] An element name in a global element declaration based on an object class must be the name of the object class, with the separators removed and with “Details” removed.

[.R 16] For every complex type definition based on an object class, its content model must be defined such that it reflects each property of the object class as an element declaration, with its cardinality and sequencing within the schema content model determined by the details of the source CC or BIE.

[.R 17] An element name in an element declaration based on a property must be the full dictionary name of the property in the syntax-neutral model, with the separators and object class term removed, and with the property term removed if it is identical or similar to the representation term. Each element name declaration will be based on the property term and qualifiers and the representation term of BIE or be based on the property term and object class of the ASBIE. If there are duplicate words in the property term and representation terms of the source dictionary entry name, then the duplicate words will be removed.

[.R 18] If the object class term would have been helpful in the resulting XML name for clarity or if needed to differentiate the element and allow it to have a different type.
Every element declaration corresponding to a property **must** be bound to a type corresponding to the property’s representation term. Where the representation term corresponds to an object class (aggregate BIE), the complex type corresponding to that object class **must** be used. Where the representation term corresponds to a Core Component Type, the complex type corresponding to that Core Component Type **must** be used. The data type of each leaf element declaration will be the data type of its source BIE.

**[R 19]**

The element representing an ASBIE will be of the complex type corresponding to its ABIE.

**[R 19A]** The element representing an ASBIE will be of the complex type corresponding to its source BBIE.

**[R 20]** A namespace schema module dedicated to defining types corresponding to the Core Component Types **must** be created.

**[R 21]** Each CCT **must** have at least one corresponding unique complex type and simple type, where the element’s content (governed by the `xsd:simpleContent` construct and the CCT’s simple type) represents the content component of the CCT and whose attributes (defined in the complex type) each represent a supplementary component of the CCTs. [NOTE: Rework on the wording]

**[R 21a]** User defined attributes SHOULD NOT be used. If used, User defined attributes Must only convey CCT Supplementary Component Information. (Not yet accepted)

**[R 22]** The complex type name corresponding to a CCT **must** be the CCT name, with the periods and spaces removed.

**[R 23]** The name of the simple type corresponding to the content component of a CCT **must** be the content component name, with the periods and spaces removed.

**[R 24]** The name of the attribute corresponding to a supplementary component **must** be the name of the supplementary component, with the periods and spaces removed. If any terms in the supplementary component is equal to the has the same meaning as the representation term of the CCT, it **must** be removed. If the final field (the “representation term” field) is “Text”, it **must** be removed. If the final field is an approved acronym or abbreviation it must be replaced with the acronym or abbreviation. [NOTE: Rework on the wording]

**[R 25]** Mixed-content elements **should not** be used.

**[R 26]** If used, the ID-attribute must be declared globally. All other attributes must be declared locally. The CCT schema module **may** define a set of one or more common attributes that apply to all elements.

**[R 27]** A common attribute **should** be declared as a global attribute only in cases where the attribute’s meaning is identical no matter what element it is used on, and where the attribute is useful on every element. This rule applies to both external (such as `xml:lang`) and specific global attributes. [NOTE: Revisit on rule 26 and 27 after decision made on use of attribute groups]

**[R 28]** The names of specific global attributes must be based on assigned object class property names, as is done for elements that are properties.
Code Lists Must not be enumerated in the core schema. Code Lists Must be enumerated in a schema module using the code list schema template. [NOTE: We need to clarify how we differentiate internal and external and how we generate internal code lists as user schema module or enumeration inside of the “core” schema]

Every type definition and element declaration must contain a structured set of annotations in following pattern, where the keyword is typically based on the spreadsheet column heading in the syntax-neutral model and the description is typically based on the content of the spreadsheet field:

The following sets of annotations are required in type definitions and element declarations: [NOTE: We want to defer]

**UID**: The unique identifier assigned to the type in the library.

**Name**: The complete name (not the tag name) of the type per the library.

**Object Class**: The Object Class represented by the type.

**Definition**: Documentation of how the type is to be used, written such that it addresses the type’s function as a reusable component.

