TTC標準
Standard

ＪＴ－Ｍ３０３０

テレコミュニケーションマークアップ言語（tML）フレームワーク

Telecommunications Markup Language (tML) framework

第１版

2006 年 6 月 1 日制定

社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目　次

＜参考＞

１．国際勧告等との関連

本標準は、ITU-T 勧告 M.3030 (08/2002)に準拠したものである。

２．上記国際勧告等に対する追加項目等

2.1　オプション選択項目

なし。

2.2　ナショナルマター項目

なし。

2.4　原勧告と章立ての構成の相違

なし。

３．改版の履歴

版　数	制　定　日	改　版　内　容
第1版	２００６年６月１日	制　定

４．工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページでご覧になれます。

５．その他

（１）参照している勧告・標準等

ITU-T 勧告： M.3013(2000)

ISO 標準：ISO/IEC10646-1:2000, ISO/IEC 11179-3:1994, ISO/IEC 11179-5:1995

W3C 勧告：eXtensible Markup Language (XML) 1.0 (Second Edition) (2000),
XML Schema Part 1: Structures (2001), XML Schema Part 2: Datatypes (2001),
Namespaces in XML (1999)

IETF 仕様：RFC2141 (1997), RFC2396 (1998)

OMG 仕様：Unified Modeling Language Specification, Version 1.4 (2001)

Committee T1 標準：T1.227-2000

TTC 標準：JT-M3010 (2001)

（２）参照している標準等の置換

本文中で参照している ITU-T M.3010 は TTC 標準 JT-M3010 に置き換える。

1. スコープ

この勧告は、テレコミュニケーションマークアップ言語(tML)フレームワークドキュメントとして参照される tML を規定する。

tMLはテレコミュニケーションのオペレーション、アドミニストレーション、メンテナンス、およびプロビジョニング(OAM&P)のドメイン[1]で利用される。tMLは WorldWide Web Consortium (W3C) eXtensible Markup Language (XML)の応用である。

tML はオーダリング、ビリング、メンテナンス、およびプロビジョニングといった機能のためのテレコミュニケーション OAM&P エンティティ間のメッセージフォーマットとして使用される。

この勧告は TMN x リファレンスポイント (ITU-T Recommendations M.3010 [1] and M.3013 [2]参照)で使用するための tML メッセージ構造を開発するためのガイドラインを含む。

このバージョンの tML フレームワークは TMN X インタフェースに焦点を合わせている。以降のバージョンでは、他の TMN インタフェースを含むようにスコープを広げるかもしれない。

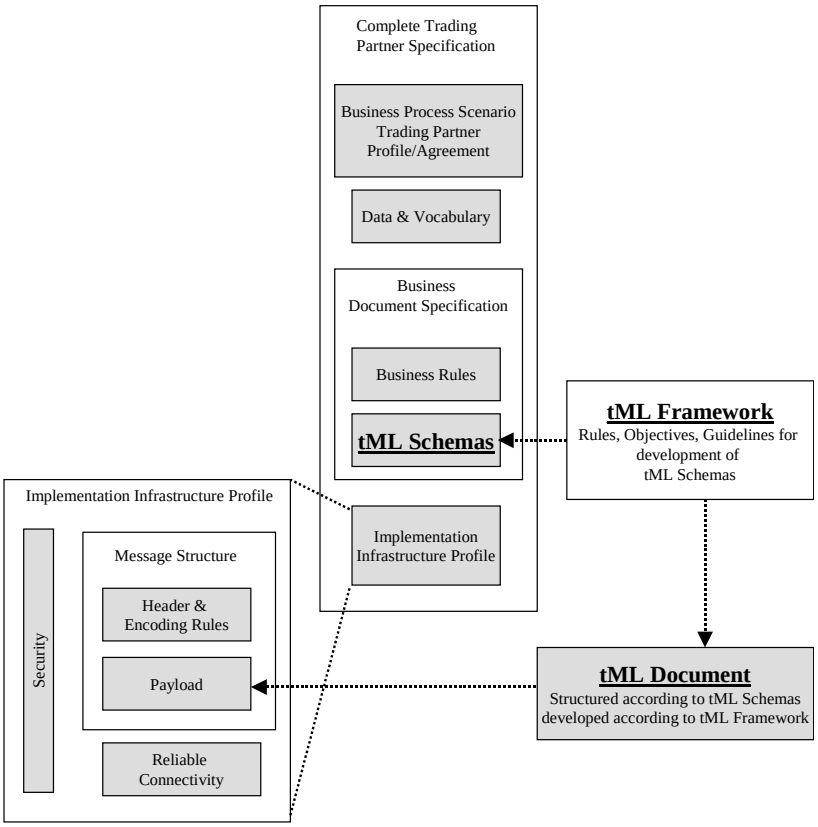

図1 /JT-M3030　tML Framework Scope
(ITU-T M.3030)

勧告(図 1)のこのバージョンのスコープは、以下のためのルール、方針、ガイドラインを含む。

[1] 現在、t-m-lいう用語は他の用途も存在するが、テレコミュニケーションマークアップ言語(tML)と混同すべきではない。tMLは小文字“t”で用語の他の用途と区別する。他のTMLの２つの例は、1)チュートリアルマークアップ言語(TML)、画面レイアウト、または出力形式(TMLはSGMLを使用して明確化された)から問題の意味内容を分けるためにデザインされた交換フォーマット；そして 2)テレフォニーマークアップ言語(TML)は、分散コンピュータテレフォニーとメッセージングアプリケーションにWebを適用する独自フレームワーク、である。

- tML の X インタフェースアプリケーションのためのビジネスドキュメント構造(すなわち、tML スキーマ)を明確化すること
- 共通のボキャブラリ構造の使用
- 名前空間の使用
- 既存の標準から tML へのマッピング
- 使用されるメタデータの仕様

この勧告のスコープでは以下の項目は、取引を行うパートナが交渉を通じて明確化するので明確化しない。：

- ビジネスプロセスシナリオ
- 実装インフラストラクチャプロフィール—特定の通信プロトコルプロフィールの仕様(信頼性、可用性、生存性、すなわち RAS の条項を含む)、およびセキュリティ、プライバシ、否認防止の条項
- データとボキャブラリの内容

2. 参照

以下の ITU-T の勧告とその他の参照は、テキストとこれを構成する条項で一貫して参照する条項を含む。出版時点で、示される版は有効であった。全ての勧告と他の参照は改訂版に従わなければならない；それゆえ、この勧告のユーザは、本勧告と以下にリストされた他の参照の最新版を適用する可能性を調査することを推奨する。現在有効な ITU-T 勧告のリストは定期的に出版される。The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.(この勧告の中で参照するドキュメントは、独立したドキュメントであり、勧告の状態は提供されない。)

2.1 ITU-T 勧告

[1] ITU-T M.3010 (2000), テレコミュニケーションマネジメントネットワークのための原理

[2] ITU-T M.3013 (2000), テレコミュニケーションマネジメントネットワークのための考察

2.2 ISO 標準

[3] ISO/IEC 10646-1:2000 "情報技術 –ユニバーサル マルチプル-オクテット コーデッド キャラクタセット (UCS) －パート 1: アーキテクチャとベーシックマルチリンガルプレーン "

[4a] ISO/IEC 11179-3:1994, "情報技術 – データ要素の仕様と標準化– *Part 3:* データ要素の基本属性"

[4b] ISO/IEC 11179-5:1995, "情報技術 – データ要素の仕様と標準化–*Part 5:*データ要素のための命名と識別原理

2.3 World Wide Web Consortium (W3C) 勧告

[5] eXtensible Markup Language (XML) 1.0 (Second Edition), 6 October 2000

[6] XML Schema Part 1: Structures, 2 May 2001

[7] XML Schema Part 2: Datatypes, 2 May 2001

[8] Namespaces in XML, 14 January 1999

2.4　IETF 仕様

[9] IETF RFC 2141 (1997), URN Syntax

[10] RFC 2396 (1998),　Uniform Resource Identifiers (URI): Generic Syntax August 1998

2.5　Object Management Group 仕様

[11] Unified Modeling Language Specification, Version 1.4. September 2001

2.6　Committee T1 標準 (Sponsored by ATIS, Alliance for Telecommunications Industry Solutions)

［12］T1.227-2000—OAM&P – フォルトマネージメントをサポートする管轄区域境界を横切ったオペレーションシステム間のインタフェースのための汎用的ネットワーク情報モデルの拡張

3. 定義

この勧告は以下の用語を定義する。

3.1　XML ベースの定義

3.1.1　属性 (attribute)

オブジェクトやエンティティの特徴(ISO/IEC 11179-3)；要素のプロパティ；要素に関する情報を渡すことができるので、属性は要素のメタデータと見なすことができる。属性は、***名前=値***の組として構造的に構成される。ここでは***名前***は属性の名前で、***値***は属性の値である。

3.1.2　文字 (character)

ISO/IEC10646 によって明確化されたテキストのアトミックな単位；ISO/IEC 10646 によって定義された、英字の１つ、数字または句読点。

3.1.3　文字データ (character data)

マークアップ文字でないすべてのテキスト文字

3.1.4　内容モデル (content model)

インスタンスドキュメントの中で、要素の構造に影響する要素の開始タグと終了タグの間のすべてのデータ。これは、要素の中の属性か他の要素であるかもしれない。(この用語は使用されない)

3.1.5 コンテキスト(context)

名前が適用されているか、またはそれが起動するアプリケーション環境や規律に関する意味または説明。(ISO/IEC11179-3)

3.1.6　データ辞書(data dictionary)

