

BILINGUAL SOFTWARE GUIDELINES AND STANDARDS



BWRDD YR IAITH
GYMRAEG • WELSH
LANGUAGE BOARD



Noddir gan
Lywodraeth Cynulliad Cymru
Sponsored by
Welsh Assembly Government



Bilingual Software Standards & Guidelines

**A WELSH LANGUAGE
BOARD DOCUMENT**

Version : 1.0
ISBN: 095353342 5

Contents

1	Introduction	7
1.1	Intended Audience	7
1.2	'Bilingual' Software	7
1.3	Scope of Applicability	8
1.4	Review and Publication	8
1.5	Standards and Guidelines	9
1.6	Terminology	9
1.7	Authors	10
1.8	Further Information	10
2	Non-Technical Overview	11
2.1	Localisation – Guidelines, Issues and Management (Section 3)	11
2.1.1	Definition	11
2.1.2	Reasons for Localisation	11
2.1.3	Project Management of Bilingual Software Development	11
2.1.4	Translation and Linguistic Quality	11
2.2	Locales, Alphabets and Character Sets (Section 4)	12
2.2.1	Using Locales	12
2.2.2	Differences between English and Welsh Locales	12
2.2.3	Alphabets	12
2.2.4	Accented Letters & Character Sets	13
2.2.5	Other Differences	14
2.2.6	Minimising the Impact of Locale Differences	14
2.3	Architecture and Design (Section 5)	15
2.3.1	Language Selection	15
2.3.2	Persistence of Language Selection	15
2.3.3	Storing Language Preferences	15
2.3.4	Web and Email Addresses	16
2.3.5	Consistent Presence of Chosen Language	16
2.3.6	Structuring Bilingualism	16
2.3.7	Messages	17
2.3.8	Integration with Language Support Tools	18
2.4	User Interface (Section 6)	19
2.4.1	Language	19
2.4.2	Design & Layout	19
2.4.3	Accessibility	20
2.4.4	User Assistance	20
2.4.5	Installers	21
2.5	Outputs (Section 7)	21
2.5.1	Monolingual or Bilingual	21
2.5.2	Layout and Format	21
2.5.3	Mail shots	22
2.5.4	Character Set	22
2.5.5	Emails	22
2.6	Data Management (Section 8)	23
2.6.1	General Guidance	23
2.6.2	Identifying Bilingual Data	23
2.6.3	Mandatory, Optional or Inappropriate Data	23
2.6.4	Handling Third Party Data	25
2.6.5	Manual Data Entry	25
2.6.6	Encouraging Bilingual Data Entry	25
2.6.7	Consistent Set of Mandatory Data	26
2.6.8	Data Storage	26
2.6.9	Unavailable Data	26
2.6.10	Transmitting Data	26
2.6.11	Metadata – “Data About Data”	26
2.6.12	Searching	27

2.6.13	Bilingual vs Multilingual	27
2.7	Postal Addresses and Geographical Information (Section 9).....	27
2.7.1	Use of Postal Addresses	28
2.8	e-Government Interoperability Framework (e-GIF) (Section 10).....	28
3	Localisation – Guidelines, Issues and Management	29
3.1	Definition of ‘Localisation’	29
3.2	Reasons for Localisation.....	29
3.2.1	The Law	29
3.2.2	Bilingualism in Wales Today.....	29
3.2.3	Business Opportunity.....	30
3.3	Project Management of Bilingual Software Development	30
3.3.1	Requirements Management	30
3.3.2	Testing & Quality Control.....	31
3.4	Translation & Linguistic Quality.....	31
3.4.1	Consistency and Use of Standard Resources.....	31
3.4.2	Working with Translators	31
3.4.3	Outsourcing Translation	32
4	Locales, Alphabets and Character Sets	33
4.1	Locales	33
4.1.1	Locale Identification	33
4.1.2	Using a Locale	34
4.1.3	Differences Between British English and British Welsh Locales	35
4.1.4	Minimising Impact of Locale Differences	36
4.2	Alphabets	36
4.2.1	Character vs. Letter Counts.....	37
4.2.2	Selecting a Sort Order	37
4.2.3	Sorting Digraphs	38
4.2.4	Combining Grapheme Joiner.....	39
4.3	Character Sets	39
4.3.1	Explicit Specification of Character set	40
4.3.2	Operating Platform Configuration	42
4.3.3	Encoding of non-ANSI Characters	42
4.3.4	User Input	42
4.3.5	Choice of Font	44
5	Architecture and Design	45
5.1	Language Selection	45
5.1.1	Initial Implicit Assumption	45
5.1.2	Dedicated Language Selection Function.....	46
5.1.3	Ubiquitous Language Selector	46
5.1.4	Configured Language Preference	47
5.1.5	Non-Graphical Interfaces (e.g. Call Centre Telephony Software).....	47
5.2	Persistence of Language Selection	48
5.3	Storing Language Preference	48
5.3.1	Application Level Profiles	48
5.3.2	Operating Platform Profiles	49
5.4	Domain Names	49
5.5	Consistent Presence of Chosen Language	50
5.5.1	Absence of Language-Elements	50
5.5.2	Predicting Absence of Language-Dependent Data and Objects.....	50
5.5.3	Identifying Inconsistency	51
5.6	User Interface Structures	51
5.6.1	Parallel/Mixed Text (Level 1)	51
5.6.2	Parallel Mirroring (Level 2)	52
5.6.3	Switchable Embedded Content (Level 3)	53
5.6.4	Resource-Based (Level 4).....	54
5.7	Translation and Content Support Tools	55

5.8	Error and Event-Driven Messages.....	55
5.9	Integration with Language Support Tools	57
6	User Interface	58
6.1	Language	58
6.1.1	Quality of Language.....	58
6.1.2	Abbreviations	58
6.1.3	Policies, Disclaimers and other Legal Text	59
6.1.4	Simultaneous Deployment.....	59
6.1.5	Monolingual Content.....	59
6.2	Design & Layout.....	59
6.2.1	Parity of Design Quality	59
6.2.2	Mixed Language Interface	59
6.2.3	Anchor Components	60
6.2.4	Prominence of Information with a Language Context	60
6.2.5	Grammar Neutrality	60
6.2.6	Labels & Placeholders	60
6.2.7	Images, Graphics and Icons.....	61
6.2.8	Other User Interfaces	61
6.3	Accessibility.....	61
6.4	Graphical User Interface (GUI)	62
6.5	User Assistance	64
6.6	Installers.....	64
7	Outputs	65
7.1	General Guidance	65
7.2	Monolingual or Bilingual.....	65
7.3	Layout and Format.....	66
7.3.1	Mixed/Parallel Text	66
7.3.2	Sequential Text.....	66
7.3.3	Alternating Pages	67
7.3.4	Separate Objects	67
7.4	Mail Merge	67
7.5	Character Set.....	68
7.6	Reports.....	68
7.7	Emails	70
7.7.1	Subject Line	70
7.7.2	Subject Line Modifiers	70
7.7.3	Disclaimers and other text.....	70
7.7.4	Attachments.....	70
7.7.5	Email Address Domain Names.....	71
7.7.6	Reply Name and Address.....	71
8	Data Management	72
8.1	General Guidance	72
8.2	Identifying Bilingual Data	73
8.3	Managing Bilingual Data	73
8.4	Handling Third Party Data.....	73
8.5	Enumerated Data Items	74
8.6	Mandatory, Optional or Inappropriate	75
8.7	Data Capture.....	75
8.7.1	Automated Data Entry/Procurement.....	75
8.7.2	Manual Data Entry	76
8.7.3	Encouraging Bilingual Data Entry.....	76
8.7.4	Providing Data Objects	78
8.8	Data Storage	78
8.8.1	Character Sets.....	78
8.8.2	Sort Orders	78
8.9	Data Display.....	79

8.10	Data Transmission and Interfaces	79
8.11	Meta-Data	80
8.11.1	Encoded Meta-Data	80
8.11.2	Creating Meta-Data	80
8.11.3	Managing Meta-Data	81
8.11.4	Using Meta-Data	83
8.12	Searching	83
8.13	Language Preference	84
8.13.1	Storing the Language Preference	84
8.13.2	Using the Language Preference	84
8.13.3	Bilingual vs. Multilingual	84
9	Addresses and Geographical Information	85
9.1	Geographic Information (GI)	85
9.2	Address Databases	85
9.2.1	Public Sector Resources	85
9.2.2	Quality of Data	86
9.3	Relevant Standards	86
9.4	Use of Addresses	86
9.4.1	User Entry of an Address	87
9.4.2	Using the Appropriate Address	87
9.4.3	Delivery	88
9.5	Storage and Analysis of Address Data	88
10	e-Government Interoperability Framework (e-GIF)	89
10.1	e-Government Metadata Standard (e-GMS)	89
10.1.1	Document Content Language	89
10.1.2	Metadata Language	89
10.2	Category Lists	90
10.2.1	Use of Category Lists for English and Welsh Data	90
10.3	XSD Schemas	91
10.3.1	Archives and Records Management	91
10.3.2	Election Markup Language (EML)	91
10.3.3	National Public Transport Access Nodes (NaPTAN)	92
11	Further Consultation	93
11.1	Sort Orders	93
11.1.1	User Interface Sort Order	93
11.1.2	Database Sort Order	94
11.2	Addresses & Geographical Information	94
11.3	eGovernment	94
11.4	Other Standards	95
11.5	New Technologies	95
11.6	Specific Application Areas	95
12	Guidance for Web Designers	97
12.1	Character Sets	97
12.2	Initial Implicit Language Assumption	97
12.3	Ubiquitous Language Selector	98
12.4	Persistence of Language Selection	99
12.5	Storing Language Preference	99
12.6	Domain Names	99
12.7	Error and Event-Driven Messages	99
12.8	Mixed Language Content	100
12.9	Meta data	100
13	Resources & Bibliography	101
13.1	Character Sets	101
13.2	Keyboard Key Sequences	101



13.3	Language Resources	101
13.4	Welsh Language Board	101
13.5	Localisation & Multilingual Computing	102
13.6	Address Management	103
13.7	e-Government Standards & Guidelines:	103
13.8	Other	103

1 Introduction

Wales is a bilingual country, where the number and percentage of bilingual (Welsh/English) speakers is increasing.¹ The equal status of English and Welsh in conducting public business in Wales is enshrined in the Welsh Language Act 1993.

In addition, the Welsh Assembly Government, in its policy document, *Iaith Pawb*, (everyone's language), has declared its ambition to create a "truly bilingual Wales, [...] where people can choose to live their lives through the medium of either or both Welsh or English and where the presence of the two languages is a source of pride and strength to us all." In order to meet both statutory requirement, as well as this laudable policy aim, it is important that high quality IT systems are developed to be of practical everyday use. *Iaith Pawb* also called for a strategy document to increase the status of Welsh in Information Technology. That strategy document, available on the Board's website, noted the need for detailed technical guidance for bilingual computing, which is the aim of the present document.

