HMRC CT Inline XBRL Style Guide

Contents

1. Introduction

The Inline XBRL Style Guide is intended to assist commercial and in-house software developers engaged in the process of producing applications that create one or both of the Inline XBRL components of an online Company Tax Return (i.e. a Tax Computation and/or a set of Statutory Accounts). It does not cover information relating to the Company Tax Return itself (the CT600 and its supplementary pages) or the mechanisms of online filing – for these, please refer to the 'Tech Pack' for CT online filing, published by HMRC's Software Developer Support Team (SDST) at http://www.hmrc.gov.uk/ebu/ct-tech-index.htm.

The information and guidance presented here is not mandatory (except where it relates to requirements of the Specification itself or the HMRC online service) but is intended to assist developers in producing Inline XBRL documents that are as complete and useful as possible.

This Guide should be read in conjunction with the Inline XBRL 1.0 Background Information and Guidance document produced by the XII[1] Rendering Working Group, which is intended as a non-normative companion to the Specification itself – the background document can be found at:
http://www.xbrl.org/Specification/inlineXBRL/REC-2010-04-20/inlineXBRL-background-REC-2010-04-20.html

In addition, this Guide builds upon, and is consistent with, the information contained in the XBRL UK Preparers and Developers Guide, which is targeted at those creating financial reports (in particular company accounts) in XBRL in the United Kingdom.

2. Conformance with the Inline XBRL Specification

The HMRC CT online service is conformant with the Final Recommendation (April 2010) of the Inline XBRL Specification, version 1.0, which can be found at:
http://www.xbrl.org/Specification/inlineXBRL-part1/REC-2010-04-20/inlineXBRL-part1-REC-2010-04-20.html

The XII Rendering Working Group (the body that controls the Specification) and HMRC have taken care to ensure that any changes from the 4th Candidate Recommendation onwards (originally introduced into the CT service in November 2009) are backwards-compatible so that Inline XBRL production software already "in the field" is not invalidated by new Recommendations.

---

[1] XBRL International Inc. A consortium of over 600 companies, government agencies and accountancy firms that oversees and manages the publication of the XBRL "family" of Specifications.

3. Document level Issues

3.1 Multiple input documents

The vast majority of Tax Computations and Accounts supplied by vendor applications are expected to be single, self-contained Inline XBRL documents that can be opened and viewed in a single browser window. However, for practical reasons, multiple Inline XBRL documents (sometimes referred to as an Inline XBRL Document Set, or IXDS) are allowed by the service (the CT600 XML Schema permits multiple iXBRL instance documents to be presented in a single submission for each of the Tax Computation and the Accounts).

The potential for generating "oversized" Inline XBRL documents is the only reason to create a document set as opposed to a single Inline XBRL document. Experience has shown that XHTML documents larger than about 5mb in size present difficulties for most current web browsers, so for the sake of your customers and HMRC staff, arrange to break up Inline XBRL documents that may approach this size.

There are however some extra things to think about with multiple input documents. First, give due consideration to navigation between documents – you may have a summary document with links to all the detailed parts, or you may simply chain two or more documents together. Either way, navigation links must be implemented with relative URLs so that they continue to function correctly within HMRC's environment - absolute URLs containing your customer's domain are not going to resolve properly. It is best to avoid sub-directories and keep the collection of documents in a single directory – that way the navigation URLs only depend on the preservation of the original filenames, which will have to be supplied as attributes (see the CT600 XML Schema for details).

One, and only one, of the documents in the set will have to be nominated as the "entry point" via an attribute on its CT600 *<InlineXBRLDocument>* or *<EncodedInlineXBRLDocument>* element. This is the document that you want the user to see first when opening and viewing the document set. Multiple input documents with no nominated "entry point" will be rejected, as will multiple input documents with more than one nominated "entry point" (for single input documents there is no need to provide the "entry point" attribute at all).

3.2 Single output document

The result of processing one, or a set of related, Inline XBRL "input" document(s) in a submission is always a single target XBRL instance document. Multiple target XBRL instance documents are not supported by the service, despite being permitted by the Inline XBRL Specification.

The consequence is that the service will always derive a single XBRL instance document for the Tax Computation (where supplied) and a single XBRL instance document for the Accounts (where supplied) even if those instance documents refer to concepts defined in multiple Taxonomies (e.g. when a private extension Taxonomy is supplied).

3.3 XHTML vs HTML

The Inline XBRL Specification is designed to allow Inline XBRL to work with either HTML or XHTML. The choice of XHTML for the HMRC CT online service is made for us by

the UK Govt Gateway, which expects submission payloads to be well-formed XML. Only XHTML meets this requirement.

As a consequence you should ensure that the <html> root element of your Inline XBRL document is in the XHTML namespace. This is usually achieved by setting the default namespace for the document, thus:

```
xmlns='http://www.w3.org/1999/xhtml'
```

in addition to all the other namespace declarations required for Inline XBRL use. In particular, an Inline XBRL document is recognised by the presence of one or more elements that have the "http://www.xbrl.org/2008/inlineXBRL" namespace name.

A DOCTYPE declaration for Inline XBRL documents **must** be omitted – in many processing situations, including the HMRC CT online service, this will lead to an unsuccessful attempt to validate the document against a DTD, resulting in failure. A full modular Schema for the Inline XBRL extension to XHTML is available, a full Inline XBRL DTD is not.