データの名前、語義の意味、関係、および型のような特徴を含むデータモデルにおける、データオブジェクト、または項目に関するメタデータの集まり

3.1.7　データ要素 (data element)

定義、識別、表現、および許容値が一組の属性によって明確化されるデータの単位(ISO/IEC11179-3)。

3.1.8　**デフォルト名前空間** (default namespace)

プレフィックスの付かない名前空間。デフォルト名前空間は、それが宣言される(その要素が名前空間プレフィックスを持たないなら)要素と、その要素の内容の中のプレフィックスのないすべての要素に適用される。デフォルト名前空間宣言における URI 参照が空であるなら、宣言のスコープの中でプレフィックスのない要素は名前空間にも存在しないと考えられる。

3.1.9　**区切り** (delimiter)

文字列かテキストフィールドの開始と終了に印を付ける特殊文字

3.1.10 **ドキュメント** (document)

データオブジェクトのクラス；印刷されたドキュメントのテキスト、一組のデータベースレコードであるかもしれない。

3.1.11 **ドキュメント要素** (document element)

インスタンスドキュメントのルート要素。ルートと呼ばれる一つの要素、すなわちドキュメント要素がある。部分的にもその他の要素の内容の中には現れない。

3.1.12　**要素** (element)

XML ドキュメントの中の論理データ構造；開始タグと終了タグは、要素の始まりと終わりを定義する。それぞれの XML ドキュメントは一つ以上の要素を含んでいる。その境界は開始タグと終了タグ、もしくは空要素用に、空要素タグによって区切られる。それぞれの要素は 、“ジェネリック識別子”(GI)と呼ばれる大文字と小文字を区別する名前によって識別された型を持っていて、一組の属性仕様を持っているかもしれない。それぞれの属性仕様は名前と値を持つ。

3.1.13　**要素宣言** (element declaration)

型と名前の関連付け。(この用語は、ローカル要素宣言の定義内だけにある。これは使用されない。)

3.1.14　**要素型** (element type)

開始、終了、または空タグに現れる名前。以下の例には、3 つの要素があるが、要素型は２つである：

```
<enduser>
    <firstname>Jo Anne</firstname>
    <firstname>Stephen</firstname>
</enduser>
```

その型の要素が子要素だけを含まなければならないとき、要素型は要素内容を持っている。そして任意に空白記号で区切られる。

3.1.15　**空要素** (empty element)

内容を持たない要素：空タグは<name></name> または<name/>のシンタックスに従う。

3.1.16　**終了タグ** (end tag)

開始タグで始まるすべての要素の終わりは、開始タグで与えられる要素の型をエコーする名前を含んだ終了タグで印されなければならない。XML プロセッサがアプリケーションと呼ばれる別のモジュールの代わ

りに仕事をしていることが仮定される。

3.1.17　**エンティティ** (entity)

名前によって識別された決まった値を持たない仮想記憶領域単位；多くの場合別々のファイルだが、文字列かデータベースレコードかもしれない。

3.1.18　eXtensible Markup Language (XML)

構造上、XML ドキュメントは SGML ドキュメントに従っている。W3C 勧告。Standard Generalized Markup Language(SGML)のアプリケーションプロファイルか制限されたフォーム。構造上、XML ドキュメントは SGML ドキュメントに従っている。SGML のサブセットである XML は Web で機能するように明確に設計されている。HTML のタグは事前に定義されるが、XML はタグがページの開発者によって定義されるのを許容する。したがって、XML で定義された Web ページはデータベースレコードのように機能することができる。

3.1.19　**ジェネリック識別子** (generic identifier (GI))

要素“型”の名前。“タグ名”という用語は、GI を参照するために使用される。

3.1.20　**グローバル要素(と属性)宣言** (global element (and attribute) declarations)

グローバル要素、およびグローバル属性はスキーマ要素の子として現れる宣言で作成される。一度宣言されると、グローバル要素もしくはグローバル属性は、他の一つ以上の宣言から参照されることができる。

3.1.21　HyperText Markup Language (HTML)

WWW ブラウザによる表示のための広範囲のドメイン(例えば、テキスト、グラフィックス、データベースクエリの結果)からのコーディング情報のシステム。タグと呼ばれるある特別なコードは、ブラウザに情報の表示方法を伝えることができるようにドキュメントに埋め込まれている。

3.1.22　**インポート** (import)

別のスキーマからの型宣言の合併を定義する。インポートメカニズムは、異なった名前空間の他のスキーマに含まれる定義と宣言が、ドキュメント内で参照をされるのを許容するために XML スキーマの中で使用される。開発者が別の名前空間からスキーマフラグメントかモジュールを参照するなら、インポートメカニズムが使用されなければならない。開発者が同じ名前空間からフラグメントかモジュールを参照するなら、インクルードメカニズムが使用されなければならない。

3.1.23　**インクルード** (include)

既存の名前空間への別のスキーマの合併を定義する。置換テキストが検索されて処理されるときエンティティは、参照自体に代わって、参照が認識されたロケーションにおけるドキュメントの一部であるかのようにインクルードされる。XML スキーマは他の XML スキーマにインクルードされることができる。そのようなインクルードスキーマドキュメントは（a）<インクルード>するスキーマドキュメントと同じ対象の名前空間を持つ、もしくは(b)対象の名前空間を持たない、のどちらかでなければならない。そのような場合<インクルード>されるスキーマドキュメントは<インクルード>するスキーマドキュメントの対象の名前空間に変換される。

3.1.24　**ローカル要素宣言** (local element declarations)

ローカル要素宣言はスキーマ構造の中で、なおいっそう入れ子にされて、ルートスキーマ要素の直接の子ではない。(この用語は使用されない)

3.1.25 **ローカル名** (local name)

ローカル名は修飾名のローカルな部分である。これは XML の名前空間でローカルな部分と呼ばれている。

3.1.26 lower camel case:

Lower Camel Case は最初の文字が小文字である大文字使用パターンで、しかし、その後全ての語頭は大文字になり、分離記号はつかわれない。これは例です(thisIsAnExample)

3.1.27 **マークアップ**(markup)

マークアップはドキュメントエンティティの最上位にある開始タグ、終了タグ、空要素タグ、エンティティ参照、文字参照、コメント、CDATA セクション区切り、ドキュメント型宣言、処理命令、XML 宣言、テキスト宣言、および空白記号の形をとる。

3.1.28 **メタデータ** (metadata)

他のデータを記述するデータ。

3.1.29 **メタ-言語** (meta-language)

他の言語を記述する言語。SGML と XML は、tML などのマークアップ言語を定義するので、メタ言語と考えられる。(この用語は使用されない)

3.1.30 **名前** (name)

文字か句読文字の 1 つで始まり、名前文字として知られている、文字、数字、ハイフン、アンダスコア、コロン、または句点で続くトークン。

3.1.31 **名前空間** (namespace)

URI または URN 参照で識別された一意な名前の概念的な集まり。[IETF RFC2396];XML ドキュメントでは要素型や属性名として使用される。名前空間は、他の名前と分離するための名前を修飾するために XML で使用される。注意：URI は URL に似たフォーマットを持つ。XML スキーマのインスタンスを見つけるために URI を解決することがいつも可能であるというわけではない。

3.1.32 **名前空間プレフィックス** (namespace prefix)

XML において要素または属性名を名前空間 URI と関連づける文字列。

3.1.33 **名前空間ルート** (namespace root)

いくつかの tML 名前空間の標準の一部。例えば　urn:int.itu/tML

3.1.34 **名前空間** URI　(namespace URI)

XML 名前空間を識別する URI。厳密に言うと、これは実際には名前空間 URI 参照である。これは XML において W3C ドキュメント名前空間の名前空間名と呼ばれる。

3.1.35 **オブジェクトクラス用語** (object class term)

それが属する論理的なデータ集合(論理的なデータモデルにおける)を表すデータ要素の名前のコンポーネント。例えば“employee.”(この用語は使用されない)

3.1.36 **親要素** (parent element)

他の要素を含む要素；親要素の中に含まれた要素は子要素として知られる。(この用語は使用されない)

3.1.37 **修飾名** (qualified name)

名前空間プレフィックスとコロン文字によって任意に先行されたローカル名(この仕様に基づき定義されるように)の連結として定義された要素または属性の名前。

3.1.38 **レジストリ**(registry)

この勧告のコンテキストの中では、リポジトリについてのメタデータのロケーションである。レジストリは、スキーマ、スキーマの所有者、およびリポジトリの中に格納された他の情報を定めたいユーザに対して提供される。

3.1.39 **リポジトリ**(repository)

1 つ以上のグローバルに分散されたロケーションである。それはスキーマ、スキーマ所有者の名前とロケーション、UML モデル、エンティティ間の情報交換を容易にするのに必要な他のデータや構成物を格納するために使用される。

3.1.40 **ルート要素** (root element)

ドキュメントの他のすべての要素を含む 1 つの要素;ルート要素はドキュメント要素であるかもしれない。

3.1.41 **スキーマ** (schema)

1 組のスキーマコンポーネント。型の定義と対象名前空間と呼ばれる特定の名前空間に属する名前の要素宣言の集まり(ボキャブラリ)。スキーマは XML ドキュメントのクラスの許容できる内容を定義する。スキーマの目的は、構成部品(データ型式、要素とそれらの内容、属性とそれらの値、エンティティとそれらの内容、および記法)の意味、用法、および関係を制約してドキュメント化するために、構成物を用いて XML インスタンスドキュメントのクラスを定義して、記述することである。ドキュメントのクラスは、スキーマのルールに適合するインスタンスドキュメントの中で、すべての可能な構造置換を参照する。