1.1 Intended Audience

This document should be read by all those with an interest in, or responsibility for, IT projects intended for use in Wales – managers, developers and users alike. Although the Board by its very nature would encourage all institutions to treat Welsh and English on a basis of equality whilst providing services to the public in Wales, the base legal document for this remains the Welsh Language Act, 1993. Please contact the Board for further advice (post@welsh-language-board.org.uk)

More specifically, the present document has three key audiences:

- Developers producing software applications (including websites) for use within Wales or that will potentially be used by Welsh speakers. For such developers the document is intended to provide guidelines and standards to guide the functional specification and development of the capabilities of these applications;
- Any individual involved in the specification or procurement of software applications that will be accessed from within Wales or by Welsh speakers. For these individuals, the document is intended to provide guidance in defining the requirements and validating the compliance of these applications against the standards;
- Policy makers and compliance officials with a responsibility to ensure that their organisation or business delivers the highest quality and compliant interface to citizens and customers.

1.2 'Bilingual' Software

The term "bilingual", as used in this document, refers explicitly to the Welsh and English languages. Note that this doesn't necessarily mean that both languages should be presented to the user at all times. Writing effective bilingual software is about empowering the user to work with an IT system in the language of their choice and having the freedom to change that choice.

¹ See results of 2001 Census on <http://www.welsh-language-board.org.uk>.

Though this document is specific to bilingual (Welsh & English) software support, wherever possible we advocate a design approach that results in a multi-lingual capability.

1.3 Scope of Applicability

Software applications are ubiquitous and occur on a broad variety of platforms being a key part of our interaction with computer systems of all forms. These standards cover all software applications that interact with users using language or that have an impact upon how other applications interact with users.

Examples of such applications include, but are not limited to:

- Web sites;
- Web based applications and interfaces;
- PDAs (Personal Digital Assistants);
- ATMs (Automated Teller Machines);
- Digital TV (idTV);
- Legacy platforms (e.g. mainframes);
- Thin client architectures;
- Client/Server and client-only applications;
- Operating systems;
- Electronic Signage;
- Cellular/3G phones;
- Embedded Systems.

As software technology advances and also its range of applications, we intend for the scope of these standards to be extended to cover these new technologies and ensure equal support and treatment of the Welsh and English languages. Examples of such technologies include (but are not limited to):

- Text to speech (speech synthesis);
- Speech recognition;
- Machine translation;
- Handwriting recognition;
- Predictive text processing;
- OCR (Optical Character Recognition);
- Smart card technology.

Where a new technology or software architecture emerges subsequent to the most recent issue of this document and is therefore not directly referenced, the standards and guidelines should be interpreted in the most practical manner possible consistent with their intent and purpose.

1.4 Review and Publication

This is the first issue of this document and is the first time that such a set of guidelines and standards have been collected into a single coherent form.

As a result, we are eager to receive all comments, suggestions and input during an ongoing consultation process. We accept that there might be a number of areas where there is a more effective approach than we have recommended, where domain experts can provide valuable input and/or that we might have overlooked.

It is recognised that the evolving nature of this field together with ongoing technological advances will lead to frequent revisions and updates of this document and we encourage all participants engaged in the development, maintenance, management and procurement of bilingual software applications to engage in an ongoing dialogue in order to guide the evolution of these standards. We will review all input received and use this to produce future versions of the document.

It is also intended that these standards will be accompanied by a number of supporting resources. Further information, together with contact details for further submissions, are available on the Welsh Language Board website.

1.5 Standards and Guidelines

This document defines both standards and guidelines to be used when developing IT solutions that will be used by bilingual (Welsh/English) users, or provide information that will be available to a bilingual audience.

The degree to which the reader and user of these standards must comply will be determined by:

- Their statutory obligations under the Welsh Language Act 1993;
- The Welsh Language Scheme currently in place for their organisation;
- The commercial and social benefits they will obtain through compliance and the provision of a bilingual system.

Standards can be easily identified through the use of the following formatting:

This is a standard that must be followed as appropriate.

Anything outside such formatting can be considered as further guidance in creating high quality bilingual software.

1.6 Terminology

Throughout this document, terminology common to and standard in the IT and software industry is used. In addition, some terms have been used consistently throughout the document and have a specific meaning. Note, that this isn't an attempt to define standard terms for use outside of this document.

Term	Meaning
Diacritic characters	Refers to all letters with accents (diacritic marks, e.g. á, ô, ÿ, etc). See 4.3 for more information.
Diacritic equivalence	Where a letter with a diacritic mark is treated as equivalent to the letter without the mark (e.g. for searching).
Language-element	Any item that is part of a language. Can be an item of text, image, sound segment, etc.
Language-neutral	Where the meaning of an item (usually an element of the user interface) is independent of a particular language.
Language-sensitive	Where an item (usually an element of the user interface) does have a relevance to a language.
Operating platform	Described in 4.1.2, refers to the environment in which a software application operates. Usually means an underlying

	operating system, embedded platform or web browser.
User interface component	Any item forming part of a user interface. This term is used to include speech and tactile interfaces as well as the more typical graphical user interface.

1.7 Authors

This document was commissioned by the Welsh Language Board and authored by Draig Technology Ltd with assistance from the Welsh Language Board and Canolfan Bedwyr, of the University of Wales, Bangor.

The Welsh Language Board and the authors would like to thank all those individuals and organisations that have also contributed to this document and submitted comments during the consultation period.

1.8 Further Information

Further information and advice regarding this document and the issues addressed can be obtained by contacting the Welsh Language Board:

The Research, Language Technology and Grants Unit
Welsh Language Board
Market Chambers,
5-7 St Mary St,
Cardiff
CF10 2AT

post@welsh-language-board.org.uk

2 Non-Technical Overview

The aim of this section is to give an overview of the standards for a non-technical audience. Every effort has been made to make the content as accessible as possible to the widest possible audience. Terms are defined, their significance discussed and an overview of the standards are also given.

Please note that this section merely constitutes a high-level view of the standards. The main standards begin with section 3.

2.1 Localisation – Guidelines, Issues and Management (Section 3)

2.1.1 Definition

This section defines ‘localisation’ as being the process by which a piece of software can be made linguistically and culturally appropriate for users who speak English, Welsh or both. This could involve ensuring that all text is available in both languages and that lists are sorted appropriately for each language’s alphabet, for example. But localisation also involves many other technical activities and architectural decisions that aren’t always visible to an end-user or a non-technical user.

2.1.2 Reasons for Localisation

The Welsh Language Act (1993) places a duty on the public sector to treat Welsh and English on an equal basis when providing services to the public in Wales. Software systems are one means by which the public sector provides services and are therefore subject to the Act. For those not in the public sector, having bilingual software systems can bring other benefits such as improving the quality of customer service, attracting new customers or increasing customer loyalty.

2.1.3 Project Management of Bilingual Software Development

A piece of software starts its life being specified (“the software will do x, y and z”) and designed (“it will do x, y and z like *this*”). The code is then written according to the design and the finished product is tested for accuracy and overall quality. All these events form part of what is known as the “software development lifecycle”.

Section 3.3 describes the importance of considering bilingual use and functionality throughout every stage of this lifecycle and recommends assigning a member of the team to promote and supervise localisation matters during the progress of a project.

It is recommended that bilingual functionality and content should be specified from the outset – i.e. from the requirements specification – and are carried forward into design, implementation and then testing. This ensures that such functionality and content is of a sufficiently high standard and not regarded as a last-minute bolt-on.

Care should be taken to ensure that changes in the software do not detrimentally affect the content and functionality in either language.

2.1.4 Translation and Linguistic Quality

Section 3.4 suggests using standard translated terminology wherever available to ensure consistency and accuracy. There has already been substantial efforts invested

in the production of such resources for frequently-used computing terms (e.g. “keyboard”, “mouse”, “window”, “icon”) as well as other general terms (e.g. countries, salutations). Section 12 contains references to where these are freely available.

Project plans should allocate time for translations to take place. Translators will require supporting contextual material to assist them in translating to a high quality (for example, knowing where and why a piece of text appears and what it means). Unless using specialist technical translators, all material should be provided to translators in a form that can be used without a high degree of technical knowledge (for example, provide a list of text items and supporting material in a spreadsheet rather than expecting the translator to sift through dense programming code).

2.2 Locales, Alphabets and Character Sets (Section 4)

A “locale” can be thought of as a bundle of characteristics that describe a language and culture in a particular geographic area. The bundle might include facts about date formats, currency symbols, how to sort lists alphabetically, and other facts.

For example, the locales for English in the United Kingdom and the United States are largely similar but subtly different. One key difference is that we have different currency symbols. We also write dates differently; “30 November 2005” is “30/11/2005” in the UK and “11/30/2005” in the US.

2.2.1 Using Locales

Section 4 of the document discusses the use of locales for both British English and Welsh. The locale for British English is easily available to software developers, however it has taken time for a standard Welsh locale to be created and this isn’t as readily available at the time of writing. The standards mandate that a software application shall make use of the characteristics described in locales whenever they are available (see 4.1.2).

Technical processes are available to “best guess” the most appropriate bundle of characteristics to use when dealing with a particular user. These should be used wherever a user’s language and cultural preferences are not already known. When those preferences *are* already known, they shall be used.

2.2.2 Differences between English and Welsh Locales

Essentially, all the cultural characteristics of a locale bundle are similar between English and Welsh – both use the Gregorian calendar, the constituent mainland countries of the UK are in the same time zone and the same currency symbol is used in each.

The main differences are language-related:

- The English and Welsh alphabets are different and this means lists need to be sorted slightly differently.
- Welsh uses accented letters whereas English generally doesn’t.
- There are several other subtle but important differences.

2.2.3 Alphabets

The Welsh alphabet has 29 letters whereas the English has 26. Welsh doesn't have the letters 'k', 'q', 'v' and 'z' but does have the additional letters 'ch', 'dd', 'ff', 'ng', 'll', 'ph', 'rh' and 'th'. These letters are known as "digraphs" – individual letters that are made up of more than one character. This means that "Llandudno" has 8 letters in the Welsh alphabet, and 9 letters in the English alphabet.

These differences impact upon sorted lists. Section 4.2.2 of the standards describes how these issues should be addressed.

As a bilingual country, words containing the letters 'k', 'q', 'v' and 'z' are widely used, even though they do not form part of the Welsh alphabet. To sort in Welsh, therefore, a superset of the English and Welsh alphabets should be used that includes these letters:

a, b, c, ch, d, dd, e, f, ff, g, ng, h, i, j, k, l, ll, m, n, o, p, ph, q, r, rh, s, t, th, u, v, w, x, y, z

When sorting in English, the English alphabet should be used without consideration of Welsh digraphs (e.g. 'th' should be treated as 't' and 'h').