**Code Lists/Standards**: A list of potential standard code lists or other relevant standards that could provide definition of possible values not formally expressed in the structural definitions.

**Core Component UID**: The UID of the Core Component on which the Type is based.

**Business Process Context**: A valid value describing the Business Process contexts for which this construct has been designed. Default is “In All Contexts”.

**Geopolitical/Region Context**: A valid value describing the Geopolitical/Region contexts for which this construct has been designed. Default is “In All Contexts”.

**Official Constraints Context**: A valid value describing the Official Constraints contexts for which this construct has been designed. Default is “None”.

**Product Context**: A valid value describing the Product contexts for which this construct has been designed. Default is “In All Contexts”.

**Industry Context**: A valid value describing the Industry contexts for which this construct has been designed. Default is “In All Contexts”.

**Role Context**: A valid value describing the Role contexts for which this construct has been designed. Default is “In All Contexts”.

**Supporting Role Context**: A valid value describing the Supporting Role contexts for which this construct has been designed. Default is “In All Contexts”.


**System Capabilities Context**: A valid value describing the Systems Capabilities contexts for which this construct has been designed. Default is “In All Contexts”.

[R 32] The nillable attribute must not be used in any schema. The element declaration of xsi:nil shall not appear in any conforming instance.

[R 33] The top-level element must be globally declared in a root schema.

[R 34] If a definition depends on named constructs found in another namespace, then that other namespace must be imported as a namespace schema module. The referenced constructs must not be directly included as Internal Schema Modules.

[R 35] A namespace may be completely specified within the Root Schema. If For larger namespaces, more schema modules may be defined – call these internal modules. The root schema for that namespace then MUST include those Internal Modules.

[R 36] The namespace names for namespaces must have the following structure while the schemas are at draft status:

urn:uncefact:names:tc::schema:name:major:minor

[R 37] The namespace names for Schemas holding specification status MUST be of the form:

urn:oasis:names:specification::schema:name:major:minor

[R 38] Schema location must include the complete URI which is used to identify schema modules.

[R 39] schema modules must be located under the committee directory:

http://www.oasis-open.org/committees/schema/<schema-mod-name>.xsd

[R 40] Every Schema Major version Must have the URI of:

urn:oasis:names:tc::Order:major-number:0

[R 41] The first minor version release of a Schema Must have the URI of:

urn:oasis:names:tc::Order:major-number:1

[R 42] For Minor version changes, the name of the version construct Must not change (short name not qualified name), unless the intent of the change is to rename the construct.

[R 43] The number scheme must be the major number is a non-negative integer and the minor number is a non-negative integer.

[R 44] The CCT types and Reusable types and their namespace, should have a version.

[R 45] Non-namespaced schema modules, will not have their own versions or namespaces, and thus must only be used within the context of and in conjunction with one specified parent.

[R 46] Each version Must have a namespace.

[R 47] Each minor version must be given a separate namespace.
[R 48] A published namespace MUST never be changed.

[R 49] When the URN changes to reflect a change in the namespace, this change will be reflected in the version number, either major or minor.

[R 50] Minor versioning must be limited to declaring new optional constructs, extending existing constructs and refinements of an optional nature.

[R 51] Changes in minor versions must not break semantic compatibility with prior versions.


[R 53] A Core Component Type without any restriction of the Content Component must be defined by a complexType. The complexType includes a simpleContent group with an extension for all relevant attributes of this Core Component Type. The base type definition of this extension must be based on one of the decided built-in datatypes (see table ###). (NOTE: Need to address the issue that when supplementary component is expressed by the existing XSD dataType than the supplementary component must not be conveyed as an attribute. Needs to be reworded, compare with R 21)

[R 54] The CCT Content Component must be defined as a simpleType.

[R 54A] The Content Component must be based on a XSD built-in datatype. If the Content Component of a Core Component Type is restricted by any kind of facets, this Content Component must be a restriction of a simpleType. – [R 54B] The name of the Content Component simpleType must be name of the Content Component with period, underscores and space removed and the word “Type”.

[R 55] The Core Component Type with the restricted Content Component must refer to the relevant named simpleType.