3.1.42　**スキーマ・コンポーネント**(schema components)

スキーマの抽象データモデルを構成する構成要素を表す汎用的用語。１３種類のスキーマ・コンポーネントがある。

名前つきのコンポーネントとして、

- 単純型定義(simple type definitions)
- 複雑型定義(complex type definitions)
- 属性宣言(attribute declarations)
- 要素宣言(element declarations)
- 属性グループ定義 (attribute group definitions)
- 識別子束縛定義(identity-constraint definitions)
- モデルグループ定義(model group definitions)
- 記号宣言(notation declarations)
- 注釈(annotations)

名前なしのコンポーネントして、

- モデル・グループ（model groups）
- 粒子(particles)
- ワイルドカード(wildcards)
- 属性利用(attribute uses)

3.1.43　**スキーマ要素**(schema element)

スキーマドキュメントのルートとなる要素

3.1.44　**セマンティックス**(semantics)

言語学上の分岐で、単語の意味をあらわす（ウェブスターより） 。

セマンティックスは、名前部分とそれらを分かつセパレータの意味に関係する(ISO 11179 より)。

3.1.45　Standard Generalized Markup Language (ISO 8879)

メタ言語で、他のマークアップ言語を構成するために使われる。XML は SGML の一方言であるが、SGML-lite として参照されることもある。

3.1.46　**開始タグ**(start tag)

XML 要素は、ひとつの開始タグから始まる。

3.1.47　**シンタックス**(syntax)

言語を表現するための構造で、言語の構造を規定するルール。文字や文字列同士の関係であり、その意味や解釈、使用方法とは独立である (ISO/IEC 11179-1)。

3.1.48　**タグ**(tags)

XML で書かれたドキュメントの中で最初と最後の印となるテキスト構造（マークアップ文字）。開始タグと終了タグの 2 種類のタグが存在する。開始タグと終了タグの名前は、要素の型を表す。これは要素型の名前である。開始タグにおける要素の型は、終了タグと合致しなければならない。開始タグと終了タグで挟まれたテキストは要素の内容と呼ばれる。

3.1.49　**タグ名**(tag name)

本来の「タグ名」ではないが、ジェネリック識別子として利用される。たとえば、<automobile>の中の automobile はジェネリック識別子である。ジェネリック識別子は、名前空間に置いて一意である。

3.1.50　**ターゲット名前空間**(target namespace)

ターゲット名前空間によって、定義や宣言をボキャブラリと区別することが出来る。たとえば、ターゲット名前空間によって、XML スキーマ言語のボキャブラリの中の要素の宣言を、仮想の化学言語ボキャブラリの中の要素の宣言を区別することができる。

3.1.51　**ユニバーサル文字セット**(universal character set, UCS)

ユニコードと ISO/IEC 10646 が定義した文字セット。これによって、世界の殆どの文字を一つの文字セットの上で符号化し、文字セットを切替えることなく、一つのドキュメント内で言語や活字を混在することが出来る。

3.1.52　**ユニコード**(unicode)

アメリカ・欧州・中東・アフリカ・インド・アジア・太平洋地域の言語を処理し表現するために設計された文字コードシステム。ユニコードは、プラットフォームやプログラムや言語に関わらず、全ての文字に一意な番号を与える。ユニコードの最新バージョンは、49,194 の独立した文字コードを含む。ユニコードは、16 ビットで ASCII 文字セットを包含している。ユニコードの標準は、ISO 646 の UCS-2 サブセットと同期している。ユニコードと ISO/IEC 10646 は同期しているが、ユニコードは実装上の制約が追加されており、幅広いプラットフォームやアプリケーションで同一に扱えるようになっている。ユニコードでは、ISO/IEC 10646 には存在しない文字仕様や文字データ、アルゴリズムや仕様が用意されている。

3.1.53　upper camel case: Upper Camel Case は、全ての単語の最初の文字が大文字である大文字使用パターンであり、分離記号は使用されない。これは例です。(ThisIsAnExample.)

3.1.54　UTF:　(UCS Transformation Format)

ユニコード標準の一つ。ISO/IEC 10646-1 は、複数オクテットで表現する文字セットを定義し、これを UCS と呼んでいる。UCS は世界中の文字システムの大半をカバーしている。しかし、複数オクテットで表現する文字列は、現在の大半のアプリケーションやプロトコルとは互換性が無いので、UTF-8 や UTF-16, UTF-32 等の UTF　(UCS transformation formats)の開発が進んでいる。テキストデータが UTF-8/16/32 でエンコードされれば、そのテキストデータは、ユニコードでエンコードしたと言える。XML は、ユニコード文字列で定義されており、UTF-8 と UTF-16 でエンコードすることが必須である。XML 1.0 仕様では、UTF-16 と UTF-8 の両方を処理できなければならない。

3.1.55　UTF-8: (UCS Transformation Format 8)

UTF-8 のエンコードは可変長であり、文字は 1 バイト、2 バイト、3 バイト、または 4 バイトでエンコードされる。UTF-8 は、現在ユーロを定義し、且つ全ての地方の文字を表現できる唯一の標準である。UTF-8 は 1 オクテットの全てのビットを使うが、US-ASCII がカバーしている。US-ASCII 文字は、通常の US-ASCII 文字を表す 1 オクテットと、US-ASCII にだけ必要な値[訳注エスケープ文字など]にエンコードされる（IETF RFC 2279）。

3.1.56　UTF-16: (UCS Transformation Format 16)

UTF-16 のエンコードは可変長であり、16 ビットの大きさを持つ。1 文字は一つまたは二つの 16 ビット単位で表される。バイト単位で言うと、1 文字は 2 バイトまたは 4 バイトで表される。

3.1.57　UTF-32

UTF-32 のエンコードは、固定長で 1 文字が 32 ビット（4 バイト）で 表される。

3.1.58　**妥当性**(valid)

定義ルールに従って宣言されたスキーマで書かれた XML ドキュメント

3.1.59　バージョン識別子(version identifier)

スキーマ仕様の版を表す識別子。

3.1.60　**ボキャブラリ（テレコミュニケーション）** (vocabulary (telecommunications))

与えられたテレコミュニケーションのコンテキストを表す、用語（名詞と動詞）と意味の集合である。
注意：要素型の名前は、特定のボキャブラリ用語で表される。

3.1.61　整形式ドキュメント(well-formed document)

XML ドキュメントの一つ、あるいはテキスト・オブジェクトの一つで、W3C XML 勧告のルールに準拠しているもの。テキスト・オブジェクトは、次の条件を満たせば、整形式 XML ドキュメントである。
(1) 全体を見ると、製造ラベルが付いたドキュメントと合致している。
(2) XML 勧告の条件を満足している。
(3)ドキュメント内で直接または間接的に参照されるエンティティが定式化されている。

3.1.62　World Wide Web Consortium(W3C)**勧告**

W3C 内のコンセンサスであり、勧告が規定したアイデアや技術は流布するに足り、W3C のミッションを後押しする。

3.1.63　XML **アウェア**(XML aware)

XML ベースのデータを認識し、XML のコンセプトを理解するソフトウェア・アプリケーション。(この用語は使用されない)

3.1.64　XML **宣言**(XML declaration)

XML ベースのドキュメントの先頭でなされるオプショナルな宣言であり、XML のバージョンを表す。たとえば、

```
<?xml version="1.0"?>
 <greeting>Hello, world!</greeting>
```

は完璧な XML ドキュメントであり、整形式だが、妥当ではない。

3.1.65　XML **の派生** (XML derivative)

XML 勧告の応用の一つで､特別な目的やバーティカルマーケットをサポートするために設計されたモノ。たとえば、CML (化学用マークアップ言語)や MathML、FPML（金融商品用マークアップ言語）、WML（無線用マークアップ言語）、tML（テレコム用マークアップ言語）など。

3.1.66　XML ドキュメント(XML Document)

XML 勧告に従って整形式されているならば、そのデータオブジェクトは XML ドキュメントである。

3.1.67　XML ボキャブラリ(XML vocabulary)

ある特定の機能に関する XML ベースのタグセット。tML や WML、MathML は XML ボキャブラリの一例である。

3.2　Internet Engineering Task Force (IETF)の定義

3.2.1　IETF

ネットワークの設計者、運用者、ベンダ、研究者が集まった大規模でオープンな国際団体である。インターネットのアーキテクチャを進化させ、インターネットをスムースに運用する。IETF は興味を持った個人に門戸を開いている。IETF は RFC や仕様を開発している。

3.2.2　URI (Uniform Resource Identifier)

IETF 標準の一つ。ウェブの基本アドレス。
URL (Uniform Resource Locators)と、将来の全てのリソース種別を含む。
URL は URI スキーマを使って作る。たとえば、http://や ftp://は URI のサブセットである。

3.2.3　URL(Uniform Resource Locator)

IETF 標準の一つ。ウェブのリソースのロケーションを表す。階層的なスキーマで表され、プロトコル(http など)とホスト名(www など)とデータパスから構成される。

3.2.4　URN (Uniform Resource Name)

ロケーションとは独立に、リソースや情報ユニットを識別する名前。URL は、URN が示すリソースのインスタンスを含むロケーションを指し示す。

3.3　Object Management Group (OMG)の定義

3.3.1　UML (Unified Modeling Language)

システムアーキテクチャを表すための OMG 仕様の一つで、単一の言語でオブジェクトの解析と設計をすることを目的とする。ソフトウェアを規定し、視覚化し、組立、ドキュメント化すると共に、ビジネス・プロセスをモデル化するために使われる。