There are some issues relating to the sorting of digraphs in Welsh and these are detailed in section 4.2.3. For example, it isn't possible to easily determine whether the sequence "ng" is the letter 'ng' or 'n' then 'g'.

- C-a-ng-e-n – Cangen;
- B-a-n-g-o-r – Bangor.

It is important that a consistent approach is used to sort digraphs in Welsh. For example, the sequences "ng", "ph", "rh" and "th" shall *always* be treated as digraph letters when encountered.

One solution to this problem is to use what is known as the "combining grapheme joiner". This is an invisible character that can be inserted between two other characters as if to say "these are two letters, not one". For example, "Ban●gor". This approach is outlined in section 4.2.4.

2.2.4 Accented Letters & Character Sets

The most commonly used accent in Welsh is the circumflex. This, and all other accents in Welsh can only appear on vowels. In Welsh, the vowels are A, E, I, O, U, W and Y. For example, 'dŵr' (water) or 'tân' (fire). Acute (á) and grave (à) accents are also used on all vowels, as is the dieresis (ä).

As all these characters can be used in Welsh, it must be possible to type them and to store them.

A computer can only deal with a range of known characters that it understands. This range is known as a "character set", and these are the subject of section 4.3. A character set can be said to have a narrow or wide range of characters.

The standards mandate that the character set chosen for use must contain all the accented and non-accented characters of the Welsh alphabet. This is most easily accomplished by choosing a wide-ranging character set, such as "Unicode". Some software systems are required to publicise which character set they use so that other

software can interact with them correctly. Section 4.3.1 mandates that wherever possible, the character set used shall be explicitly specified.

Where it isn't possible to choose a character set that lends itself to the Welsh language, it is often to work around the associated technical problems by writing text in a specially-encoded way. Section 4.3.3 gives technical detail on this approach.

It isn't always possible for a user to type accented letters, especially on a standard UK keyboard. The standards therefore dictate that it should be possible to approximate accented characters using what is known as 'diacritic transliteration'. In essence, this means that the software shall interpret the sequence "w^" as being equivalent to "ŵ", for example. Further detail is given in section 4.3.4.2.

2.2.5 Other Differences

The abbreviated days of the week contain digraphs in Welsh:

Day	English	Welsh
Monday	M	LI
Tuesday	T	M
Wednesday	W	M
Thursday	T	I
Friday	F	G
Saturday	S	S
Sunday	S	S

It is therefore important that software systems can cope with abbreviations longer than one character in this situation.

It is also not possible to take the first letter of each day of the week to create an abbreviation in Welsh, as the equivalent of the "-day" suffix in English occurs as a prefix in Welsh:

- **Monday;**
- **Dydd Llun.**

Section 4.1.3.1 therefore mandates that abbreviations shall be both grammatically correct and cater for digraph letters.

Ordinals are also different in Welsh. For instance, 'First Tuesday' and 'Second Tuesday' in English would be 'First Tuesday' and 'Tuesday Second' under Welsh grammar.

2.2.6 Minimising the Impact of Locale Differences

The standards suggest ways of minimising the impact of locale differences in section 4.1.4. These include:

- Using language-neutral short dates (e.g. 30/11/2005) instead of worded dates;
- Avoid using ordinals wherever possible.

A font should be chosen that is equally aesthetically-suitable for both English and Welsh and that also supports all the accented characters included in the software's chosen character set. Guidance on choosing such a font is given in section 4.3.5.

2.3 Architecture and Design (Section 5)

2.3.1 Language Selection

When a user accesses a software application, a determination will need to be made of which language to display. This can be achieved by:

- An implicit assumption where the user has not yet identified themselves or their preference; or
- The use of an initial language choice (displayed bilingually) when first accessing the application;
- If it is possible to identify the user, his/her explicit language preference shall be used.

Identification of a user or implicit language preferences can be carried out in several different ways, some more technical than others. One basic means of identifying a language preference is by a web address. For example, if a user enters a web site via a Welsh internet address (e.g. <http://www.bwrdd-yr-iaith.org.uk/>) it could possibly be assumed that their preference is for Welsh content. This and more technical approaches are detailed in sections 5.1.1 and 5.4.

During further use of the software it shall be possible to switch languages at any time using a language selector located in a uniform place throughout the software. Languages shall be displayed in their native names (e.g. 'English' and 'Cymraeg').

If possible, the software shall switch languages immediately without any loss of data. Where it isn't possible to do this for technical reasons, an explanation shall be given and the user should be given the chance to reverse their decision.

Further details are given in section 5.1.3 of the document.

2.3.2 Persistence of Language Selection

In general, whenever a screen switches to a language as a result of an explicit request from the user, the new preference should be remembered until the user finishes with the software or changes their preference again. See section 5.2 for further detail.

2.3.3 Storing Language Preferences

It is possible to store language preferences in different ways when software is required to remember a language preference between uses. Several approaches are available in order to accomplish this.

In short, the available approaches for storing this preference include:

- Recording the preference in the software system's own permanent memory;
- Recording the preference on the user's computer system in some fashion;
- Recording the preference in a user directory or shared resource.

Some of these options may not be viable because of security, privacy or other operational concerns. Whichever option is chosen, if a language preference cannot be found by the software, the software should handle this situation gracefully.

Further detail is given in section 5.3.

2.3.4 Web and Email Addresses

If the software system is internet-based, two options are available for web and email addresses:

- Use a language-neutral address (e.g. google.com);
- Use **both** English and Welsh addresses (e.g. rhywle.gov.uk, somewhere.gov.uk).

It isn't currently possible to purchase .uk addresses with accented letters although this is likely to change in the future. At that point, it is recommended that appropriate addresses are also purchased (e.g. llanllŷr.gov.uk) in **addition** to non-accented variations, but not in **replacement**.

See section 5.4 for further details.

2.3.5 Consistent Presence of Chosen Language

Section 5.5 of the document states that functionality and content shall be available equally in both Welsh and English such that a user, having chosen a preferred language, is able to navigate and use the software solely in the language of their choice.

The unintentional absence of a language element – that is to say, any language sensitive item such as text or an image – should be audited automatically by the software for an administrator user to follow up. In this situation, the content from the alternate language should be displayed. For instance, if the Welsh text is missing, the English should be displayed if available.

If it is known in advance that a resource isn't available in the user's preferred language, the user should be notified. For example, if a particular document is only available in English on a web site, the link to the document should include "(English Only)", automatically if possible.

It is recommended that the software is designed to include the capability to search for and highlight text that is missing in particular languages. This will help reduce inconsistency in the software system.

2.3.6 Structuring Bilingualism

How do you create bilingual software? Create two separate software systems, one for each language or create one that supports both languages? This question is addressed in section 5.6.

Several options are available:

1. Have one system that includes English and Welsh either side-by-side or in alternate paragraphs;
2. Have two versions of the software – one for English, one for Welsh – and the ability to switch between the two versions;
3. Have one system that includes the text for every language but only shows the user's preferred language and hides text from any other language;
4. Have one language-agnostic system that pulls in text in the user's preferred language on a just-in-time basis from a shared resource (e.g. a database).

The document discusses these in terms of Level 1, 2, 3 and 4 architectures (resp.).

Level 4 is the most powerful, flexible, reliable and provides the most capabilities in terms of language management and maintenance.

However, all four approaches have the potential to produce applications that will satisfy the standards. The trade-off is balancing the initial development cost against the ongoing maintenance and management cost when trying to achieve the same level of quality, capability and compliance.

Level 1 is suited to static content such as web sites, however it can lead to poor design quality, rendering the system difficult to use.

Level 2 is again suited to static content but only on a small-scale. All work must be performed twice and there is a danger of asymmetric content between the English and Welsh systems. It is also a labour-intensive job to ensure that corresponding pages in English and Welsh are somehow linked together so that language switching can take place. Loss of data when switching languages is also a common problem.

Level 3 doesn't require parallel software systems in each language and only displays the language of interest to the user. In this sense, it is an improvement over Levels 1 and 2. However, it may mean that identical content may exist in multiple places throughout the system, rather than having language content defined once and re-used many times.

Level 4 ensures that the user only ever sees the language of their choice whilst making efficient use of the language content available. This architecture also makes it easier to run analysis reports on language content to identify inconsistencies or missing items. It is also scalable to a multilingual solution.

2.3.7 Messages

Section 5.8 states that every effort should be made to ensure that the user only ever sees messages (e.g. "your information has been saved" or "sorry, an error occurred") in their chosen language.

All messages coming from the bilingual software system should be available to the user in both languages. If a message is received from *outside* the bilingual software system, if it can be changed to appear in the user's chosen language, it should be. If it isn't feasible for this to happen for technical or other reasons, there is little else that can be done in this situation.

Note that it is unlikely that the person who encounters an error will be the same person who fixes the cause. Likewise, there is no guarantee that the two people both speak

the same language. As such, the software system should audit any errors in a way that the appropriate information is available in both languages.

2.3.8 Integration with Language Support Tools

Language Support Tools include things such as spelling or grammar checkers. The standards state (in 5.9) that any language support tools used shall be available to users in both English and Welsh.

2.4 User Interface (Section 6)

The “user interface” is the means by which somebody interacts with a software system. It can consist of text, images and places allowing a user to enter details (fields), for example. The term is generic, but specific examples include:

- A web page;
- A computer program such as Microsoft Word (a “window”);
- The screen on which you set up the timer for a video recorder;
- The screen and controls of a cash machine;
- The screen and keypad on a mobile phone.

2.4.1 Language

The language used in a software system shall be of an equal quality in both English and Welsh. Quality can be measured through grammar, spelling and comprehensibility. This may require the outsourcing of material for translation when it is deemed that the appropriate skills are not held in-house.

Crystal Mark (Plain English Campaign) and Cymraeg Clir (Canolfan Bedwyr) are useful resources in language clarity. Details are given in section 13.3.

Abbreviations should be meaningful, accurate and consistent with existing conventions. Note that abbreviations and acronyms are often different in English and Welsh.

Equality of language extends to all text, including all text that isn’t a functional part of the software application such as legal disclaimers or terms and conditions. It is also recommended that the organisation’s policy on bilingualism is published and referenced in the software.

When software is updated, both languages should be taken into consideration. New or modified functionality shall be made available in both languages simultaneously. In situations where monolingual content is justifiable, an explanation should be presented in the missing language.

See section 6.1 for further details.

2.4.2 Design & Layout

The design and layout of the user interface shall be of equal quality in both the English and Welsh versions.

Where Welsh and English text is both shown, both shall be given equal prominence. Care should be taken to separate the content for each language.

Common user interface components – menus, data entry fields and language selectors, for example – should be placed in the same place in both languages. This assists the user in accomplishing tasks quickly in either language or improving skills in their weaker language.

So far as is possible, the layout of a user interface should be language-neutral. This avoids the need to swap the location of data fields when switching languages.