The XHTML *version* attribute can be used to identify a document as Inline XBRL, but its use is optional. If used, the value **must** be the Formal Public Identifier "-//XBRL International//DTD XHTML Inline XBRL 1.0//EN".

The use of XHTML also allows more rigorous checking of the mark-up (including the Inline XBRL mark-up elements) against the XHTML modular Schema, reducing the likelihood of HMRC accepting Inline XBRL that renders slightly differently in different browsers.

3.4   Document Encoding & Use Of "non-standard" Characters

The UK Govt Gateway expects submission payloads to be UTF-8[2] encoded, and so Inline XBRL documents submitted to the HMRC CT online service must also be UTF-8 encoded by default.

If your Inline XBRL document is to be used standalone (e.g. as a saved version of the submitted document) then it is advisable to provide an initial XML declaration that explicitly sets the document encoding to UTF-8, thus:

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

This will ensure that web browsers treat the document as intended and render any embedded, multi-byte UTF-8-encoded characters correctly. HMRC systems will always treat the content as UTF-8 encoded, so this ensures that character renderings are consistent on both sides of the fence.

When embedding an Inline XBRL document in the CT600 *<InlineXBRLDocument>* or *<EncodedInlineXBRLDocument>* element the XML declaration (and any leading Byte-Order Mark or BOM) must be removed. The complete XML transaction (rooted at *<GovTalkMessage>*) should have an initial XML declaration containing 'encoding="utf-8"' which will apply to the transaction document as a whole.

---

[2] 8-bit UCS/Unicode Transformation Format – see http://www.unicode.org, http://wikipedia.org/wiki/Unicode and http://wikipedia.org/wiki/UTF-8

There are two principal methods of including "non-standard" (a euphemism for characters that fall outside of the ASCII 7-bit character set) characters in Inline XBRL documents (such characters may include currency symbols, simple graphic symbols, accented characters, etc). The first is to use the appropriate UTF-8 encoding, which will usually (but not always) be a two-byte value, both of which will have their top (i.e. 8th) bit set. So long as the document encoding is correctly set then browsers or other display tools will be able to interpret these bytes correctly as the appropriate character. However, be aware that text editors and other display tools that don't understand document encodings and treat the content as 7-bit ASCII will not display UTF-8 encoded characters correctly, and if used to modify a document with embedded UTF-8 encoded characters may actually destroy or alter the encoding such that it no longer renders correctly in UTF-8-aware applications.

To avoid such eventualities, if needs be, the second method may be employed. XML (of which XHTML is an application) provides for escaped character references of the form "&#xxxx;" where "xxxx" may be replaced by a decimal number representing the Unicode code point (see http://wikipedia.org/wiki/Code_point for more details) of the character glyph to be displayed. For example, here are some common symbol glyphs and their character reference equivalents:

£ (pound sign)	£
€ (euro sign)	€
© (copyright)	©

Such code point values are also encoded into the bytes of a UTF-8 encoded multi-byte character – you can think of an XML character reference and a UTF-8-encoded character as two alternative ways of specifying a Unicode code point, which in turn represents a particular character glyph.

The advantage of the second method is that non-UTF-8-aware text editors and text display tools will not disrupt or damage the character references, though they will not be displayed as intended.

The exceptions to the rule are the five characters special to XML mark-up: '<', '>', '&', ''' and '"'. Their ASCII character representations are UTF-8 compatible, so to protect them from interpretation by XML-aware processors, XML provides five named character entities to escape them ("<", ">", "&", "'" and """). You can use these named character entities in preference to the five equivalent numeric character references if you wish.

It is recommended that you avoid using named character entities aside from these five "special" ones. Named character entities are usually defined in a DTD (Document Type Definition) and the XHTML DTD defines quite a number of them. However, Inline XBRL does not have a fully defined DTD (the preference being for the modular XHTML Schema) meaning that the interpretation of these named entities becomes browser dependent.

3.5   Java Applets, JavaScript

Java applets, JavaScript, and any other HTML *<script>* content, is embedded code executed by a browser or other rendering engine, and presents an unacceptable security risk for HMRC internal systems. It is therefore not possible to submit Inline XBRL

documents with embedded applets or script fragments of any kind either to animate the document or enhance the look & feel of the document.

Tax Computations and Accounts are not documents that have traditionally been adorned with dynamic content or sophisticated presentation (at least not for regulatory purposes!). House styles or particular look & feels are best achieved with CSS – see below.

3.6   CSS styling

Cascading Style Sheet (CSS) language is the *de facto* standard means by which stylistic information is added to the rendering of (X)HTML documents by web browsers and users of the HMRC CT online service have free rein to style Inline XBRL documents using it in any way they see fit.

If CSS is required it must be embedded in the Inline XBRL document directly. It is not possible to provide a separate (.css) file for inclusion, even with multiple Inline XBRL input documents. In these cases, duplicate the CSS declarations as necessary in each input document. If you are using multiple Inline XBRL input documents because of the size of the submission document the duplicate CSS script is hardly going to make matters worse.