3.4　この勧告の追加用語

3.4.1　グローバル・テレコム・データ辞書(Global Telecommunications Data Dictionary, GTDD)

公開されたデータベースの一つで、テレコムのオペレーション、アドミニストレーション、メンテナンス、およびプロビジョニング(OAM&P)を行うアプリケーションの名前やソース、バージョン、説明、セマンティックス、値、型、名前空間を含む。tML は GTDD のクライアント・アプリケーションの一つである。

3.4.2　実装のフレームワーク(Implementation Framework)

仕様またはアーキテクチャ要素の集合で、エンティティ間で tML インスタンスドキュメントを運ぶために使われる。

3.4.3　tML (telecommunications Markup Language)

W3C XML 勧告から派生したボキャブラリの一つ。tML はテレコムの OAM&P インタフェースを持つアプリケーションのメッセージ形式として使われる。

3.4.4　tML ベースライブラリ(tML Base Library)

再利用可能な共通データ・コンポーネントの集合。一つの領域では、複数のライブラリがあっても良い。ライブラリは、自身の名前空間を持たずに、tML スキーマに含まれても良い。あるリージョナルベースライブラリの中のデータ・コンポーネントは、一意な名前を持たなければならない。

3.4.5　tML のフレームワーク(tML Framework)

ITU-T 勧告で、TMN の中で tML を開発し実装するためのルールや勧告、ガイドラインとなるもの。この勧告は、“tML Framework document”と呼ばれる。

3.4.6　tML 名前空間(tML Namespace)

URI 規格(RFC-2396)が規定する名前の集合。tML ドキュメントの中で使われ、要素の型と属性を表す。

3.4.7　tML リージョン(tML Region)

tML のフレームワークを使う標準開発組織。ITU や ANSI T1、ETSI や JSI などのこと。

3.4.8　tML スキーマ(tML Schema)

tML インスタンスドキュメントの構成部品の意味や使い方、関係をドキュメント化したモノ。tML スキーマは、W3C XML スキーマ勧告に準拠している。

3.4.9　tML タグ(tML Tag)

ドメイン固有な要素と属性、エンティティを表す名前。tML ドメイン内で一意でなければならない。

3.4.10　ボキャブラリ(Vocabulary)

この勧告のコンテキスト。取引を行うパートナ同士で情報をやりとりするために使う。ボキャブラリを形作る用語は、アプリケーションに依存する。ボキャブラリを形作る用語は一カ所以上存在し、GTDD に含まれる。tML ボキャブラリは、テレコム固有の用語を表し、GTDD の中に存在する。

4. 略語

この勧告では次の略語を使う。

ANSI	American National Standards Institute
B2B	Business-to-Business
ETSI	European Telecommunications Standard Institute
GTDD	Global Telecommunications Data Dictionary
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
OAM&P	Operations, Administration, Maintenance and Provisioning
OO	Object-Oriented
OS	Operations System
OSS	Operational Support System
SDO	Standards Development Organization
TA	Trouble Administration
tML	telecommunications Markup Language
TMN	Telecommunications Management Network
UCS	Universal Multiple-Octet Coded Character Set
UID	Unique Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTF	UCS Transformation Format
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

5. 規約

この勧告では、次の決まりに従ってドキュメントを書き、読者にテキストの重要性を喚起させなければならない。

5.1 要求

“must”とか”shall”という助動詞で表される段落は、必須の要求を表す。
要求は、ボールド・フォントで“ルール”と表し、簡潔なドキュメントで要求を述べなければならない。

5.2 方針

ある段落は方針を表す。これらの段落では行動指針(course of action)または最善の手法(best practice)を、(～する)べきである[should]を使って表す。

5.3 ガイドライン

ある段落はガイドラインを表す。これらの段落では、(～し)てもよい[may]を使って表される。

5.4 フォント

要素と属性は、Courier New フォントで示す。

5.5 「(～し)なくてはならない (shall) 」という用語

勧告に合致須していることを示すため、(～し)なくてはならない[shall]を使う。

6. 一般定義

6.1 この勧告の目標

6.1.1 この勧告の主要な目標

a) tML スキーマを開発するための統一的な指針を与えること

b) XML を使って TMN の X インタフェースを表し、取引を行うパートナ同士の相互作用を容易にすること

6.1.2 第二の目標

a) テレコム業界で、プラットフォームに独立な B-to-B のフレームワークを利用可能にすること。これは、この勧告の中の tML スキーマを使ってテレコム業界の仕事を書き表すことによって、成し遂げられるはずである。tML スキーマの生成は、取引を行うパートナ間でデータをやりとりする一つの方法に過ぎない。

b) GTDD のための基礎を構築すること。これは、tML が出来るだけフルセットのセマンティック情報を含むことで実現できる。tML スキーマは、GTDD のための情報ソースに過ぎないが、将来の tML スキーマは GTDD から生成されるかもしれない。

6.1.3 M.3030 に将来追加されるのは

GTDD における tML の利用と、tML の登録、tML の保管

6.2 用語 tML の語源

この勧告で使われる tML と言う用語は、W3C XML 勧告に由来するマークアップ言語を参照している。tML 言語のボキャブラリとセマンティックは、GTDD の中で定義される。tML 言語のボキャブラリ(定義セクション参照)とセマンティックスは、グローバルテレコミュニケーションデータ辞書(GTDD)中で定義される。ユニークな識別子(UID)が、tML スキーマの中の各要素の名前に対して付与される。UID は、GTDD の中で、メタデータに対して要素の名前との対応関係を定義する。このメタデータは、その定義(セマンティックス)を含んでいる。

各要素およびそれと関連するメタデータの定義は、ITU-T に準拠する tML スキーマの各要素に含まれるものとする。tML スキーマの中で規定されるメタデータについての更なる情報は、Annex1 を参照のこと。tML スキーマに含まれる定義とメタデータは、後々、GTDD エンティティのソースとなりうる。

tML は W3C XML ベーススキーマの階層構造(図 2)である。それは以下に示す順序で、より特化されながら互いに積み上げられている：

1) すべてのドメインでの使用するための共通のデータ要素から成る基本(コア)ボキャブラリ

2) オペレーションに特定したアプリケーションドメインのボキャブラリセット、例えばプロビジョニング、トラブル管理、およびビリング

3) サービス／テクノロジに特定したドメインのボキャブラリセット、例えば DSL、SDH、および ISDN

これらのドメインへのオーバーレイには国際およびリージョナル特有のバリエーションがある。

6.3 tML スキーマの構造

図２に示す tML スキーマ構造図は、どのスキーマが tML に包含されるかによってメソッドを例示している。名前空間はこの構造（すなわち、リージョナル／アプリケーション／テクノロジ）に従ってこれらのスキーマに割り当てられるであろう。図２は、共通データ要素からなるそのリージョナルベースライブラリと、アプリケーションおよびテクノロジドメインの間の関連を示している。領域は ITU、ANSI、ETSI または JSI

SDOs あるいは共通の興味を持つ他の分野により構成されるかもしれない。各々の領域に関して、（国際は単に標準化機関領域の特別なケースである）‘リージョナル’ボックスによって囲まれた完全な構造が各々の tML リージョンに対して存在するだろう。これは、各々の領域が、他の地域に考慮することなく標準スキーマを開発することを可能にするであろう。同時に、それらのスキーマの採用を別の領域から妨げられることはない。これは、‘国際的’要求によって束縛されることなしに、地域的な要求に適合するコントリビューションを促進することを可能とするだろう。同時に、これらのリージョナルなコントリビューションがより広く使われることを可能とする。

図２において外部リソースライブラリは、tML スキーマに取り入れられる潜在的な情報源の素材を表している。tML スキーマは、どのような外部リソースライブラリまたはスキーマからもインポートしないこととしているが、新しい tML データ構成要素が創造されるような場合にはそれらから有益な用語を採用するものとする。すなわち、tML スキーマは tML に準拠しないどのようなスキーマも参照しないこととしている。

この勧告の目的のために、‘国際’は他の領域と同等とみなされている。これは、国際的な tML スキーマのセットが、あるリージョナル tML スキーマを超える優先権を持たないことを意味している。しかし、複数の領域に対して共通の tML スキーマは、国際的な tML スキーマになるかもしれない。例えば北アメリカの ANSI トラブル管理 tML スキーマ(ANSI/TA)は国際的なトラブル管理 tML スキーマ(ITU/TA)から属性を継承しないかもしれない。これは、その使用が適切な場合において ANSI 標準が国際標準の機能を採用することを妨げない。この主要な理由の 1 つは、tML 名前空間の仕様を可能な限り扱いやすくしておくことにある。この目的のためにリージョナルベースライブラリは独立した名前空間を持つべきではないが、適宜追加されるべきである。

データコンポーネントは、どのような領域の中のリージョナルベースライブラリのものでもユニークにネーミングされるものとする。このネーミングの要求は、領域の中で適切な標準化機関によって管理されるものとする。

tML スキーマ要素を最大限再使用するために、アプリケーションドメインがテクノロジドメインより優先権を持つように意図されている。tML フレームワークは、テクノロジドメインから共通データ要素のライブラリの使用を除外しない。しかしながら、それはこれらが適切なアプリケーションドメイン名前空間の中に含まれている場合に限定している。そのようなライブラリはそれ自身の名前空間を持たないであろうが、それらが含まれるアプリケーションドメイン名前空間の一部になるであろう。

図２／JT-M3030　tML スキーマ構造

(ITU-T M.3030)