Consider, for example, the ordinals example given earlier. The following layout may be fine for English but it isn't flexible enough to accommodate "Tuesday Second", as demanded by Welsh grammar:

The of of the month.

When text in a particular language is pulled-in for use on a just-in-time basis, each piece of text is usually placed inside a placeholder of some description. The layout of user interfaces shall be designed to ensure that there is sufficient space allowed for the display of text in either language.

See section 6.2 for further detail.

Images should be culturally neutral wherever possible, and the use of flags to denote languages is discouraged. Care should be taken not to use images that hinge on idioms in either language. Idioms rarely translate exactly, if at all.

Section 6.4 expands on the use of images and branding in bilingual software. It states that any logo or branding should be appropriate for the language being displayed. This means that an English logo and branding should be used when viewing the software in English and Welsh logo when using the software in Welsh, unless:

- The brand or logo is language-neutral (e.g. Google);
- The brand or logo is bilingual anyway.

If monolingual text is used inside an image, both English and Welsh versions of that image should be made available.

When the software links to external resources, it should link to a resource that is appropriate for the current language, if possible. For example, if a link is placed on a web page, it should point to an English web site when the language is English or to a Welsh web site when the language is Welsh.

2.4.3 Accessibility

The term 'accessibility' refers to whether a person is able to access a software system regardless of physical, economic or technical factors. The main driver for accessibility is equal opportunity. An accessible system would be usable by a disabled person or somebody who doesn't own a state-of-the-art computer, to give just two examples.

In section 6.3, the standards state that where accessibility support and functionality is included in an application, an equal level of support shall be provided for each language.

It should also be possible for accessibility tools to discriminate between text in each language. This is advantageous to screen-reading and text-to-speech software, which needs to know how to pronounce character combinations correctly (e.g. 'll' as in the English "wall" or 'll' as in the Welsh "llefrith").

2.4.4 User Assistance

'User Assistance' is a generic term that describes things such as documentation, "help" files, lists of frequently asked questions, etc.

Section 6.5 states that any user assistance material should be available bilingually. The content of both the English and Welsh should be identical at all times. Care should be taken to ensure that the terms used in the material should match the terms used in the software exactly.

When pictures (diagrams or copies of screens from the software) are used to explain the software, text in these images should be in the same language as the text.

2.4.5 Installers

An installer is the means through which software is added to a computer. They typically ask the user to make certain decisions about how the software should be configured.

The standards are also applicable to installers since they are examples of software themselves. Installers should therefore be bilingual, use standard terminology where available and have legal disclaimers in both English and Welsh.

Installers should not assume that the language chosen for installation will be the same as the language in which the software will be used.

2.5 Outputs (Section 7)

Outputs of a software system may include things such as emails, lists, charts or printed materials.

In section 7.1, several general observations are offered, the most notable being that recipients of emails and other outputs can easily forward them to others who might have different language preferences or abilities.

2.5.1 Monolingual or Bilingual

Section 7.2 states that the best approach is to produce bilingual outputs, although there may be situations in which monolingual output is justifiable or appropriate. These situations can include single recipients with a known language preference, personal communications or when a group of recipients will, by definition, be competent in a language (e.g. job adverts where competence in Welsh is required).

Bilingual outputs should be used whenever a language preference is unknown. The language given prominence (i.e. that appears first or on the left if in column format) should be decided upon based upon the nature of the software system, any known demographics or the subject matter concerned.

2.5.2 Layout and Format

Outputs from a software system should comply with all the relevant standards for user interfaces (see above). In addition to section 7.3, further help on producing bilingual outputs can be found in the Welsh Language Board document 'A Guide to Bilingual Design'.

Options for layout and format include:

- Parallel / Mixed text – this involves the placement of both languages side-by-side or mixing sentences (e.g. first sentence in Welsh, followed by a second in English – or vice versa).
- Sequential text – where the full body of text in one language is followed by the full body of text in the other language. If the recipient’s language preference is known, the preferred language should be placed first. Some indication should be given that the other language follows. This is particularly suited to email.
- Alternating pages – this is particularly suited to printed material, although care should be taken that the reader doesn’t have to turn more pages for one language than the other.
- Two separate outputs – this could be one document in English and a second in Welsh, for example.

2.5.3 Mail shots

A mail shot (or “mail merge”) is typically created on a “fill-in-the-blanks” basis, pulling the appropriate information from the software and placing it in the appropriate blanks (e.g. “Dear _____ ,”).

The quality of mail shots produced in this way shall be high and equal between the two languages. No English text should appear in a Welsh document, and no Welsh should appear in an English document due to technical limitations.

Refer to section 7.4 for further details.

2.5.4 Character Set

The outputs should be able to display/print every accented letter of the Welsh alphabet. In many cases, this will involve choosing an appropriate character set (as discussed earlier). See section 7.5 for details.

2.5.5 Emails

The standards defined in section 7.7 apply to software systems that transmit, forward, store or display emails.

Subject lines should be bilingual with mixed text (i.e. language 1 / language 2), placing the preferred language first when known. When an email is replied to or forwarded, the following prefixes shall be used:

	English	Welsh
Reply	Re:	Atb:
Forward	Fw:	Yml:

For example, if an email reply is sent to an individual with an expressed preference for English, the prefix used should be: ‘Re/Atb:’.

Any disclaimers used in the email shall be bilingual.

Attachments shall be language-neutral, contain bilingual content (complying with the standards) or have one attachment for each language.

As discussed earlier, bilingual web and email addresses should be used. Note that the name before the “@” may also need to be bilingual, for example: contact@... and cyswllt@...

When specifying an email address to reply to, this should either be in the preferred language of the user or the prominent language of the email.

2.6 Data Management (Section 8)

The section on Data Management deals with many software implementation and abstract details. For the purposes of this overview, only the high-level aspects will be highlighted. Should a greater degree of detail be required, it is recommended that section 8 is read in its entirety.

Any matter obtained, used or displayed by the software is known as “data”. The term ‘data management’ refers to *how* this data is stored, used or displayed in a bilingual context.

2.6.1 General Guidance

Specific concerns include:

- That the data storage system can fully support and manage bilingual content;
- That the user is encouraged (or enforced) to enter data bilingually, if possible;
- That English and Welsh data are semantically consistent;
- That a user has access to data in another language if it isn’t available in their preferred language;
- That the bilingual functionality is a functional asset rather than a liability.

An individual is entitled to communicate solely in the language of their choice. No assumption should be made about which is the preferred or mandatory language unless the user’s language preference is already known.

2.6.2 Identifying Bilingual Data

Bilingual data can include text, photos, pictures, audio or video.

Data can be “structured” (i.e. a user can make a choice from a list) or “free” (i.e. the user can type whatever they like).

When the data is structured, it is usually easy to provide translated lists with entries in common. For example:

Number	English	Welsh
1	Anglesey	Ynys Môn
2	Flintshire	Sir Y Fflint
3	Cardiff	Caerdydd

When a user selects their county in either language, the software will store the county number. That way, the county name can always be retrieved in either language.

2.6.3 Mandatory, Optional or Inappropriate Data



There will be circumstances where the nature of the system or the user community is such that it makes bilingual data entry a mandatory requirement (e.g. in a dictionary system). In this case, the user should be required to enter the data in both languages regardless of personal language preference.

2.6.4 Handling Third Party Data

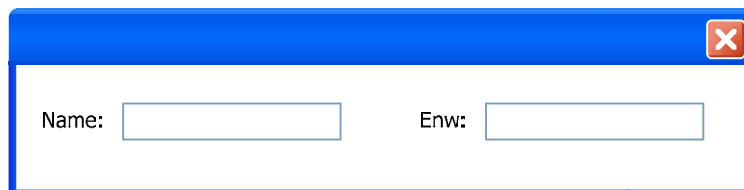
When third party data is available to the system and it is available in both English and Welsh, the data for both languages shall be used and fully supported.

Note, however, that it is not the responsibility of the software owner to ensure that data is provided in a user's preferred language when it is only available in a single language (further discussion of this point can be found in section 8.4).

2.6.5 Manual Data Entry

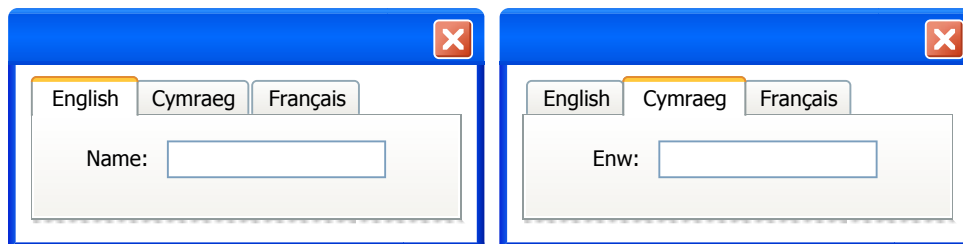
The system should allow data to be entered bilingually. The means of doing this can vary depending on whether the software is a web site, a window application, or a mobile phone, etc.

Some data entry options include placing corresponding fields in parallel or using “tabs”. Using tabs is more scalable as a multilingual solution, if only because it requires less screen space.



A screenshot of a software window with a blue title bar and a close button. Inside the window, there are two input fields side-by-side. The first is labeled "Name:" and the second is labeled "Enw:". Both fields are empty text boxes.

Parallel Data Entry



Two side-by-side screenshots of software windows. Each window has a blue title bar and a close button. The left window has three tabs at the top: "English" (selected), "Cymraeg", and "Français". Below the tabs is a "Name:" label and an empty text box. The right window has the same three tabs, but "Cymraeg" is selected. Below the tabs is an "Enw:" label and an empty text box.

Entry using “tabs”

Data will occasionally be semantically linked together in both languages. For example, if “Anglesey” is selected as a county on an English screen, “Ynys Môn” will need to be selected on a Welsh screen for the same item. On other occasions, the data may be different for each language.

2.6.6 Encouraging Bilingual Data Entry

Bilingual software systems shall employ whatever mechanisms are possible to ensure equal treatment of all languages and encourage entry of data for all supported languages.

Where the layout of the data entry screen doesn't make it completely clear and evident that there is the capability to enter both languages, indicators shall be located alongside those fields that require bilingual entry.

It may also be possible to alert the user to the fact that data can be entered bilingually as they attempt to save what they've entered.

It is also good practice to ask why data has not been entered bilingually. This gives two advantages:

- It provides the software system with an explanation that can be used when displaying the data;
- It reminds, encourages and reinforces the entry of bilingual data.

Another option is to add text to a disclaimer detailing that the software owner isn't responsible for the availability of the data bilingually.

2.6.7 Consistent Set of Mandatory Data

Some fields will require data; in others it will be optional. The software should ensure that data has been provided in all the mandatory fields of at least one language.

2.6.8 Data Storage

However the data is stored, it should be able to record all the accented letters of the Welsh alphabet correctly. This will entail checking all storage locations use an appropriate character set (as defined earlier).