3.7   Pagination

Inline XBRL documents submitted to the HMRC CT online service will not be printed within HMRC as a matter of course, but there will be occasions when it is appropriate or necessary to do so. To facilitate this, page breaks should be inserted at appropriate points in the Inline XBRL document to ensure that printed output looks as neat and as presentable as possible. Page breaks should be inserted before major headings, tables, front-sheets, etc. to ensure that, as much as possible, important information isn't spread over a physical page boundary. These breaks should mirror the page breaks inserted by generating applications when producing local renderings in other formats (e.g. plain text, PDF, Word).

In XHTML, "page break before" and "page break after" styles can be applied to mark-up elements directly, thus:

```
<p style="page-break-after: always">……</p>          (1)

<table style="page-break-before: always">……</table>    (2)
```

They can also be included in macros and bound to elements of a particular class. Example (1) above forces a page-break *after* the content between the paragraph tags has been output, and example (2) above forces a page-break *before* the table is output. For a full list of page-breaking styles, their values and their resulting behaviours please refer to documentation or reference material for CSS.

3.8   Images, logos, etc

Within the HMRC environment there is no scope to refer to external image files by URL, so any logos or small images that are integral to the look & feel of a submitted Inline XBRL document should be inlined as a base64-encoded string in the URI value normally used to reference an image file using the 'data' URI scheme. E.g.:

```
<img src="data:image/png;base64,xxx…xxx" alt="TestCo Ltd logo" />
```

where 'xxx…xxx' is the base64-encoding of the image whose mime-type is 'image/png' (GIF, JPG and PNG are the only widely supported image formats).

This technique is manageable with relatively small images, logos, etc., but should be used sparingly to avoid bloating the overall Inline XBRL document. If an image is required in more than one location it is possible to create a CSS macro utilising the background:url style and passing it the inlined URI. E.g. (assuming a 100x100 pixel image):

```
DIV.img {
    width:100px;
    height:100px;
    background:url(data:image/png;base64,xxx…xxx);
}
```

This technique ensures that only one physical instance of a base64-encoded image is included in an Inline XBRL document. Support for inlined images in web browsers is widespread but not universal – check your intended target browser audience carefully before embarking upon the use of this feature.

3.9   XML Canonicalisation & Encoding Submissions

Since XHTML is legal XML it will canonicalise (for the purposes of HMRC IRmark creation) successfully, though as with other HMRC services that support IRmark, the explicit canonicalisation step can, with care, simply be avoided by generating canonical XML (XHTML) in the first place.

With large Inline XBRL (i.e. XHTML) documents that cannot be generated in canonical form the performance of client-side canonicalisation may leave something to be desired – if this is the case the canonicalisation step can be avoided, at the cost of an encoding step, by using the CT600 XML *<EncodedInlineXBRLDocument>* element in preference to the *<InlineXBRLDocument>* element and providing a base64-encoded string version of the Inline XBRL document instead. Base64-encoded Inline XBRL documents are decoded by the HMRC CT online service and treated identically to "ordinary" Inline XBRL documents (except where the Inline XBRL document is not well-formed XML – an error will be raised by HMRC's validation system rather than by the Govt Gateway). Take care not to include any Byte-Order Mark (or BOM) that may be present when encoding – its removal is the responsibility of the encoding/filing application.

Inline XBRL introduces a number of extra and unfamiliar namespace declarations and attributes – care should be taken to make sure that all namespace declarations and attributes are ordered correctly and all superfluous namespace declarations are removed (see the document ordering section of the XML Canonicalisation Spec: http://www.w3.org/TR/xml-c14n#DocumentOrder for details).

3.10   HTML <title> Element

The (X)HTML <title> element (placed in the document <head> part) provides an opportunity for a document author (or creator application) to set the title of the window in which the document is displayed in a typical desktop window-based operating system – the title is usually displayed in the header or title bar of the window, and may be used to identify the window in other contexts (menus of open windows, context switchers,

etc). It is therefore desirable that the browser (or other application) window is readily identifiable amongst a number of other, potentially similar, open windows.

We recommend, therefore, that you avoid static and innocuous-sounding titles (such as "iXBRL doc" or "Computation") and use a combination of the Company Name and the financial report type, e.g.:

```
<title>ACME Boats, Ltd – Computation</title>
```

or

```
<title>ACME Boats, Ltd – Accounts</title>
```

This will ensure document windows can be quickly located and uniquely identified by your users (who may be agents dealing with a number of customers) and HMRC staff alike.

3.11   Hidden data items

Inline XBRL provides a mechanism (*ix:hidden*) for marking up data items that do not appear on the face of a document. This mechanism should be used as sparingly as possible – the more data that is "hidden" the harder it is to assure that the human and computer readable data items in the document are aligned – when they are one and the same value, alignment is assured.

The *ix:hidden* section of an Inline XBRL document should be reserved for:

- The declaration of tuple container elements
- Document metadata, including creator information (see §5.3)
- Boolean items not directly associated with statements on the face of the document (see §4.9)
- "Flag" items not directly associated with facts on the face of the document (see §4.10)
- Facts that do not normally appear on the face of the document

Furthermore, the use of HTML or CSS text hiding mechanisms other than that conventionally used to protect the *ix:hidden* section from view or implement the Boolean item mark-up technique described in §4.9 are strongly discouraged.