トラブル管理[2]に対する北アメリカ標準T1.227 に基づく寄書を例として示す(図 3)。これは、何等かの特定のテクノロジの参照を必要とせず、障害管理に対する主要な‘汎用的’標準の中に存在している。したがって、一般的な部分はANSI／障害管理ドメインの中にあるであろう。しかし、特定のテクノロジ（特殊なサービス回線などの）にのみ有効であるような特有のタイプの障害のために参照するT1.277 の詳細の中のいくつかは、ANSI／障害管理／特殊サービス回線ドメインの中にあるであろう。これらが相互にどのように関連しているかについて図３に示す。

[2] T1.227-2000－OAM&P- 障害管理への支援を提供するための管轄区域境界を横断するオペレーションシステム[OSs]間のインタフェースのための、汎用的ネットワーク情報モデルへの拡張。この標準の範囲は、ネットワーク管理のために使われ、且つ異なる管轄区域境界に置かれたOSsのためのオペレーションシステムインタフェースに対するオペレーションシステムに限定される。

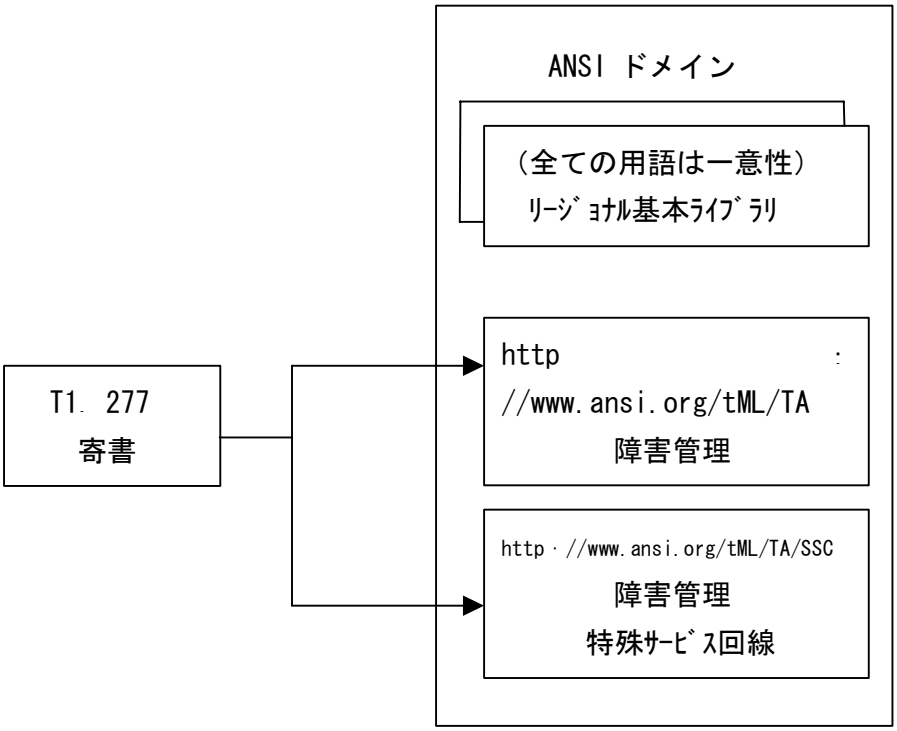

図３／JT-M3030　tML スキーマ構成に対する寄書マッピングの例
(ITU-T M.3030)

6.4　TMN x 参照点における tML の使用

tML は、当初 TMN x 参照点に適用されるだろう。

6.4.1　TMN x 参照点における tML

tML は、ハードウェアプラットホーム、オペレーテイングシステムのソフトウェア、または情報を取り交わしている２つかそれ以上の管理部門によって使用されるプログラミング言語にかかわらず、データフォーマットを表現するための普遍的な言語のように X インタフェース（TMN x 参照点において）上で使用され得る。tML の使用は、ウェブベースのシステムにアプリケーションを統合しようと望んでいる管理部門にとって重要である。TMN X インタフェースを通して取り交わされる tML ドキュメントは、統合オペレーショナルサポートシステム（OSS）と企業インフォメーションテクノロジ（IT）システムのためのあらゆる戦略計画（それらは XML メッセージフォーマットに基づいている）の一部を形成し得る。

X インタフェースにおける tML のアプリケーションドメインの例を以下に示す。(図 2 を参照)

- フルフィルメント
    - プレオーダ
    - オーダ
    - プロビジョニング
    - その他.
- アカウント管理
    - ビリング
    - アカウント情報のメンテナンス

    - その他
- 品質管理
- 障害管理
    - 試験
    - 保守管理
    - その他
- 構成管理
- その他

X インタフェースに対するアプリケーションドメインの中身についての将来の情報源は、勧告M. １４００である。

7 tML 技術仕様

7.1 勧告への適合性

本節では、tML スキーマ標準を作成するために、ルール、方針、および関連しているガイドラインを直接的または間接的に提供する。

標準化に従ったスキーマは、当勧告の中でそれらが全ての要求（R アイテム）を満たすことを規定した当勧告に対して整合しているとみなされるべきである。

ルールは定められた手法であり、守られるべきものである。当勧告と整合をとるため、示されたルールは手法として用いられるべきである。実装は、メッセージ(起動者または応答者としてのどちらでも)を別の実装と交換するたびに、もし実装が常にこの勧告の中のルールによって規定されているようにふるまうならば、厳密にこの勧告に整合するものとする。

方針は任意の手法である。この勧告と整合をとるために方針に準拠する必要はない。

ガイドラインは、アクションを決定する際のスタイルの選択である。この勧告と整合をとるためにガイドラインに準拠する必要はない。

7.2 tML スキーマ — 概要

W3C スキーマ言語は、tML スキーマドキュメントを作るための基礎として使われる W3C 勧告である。tML スキーマは、それがインスタンスドキュメントのシンタックスを認証できるように、tML インスタンスドキュメントのプロセッサ用に使用可能にされる。tML スキーマは業界標準として作られる。最も基本的なアプリケーションの中で、特定の tML スキーマが、ドキュメントのそのクラスのための認証メカニズムとして tML ドキュメントのクラスに適用される。tML ドキュメントのクラスのインスタンスは、インスタンスドキュメントがインスタンスドキュメント構造とシンタックスに関連付けられた tML スキーマの中で定義された特定のルールセットに適合するかどうかを決定するために、tML スキーマと比較される。

7.3 tML インスタンスドキュメントの認証

エンティティがtMLインスタンスドキュメントを交換する場合、それは、そのドキュメントがtMLスキーマに対して認証されていることを必要としないかもしれない。tMLインスタンスドキュメント交換に対する唯一のルールは、tMLドキュメントが整形式であることであり、それらがW3C XML勧告に合致していることを意味している。それらは認証済みである必要がない。しかし、ほとんどの場合、インスタンスドキュメントを処理するアプリケーションが構造品質の標準的な評価基準を持っているために、エンティティはtMLスキーマに対してtMLインスタンスドキュメントの認証を必要とするであろう。この評価基準はエンティティ

間の相互運用性の可能性を高める。さらに、tMLスキーマの使用は生産性を改善する。なぜなら、スキーマは標準本体によって事前承認されているであろうし、しかもインスタンスドキュメントの中でデータ構造の品質を確認するために各エンティティがそれ自身のメカニズムを開発する必要が無いだろうからである。スキーマ認証を、少ないコストであるいはコストをかけないでより容易に遂行するために、公共のインターネット上で入手可能なXMLスキーマ認証ツールがある。

7.4 他のゴール

tML スキーマ開発者は以下のゴールのために努力するべきである：

相互運用性

再利用性

拡張性

複雑性の軽減

読み手のための可読性の強化

7.5 ルール、方針およびガイドライン

7.5.1 スキーマ認証

ルール1 : tMLスキーマは、２００１年5月のＷ３Ｃスキーマ勧告パート１と２に従って妥当性があるものでなくてはならない[shall]。

ルール2 : tMLスキーマはこの勧告におけるすべてのルールに適合するものでなくてはならない[shall]。

ガイドライン1 : tMLスキーマはＵＴＦ－１６文字セットの中でエンコードされるものであるべきである[should]。

UTF-8はUTF-16のサブセットなので、ペアワイズアグリーメントによって、オプショナルで使われるかもしれない。

ガイドライン2 : tMLスキーマは、この勧告に準拠していることの証明を促進するために、厳密なレビュープロセスを受けるべきである[should]。

7.5.1.1 ローカルおよびグローバル宣言

ルール3 : tMLスキーマは、elementFormDefaultを有資格に設定するものでなければならない[shall]。

ルール4 : tMLスキーマは、attributeFormDefaultを無資格に設定するでなければならない[shall]。

7.5.1.2 XML スキーマ 名前空間宣言

ルール5 : XMLスキーマ名前空間はhttp://www.w3.org/2001/XMLSchema として宣言されなければならない[must]。

ガイドライン3 : プリフィックスxsd：この名前空間を示すために使われるべきである[should]が、実際の場ではどのようなプリフィックスでも使用され得る。

ガイドライン4 : すべてのドメインに特有なtMLスキーマは、少なくとも２つの名前空間を持つべきである[should]。：targetNamespace および XMLSchemanamespace

(http://www.w3.org/2001/XMLSchema)。

7.5.2 ‘外部’および‘ベース’ライブラリの使用

ガイドライン5：tML スキーマはどのような外部リソースライブラリまたはスキーマから**インポート**をしてはいけない[shall not]とするが、新しいtMLデータコンポーネントを作成するため、それらから有益な用語を適用しなくてはならない[shall]。