Data shall be sorted using the appropriate alphabet (English or hybrid Welsh/English) as discussed earlier, wherever technically possible.

2.6.9 Unavailable Data

On no account shall data be withheld from a user because it isn't available in their preferred language. An explanation for the monolingual content should be given, as already discussed. If the data is unintentionally missing, this should be logged.

2.6.10 Transmitting Data

Whenever data (information) is shared between software systems, all the data should remain intact linguistically. This means no accented letters should be lost during transmission (a common problem).

2.6.11 Metadata – “Data About Data”

The term “metadata” can best be described as “data about data”.

As already mentioned, audio material can be considered to be data. The title and writer of a piece of music can be considered meta-data – information (data) about the music (data).

Although this sounds very abstract, metadata is used widely in software systems. Here is a further example:

Data	Meta-Data
Canllawiau a Safonau Meddalwedd Dwyieithog	Iaith: Cymraeg Pwnc: Meddalwedd
Bilingual Software Standards & Guidelines	Language: English Subject: Software

The software system should allow equivalent metadata to be provided for each language.

See section 8.11 for further details.

2.6.12 Searching

A bilingual system will contain and manage data in both English and Welsh. Ideally, all data will be held in both languages, however the reality is that some data may be available in one language and not the other.

A user should be able to choose whether to search in English-only mode, Welsh-only mode or bilingually. The default should be the current user interface language.

The following table describes what results should be returned depending on the availability of bilingual data:

Chosen Languages	Available in English	Available in Welsh	Action
English Only	Yes	Yes	Return English
English Only	No	Yes	Inform the user that additional results are available in Welsh
English Only	Yes	No	Return English
Welsh Only	Yes	Yes	Return Welsh
Welsh Only	No	Yes	Return Welsh
Welsh Only	Yes	No	Inform the user that additional results are available in English
Both	Yes	Yes	Return Both
Both	No	Yes	Return Welsh
Both	Yes	No	Return English

For each result, some indication of its language should be given.

Ideally, systems should treat accented and non-accented letters identically (e.g. w and ŵ are identical).

Refer to section 8.12 for further detail.

2.6.13 Bilingual vs Multilingual

The requirements of a bilingual software system are slightly different to those for a multilingual approach. A multilingual system typically has a base language with the ability to support a number of alternate languages. A bilingual system has two base languages and though it may also support additional/alternate languages, the two main languages (i.e. English and Welsh) must always have equal support and neither used without the other.

Consult section 8.13.3 for further detail.

2.7 Postal Addresses and Geographical Information (Section 9)

It is possible to obtain third party resources allowing a software system to use geographical information. For example, if you were to type in a post code, a map could be displayed of the area in question.

Section 9 details several different types of Geographic Information (GI) types but concentrates mainly on address data.

Address information is useful for several different reasons, including the capability to find a full address from just a post code or ensure an address has been entered correctly. There are several sources of this data in both the private and public sectors and details are given in section 9.2. It should be noted that the quality of this address information can vary from source to source.

2.7.1 Use of Postal Addresses

Wherever possible, a software system should allow users to enter addresses in whichever language (or combination thereof) that they prefer. Where an address has not been provided by a user, best efforts should be made to use an address consistent with their preferred language. Where this isn't possible, bilingual addressing should take place.

Note that the delivery capabilities of the chosen courier should be considered when making addressing decisions. Royal Mail are capable of delivering mail based upon a valid post code and building name or number.

2.8 e-Government Interoperability Framework (e-GIF) (Section 10)

The e-Government Interoperability Framework is a means through which computers and software systems can communicate with each other. It stipulates many standards with the hope that if all software systems follow them correctly, each should be able to communicate efficiently.

Many of these standards are based around metadata (as described earlier). Standardised sets of metadata are available for subjects as diverse as election and public transport-related software systems.

Section 10 discusses e-Government at a fairly low level. Should further detail be required, please consult this section directly.

3 Localisation – Guidelines, Issues and Management

This section provides general guidance on best practice when developing bilingual software applications for use within Wales, and by Welsh speakers.

3.1 Definition of ‘Localisation’

Localisation is *not* the same as translation, although translation does form a substantial part of it. At a general level, it involves addressing issues such as:

- ensuring date and time formats are appropriate for the user’s culture;
- ensuring data and lists are sorted using the appropriate alphabet;
- ensuring number formats are correct for the current culture – is ten Euros and fifty cents “€10.50”, “€10,50” or “10,50 €”?
- ensuring the user interface is flexible enough to support a specific language and still makes sense;
- ensuring that data in different alphabets can be stored accurately;
- translation of all text – labels, help files, alerts, etc – into a specific language;
- ensuring that users of a system are able to communicate in their preferred language, on an equal basis with other languages supported by the system.

The focus of this document is Welsh and English in the UK. Some of these items are not an issue – the date and number formats are the same, for example. However, there are some key issues caused by differences between the two languages. Such differences are addressed later in the document in more detail.

3.2 Reasons for Localisation

There are a number of drivers and motivations for localising a software application. Understanding the reason for the effort helps to interpret the standards documented later and the extent to which they should be applied in a particular situation.

3.2.1 The Law

By far the most significant legislation to date in respect of the Welsh language is the Welsh Language Act, 1993.

In simple terms, the Act specifies three things:

- it places a duty on the public sector to treat Welsh and English on an equal basis when providing services to the public in Wales;
- it gives Welsh speakers the right to speak Welsh in court;
- it establishes the Welsh Language Board to promote and facilitate the use of the Welsh language.

3.2.2 Bilingualism in Wales Today

Research conducted on behalf of the Welsh Language Board shows that the overwhelming majority of those questioned felt that the language was something to be proud of and supported its use.

Support for the Welsh language therefore, by individuals and organisations, is broad and widespread. Public bodies and major businesses, as well as smaller companies

and voluntary organisations are making increasing use of the language in all areas.

This situation will continue to develop, in response to customers' wishes and expectations and there is an increasing demand for the software applications used by this bilingual population to be capable of supporting the Welsh language.

3.2.3 Business Opportunity

Throughout Wales more and more organisations (in the public, private and voluntary sectors) in response to the growing expectations of their customers, are becoming aware of the benefits of using the language, such as:

- improving the quality of customer service;
- attracting new customers;
- increasing customer loyalty;
- harnessing employee and customer goodwill at relatively low cost;
- gaining a marketing edge over competitors;
- enhancing public relations efforts.

With organisations in all sectors offering more and more bilingual services, it's important that software systems are flexible and of a sufficiently high standard to be of practical everyday use.

It is also important to note that, once an application is *capable* of being localised (i.e. that a scalable approach and architecture have been used), additional languages and cultures can usually be added at relatively low cost to create a multilingual capability. This presents business opportunities in other markets for an organisation utilising a software application that embraces these standards.

3.3 Project Management of Bilingual Software Development

Software development teams are often dynamic, changing with project and company requirements. It is therefore important to make all team members aware of all the issues concerning bilingual software development and localisation at the start of a project. It may be advisable to assign a team member to promote and supervise localisation matters during the progress of the project.

3.3.1 Requirements Management

Once the decision has been made to localise an existing package or to ensure a new development is to be compliant with these standards, the technical challenge is minimal since all standards can then be defined as functional requirements.

It is strongly recommended that Welsh language support and bilingual capabilities are defined along with the main set of requirements for the application. A reference to this standards document might work from a contractual perspective, but will not resolve any ambiguities or interpretation required to apply these standards to a specific situation.

Further, not having explicit and detailed references to how the capabilities will be implemented can easily result in a system design that will 'bolt on' the language capabilities as an afterthought. This usually results in poor quality support, limited flexibility and additional ongoing cost in maintaining the bilingual quality of the system.

3.3.2 Testing & Quality Control

It is important to ensure that testing is done equally for both Welsh and English versions of software. Ideally, a single version of the software will be created with the language specific aspects coming from a single resource or database (see 'Architectural Approach', section 5).

It should be noted that any changes or fixes may affect the stability or functionality of one language detrimentally and should be performed in a controlled manner, with due attention given to both Welsh and English. Again, the project and requirements management approaches outlined above and the architectural suggestions in section 5 will help to mitigate this problem.

Acceptance and regression test cases should be defined that will check and ensure that the quality of language support for an application remains high. Specifically, test cases should be included that verify that the application is consistent with the requirements of these standards.

Essentially, processes to ensure compliance with these standards should become an intrinsic part of all quality control processes related to the production and management of the software application.

3.4 Translation & Linguistic Quality

Though this document doesn't intend to set standards for translation or linguistic quality, awareness of the need to produce high quality translation is important.

Throughout the software development lifecycle, an understanding of the translation process is important to ensure, encourage and accommodate high linguistic quality of the user interface and of data management by the resulting application.

This section will outline some key issues to be considered. Throughout the remainder of the document there will be ongoing references to the architecture, functionality and standards that will support the translation activity.

3.4.1 Consistency and Use of Standard Resources

There has been substantial effort already invested into the production of bilingual software. This has resulted in comprehensive terminology databases as well as standard lists produced for common uses (county names, salutations, etc). Section 12 contains references to where these are freely available.

Every effort should be made to utilise this standardised terminology wherever possible. Though this document doesn't intend to set standards for linguistic quality, the need to maintain consistency with established terminology for the translation of user interfaces does fall within its scope. It is also the Welsh Language Board's policy that only accredited and qualified translators should be used.

3.4.2 Working with Translators

When providing text for translation, it is best to provide as much contextual information as possible. This may take the form of screenshots, annotations or a pre-release

version of the software. Annotations are extremely important and help to resolve potential ambiguities. For example, does “Clear Text” mean “text that is clear” or “clear the text away”, i.e. is it adjectival or imperative? Is “copying” a noun or adverbial? In providing a pre-release version of any software, translators can also be used as non-technical beta testers.

It is important to supply the translation and supporting material in a form that is understood by the translators. For example, provide one table of text strings or a translation user interface, rather than expecting the translator to understand and work directly with the underlying database.

Ensure that the project plan allows time for translation work to occur. Also ensure that the translators appreciate how their work fits into the larger plan. Do not just assume that the translators will be able to do the translation whenever the first cycle of development happens to be complete.

You should also consider how appropriate is the use of terminology management and translation memory software in the outsourcing of translation.

3.4.3 Outsourcing Translation

Where external resources are used for the translation activity, every effort should be made to ensure that they are adequately skilled and capable. Evidence of prior work should be sought and membership of professional bodies such as the Association of Welsh Translators – CCC (see 13.3) should be a prerequisite.

4 Locales, Alphabets and Character Sets

This section will first discuss what a locale is, why it is significant to a software application and the standards and guidelines relating to how locales should be used in a bilingual software system.

The English and Welsh alphabets will then be examined, highlighting their differences and defining how a bilingual system should support the use of both alphabets, usually simultaneously.