If in doubt follow the general principle that if a fact appears on the face of a document then it should be marked-up there and not by any method that involves the hiding of data and mark-up.

4. Element level Issues

4.1   Transformation rules

The Inline XBRL Transformation Rules Registry, which is Part 3 of the Inline XBRL Specification, defines all the transformation rules that are applied to displayed fact values in an Inline XBRL document to turn them into canonical XSD type representations suitable for inclusion in the resulting XBRL target document. For completeness, the current transformation rules, at the time or writing, are enumerated here with an explanation of the effect of each one:

- numcomma   Re-formats numbers using comma as fraction separator into XSD non-negative decimal type
- numcommadot   Re-formats numbers using commas as thousands separator and dot as fraction separator into XSD non-negative decimal type
- numdash   Re-formats accounting notation '-' as value zero in XSD non-negative decimal type
- numdotcomma   Re-formats numbers using dot as thousands separator and comma as fraction separator into XSD non-negative decimal type
- numspacedot   Re-formats numbers using space as thousands separator and dot as fraction separator into XSD non-negative decimal type
- numspacecomma   Re-formats numbers using space as thousands separator and comma as fraction separator into XSD non-negative decimal type
- dateslashus   Re-formats a US-style slash-separated date (MM/DD/(YY)YY) into XSD date format
- dateslasheu   Re-formats a EU-style slash-separated date (DD/MM/(YY)YY) into XSD date format
- datedotus   Re-formats a US-style dot-separated date (MM.DD.(YY)YY) into XSD date format
- datedoteu   Re-formats an EU-style dot-separated date (DD.MM.(YY)YY) into XSD date format
- datelongus   Re-formats a US-style long date (Month DD, (YY)YY) into XSD date format
- datelonguk   Re-formats a UK-style long date (DD Month (YY)YY) into XSD date format
- dateshortus   Re-formats a US-style short date (mon DD, (YY)YY) into XSD date format
- dateshortuk   Re-formats a UK-style short date (DD mon (YY)YY) into XSD date format

Note that the Registry may grow over time as new transforms are added, and you are advised to check the latest version for an up to date list.

For all versions of the Inline XBRL Specification prior to the final Recommendation the namespace for the Transformation Rules Registry was:

http://www.xbrl.org/2008/inlineXBRL/transformation

The Transformation Rules Registry namespace for the final Recommendation is:

http://www.xbrl.org/inlineXBRL/transformation/2010-04-20

The change to introduce a date component was intended to allow future updates to the Registry to be separately versioned and identified. To ensure backwards-compatibility the unversioned namespace will continue to be supported for a period of at least a year from April 2011 to allow for Inline XBRL generation software to be migrated.

4.2   Duplication of marked-up facts

4.2.1   De-duplication

The Inline XBRL Specification provides a mechanism to allow conforming processors to de-duplicate identical tuple-member facts prior to XBRL generation to avoid potential violations of a tuple's content model. This allows document creators to mark-up *all* duplicate facts regardless of their status as tuple members or independent concepts. Duplicate facts that are not tuple members may, as a consequence, be present in the target XBRL document, but this is not illegal XBRL.

4.2.2   Equivalence

Identical tuple-member facts are identified as such by a simple string equivalence test – fact values, irrespective of their type, are compared, character by character, after being *whitespace-normalised*. Whitespace normalisation, which is equivalent to the XML Schema *whiteSpace* facet being set to *collapse,* involves the substitution of tabs, linefeeds and carriage returns by the space character, and the subsequent conversion of multiple spaces to a single space. Finally, any leading or trailing spaces are eliminated. For numeric data this kind of normalisation will usually have no effect, but for text it ensures that insignificant whitespace (line breaks, double spaces, etc) is generally disregarded for comparison purposes.

This kind of simple-minded equivalence test avoids the need for Inline XBRL processors to be aware of the type of the values being compared (becoming Taxonomy-aware would add significant overhead to the performance of Inline XBRL processors) but there are potential pitfalls – numbers expressed to different levels of precision, or in different mathematical notations, will not appear to be equivalent when semantically they are.

4.3   Escaped HTML in non-numeric data

The Inline XBRL Specification allows textual data in a nonNumeric marked-up fact to contain 'escaped' (X)HTML formatting. This is not a feature of the Specification that will be supported by the HMRC CT online service.

Normally, any (X)HTML mark-up in a nonNumeric element will be eliminated, and leading and trailing whitespace removed, before transformation to the target XBRL instance document value. (X)HTML mark-up in any other Inline XBRL element is not permitted.

4.4   Number Signs

Numbers represented in XBRL (and Inline XBRL) are generally positive in value, notwithstanding the value of the 'balance' attribute in the concept's definition, which is

usually of use only in calculation relationships. Where a genuine negative value is required in the target XBRL document (e.g. to show a loss for a concept that is defined in terms of profit) the 'sign' attribute in the Inline XBRL mark-up should be used to indicate this.

The representation of the sign in the Inline XBRL document rendering is completely separate and is a function of the XHTML mark-up that surrounds the marked-up fact. That mark-up might indicate a negative value with surrounding brackets, or by a leading minus sign, or by changing the font to red (or a combination of these) – the point is that none of these adornments, by themselves, have any influence over the sign of the value that ends up in the target XBRL document, which by default will be positive unless the sign attribute is used as an indicator.