ガイドライン6：tMLスキーマは、どのような名前空間もtMLフレームワークの範囲を超えて参照してはいけない[shall not]。

ガイドライン7：この目的のために、リージョナルベースライブラリは独立な名前空間を持つべきではない[should not]が、適切な所で含まれるべきである[should]。これは、ネーミングの独自性を保証するためにリージョナルＳＤＯが名前空間を使うことを妨げないけれども、それがtMLドキュメントにたくさんの数の名前空間を参照することをもたらすおそれがある時にはやめる。

ルール6：どのような領域の中のリージョナルベースライブラリのデータコンポーネントであってもユニークにネーミングされなければならない[shall]。

ルール7：ベースライブラリのネーミング要求は、その領域の中で、関連した標準化機関によって管理されなければならない[shall]。

7.5.3 名前空間の名前付け（Namespace naming）

tML は、あるスキーマがこの勧告に整合していることを示すために指定された一つのセクションを除き、完全な名前空間を定義しない。

ルール 8：名前空間のルート（根）はそのリージョナル標準化機関が責任を持たなくてはならない[shall]（管理しなくてはならない[shall]）

例：

http://www.ansi.org/

urn:int.itu/

http://etsi.org/

ガイドライン 8：URL として名前空間 URI を利用することによってアクセスされないようにする tML スキーマは URN を使うべきである[should]。

例：

http://www.ansi.org/tML/…　は web のロケーション(URL)と思われるかもしれないが、これは名前である。

urn:int.itu/tML/…　はユニークな名前であり、web アドレスに思われないだろう。

ルール 9：tML スキーマに適合している名前空間は tML/を含まねばならない[shall]、従い、先ほど示した名前空間のルートは次のようになる。

http://www.ansi.org/tML/

urn:int.itu/tML/

http://etsi.org/tML/

ガイドライン 9：tML スキーマは次に示す方法により追加的に名前をつけてよい[may]。

ApplicationDomain/TechnologyDomain

ここで、ApplicationDomain は TroubleAdminisitration のようなスキーマに属しているアプリケーションドメインの名前である。

TechnologyDomain はサービスまたはテクノロジに属しているドメインの名前である。これは完全なアプリケーションドメインに汎用的なスキーマの場合には発生しない。

名前空間のルートを含む完全なスキーマ名の例を挙げる。

urn:int.itu/tML/TroubleAdministration － これは、International Trouble Administration 向けの汎用的スキーマである。

urn:int.itu/tML/TroubleAdministration/DSL － これは DSL 製品のみに関連する International Trouble Administration 向けの要素である。

http://www.ansi.org/tML/TroubleAdministration － これは、American Trouble Administration 向けの汎用的スキーマである。

http://www.ansi.org/tML/TroubleAdministration/DSL － これは、DSL 製品のみに関連するトラブル管理のための要素である。

http://etsi.org/tML/FulFilment – これはヨーロッパの Fulfilment スキーマであり、そして

http://etsi.org/tML/FulFilment/DSL – これはヨーロッパの DSL に関するスキーマである。

一つ以上のスキーマがエリアのいずれかにあるならば、アプリケーションかテクノロジのために、それは’ドット’、そしてスキーマ名が続くアプリケーション、またはテクノロジを前に置くことによって示されるべきである：例えば

urn:int.itu/tML/TroubleAdministration.ErrorCodes **or**
urn:int.itu/tML/TroubleAdministration/DSL.AcceptanceCodes

7.5.4 名前付けの規約（Naming conventions）

この勧告での名前付けルールとガイドライン及び方針は、要素と属性名及びタイプ定義（複雑型及び単純型）を含む tML スキーマにおいて利用される名前付けられたすべての作成物の名前付けをもとに定義されている。

7.5.4.1 名前付けの考慮（Naming considerations）

名前付けは要因(factor)の数によって影響を受ける。データモデルを明確にすることはアプリケーションの開発者にとって重要である。一般に良いデータモデルはアプリケーションを長持ちさせると信じられている。

レガシーなデータモデルは存在し、開発者は既に存在している名前を利用することを求められるだろう。

ガイドライン 10：名前は人が読める（意味がある程度理解できる）ものであるべき[should]。

ガイドライン 11：業界が認める用語は新しく作られる用語よりも名前にふさわしい[may]。あなたの業界の一部が他の領域や新規参入者が理解するであろう用語を理解すると仮定してはいけない。

ガイドライン 12：その意味が疑わしくなる可能性がある場合は略字を展開しなさい。時間やスペースの問題で、アプリケーションプログラムやスクリーンに表示するために略字を使うのはよいだろう[may]。

ガイドライン 13：その名前を理解しやすくするために複数の単語を繋ぎ合わせてもよい[may]。または、その連結されたものが一つのオブジェクトクラスをあらわすという理由や、異なる意味を持つ似たような名前と区別したいという理由でもよい。

7.5.4.2 一般規約（General conventions）

この勧告でのタグによる名前付けルールとガイドライン及び方針は、ISO11179 Part5 の一部である”Naming and Identification Principles for Data Elements."に基づいている。

ルール 10：tML タグ名の割り当てはビジネス要求が他の名前付け規約の利用を命令しない限り、この勧告にある規約に従わなくてはならない[shall]。

方針１：名前はたとえ長くなっても直感的なものにするべきである[should]。

方針２：名前は内容が分かる範囲でできるだけ短くするべきである[should]。

方針３：その用語が既に定義されているかどうか気をつける。用語を決定する前に業界のデータ辞書を参照せよ。

方針４：略字は避けるべきである[should]。

方針５：同じドメインの他の名前と混同されないように選ぶべきである[should]。

方針６：用語の中から似たような名前を使うことで、名前の型は一貫性をとる。

方針７：”Type”や”Group”を添え字に使うことで、タイプ名と要素名及びグループ名を区別する。

方針８：データモデルの中で混同を導くため、一つの要素や表記法及びエンティティに同じ名前をつけるのは避ける。

ガイドライン 14：単純型及び複雑型は通常名前を持つべきである[should]。

ルール 11：tML スキーマで利用される用語は業界の参加者によって認められている既存の用語から派生したものとしなくてはならない[shall]。

ルール 12：各ドメインで使われる用語は唯一としなくてはならない[shall]。

ルール 13：同じドメインにおいてはその目的があいまいとなるような用語を使ってはいけない[shall not]。例えば、もし address タグが住所、例えば顧客住所や宛先住所の識別のために定義される場合は、話し手が聞き手に対する演説の意味の”address”を使ってはならない。

<speaker>Abraham Lincoln</speaker>

<address>The White House</address>

<address>Gettysburg Address</address> <!-- not allowed（認められない） -->

ガイドライン 15：tML スキーマ及び tML タグを正確で完全に定義することは重要である。同じ tML タグは同じドメインにおいて異なるもののために決して利用するべきではない[should never]。

ルール 14:ある要素の属性と同じ要素の子に対して同じ tML 用語を決して使ってはいけない[shall never]。

7.5.4.3 **要素と属性の名前付け**（Element and attribute naming）

7.5.4.3.1 **要素の名前付け**（Element naming）

ルール 15：要素名にはコロンを含んではいけない[shall not]。

その理由は名前空間の仕様との混乱を引き起こすかもしれないためである。つまり、ある要素型または属性名を foo:bar のようにしてしまうと、プロセッサによって XML 名前空間のプレフィックス foo とローカル名 bar との間にセパレータがあると翻訳される可能性があるためである。

ルール 16:さまざまなリージョナル言語に対応するため要素名はアルファベットと数字のみからなる文字で構成されなくてはならない[shall]。句読点は含んではならない。

ガイドライン 16：要素名はPurchaseOrder3のようにUpper Camel Caseであるべきである。

7.5.4.3.2 **属性の名前付け**（Attribute naming）

ルール 17：属性名にはコロン":"を含んではならない[shall not]。

その理由は名前空間の仕様との混同を引き起こすかもしれないためである。つまり、ある要素型または属性名を foo:bar のようにしてしまうと、プロセッサによって XML 名前空間のプレフィックス foo とローカル名 bar との間にセパレータがあると翻訳される可能性があるためである。

ルール 18：属性名は次のルールに従うべきである[should]。さまざまなリージョナル言語に対応するため属性名はアルファベットと数字のみからなる文字で構成されなくてはならない[shall]。句読点は含んではならない。

ガイドライン 17：属性名は、partyIdentifier3 のように最初が小文字の lower Camel Case とするべきである[should]。

3 There are existing names in common use that prevents this from being a Rule, and there are cases where new element names are to be developed in keeping with past camel case usage rules that may conflict with this Rule.(これがルールであることを妨げる共用の既存の名前があり、新しい要素名がこのルールと衝突するかもしれない過去のcamel caseの使用ルールに合わせ開発されることになっているケースがある。)

7.5.5　tML スキーマ注釈（tML Schema Annotations）

注釈の目的は、ある人が必要とする情報を外部的に捜し求める手間を減らすため、tML スキーマ(あるいはインスタンスドキュメント)内に実際的であるようにドキュメンテーション(documentation)を維持することである。ドキュメンテーション(documentation)とコメントは名前付けられた複雑型や名前付けられたモデルグループ及び名前付けられた属性グループのような再利用可能なアイテム内で有用である。

ルール 19：<documentation>要素はスキーマ文法が認める限り利用されなくてはならない[shall]。しかし、そのテキストが挿入される要素の利用を明確にするために必要な情報が要求するあらゆる要素を即座にインラインで処理することが望ましい。

ルール 20 : xml:lang 属性はその情報の言語を示すために<documentation>要素の中で使われなくてはならない[shall]。

タグ名を注意深く選択すると、tML ドキュメントが読みやすくなる。スキーマ定義内でのコメントの配慮ある利用はそのスキーマの理解を非常に助ける。

ガイドライン 18：すべての tML スキーマはその構成とそのスキーマによって記述される tML ドキュメントの内容に関するコメントを含むべきである[should]。