Character sets are then discussed, which ones should be used, associated issues, how a user should be enabled to enter less common characters and the factors influencing the selection of a suitable font.

4.1 Locales

A locale is the definition of a set of characteristics that describe the operating environment for a software application that enable the application to meet the cultural, language, interface and formatting expectations of the user.

A locale will typically specify:

- Language (including language formatting issues, such as text direction);
- Number, currency, date and time formats;
- Sort order;
- Keyboard layout;
- Calendar and Time Zone specific information;
- Information on a range of cultural conventions.

The locale is core to the concept of localisation of a software application. This section will define the standards associated with the use and interpretation of locales for bilingual software applications.

4.1.1 Locale Identification

The locale defines the language and the characteristics of the location. However, there can be multiple languages spoken in a single location as well as many locations where a particular language is spoken. Therefore, locale identifiers reflect both of these aspects.

The most common means of referring to a locale is by the pairing of an ISO-639 two-letter language element to an ISO-3166 country element. Using this notation, the two locales relevant to bilingual systems used in Wales are:

- English (United Kingdom), en-GB, etc
- Welsh (United Kingdom), cy-GB, etc

This notation is commonly found in content transmitted over networks – HTML, XML, etc.

A three-letter ISO-639 notation is also sometimes used when only referring to the language (and not the culture):

- English – eng;
- Welsh – cym;

British English and Welsh locales can also be referred to using their hexadecimal Locale Identifiers (LCIDs) as follows:

- English (United Kingdom), 0809;
- Welsh (United Kingdom), 0452.

LCIDs are commonly used for code-level operations, particularly when interfacing with an operating system.

The relevant identification approach will depend upon the operating platform and the interface to the locale management on that platform.

This version of this document will focus on the Welsh and English language locales for the United Kingdom only. However, the discussion will be general enough to allow its application to other locales where the Welsh and/or English language is spoken (e.g. U.S.A., Patagonia, Australia, etc.). Whether formal locales are defined in these regions is a local issue and should be addressed with the software producers for those regions that contribute to the operating platform defining the locale.

4.1.2 Using a Locale

A locale can be defined in many places. It is typically defined by the operating system (selected by the user at installation time, or modified later). It is also commonly defined as part of the user profile, by a web browser or by some other hardware or software environment that hosts the application. We will refer to this collective and often hierarchical environment as the 'operating platform'.

Wherever possible, a software application should seek and utilise all such available information and take all environment and contextual clues to determine the preferred means to interact with the user. Such an approach can end with conflicts. Where this occurs an order of precedence should be established, documented and used to guide the development of the application (see 4.1).

Explicit Language Selection

A user's explicitly defined language preference shall always override any implicit assumptions of language preference.

Though it is preferable for a software application to obtain the locale specific information from the operating platform, this isn't essential, since this information can be 'hard coded' into the application where the number of locales to be supported is fixed and known in advance.

Having said this, it is strongly recommended that an application is designed to be flexible enough to identify the locale and obtain all locale information from the operating platform. This will allow for more robust software designs and result in broader multi-locale and multi-lingual capabilities.

Use of Locale

It is essential that any software application is capable, in some way, of understanding the locale preference of the user and ensuring that it adopts the appropriate rules for that locale.

Architectural and design considerations for locales are discussed in section 5.3.

4.1.3 Differences Between British English and British Welsh Locales

The English and Welsh locales for the UK are, as one would expect, very similar. The date and time formats are the same, the calendar and time zone information is the same and most cultural, social and political aspects are also held in common. Essentially, all 'location' aspects of the locale are identical.

The differences lie in the 'language' aspect of the locale definition. Again, there are many commonalities, such as the text direction and the base character set (i.e. Latin). This section will identify the differences that should be accommodated by bilingual software applications.

The differences are:

1. Language. A Welsh (UK) locale will define the Welsh language as being the default language to be used for all aspects of user interaction;
2. Sort order. The Welsh alphabet has a different sort order to the English alphabet (see 4.2.3);
3. Keyboard. Though the keyboard has the same base layout, a Welsh locale might define alternate key sequences to assist in the entry of diacritic marks (see 4.3.4);
4. Calendar. Though the UK location defines the calendar (Gregorian) and time zone, some accommodation is required to support the different naming used for days of the week (see 4.1.3.1);
5. Ordinality. The grammar of the Welsh language varies from English when describing an ordered sequence of objects (see 4.1.3.2).

4.1.3.1 Calendar Abbreviations

When working with calendars, it is common practice to abbreviate day and month names to just use the initial letter. When doing this, there are two considerations in accommodating the Welsh language.

Firstly, for day names, where English uses the 'day' as a post-fix, thereby making the first character of the name for each day unique (or almost), the grammatical structure of Welsh dictates the opposite. The practice is therefore to abbreviate the first letter(s) of the unique part of the day name (the second word), resulting in the abbreviations 'Ll, M, M, I, G, S, S' or 'Ll, Ma, Me, Ia, Gw, Sa, Su).

Grammar for Abbreviations

The grammatical structure of the language shall be accounted for when forming abbreviations and acronyms.

Secondly, the use of digraph letters as the initial letter of some months and days means that the abbreviation should allow for the use of two characters (e.g. the letter 'Ll as an abbreviation for 'Llun' and not just the 'L' character).

Digraphs in Abbreviations

Digraphs, shall be accounted for when forming abbreviations and acronyms.

(Note: Though the abbreviations for days of the week have been considered here, sensitivity to the grammatical structure of the language is important with all abbreviations. This is covered in a more generic manner in section 6.1.2).

4.1.3.2 Ordinals

When using ordinals, Welsh grammar is more variable than English. For instance, where in English you would describe the 'First Tuesday', 'Second Tuesday' or 'Third Tuesday' of the month, Welsh grammar will say the equivalent of 'First Tuesday', 'Tuesday Second' and 'Tuesday Third'. The difference being the movement of the ordinal number from preceding the object to succeeding it.

On the face of it, this is just one aspect of the grammar of the language and in most cases, it is just a translation issue. However, it is particularly relevant for software applications that provide calendar management, and where support is provided for scheduling events. In cases such as this, it is important that the user interface provided is neutral to this issue by not requiring a fixed relationship between the ordinal number and the object. This issue is discussed further in section 6.2.5.

4.1.4 Minimising Impact of Locale Differences

Data Management (section 8) will provide a detailed discussion on how to minimise the effort required developing, using and maintaining data in a bilingual software application.

However, there are recommendations that are specific to locales, particularly for calendars. These are:

- If a fully-worded date is desirable for aesthetic or other reasons, this should be retrieved from the operating platform where this may have already been localised for the relevant language;
- Where possible and where it doesn't detract from the usability of the software, using numbers in place of month and day names minimises the translation effort and functional differences between the user interfaces for each language. For instance, 'Wednesday, December 1st, 2004' will require translation, whereas '1/12/2004' is language-neutral;
- When using ordinal numbers in dates (indeed in any context), avoiding the use of the ordinal suffix (e.g. st, nd, rd, etc) avoids the need to maintain a translation and making the ordinal number language-independent. For instance, using '1 December 2004' instead of '1st December 2004'.

This approach also provides benefits by making the application more flexible when used in locales with more variation in date formats where the underlying locale information will provide a better presentation of just numeric dates than those with textual information.

4.2 Alphabets

The Welsh alphabet has 29 letters whereas the English alphabet has 26. The difference is accounted by:

- Digraph letters that occur in the Welsh alphabet: *ch, dd, ff, ng, ll, ph, rh, th*;
- Letters of the English alphabet that are not in the Welsh alphabet: *k, q, v, x, z*.

Digraph letters are composed of two characters, but are treated as a single letter. The full Welsh alphabet is therefore:

a, b, c, ch, d, dd, e, f, ff, g, ng, h, i, j, l, ll, m, n, o, p, ph, r, rh, s, t, th, u, w, y

These differences result in two main issues that need to be addressed:

- The sort order is different from the English alphabet and is further complicated by the presence of digraph letters;
- For a text string, the character count will likely be different to the letter count.

Welsh vowels can also be modified by diacritic marks (e.g. *â, ô, w̃*, etc.). This issue will be covered in the discussion of character sets (section 4.3)

4.2.1 Character vs. Letter Counts

Where a count is made of the length of a text string, it is important to be aware that the number of characters will typically be less than the number of letters due to the potential inclusion of digraph letters.

This is unlikely to cause a challenge in the majority of cases, since software applications conventionally just refer to the number of characters in a string and therefore already have a language-neutral approach. This is also the recommended approach whenever appropriate.

However, in a situation where a letter count is necessary, it is important that any processing of, or reference to, Welsh language text is sensitive to the use of digraphs and counts them as a single letter. This will likely require a reference to dictionaries to achieve and this requirement should be factored into the design of a system.

4.2.2 Selecting a Sort Order

It is common (particularly when using proper names) for both English and Welsh words to be intermingled. However, a valid sort is still required.

Although the alphabets have differences, if a superset of all letters is used, there is no conflict in the sort order with the modified alphabet therefore becoming:

a, b, c, ch, d, dd, e, f, ff, g, ng, h, i, j, k, l, ll, m, n, o, p, ph, q, r, rh, s, t, th, u, v, w, x, y, z

This should be used for a Welsh language interface and to sort Welsh language data. However, when sorting in an English language interface, the English alphabet should be used without the digraph letters to avoid confusing English language users.

Welsh Sort Order

When displaying a Welsh user interface and/or working with Welsh language data, the sort order shall be based upon a superset of the English and Welsh alphabets.

English Sort Order

When displaying an English user interface and/or working with English language data, the sort order shall be based upon the English alphabet alone.

To further clarify this point, the sort order to be used depends upon the language being used for the user interface, not the language of the text being displayed.

4.2.3 Sorting Digraphs

The difference in alphabets means that data needs to be sorted differently, depending upon the language being sorted.

Though the single character letters are in the same order in both languages, the digraph letters create a challenge since the second character doesn't get compared to the second character of other words in order to determine the sort order.

For example, *label*, *lori* and *llefrith* are sorted correctly in Welsh.

Two digraphs (*Ng* and *Rh*) cause further challenges since there are cases where it is impossible (without reference to an annotated dictionary) to know whether a word contains 'ng' or 'n' then 'g'. For example:

- C-a-ng-e-n – Cangen;
- B-a-n-g-o-r – Bangor.

In situations like this, it is important to adopt a consistent strategy. In the case of both 'ng' and 'rh', the digraph occurs far more frequently in Welsh than the 'n', 'g' (or 'r', 'h') sequence of letters and therefore treating all instances as the 'ng' (or 'rh') letter is generally accepted as the most appropriate. I.e. the recommended approach is to treat all plausible occurrences of a digraph as a digraph.

Though this will introduce a small amount of inaccuracy in the sorting, without reference to a lookup dictionary it isn't possible to gain a better degree of accuracy.