The following examples should illustrate the differences[3]:

a.   *Negative underlying value with negative portrayal*

The monetary fact representing a loss in a statement of the form:

> Profit (Loss) before Tax   (345,678)

in the Inline XBRL rendering of a set of Accounts might be marked up in the Inline XBRL document as a table data item as follows:

```
<td>(<ix:nonFraction name="PBT" unitRef="gbp" sign="-"
   format="ixt:numcommadot">345,678</ix:nonFraction>)</td>
```

Note that the brackets are outside the Inline XBRL mark-up so as not to interfere with the value that is transformed for the target XBRL instance document. The presence of the *sign* attribute results in a negative value in the resulting XBRL instance document that correctly indicates that the value represents a loss rather than a profit:

```
<PBT unitRef="gbp">-345678</PBT>
```

The item *PBT* would normally be defined in the Taxonomy with its *balance* set to 'credit' to indicate that where a profit is reported, it is naturally thought of as an increase in funds (i.e. a credit), and so a loss must be explicitly negated.

b.   *Positive underlying value with negative portrayal*

The monetary fact representing cost of sales in a statement of the form:

> Cost of Sales   (123,456)

in the Inline XBRL rendering of a set of Accounts might be marked up in the Inline XBRL document as a table data item as follows:

```
<td>(<ix:nonFraction name="CostSales" unitRef="gbp"
    format="ixt:numcommadot">123,456</ix:nonFraction>)</td>
```

Note that the brackets are outside the Inline XBRL mark-up so as not to interfere with the monetary value that is transformed for the target XBRL instance document.

---

[3] Items names deliberately shortened and stripped of namespace prefixes, and some attributes omitted, to avoid complicating the examples.

However, in this case, the lack of a sign attribute results in a positive value in the resulting XBRL instance document that correctly indicates that the value represents a cost amount:

```
<CostSales unitRef="gbp">123456</CostSales>
```

The item *CostSales* would normally be defined in the Taxonomy with its *balance* set to 'debit' to indicate that where a cost is reported it is naturally thought of as a decrease in funds (i.e. a debit) and so doesn't need to be explicitly negated.

c.   Negative underlying value with positive portrayal

The monetary fact representing a cost of disposal in a statement of the form:

Disposal Cost   45,678

in the Inline XBRL rendering of a set of Accounts might be marked up in the Inline XBRL document as a table data item using a revenue tag as follows:

```
<td><ix:nonFraction name="DisposalRevenue" unitRef="gbp" sign="-"
    format="ixt:numcommadot">45,678</ix:nonFraction></td>
```

Note that in this (slightly contrived!) example the cost is being tagged as a negative revenue. The presence of the sign attribute results in a negative value in the resulting XBRL instance document that correctly indicates that the value represents a cost amount:

```
<DisposalRevenue unitRef="gbp">-45678</DisposalRevenue>
```

The item *DisposalRevenue* would normally be defined in the Taxonomy with its *balance* set to 'credit' to indicate that where revenue is reported it is naturally thought of as an increase in funds (i.e. a credit) and so in this case needs to be explicitly negated to show a cost.

4.5   Scaling and Precision

Numbers may be conveniently scaled in human-readable reports to show, for instance, figures in round thousands or millions. When generating human-readable output programatically it is of course straightforward to ensure that if scaling is required the correct precision is declared and the correct rounding is applied to the value to ensure that all occurrences of the same fact (whether appearing scaled or not) remain equivalent within the limits of their declared precisions.

For instance, an underlying value such as £1,234,567.89, which is accurate to 2 decimal places, may appear rounded to the nearest pound with a *decimals* value of '0' (i.e. £1,234,568), rounded to the nearest thousand with a *decimals* value of '-3' and a *scale* value of '3' (i.e. £1,235), or rounded to the nearest tenth of a million with a *decimals* value of '-5' and a *scale* value of '6' (i.e. £1.2). The latter, when un-scaled and converted to canonical form for XBRL would result in a value of £1,200,000 (accurate to -5 decimal places), which nevertheless is "equivalent" to £1,234,567.89 (accurate to 2 decimal places) within their stated accuracies. However, any 'higher' value for *decimals* (e.g. '-4') would result in an inconsistency.

Care, however, should exercised when manually marking up pre-existing financial reports. The original author will have applied the appropriate scaling and rounding

(correctly, one hopes) and it is the responsibility of the user to identify the correct scale factor to declare – from this and the value being marked up, it should be possible for manual tagging applications to determine the necessary *decimals* value to set an appropriate precision. If rounding has been mis-applied to one or more occurrences of duplicate values in the original document (as may occur when summarising more accurate financial data from the body of a report) then the duplicate values may be inconsistent within the bounds of their stated accuracies. Such inconsistencies may result in submission rejection or increased scrutiny during risk analysis within HMRC.

4.6   Percentages

The conventional representation of a percentage value in an Inline XBRL document can be achieved easily, whilst still preserving the underlying meaning of the value as a decimal fraction (for calculation relationships, etc). However, in presentation terms, the choice of one or the other is down to the preference of the preparer (or the preparer's software).