ルール 21 : 型やグループ、要素または属性は<documentation>要素のなかでつぢの情報と共に十分に定義されなくてはならない[shall]。

Unique IDentifier (UID)

Short Definition

Full Definition

Source of the tag including version

Name

Basic or Aggregate

Business Term(s)

Status

ルール 22：tML スキーマは次のメタデータを持たなくてはならない[shall]。

完全なバージョン履歴

変更リスト

- 日付
- 変更者
- 変更理由

原作者

生成日

このスキーマに責任を持つ標準開発組織名

7.5.6　スキーマ要素と属性（Schema Elements vs. Attributes）

多くの場合、要素と属性の選択はスキーマの作者の考えに基づく。しかしながら、他の場合は選択は必須である。

7.5.6.1　**要素** (Elements)

7.5.6.1.1　**要素の利用** (When to use Elements)

ガイドライン 19：あるアイテムを利用する方法が疑わしい場合は、要素を作る。

ガイドライン 20：ある独立したオブジェクトとして考えられる情報の断片を表現するために要素を使う。

ガイドライン 21：その情報が親子関係により他の情報の一部に関連する場合に要素を使う。

ガイドライン 22：そのデータが頻繁に変更される場合に要素を使う。XML プロセッサが属性を用いて要素データの発見と変更及び比較することが容易になる。

ガイドライン 23：変更や見直し、改定がある場合は要素を使う。

7.5.6.1.2　**要素が必須のとき**（When Elements are mandatory）

ルール 23：データ項目間のリンクがある場合は要素を使わねばならない[must]。

ルール 24：再利用可能な構造を表現するために要素を使わねばならない[must]。

7.5.6.2　**属性**（Attributes）

7.5.6.2.1　**属性の利点**（Advantages of attributes）

- 他のデータ項目を修飾するのに便利である。
- 属性は本来、値の列挙されたリストに関連するデータをカバーするものである。これは XML スキーマに限った話ではない。

7.5.6.2.2　**属性の欠点**（Disadvantages of attributes）

- 属性は複数の値を含めることができない（子要素は可能）
- 属性は拡張が容易ではない（将来の変更のため）
- 属性は構造を表現できない（子要素は可能）

7.5.6.2.3　**属性を使う場合**（When to use Attributes）

ルール 25：属性は要素の識別のために利用されなければならない[must]。

ガイドライン 24：単純な情報のために属性を使うこと。

ガイドライン 25：一つの要素に関連付けられた属性の数は最低限にとどめること。属性が沢山あるとドキュメントが読みづらくなる。

ガイドライン 26：言葉のように要素を修飾する情報として属性を使うこと。

ガイドライン 27：処理される要素のコンテキストを制御するために属性を使うこと。

ガイドライン 28：人に対する読みやすさだけでなくコンピュータに対してもその要素が正しく処理できる情報を格納するために属性を使うこと。

ガイドライン 29：固定値のようにデータがめったに変更されない単純な文字である属性を使うこと。

7.5.6.2.4　**属性を使ってはいけない場合**（When not to use Attributes）

ガイドライン 30：ドキュメントの構成を特定するために属性を使ってはいけない。

ガイドライン 31：それが適切かどうか疑わしい属性は使ってはいけない。

ガイドライン 32：ある属性によって表現されるデータが、将来の拡張によって更に修飾することが要求される可能性のある属性は使っていけない。

7.5.7　バージョニング（Versioning）

ルール 26：tML スキーマの版数管理はスキーマ要素の中に格納される versionID 属性により実施しなけれ

ばならない[shall]。このようにバージョン番号を管理することにより、tML ドキュメントの後方互換性を得る。例えば、<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" versionID="1.0">

方針９：古い tML スキーマバージョンは、それらのバージョンに対するドキュメントの適正認証に対して有効であり、アプリケーションは、それらの古いバージョンの存在を中継できるべきである[should]

ルール 27：tML スキーマのマイナーチェンジは後方互換でなければならない[shall]。あるいは、このルールを満足させるためのバージョン番号管理能力を持たねばならない。

7.5.8 再利用可能な内容（Reusable content）

tML スキーマ構造の再利用は次の２つの基本的なカテゴリに分類される。

１）対象とする tML スキーマドキュメントに既に我々が定義したコンポーネントの再利用、

２）どこかで定義されているグローバルなコンポーネントの再利用。内部の tML スキーマ構造の再利用は名前付けの利用を用いたいくつかの方法で実現できるであろう。

7.5.8.1 名前付けられた複雑型（Named complex types）

XML スキーマ型の定義はインスタンスの宣言とは独立です。型定義は同じインスタンスのドキュメント内での異なるノードを記述するために異なるコンテキストにおいて再利用できます。この方法により、あなたは一貫性のある、型定義された方法で内容モデルを構築できます。

方針 10：名前付けられた複雑型はドキュメントの構造でオブジェクト指向クラスを表現するために重要である場合に限り使うべきである[should]。

新しい型は複雑型を宣言したり、拡張することによって名前付けられた複雑型から派生してできる。この導出は継承のように内容モデルにおける OO コンセプトを使うことが望ましい場合に有効である。しかし、名前付けられた型の利用はそのスキーマの開発者があるドキュメント構造においてクラスを表現した場合や継承のような OO コンセプトが要素の無いように必要とされる場合にのみ有効である。この節の名前付けられたモデルグループのところで説明したように。OO 原則がドキュメント構造で要求されない限り、名前付けられた複雑型は一緒に使うことが難しい。

方針 11：tML スキーマを読みやすくするため、型定義には<simpleType name="AlarmType">のように Type というキーワードをつけるべきである[should]。

方針 12：名前付けられ、再利用される複雑型とモデルグループは版数管理の問題のため、安定であるべきであり、変更しないようにするべきである[should]。

7.5.8.2 名前付けられたモデルグループ（Named model groups）

モデルグループは要素の内容モデルにおいて再利用されうる内容モデルである。モデルグループの内容モデルは内容モデルとしてそれ自身によって再利用されうるし、他の要素の内容モデルにおいても再利用されうる。ある場合では、モデルグループは名前付けられた複雑型定義と比較して、スキーマを構築するための再利用可能なビルディングブロックをつくるための推奨される方法であるかもしれない。複雑型は型の最後においてのみ拡張できる。他の匿名の複雑型またはモデルグループ内で入れ子にすることで名前付けられたモデルグループと共に内容モデルを拡張できる。

方針 13：名前付けられたモデルグループは tML スキーマにおける再利用可能なビルディングブロックを

作る主となる方法として考えるべきである[should]。

方針 14：tML スキーマを読みやすくするため、一つのグループの定義は<xsd:group name = “NameGroup”>.のように Group をキーワードとして添えるべきである[should]。

7.5.8.3　**名前付けられた属性グループ**（Named attribute groups）

***ガイドライン* 33：**属性のグループは名前が与えられた提供済みの複数の要素の中で再利用されてもよい[may]。

7.5.8.4　XML **スキーマの制約**（XML Schema constraints）

***ガイドライン* 34：**制約を実行する実装はスキーマのツールキットによって色々あるため、XML スキーマはスキーマ内でビジネスルールを実行するための制約を利用するべきではない[should not]。

7.5.8.5　**列挙**（Enumerations）

方針 15：tML スキーマにおいて列挙型の利用は良く知られているか、値の集合が安定しているものに限るべきである[should]。数え方の一般的なルールとして、もし正しい値の集合が頻繁に変わるのであれば、それは一つの属性に列挙するべきではない[should not]。

7.5.8.6　**型と要素**（Type vs. Element）

方針 16：tML スキーマの開発者は特定のアイテムが再利用されるかどうか疑わしいときは、要素のかわりに型を生成すべきである[should]。一つの要素または型は派生されることができるし、基本型から拡張されることもできる。

7.5.9　**相互接続性のテスト**（Interoperability testing）

***ガイドライン* 35：**ドラフト版の tML スキーマを承認する前の tML スキーマの相互接続性のテストに対するすべての要求は、その標準化団体における知られているワークグループでの裁量で行われなくてはならない[shall]。もし、ある標準化団体のあるワークグループが相互接続性のテストが必要であると考えた場合は、そのワークグループ内での参加者はインスタンスのドキュメントを提供するべきである。