Using a dictionary as a lookup resource is the ideal approach and this should be done wherever possible. Where it isn't, a consistent strategy should be used to sort digraph letters.

This can be summarised as three levels of sort order capability:

1. Do not recognise the Welsh digraphs at all and just use the English alphabet. This is the recommended approach for an English language interface (see 4.2.2), but is discouraged for a Welsh language interface (and will contravene these standards);
2. Sort all text in a Welsh language interface with any plausible digraph being assumed to be, and treated as, a digraph. This is the minimum approach to be used in order to be compliant with these standards;
3. Use a dictionary lookup resource to distinguish digraphs when sorting in a Welsh language interface. This is the ideal approach, but isn't always practical and should be used when authentic handling of the language is required and appropriate.

Sorting Digraphs

A consistent strategy and algorithm for the correct sorting of digraph letters shall be defined and used throughout the application wherever textual information is sorted.

Diacritic marks do not affect the sort order; a vowel with a diacritic mark is sorted in the same order as the original vowel.

Sorting Characters with Diacritic Marks

Characters with diacritic marks should be treated as equivalent to the original vowel when sorting and should be sorted in the same order.

4.2.4 Combining Grapheme Joiner

Unicode provides an alternate solution to the digraph sorting problem, the 'Combining Grapheme Joiner'. This is a special (not ordinarily visible) character (+u034F) that can be inserted between two characters that would otherwise be assumed to be a digraph.

E.g. the 'll' in 'Williams' can be separated as follows: 'Wilu034Fliams'

Though this is a useful capability, it is fairly obscure and might not be expected to be in general usage by a typical user community. However, it is part of the Unicode standard and should be treated as acceptable input and also handled properly where it is present.

Where it is possible, a software application should allow for a user to enter the Combining Grapheme Joiner character in any textual field and for this character to be stored and processed with integrity along with the rest of the text string.

When the sort order takes account of digraphs (i.e. in a Welsh language interface), the algorithm used should account for the Combining Grapheme Joiner character and allow it to affect the sort order.

4.3 Character Sets

Though the Welsh digraph letters can be represented by the use of two characters per letter, the letters with diacritic marks can cause problems if the character set used doesn't support them.

In Welsh, the diacritic marks used are the circumflex (â), the acute (á), the grave (à) and the dieresis (ä) and can appear above vowels (a, e, i, o, u, w & y). This results in 28 characters (56 if you include both upper and lower case) that the character set must support.

Not all character sets support all of these characters. For instance, the ANSI standard only supports those characters that are common to a number of major languages (e.g. á, ô, etc).

Character Sets

A bilingual software application must use a character set that supports all 56 diacritic characters as well as all letters of the English alphabet. This character set should be

a commonly used and internationally recognised standard to ensure the ability to share text data with other applications.

Of these, Unicode (also defined by ISO/IEC 10646:2003) is the most widely used and understood standard. Further information on Unicode can be found at www.unicode.org.

The full set of diacritic characters used in Welsh and their Unicode character code is:

Circumflex		Acute		Grave		Dieresis	
Â	0194	Á	0193	À	0192	Ä	0196
Ê	0202	É	0201	È	0200	Ë	0203
Î	0206	Í	0205	Ì	0204	Ï	0207
Ï	0212	Ó	0211	Ò	0210	Ö	0214
Û	0219	Ú	0218	Ù	0217	Ü	0220
Ŵ	0372	Ŵ	7810	Ŵ	7808	Ŵ	7812
Ŷ	0374	Ý	0221	ÿ	7922	ÿ	0376
â	0226	á	0225	à	0224	ä	0228
ê	0234	é	0233	è	0232	ë	0235
î	0238	í	0237	ì	0236	ï	0239
ô	0244	ó	0243	ò	0242	ö	0246
û	0251	ú	0250	ù	0249	ü	0252
ŵ	0373	ŵ	7811	ŵ	7809	ŵ	7813
ÿ	0375	ý	0253	ÿ	7923	ÿ	0255

For reference, these symbols can be found in the following Unicode code pages:

- Latin-1 Supplement
- Latin Extended-A
- Latin Extended Additional

Alternatively, the ISO-8859-14 character set can be used. However, this character set has not been adopted as widely as Unicode.

4.3.1 Explicit Specification of Character set

The necessity of ensuring that a HTML or XML document supports the full set of Welsh characters requires that the character set (or encoding) to be used is explicitly defined.

When using Unicode in HTML documents, this is achieved by inclusion of the 'charset=utf-8' declaration in the <meta> tag. For example:

```

.....
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
.....

```

Within XML, use the 'encoding="utf-8"' declaration:

```

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

```

Other approaches will exist for other technologies and languages.

Explicit Specification of Character Set

Wherever it is possible to explicitly define the character set to be used, this should be done to ensure that the full set of both Welsh and English characters can be supported.

4.3.2 Operating Platform Configuration

Some operating platforms (e.g. some web servers) will require specific configuration in order to correctly identify and manage a UTF-8 character set.

Operating Platform Character Set Configuration

Where necessary, the operating platform must be correctly configured to recognise and correctly process UTF-8 character sets.

4.3.3 Encoding of non-ANSI Characters

There are still a large number of software applications and file formats that are ANSI based (i.e. do not support the Unicode character set). When faced with this situation, it is often possible to circumvent the lack of support through encoding the character so that, even though the software application being used will not recognise it, a Unicode-capable application will be able to.

For example, even though the 'charset' and 'encoding' specifiers (see section 4.3.1) will direct the HTML and XML to handle the full Unicode character set, it is still necessary for the tools used to edit the HTML and XML to support these characters. Unfortunately, many of these tools are still ANSI based and it is therefore necessary to encode any non-ANSI characters so that a Unicode based web browser will be able to interpret this encoding and display the characters correctly.

The HTML & XML character code is simply the Unicode character code (as shown in section 4.3) preceded by the &# sequence. For example, the ŵ and ŷ codes are:

Character	HTML Code
Ŵ	Ŵ
ŵ	ŵ
Ŷ	Ŷ
ŷ	ŷ

4.3.4 User Input

This section will provide recommendations and establish some basic standards relating to the ability for a user to enter diacritic marks into a software application.

4.3.4.1 Keyboard Data Entry

A number of standard UK keyboards do not have provision for the easy entry of characters with diacritic marks. Typically sequences of key presses are defined to facilitate the typing of these characters.

There are currently several established schemas in use for these sequences and this document will not define an overall standard or show a preference for a schema. However, there are a number of standards that software applications should meet in this area:

- Any key sequence schema should make the entry of the most commonly used characters (those with a circumflex mark) as simple as possible (i.e. without the use of ‘dead keys’, though these might be required to support the less common characters);
- Unless the user has explicitly requested to do so, no application shall intentionally impose a key sequence on a user that will override any existing configuration (particularly the Alt+nnnn sequences which should not be disabled under any circumstance);
- If a keyboard schema is provided by an application, the user should be provided with an easily accessible facility to temporarily or permanently disable the schema;
- Where a software application uses key sequences for other purpose (e.g. hot keys, shortcuts, etc), there should be options for users to either disable this capability or modify the sequences;
- Where a software application (typically an Operating System) installs keyboard support, the default keyboard configuration set up should be one that supports the Welsh characters if the user’s locale is set for Wales and/or the Welsh language;
- Any keyboard schema that is identified as a Welsh keyboard shall support all 56 of the Welsh diacritic characters.

An alternate and commonly used approach is to allow a user to define their own key sequence via a configuration utility. If this approach is used, the default schema supplied and any user permitted configuration should still adhere to the guidelines above.

Two popular schemas are the Microsoft schema introduced by Windows XP service pack 2 and the ‘To Bach’ (“circumflex”) schema defined by Draig Technology and also used in several open source implementations such as that provided by Meddal (see 13.2).

4.3.4.2 Transliteration of Diacritics

Where an application is unable to access the keyboard stream (e.g. for applications hosted within a browser), a transliteration can be used (e.g. replacing the character sequence ‘w^’ with the ‘ŵ’ character).

Diacritic Transliteration

When an application is unable to explicitly define keyboard sequences to support the diacritic characters it shall provide a transliteration capability as follows:

Accent Type	Transliteration	Example
Circumflex	char & ^	w^ = ŵ
Acute	char & /	a/ = á
Grave	char & \	a\ = à
Dieresis	char & “	O" = ö

The application shall then replace the transliteration with the appropriate character.

Such an approach should comply with the same approach outlined for key sequences in 4.3.4.1 (as applicable).

4.3.4.3 Other Data Entry Modes

Other data entry modes (e.g. speech, touch screen, handwriting, graffiti, TV remote, etc) should provide the capability for a user to enter diacritic marks or provide easy transition to a mode to permit this.

If the technology or constraints of the data entry mode do not permit for this, then the application should make use of a lookup dictionary and provide the user with the ability to select alternate spellings so that it is always possible for the user to correctly spell words.

4.3.5 Choice of Font

Section 4.3 discussed the character set and encoding standards to be used to support the Welsh language. Complying with this requirement usually provides a range of candidate fonts that can be used. Careful selection of a font is important to ensure readability, accessibility and interoperability with other applications.

Guidelines for the selection of an appropriate font are:

- Do not use fonts with long ascenders, descenders or those that are too round or angular;
- The font must cater for the full Welsh alphabet (including the 56 diacritic characters);
- Do not use a designer font that uses a non-standard approach for digraph letters and diacritic marks;
- Do not use pseudo-Celtic or calligraphic fonts;
- Use commonly available fonts (i.e. UTF-8 Serif and Sans Serif) to ensure compatibility with other applications;
- Consider accessibility issues, particularly for the visually impaired, in the Context of the Disability Discrimination Act 1995.

Further information on the selection and usage of fonts can be found in the document 'A guide to bilingual design' published by the Welsh Language Board.

5 Architecture and Design

Before proceeding with further aspects of bilingual functionality, it is important to discuss and present the standards and guidelines that will influence the architecture of a software application to enable it to support multiple languages.

This section will address these issues and cover areas such as the ability to switch between languages, ensuring consistency of content between the languages and the provision and integration with language support toolsets.

5.1 Language Selection

When a user accesses a software application, a determination will need to be made of which language to display. This can be achieved through various methods.

When the user first accesses the application and they cannot be readily identified, either:

- An implicit assumption where the user has not yet identified themselves and therefore it isn't possible to refer to an explicitly stated language preference;
- The use of an initial dedicated language selection function, presented bilingually.

During further use of the application, or when the user can be positively identified:

- A ubiquitous language selector allowing the user to switch languages at any time;
- Reference to a stored/configured language preference for the user (only possible if the user is explicitly or implicitly identified).

5.1.1 Initial Implicit Assumption

When a user first accesses a software application, unless the user can be implicitly identified, then an assumption will need to be made as to their preferred language.