[R 56] A restricted Supplementary Component (local attribute) within a Core Component Type must have a restriction of its simpleType. The base type definition of the restriction must refer to one of the decided built-in datatypes (see table ###). The restriction itself should have all relevant facets.

[R 57] Basic Core Components and Basic Business Information Entities must be defined as a complexType.

A complexType of a Basic Core Component as well as Basic Business Information Entity without any additional restrictions must be a extension of a simpleContent. The base type definition of the Basic Core Component as well as Basic Business Information Entity must refer to the complexType of the relevant Core Component Type.

[R 58] A complexType of a Basic Core Component as well as Basic Business Information Entity with any additional restrictions must be a restriction of a simpleContent. The base type definition of the restriction must refer to the complexType of the relevant Core Component Type. The element group of the restriction includes all required facets.

[R 59] Where required to changed facets of an existing content component a new data type will be defined.

If a global attribute or a Supplementary Component (local attribute) should be restricted within a Basic Core Component as well as a Basic Business Information
Entity, there must be a restriction of a simpleContent. The base type definition of the restriction must refer to the complexType of the relevant Core Component Type. This restriction includes the attribute or attributes, which should be restricted. The simpleType of each attribute must be a restriction, again. This restriction includes all relevant facets.

[R 60] Where required to remove or modify a supplementary component in an existing type definition, this must be done using the XSD restriction mechanism in the specific BCC or BBIE complexType definition. If a Basic Core Component as well as a Basic Business Information Entity should have one or more restricted Supplementary Components (local attributes) and a restricted Content Component, the simpleContent of the complexType must be a restriction. This base type definition of the restriction must refer to the complexType of the relevant Core Component Type. This restriction must include all facets and restricted attributes. The simpleType of each attribute must be a restriction, too. This restriction should have all relevant facets of each restricted attribute.

[R 61] Libraries and Schemas MUST only use UN/CEFACT approved Core Component Types.

[R 62] The Library should identify and use external standardized code lists rather than develop its own -native code lists.

[R 63] The Library may design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists.

[R 64] If a code list is created, the lists should be globally scoped (designed for reuse and sharing, using named types and named spaced schema modules) rather than locally scoped (not designed for others to use and therefore hidden from their use).

[R 65] For each construct containing a code, the documentation should identify the zero or more code lists that must be minimally supported when the construct is used.

[R 66] Users of the Library may identify any subset they wish from an identified code list for their own trading community conformance requirements.

[R 67] Both standardized and proprietary identifiers within a message are exchanged.

[R 68] For each specific point in time the built in datatype from XML schema (Part 2) must be used. These are xsd:time, xsd:date, xsd:dateTime.

[R 69] The expression of duration requires the use of an additional secondary Representation Term called Duration. Type.

[R 70] For the expression of the Representation Term Duration. Type the XSD built in datatype xsd:Duration must be used.

[R 71] A period can be expressed using the Aggregate Core Component (ACC) PeriodDetails. The ACC is divided into 3 representation types, Date, Time and DateTime. One of these must be selected. Each option has a start and end date, start and end time or start DateTime and end DateTime.
For each representation term the equivalent data type must be used i.e. if the representation term Date is used, then the corresponding built in datatype xsd:date must be used.

The start and end times may be represented by the BCCs i.e. StartTime, EndTime, StartDate, EndDate etc.

The recurrence of these periods in time may be represented by the BCC RecurrenceValue.

The start and end times may be represented by the BCCs i.e. StartTime, EndTime, StartDate, EndDate etc.

The intervals in a point in time should be represented by a single BCC indicated by the choice operator i.e. FrequencyDuration, FrequencyYear etc.

Duration may be expressed by the BCC Duration.

The number of recurrences may be expressed by the BCC RecurrenceValue.

Duration may be expressed by the BCC Duration.

The intervals in a point in time should be represented by a single BCC indicated by the choice operator i.e. FrequencyDuration, FrequencyYear etc.

A message set may be extended where desirable if the business function of the original is retained, but the message exists within its own business context.

Documents must use the same legal characters in XML character data that are listed in the XML Recommendation. Including tab, carriage return, line feed, and the legal characters of Unicode and ISO/IEC 10646.