7.5.10　tML ドキュメント—整形式（tML Document—well-formed）

***ルール* 28：**ある tML ドキュメントは本質的に一つの XML ベースのドキュメントである。ゆえに、W3C の XML 勧告に従った整形式でなくてはならない[shall]。

***ルール* 29：**ある tML ドキュメントは tML スキーマによって定義された定義の集合に準拠するべきである[will]。

Annex A　tML Schema Metadata　tML スキーマのメタデータ

このスキーマはすべての tML に準拠しているスキーマが含むべきメタデータの集合の定義を含んでいる。そのメタデータは完全なスキーマに関連していくつかの部分に分けられる。すなわち、この節とあるスキーマにおける要素に関連している部分である。このスキーマの目的のために、用語のコンポーネントは意味上の定義から得られるであろうあらゆるスキーマの構造的な要素を参照する。このスキーマ自体は tML フレームワーク(M.3030)に準拠する。

```
<?xml version="1.0" encoding="UTF-16"?>
<xsd:schema targetNamespace="urn:int.itu/tML/tMLSchemaMetadata"
xmlns:tML="urn:int.itu/tML/tMLSchemaMetadata" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
version="1.0">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-GB">
      <tML:SchemaMetadata>
        <tML:OriginalAuthor>Martin Roberts - BT plc</tML:OriginalAuthor>
        <tML:CreationDate>08-03-2002</tML:CreationDate>
        <tML:Description>This schema contains the definitions of the set of
                Metadata that all tML compliant schemas must contain.
                The metadata is divided into that which is associated with the complete schema, i.e.
                this section and that which is associated with a component within a schema.  For the
                purposes of this schema the term component refers to any schema structural element
                that might benefit from semantic definition.
                This schema is itself comformant to the tML Framework (M.tML)</tML:Description>
        <tML:Source>M.tML</tML:Source>
        <tML:SchemaHistory/>
      </tML:SchemaMetadata>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="SchemaMetadata">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="OriginalAuthor">
          <xsd:annotation>
            <xsd:documentation xml:lang="en-GB">
              <tML:ComponentMetadata>
                <tML:UID>tML000001</tML:UID>
                <tML:Definition>The creator of the original schema</tML:Definition>
                <tML:ComponentType>Basic</tML:ComponentType>
                <tML:Status>Active</tML:Status>
              </tML:ComponentMetadata>
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="CreationDate" type="xsd:date">
          <xsd:annotation>
            <xsd:documentation xml:lang="en-GB">
              <tML:ComponentMetadata>
                <tML:UID>tML000002</tML:UID>
                <tML:Definition>The date the Schema was created</tML:Definition>
                <tML:ComponentType>Basic</tML:ComponentType>
                <tML:Status>Active</tML:Status>
              </tML:ComponentMetadata>
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="Description">
          <xsd:annotation>
            <xsd:documentation xml:lang="en-GB">
              <tML:ComponentMetadata>
                <tML:UID>tML000003</tML:UID>
                <tML:Definition>A textual description of the intended purpose of the
schema</tML:Definition>
                <tML:ComponentType>Basic</tML:ComponentType>
                <tML:Status>Active</tML:Status>
              </tML:ComponentMetadata>
            </xsd:documentation>
          </xsd:annotation>
```

```
</xsd:element>
<xsd:element name="Source" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-GB">
      <tML:ComponentMetadata>
        <tML:UID>tML000004</tML:UID>
        <tML:Definition>The source of the schema e.g. an existing standard</tML:Definition>
        <tML:ComponentType>Basic</tML:ComponentType>
        <tML:Status>Active</tML:Status>
      </tML:ComponentMetadata>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SchemaHistory" minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-GB">
      <tML:ComponentMetadata>
        <tML:UID>tML000005</tML:UID>
        <tML:Definition>The full life change history of the schema</tML:Definition>
        <tML:ComponentType>Aggregate</tML:ComponentType>
        <tML:Status>Active</tML:Status>
      </tML:ComponentMetadata>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="VersionNumber">
        <xsd:annotation>
          <xsd:documentation xml:lang="en-GB">
            <tML:ComponentMetadata>
              <tML:UID>tML000006</tML:UID>
              <tML:Definition>The version of the schema after the change</tML:Definition>
              <tML:ComponentType>Basic</tML:ComponentType>
              <tML:Status>Active</tML:Status>
            </tML:ComponentMetadata>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="ChangeDate" type="xsd:date">
        <xsd:annotation>
          <xsd:documentation xml:lang="en-GB">
            <tML:ComponentMetadata>
              <tML:UID>tML000007</tML:UID>
              <tML:Definition>The date the change was completed</tML:Definition>
              <tML:ComponentType>Basic</tML:ComponentType>
              <tML:Status>Active</tML:Status>
            </tML:ComponentMetadata>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="Reason">
        <xsd:annotation>
          <xsd:documentation xml:lang="en-GB">
            <tML:ComponentMetadata>
              <tML:UID>tML000007</tML:UID>
              <tML:Definition>The reason for the change</tML:Definition>
              <tML:ComponentType>Basic</tML:ComponentType>
              <tML:Status>Active</tML:Status>
            </tML:ComponentMetadata>
          </xsd:documentation>
```

```
                </xsd:annotation>
              </xsd:element>
              <xsd:element name="ChangedBy">
                <xsd:annotation>
                  <xsd:documentation xml:lang="en-GB">
                    <tML:ComponentMetadata>
                      <tML:UID>tML000008</tML:UID>
                      <tML:Definition>The person or organisation who performed the
change</tML:Definition>
                      <tML:ComponentType>Basic</tML:ComponentType>
                      <tML:Status>Active</tML:Status>
                    </tML:ComponentMetadata>
                  </xsd:documentation>
                </xsd:annotation>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="ComponentType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-GB">
        <tML:ComponentMetadata>
          <tML:UID>tML000009</tML:UID>
          <tML:Definition>A enumerated list used to indicate the type of component.</tML:Definition>
          <tML:Description>Aggregate means the component is made up of other components.
                Basic means the component is atomic
              A component is any structural part of a schema.</tML:Description>
          <tML:ComponentType>Basic</tML:ComponentType>
          <tML:Status>Active</tML:Status>
        </tML:ComponentMetadata>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Basic"/>
      <xsd:enumeration value="Aggregate"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="ComponentMetadata">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-GB">
        <tML:ComponentMetadata>
          <tML:UID>tML000010</tML:UID>
          <tML:Definition>The set of metadata required for any component</tML:Definition>
          <tML:ComponentType>Aggregate</tML:ComponentType>
          <tML:Status>Active</tML:Status>
        </tML:ComponentMetadata>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:annotation>
          <xsd:documentation xml:lang="en-GB">
            <tML:ComponentMetadata>
              <tML:UID>tML000011</tML:UID>
              <tML:Definition>A unique identifier of the component</tML:Definition>
              <tML:Description>This will become the primary key for the component
                  within the Global Telecommunications Data Dictionary.
```

```
                    It is made up of two parts:
                        - three alpha-numeric character indicating the group responsible
                          for component, such as an Standards Development Organisation.
                        - A six digit number.</tML:Description>
                    <tML:ComponentType>Basic</tML:ComponentType>
                    <tML:Status>Active</tML:Status>
                  </tML:ComponentMetadata>
                </xsd:documentation>
              </xsd:annotation>
              <xsd:element name="UID"/>
              <xsd:element name="Definition">
                <xsd:annotation>
                  <xsd:documentation xml:lang="en-GB">
                    <tML:ComponentMetadata>
                      <tML:UID>tML000012</tML:UID>
                      <tML:Definition>A short sentence indicating the semantic meaning of the
component</tML:Definition>
                      <tML:ComponentType>Basic</tML:ComponentType>
                      <tML:Status>Active</tML:Status>
                    </tML:ComponentMetadata>
                  </xsd:documentation>
                </xsd:annotation>
              </xsd:element>
              <xsd:element name="Description" minOccurs="0">
                <xsd:annotation>
                  <xsd:documentation xml:lang="en-GB">
                    <tML:ComponentMetadata>
                      <tML:UID>tML000013</tML:UID>
                      <tML:Definition>An optional textual description of the component.</tML:Definition>
                      <tML:Description>This may include examples and general remarks</tML:Description>
                      <tML:ComponentType>Basic</tML:ComponentType>
                      <tML:Status>Active</tML:Status>
                    </tML:ComponentMetadata>
                  </xsd:documentation>
                </xsd:annotation>
              </xsd:element>
              <xsd:element name="Source" minOccurs="0">
                <xsd:annotation>
                  <xsd:documentation xml:lang="en-GB">
                    <tML:ComponentMetadata>
                      <tML:UID>tML000014</tML:UID>
                      <tML:Definition>The source of the component</tML:Definition>
                      <tML:Description>This would only be included alongside the component
                              if the source is different from the source of indicated in the
                               SchemaMetadata</tML:Description>
                      <tML:ComponentType>Basic</tML:ComponentType>
                      <tML:Status>Active</tML:Status>
                    </tML:ComponentMetadata>
                  </xsd:documentation>
                </xsd:annotation>
              </xsd:element>
              <xsd:element name="Type" type="tML:ComponentType" default="Aggregate">
                <xsd:annotation>
                  <xsd:documentation xml:lang="en-GB">
                    <tML:ComponentMetadata>
                      <tML:UID>tML000015</tML:UID>
                      <tML:Definition>The indicator of whether the component is Aggregate or
Basic</tML:Definition>
                      <tML:ComponentType>Basic</tML:ComponentType>
                      <tML:Status>Active</tML:Status>
```

```
                </tML:ComponentMetadata>
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
          <xsd:element name="Status" type="tML:StatusType">
            <xsd:annotation>
              <xsd:documentation xml:lang="en-GB">
                <tML:ComponentMetadata>
                  <tML:UID>tML000016</tML:UID>
                  <tML:Definition>The status of the component with regards to the life cycle of a
component</tML:Definition>
                  <tML:Description>The actual values of this component will be defined by the GTDD
project.
                        The only allowed values until this point are Active and
inActive</tML:Description>
                  <tML:ComponentType>Basic</tML:ComponentType>
                  <tML:Status>Active</tML:Status>
                </tML:ComponentMetadata>
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="StatusType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en-GB">
          <tML:ComponentMetadata>
            <tML:UID>tML000017</tML:UID>
            <tML:Definition>A enumerated list indicating the allowed values of the a
Status</tML:Definition>
            <tML:Description>Active means that the definition is still in use.
                  Inactive means that the definition is either not fully agreed or is no longer to be used.
                   Inactive terms are retained for backward compatability</tML:Description>
            <tML:ComponentType>Basic</tML:ComponentType>
            <tML:Status>Active</tML:Status>
          </tML:ComponentMetadata>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Active"/>
        <xsd:enumeration value="Inactive"/>
      </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```