The Inline XBRL *scale* attribute can be used to effect a transformation from the conventional representation in the Inline XBRL document to a decimal fraction in the target XBRL instance document. A scale of '-2' will transform a value such as '28%' into the equivalent decimal fraction '0.28'. A unit based on 'pure' should be applied to any percentage values. For example, given the following line item in an Inline XBRL rendering:

First Year tax Rate   28%

the value 28% is marked up as follows:

```
<td><ix:nonFraction name="Year1Rate" contextref="AP1"
      unitref="PureUnit" decimals="2"
      scale="-2">28</ix:nonFraction>%</td>
```

Note that the percent sign itself occurs outside of the Inline XBRL mark up. Also note that as per the previous section on Scaling and Precision, a *decimals* value of 2 has been set, which is appropriate for the two significant decimal places of the target value 0.28 – it bears no relationship to the pre-scaled value that appears in the rendering.

4.7   Nesting Mark-up

The Inline XBRL Specification allows the nonNumeric element to contain other Inline XBRL mark-up so that where narrative text is marked up as a single block of text within a single string-type item using *ix:nonNumeric* it is still possible to mark-up embedded facts so that they appear as discrete items in the target XBRL instance document, in addition to the item representing the text block as a whole. This is particularly useful where a value or a significant name is mentioned in the middle of a paragraph of text (such as the Director's Report or one of the Notes to the Accounts).

For example, the description of a contingent liability with an embedded current period liability figure might be marked up as follows:

```
<td>
 <ix:nonNumeric name="DescriptionLiability" contextref="CY">The company
 operates a confidential invoice discount facility. The balance due on
 this facility was £<ix:nonFraction name="AmountLiability"
 contextref="CY" unitref="GBP"
 format="ix:numcommadot">5,000</ix:nonFraction> at 31st March
 2010.</ix:nonNumeric>
```

```
</td>
```

This would result in <u>two</u> distinct items in the target XBRL document:

```
<DescriptionLiability contextref="CY">The company operates a confidential
invoice discount facility. The balance due on this facility was £5,000 at
31st March 2010.</DescriptionLiability>
```

and

```
<AmountLiability contextref="CY" unit="GBP">5000</AmountLiability>
```

Note that the figure of £5,000 is repeated, once in its original format as part of the text description item (as if the *ix:nonFraction* mark-up was not there) and once in its canonical form in the numeric item.

This technique can also be used to avoid the necessity for hidden duplicate description items where tuples are used to mark up tabular data. In the following table a single description relates to two separate monetary values for current and previous periods:

	2010	*2009*
Line Description	**£100**	**£200**

This may need to be marked-up as two separate tuples – one for each period, with the description repeated for each one. However, only one description appears on the face of the document. The solution is to nest the mark-up of the description:

```
<td>
 <ix:nonNumeric name="DescLine" contextref="CY" tupleref="t1"
 order="1"><ix:nonNumeric name="DescLine" contextref="PY" tupleref="t2"
 order="1">Line Description</ix:nonNumeric> </ix:nonNumeric>
</td>
<td>
 <ix:nonFraction name="ValueLine" contextref="CY" unitref="GBP"
 tupleref="t1" order="2">100</ix:nonFraction>
</td>
<td>
 <ix:nonFraction name="ValueLine" contextref="PY" unitref="GBP"
 tupleref="t2" order="2">200</ix:nonFraction>
</td>
```

This results in <u>four</u> items, two in each tuple, in the target XBRL instance document. Each tuple contains a description item and a value item (in that order) where the description item content is repeated in both.

Note that the contextrefs for the items in tuple 't1' are both "CY", and for the items in tuple 't2' are both "PY". This is the preferred arrangement (as opposed to the description item in one or both cases referring to the alternate contextref).

4.8 Tuple Member Ordering

The Inline XBRL Specification deliberately avoids the need for conformant processors to be Taxonomy-aware. As a consequence, an Inline XBRL processor is not aware of the constraints of the content model of a Taxonomy tuple definition. It is therefore up to the creator of the Inline XBRL mark-up to explicitly provide information about the order in which tuple members are to appear in the target XBRL instance document (the order of tuples themselves and other primary items is, of course, irrelevant).

The 'order' attribute exists to carry this information and it can take any decimal value. The output order of tuple members is imposed by the natural ordering relation defined by decimal values of increasing magnitude - it is not necessary for the values to be consecutive integers. This allows an application to assign static order values to each member of a tuple, corresponding with the order defined by the ComplexType definition in a Taxonomy, regardless of whether every tuple member appears in the output.

The fractional part of the decimal value can be used to represent the order of members in a tuple nested inside another tuple. For example, if a tuple were the 4th child member of an enclosing tuple its member content could be ordered thus: 4.1, 4.2, 4.3, etc – in terms of the sub-tuple the constant integer part is irrelevant; the ordering of members is defined by the fractional part.

4.9   Marking up Boolean items

There are a number of Boolean-type concepts in the UK Accounts Taxonomies that correspond to statements that by convention appear on the face of the document: claims for exemptions (from the need for an audit for instance), statements of fact ("the company is trading"), etc. Aligning the mark-up of these statements with the value that is to appear in the target XBRL instance (i.e. "true" or "false") can be problematic, especially for conversion or manual tagging tools.