Trading partners may agree on other character encodings to use among themselves. It is recommended in all cases that encoding declarations be provided in the XML declarations of documents.

Messages must express semantics fully in schemas and not rely merely on well-formedness.

Instances conforming to schemas should be readable and understandable, and should enable reasonably intuitive interactions.

In the context of a schema, information that expresses correspondences between data elements in different classification schemes (“mappings”) may be regarded as metadata. This information should be accessible in the same manner as the rest of the information in the schema.

Element and Simple and Complex Type Names must not use acronyms, abbreviations, or other word truncations, except those in the list of exceptions pished in Section XX. [Editor’s note: Section xx to be a section in the NDR document. Currently this section only includes ID for Identifier, DUNS, and URI.]

Acronyms and abbreviations will only be added to the approved list after careful consideration for maximum understanding and reuse.
Acronyms and abbreviations must only be taken from the latest version of the Pocket Oxford English Dictionary. The first occurrence listed for a word will be the preferred item to be used.

The Abbreviations and Acronyms listed in Section XX must always be used.

All type declarations MUST be global.

All element declarations MUST be global with the exception of ID and Code which MUST be local.

Processing Instructions MUST NOT be used.

The *nullable* attribute MUST NOT be used.

Wildcards MUST NOT be used. Editors Note – does this include Any and AnyAttribute?

Two schemas shall be developed for each standard. One schema shall be a run-time schema devoid of documentation. One schema shall be a fully annotated schema that employs XHTML for the annotations.

Mixed content MUST NOT be used (excluding documentation).

Built-in Simple Types SHOULD be used wherever possible.

Simple Type restriction MAY be used wherever possible.

ATG: Union technique MAY be used to merge data types. *Not applicable. Therefore, SHOULD NOT be used. (Code lists are excluded from this rule.)*

Complex Types MUST be named.

The absence of a construct or data MUST NOT carry meaning.

Substitution groups MUST NOT be used.

Attribute Groups MAY be used.

ID/IDREF MUST NOT be used.

Key/KeyRef MAY be used for information association.

The XSD prefix MUST be used. (xmlns:xsd=http://www.w3.org/2001/XMLSchema)

The XSI prefix SHALL be used where appropriate.

Abstract Complex Types MAY be used *(for ur-schema).*

*(not finalized) Complex Type extension SHOULD be used where appropriate.*

*(not finalized) The *final* attribute SHALL be used to control extensions to defined complex types.*
The final attribute SHALL be used to control restrictions to defined complex types.

The \textit{block} attribute SHALL be used to control extensions to defined complex types.

Complex type restriction SHOULD be used.

Notations MUST NOT be used.

All documents shall have a container for metadata and which proceeds the body of the document and is named “Head” \textit{\ldots} (anything but header)

All elements with a cardinality of 1..n, (and lack a qualifying structure) must be contained by a list container named "\textit{name of repeating element)}List", which has a cardinality of 1..1.

Other Areas we need to address:

- Use of built-in schema attributes for elements not already addressed – including abstract, block, default, final, fixed, form, maxOccurs, minOccurs
- Use of built-in schema attribute \textit{abstract} for \textit{complexTypes}
- Use of Instance attributes – nil, noNamespaceSchemaLocation, schema Location, type
- Guidelines for use of All, Sequence, and Choice
- Use of extension element – restrictions on introduction of elements and/or attributes in complexContent
- Use of \textit{field} to specify XPath note
- Use of \textit{import} and \textit{include}
- Use of \textit{list} type
- Use of \textit{model groups}
- Use of \textit{notation}
- Use of \textit{redefine}
Use of ‘restriction’ – in simple types, in complex types using simple content, in complex types using complex content.

Use of ‘schema’ (parent element) attributes – attributeFormDefault, blockDefault, elementFormDefault, finalDefault, targetNamespace, version, xml:lang

Use of ‘selector’

Use of ‘simpleType’ ‘final’ attribute

Any restrictions on the use of ‘union’ technique and its ‘memberTypes’ attribute

Use of ‘unique’