A medium-term solution to this is in the process of being standardised by the XII Rendering Working Group in the form of a pair of Transformation Rules Registry transforms called *ixt:booleantrue* and *ixt:booleanfalse*, but these will not be available in the CT online service for some time. The intention of these transforms is to allow some arbitrary text (the true/false statement) to be marked-up as normal, but for the text to be transformed to either "true" or "false" in the target XBRL instance, depending on which transform is applied. This should allow such mark-up to be undertaken in the same way as any other value that might require a transform, and avoid special-purpose code in tagging tools designed only to handle Boolean items.

In the meantime, the *ix:exclude* element can be used to simulate this behaviour at the cost of a little additional complexity in the mark-up (which nevertheless follows a pattern that applications can easily apply).

If we assume that the sentence "The company is trading." appears on the face of the document then the following "pattern" will allow the Inline XBRL mark-up of this text with a Boolean item such as *uk-bus:EntityTrading*:

```
<ix:nonNumeric name="uk-bus:EntityTrading contextRef="CY">
    <ix:exclude>The company is trading.</ix:exclude>
    <div style="display:none">true</div>
</ix:nonNumeric>
```

The nested *ix:exclude* element allows part of the content within the *ix:nonNumeric* element (the statement itself) to be excluded from the item in the target XBRL instance document, whilst the CSS style on the div element suppresses display of the actual Boolean value on the face of the document, which is the part of the marked-up value that *is* carried into the target XBRL. At the same time, visible correspondence between the marked-up statement and the resulting data value is maintained, improving document assurance.

In time, this pattern can be substituted by a simpler one that uses the *ixt:booleantrue* format transform (when it is available) on the *ix:nonNumeric* element and does away with the need for any nested mark-up.

4.10   Marking up empty "flag" items

There are a handful of concepts (of type *fixedItemType*) in the UK Accounts Taxonomies whose presence in the target XBRL instance document is sufficient to convey a piece of information (hence the term "flag") – no value is necessary; in fact no value is allowed. A prime example is *uk-bus:DirectorSigningReport*, which associates the signing of the Accounts with a particular Director via a specific member of the *EntityOfficers* dimension in the context of the item. Aligning the mark-up of these items in the target XBRL instance document with facts reported on the face of the document can be problematic, especially for conversion or manual tagging tools.

A medium-term solution to this is in the process of being standardised by the XII Rendering Working Group in the form of a Transformation Rules Registry transform called *ixt:nocontent*, but this will not be available in the CT online service for some time. The intention of this transform is to allow some arbitrary text (e.g. the name of the Director signing the Accounts) to be marked-up as normal, but for the text to be transformed to an empty element (associated with an appropriate context) in the target XBRL instance. This should allow such mark-up to be undertaken in the same way as any other value that might require a transform, and avoid special-purpose code in tagging tools designed only to handle "flag" items.

In the meantime, the *ix:exclude* element can be used to simulate this behaviour at the cost of a little additional complexity in the mark-up (which nevertheless follows a pattern that applications can easily apply).

If we assume that the name "Joe Bloggs" appears on the face of the document as the Director signing the Accounts then the following "pattern" will allow the Inline XBRL mark-up of this text with a "flag" item such as *uk-bus:DirectorSigningReport*:

```
        <ix:nonNumeric name="uk-bus:DirectorSigningReport"
contextRef="CY"><ix:exclude>Joe Bloggs</ix:exclude></ix:nonNumeric>
```

The nested *ix:exclude* element[4] results in the total exclusion of the Director's name from the item in the target XBRL instance document. At the same time, visible correspondence between the marked-up name and the resulting "flag" item is maintained, improving document assurance.

In time, this pattern can be substituted by a simpler one that uses the *ixt:nocontent* format transform (when it is available) on the *ix:nonNumeric* element and does away with the need for any nested mark-up.

---

[4] Note: unlike the similar pattern for the mark-up of Boolean items (see § 4.9) it is not possible in this case to add any whitespace (including line breaks) within *ix:nonNumeric* but outside *ix:exclude* – the *fixedItemType* of *uk-bus:DirectorSigningReport* preserves whitespace in the element content and any residual whitespace violates the *maxLength* facet, which for *fixedItemType* is zero.

5. XBRL-related Issues

5.1   Minimum Tagging Requirement

In recognition of the effort that some software developers have to expend to provide a mapping between their internal application information model and the information models defined by the available Accounts filing Taxonomies, HMRC have created, in collaboration with XBRL UK, 'minimum tagging requirements' for UK GAAP and UK IFRS. These requirements, expressed as Presentation Linkbases in each Taxonomy, define a much-reduced set of XBRL tags for developers to work with, though if the mapping task is not so onerous for particular applications there is no reason why the full Taxonomy cannot be used from the outset.

These Accounts Taxonomy minimum tagging requirements are transitional arrangements. By 2013 (probably), HMRC will phase out the minimum tagging requirements and expect full tagging of concepts that appear in the base Accounts Taxonomies (i.e. if you need to report a fact, and its concept definition appears in the relevant Accounts Taxonomy, it will need to be marked up regardless).

The minimum tagging requirements for the Accounts Taxonomies are as follows:

Taxonomy	Full List of Concepts	Minimum Tagging Reqmt
UK GAAP 2008	4,375	1,182
UK GAAP 2009	5,242	1,257
UK IFRS 2009	3,805	1,634

The companion Common Data Taxonomy (which is actually a component part of the 2009 Accounts Taxonomies), which contains 972 concept and domain member definitions, does not have a minimum tagging requirement, but typical Statutory Accounts documents will use only a handful of CD Taxonomy concepts anyway (marked up facts in an Inline XBRL document may be drawn from more than one namespace, and therefore more than one Taxonomy).

A similar minimum tagging requirement exists for the HMRC Tax Computation Taxonomy, but in this case the arrangement is not temporary. The Tax Computation Taxonomy dates from a time before Inline XBRL was envisaged, and was an attempt to provide a comprehensive Taxonomy for the complete XBRL mark-up of a tax computation (even then private extension taxonomies were considered inevitable). The minimum tagging requirement for the Tax Computation Taxonomy is, in effect, the Tax Computation Taxonomy that *would* have been created in the Inline XBRL era. It contains just those concept definitions of interest to HMRC, eliminating a lot of detail that can now be provided simply as XHTML. However, the full Taxonomy remains in case software developers wish to mark up more detail (for whatever reason).

The Tax Computation Taxonomy contains about 4,500 concept definitions in total, whereas the minimum tagging requirement contains only 1,350 of these concept definitions.

5.2   Entity Identifier Scheme for HMRC submissions

All contexts referenced by marked up data items in submissions to HMRC should use the relevant Companies House Registration Number as the entity identifier value, and the scheme name 'http://www.companieshouse.gov.uk/'. E.g:

```
<xbrli:context id=contextID'>
    <xbrli:entity>
          <xbrli:identifier
     scheme='http://www.companieshouse.gov.uk/'>12345678</xbrli:identifier>
          <xbrli:segment>
                <!—dimension declarations -->
          </xbrli:segment>
    </xbrli:entity>
    <xbrli:period>
          <xbrli:startDate>2007-04-01</xbrli:startDate>
          <xbrli:endDate>2008-03-31</xbrli:endDate>
    </xbrli:period>
</xbrli:context>
```

Note that the scheme URI should appear **with** a trailing '/' – this is simply a convention that we have chosen to adopt and agreed with both XBRL UK and Companies House. It is important to settle on a single form for the scheme because all the characters of the URI are significant – variants with or without the trailing '/' are, strictly speaking, different schemes.

The variant with the trailing slash is the normalised form according to IETF RFC 3986 – URI Generic Syntax[5].

5.3   Document Creator Information

All software developers who have existing HMRC filing products will know that HMRC's Software Developer Support Team recommend that vendor, product and version information is embedded in the GovTalk header of Gateway submissions. This enables SDS Team to provide filing statistics to individual vendors which identify how well (or badly!) their product is performing in the hands of real customers and where, if necessary, remedial action or improvement is required. The CT online service is no different in this regard, but the inclusion of Accounts produced by other software products into the filing envelope of another product presents difficulties[6].

To enable SDS Team to provide meaningful statistics to vendors of standalone Accounts production software packages we recommend that product and version information is embedded in the generated Inline XBRL document using the document metadata tags from the Common Data Taxonomy or Common Data module (vendor web site URL is not necessary).

For information relating to the product name and version number of an Accounts production software package using the 2009 (or later) versions of the UK Accounts Taxonomies we recommend using the single *XBRLDocumentAuthorGrouping* tuple, identifying the author as a software package with the value "Software" in the 'Description or Title of Author' tag:

```
<uk-bus:XBRLDocumentAuthorGrouping>
   <uk-bus:NameAuthor>Acme Accounts v1.1</uk-bus:NameAuthor>
   <uk-bus:DescriptionOrTitleAuthor>Software</uk-
bus:DescriptionOrTitleAuthor>
  <uk-bus:XBRLDocumentAuthorGrouping>
```

---

[5] Section 6.2.3, Scheme-based Normalisation, http://www.ietf.org/rfc/rfc3986.txt

[6] Separate Tax Computation preparation software packages are unlikely – this function is usually closely associated with the filing application that creates and submits the CT600 XML document.

When seeking recognition for Accounts production products it is important that software developers use the product name string in *NameAuthor* that they intend to embed in live submissions so that SDS Team can identify product names, associate them with product vendors and provide meaningful statistics in their feedback.

5.4　The Proper Use of Tuples

The primary purpose of a tuple in XBRL is to bind a collection of related facts together and provide a mechanism for repeated occurrences of that collection, much like a table with multiple columns and one or more rows. However, in XBRL terms, the concepts that are intended to be bound together in a tuple rank equally with concepts that are used outside of tuples, which means that it is permitted (by the XBRL Specification) to use these concepts to represent facts outside of any tuple structure.

The temptation, therefore, when only one collection of facts is required (i.e. a "table" with only one "row") is to omit the tuple "container" and simply mark-up the individual facts. This is then carried through to the Inline XBRL mark-up where the corresponding *ix:tuple* element is omitted, along with the *order* and *tupleRef* attributes on the member facts. Such behaviour by applications that generate Inline XBRL is strongly discouraged. When a tuple is expected it should be used even if, in terms of distinguishing between multiple collections of facts, it is redundant. This should simplify both generating and consuming applications and avoid complications with analysis applications that expect to see certain facts inside tuples regardless of their cardinality.