There are a number of indicators that can be used to guide this assumption:

- The locale and/or language preference (e.g. cy-GB) defined for the user profile (or on the operating platform). This might come from the operating system itself or the runtime environment hosting the client side application (e.g. a browser, which will pass this information in the HTTP header). Section 4.1 addresses this in more detail;
- The means of accessing the application. This might contain an explicit or implicit indicator of the preferred language. Some examples of this are:
 - The use of a particular domain name to access a web application where both Welsh and English domain names are available (see 5.4);
 - The presence of two software application start-up files, one in each language where the user is able to select the one most relevant to their language preference;
 - An explicit parameter passed in a query string, as a command line switch or some other start-up input recognised by the application;
 - The use of a link or shortcut that indicated the users preferred language whilst using a referencing application;

Any determination of the preferred language through such an implicit method should not be recorded for future use and should only determine the initial language and only for that session.

5.1.2 Dedicated Language Selection Function

An alternate approach to determine the initial language for a user is to present a dedicated language selection function to enable the user to make an explicit language selection. Common examples of this approach are ATMs, Information Kiosks and a number of web sites.

When adopting this approach, the following issues should be considered:

- The language selection interface should be as simple as possible, with no other functionality available at the same time (other than to exit);
- Use of graphics should be limited to a simple welcome message and basic branding so as not to distract from the need for the user to select a language;
- All language-sensitive elements should be bilingual with every care taken to ensure there are no prominence or preference issues;
- Once the initial language selection has taken place, the user should not need to return to this function since the ability to switch to an alternate language ‘on the fly’ should appear throughout the remainder of the user interface.

Further guidance and recommendations on design considerations can be found in Appendix A and also in the ‘Snapshot Survey’ of websites and other resources in section 12.

5.1.3 Ubiquitous Language Selector

This approach will require a language selector to be present at all times that an interactive user interface is present. Generally this is the required approach (the exception being that discussed in 5.1.4) and should be used irrespective of the method used to determine the initial language preference (in 5.1.1 and 5.1.2 above).

Ubiquitous Language Selector

The user shall be given the means to switch language between Welsh and English at any time and in any stable state of their interaction with a software application.

Language Selector

The language selector shall be equally visible, accessible and placed in the same location throughout the entire software application. It should be equally prominent irrespective of the current user interface language.

The language selector must be as accessible and intuitive for the user as possible. Specifically, the representation used for the selector should be obvious to the user of the language that the selector represents (as opposed to the current language of the interface).

For graphical user interfaces (including web sites), the convention is to place the language selector in the top right for left-to-right languages (the “sweet spot”).

In the case of a bilingual system, the selector will simply be a single selection, that being for the alternate language. For a multilingual system, the selector shall be a list of all alternate languages. For example, if a textual selector is used, it should be 'Cymraeg' to select the Welsh language whilst in an English interface and 'English' to select the English language whilst using a Welsh interface.

The use of national flags to denote language is strongly discouraged (see 6.2.7).

Immediate Language Switch

When a language selection takes place, the interface should switch immediately to the requested language. In so doing, the interface presented shall be equivalent, represent the same state of functional interaction and maintain the same data and context as immediately before the selection.

If the user is partway through a multi-state process or has partially entered data, then the interface will remain in the same state and no data loss will occur as a result of the switch. For example, if a user has entered data into several fields on a form without having saved the data and then switches language, that data should not be lost.

There is an acceptable exception to this standard where an immediate switch will cause an atomic or critical transaction to fail (such as processing a credit card transaction). In such a situation, it is appropriate to defer the request until the transaction has completed. However, the language switch should occur as soon as is practically possible after this point.

Also, there are often architectural and/or operating platform limitations that might restrict the ability to perform an instantaneous switch. In such cases, this standard should interpret 'immediate' as being the next action to take place rather than instantaneous.

Data or Context Loss Advisory

If it isn't possible to avoid the complete or partial loss of state or data context, the user must be advised of the degree and impact of this and provided with the option to accept this loss and proceed or to decline the language switch and continue with the current language.

5.1.4 Configured Language Preference

Where a user has made a positive identification to a software application (i.e. logged in), or where a software application is on a device with a dedicated user (e.g. mobile phone, PDA, etc), the ubiquitous selector can be replaced by an ability to explicitly configure the default language.

This will require the user to be made aware that this capability exists and also for the configuration menu/facility to be easily accessible in the event that a user isn't fully competent with the current interface language.

5.1.5 Non-Graphical Interfaces (e.g. Call Centre Telephony Software)

Where a non-graphical user interface is used (i.e. voice, etc), an approach should be used to enable a user to select their preferred language.

For instance, in the case of a contact centre, we recommend an approach where the contact centre software offers the caller a language choice as early as possible in the call. Thereby averting the need to maintain two separate phone numbers which is a less desirable approach.

5.2 Persistence of Language Selection

Persistence of Language Selection

When a user interface switches to a language as a result of an explicit request, the user interface will remain in that language for the remainder of the user session or until the user makes another explicit request.

Note: there is an exception to this standard and this is when a user, after making a language selection, explicitly elects to return to an earlier application state (i.e. a back button on a browser). In such a circumstance, the user interface language should also revert to that which was used in the state that the user returned to. I.e. a back button can also undo a language selection.

5.3 Storing Language Preference

Whenever a user explicitly selects their preferred language, the most immediate consideration is to ensure that the software application switches to that language. However, there is also an important secondary action that should take place, which is to note this expression of their preference so that future sessions with that user can default to their preferred language.

How this is done will depend upon the architecture and facilities of the operating platform and also the capabilities of the application itself. This section will outline some approaches to this.

5.3.1 Application Level Profiles

If the user has logged in, or otherwise clearly identified themselves to the application, and a record of that user and their profile is maintained by the application, then this is the most relevant place to record the language preference information. This way, whenever that user logs into the application in the future, their preferred language (or, at least, their last known preferred language) shall be used. This record of the preferred language should be updated each time that the user selects a language. This stored language preference can then be used when the application starts up in the following manner:

- If a user has not expressed an explicit preference (i.e. set their preferred language in their user profile for the application), then their 'last selected' language can be assumed to be their default preference;
- If a user has explicitly stated their preferred language in their user profile for the application, then this will be used and will not be overwritten by a language switch. In this case, the language selection will only persist until the end of the current user session or a further language selection is requested by the user.

5.3.2 Operating Platform Profiles

The user's language preference can also be recorded in some form of persistent storage associated with the operating platform. If the operating platform allows for the management of user profiles, then the preference should be stored in a user specific area so that it doesn't impact other users of the platform.

Some examples of how this can be achieved are:

- The use of cookies for web based applications;
- Registry entries for Windows based applications;
- Start-up or configuration files (e.g. .ini files) specific to the application or user;
- Locale settings, providing that the application has the 'authority' to modify these (see 4.1.2);
- Use of a user directory accessible by the operating platform (such as Active Directory, LDAP compliant directory or other form of user data store);
- Other approaches, depending upon the capabilities of the operating platform.

The benefit of this approach is that it allows the users preferred language to be identified by a software application before the user has 'logged in'. Further, this preference information can be shared by other applications.

There are two key considerations with this:

Firstly, for a software application to rely upon information stored outside of its scope of control the application must be prepared to handle the absence of the information.

Secondly, depending upon the operating platform, user expectations, privacy and security issues, it isn't always possible or appropriate for a software application to record persistent information outside the scope of its own data stores. These issues should be taken into consideration when making the decision of how to record this information.

5.4 Domain Names

Bilingual Domain Names

If the domain name used to access an application isn't language-neutral, there must be one name for each language. Each domain name should be used and promoted equally and can be used to assume an initial preferred language for a user in the absence of a more definitive indicator (see 5.1.1).

For example, Business Eye uses both www.businessseye.org.uk and www.llygadbusnes.org.uk.

Internationalised Domain Names (IDNs) will allow diacritic characters to be used in domain names (e.g. www.lônfawr.co.uk). When these are used, an equivalent standard or language-neutral domain name should also be registered and used. This is to ensure that the application remains accessible to users who aren't aware of this (or have it disabled for security reasons) and to older technology platforms that are unable to utilise this relatively recent capability.

5.5 Consistent Presence of Chosen Language

Consistent Availability of Chosen Language

Functionality and content shall be available equally in both Welsh and English, such that a user, having chosen a preferred language, is able to navigate and use the software solely in the language of their choice.

5.5.1 Absence of Language-Elements

This is where any language dependent element (be it a whole area or just a single item of text/graphic) of the user interface isn't available in the required language. A software application should be self-auditing in this respect. Where the application would normally log an error or exception for a functional failure, then the language absence should be treated in the same manner and at the same severity level as a non-catastrophic error.

Absence of Language-Elements

A user with a preference for a particular language interface shall not have to use an alternate language interface or any alternate language text unless it is unavailable in their chosen language. Where this occurs, it shall be treated as any other failure with the incident being logged and flagged for attention by an operator or system administrator where possible.

If the language-dependent element is available in the alternate language, then the application should use the user interface element from that language in order to enable them to achieve their objectives in using the software. There should be some acknowledgement to the user that this has been done.

This should be logged and reported as a failure. However, the notification of the problem to the user should not interrupt the users functional use of the application, i.e. an explanatory comment located near the incorrect text is appropriate but a pop-up dialogue would just irritate the user further.

5.5.2 Predicting Absence of Language-Dependent Data and Objects

Where a software application manages bilingual data or objects it should always attempt to provide the data or object suited to the preferred language of the user.

Where this isn't possible due to the data or object being unavailable in the preferred language, then the data or object for the alternate language should be used. In this event, some indication should be provided to the user that this is the case along with an explanation of the cause for this.

If the absence of the data or object can be determined in advance, then the user should be advised of this before selecting the function to display the data or object so that they are able to make as informed a decision as possible.

E.g. if a user is presented with a link to view a document in a Welsh interface (e.g. in a list of search results) and that document is only available in English, then this should be indicated in the link. For example: 'document-name (English only)'

5.5.3 Identifying Inconsistency

An essential quality check for a bilingual application is the ability to determine whether all user interface elements are available in each language.

Where possible, there should be an ability to conduct an offline analysis (i.e. report) of all user interface elements to identify those that are not available in all supported languages.

5.6 User Interface Structures

Structure of User Interface

Bilingual software applications shall structure the user interface to ensure and promote equality of functionality and content across both languages.

There is a range of structuring approaches that can be used, four of these will be discussed. In order of sophistication and capability these are:

1. **Parallel/Mixed Text.** One version of the application is produced containing equivalent English and Welsh displayed either as mixed or parallel text;
2. **Parallel Mirroring.** Two versions of the application are produced, one containing English and the other containing Welsh with the ability to switch between the languages;
3. **Switchable Embedded Content.** One version of the application is produced, with the text for the two languages embedded within the application, usually at the user interface layer;
4. **Resource-Based.** One version of the application is produced with the text for each language stored in a data store outside the application.

This numbering shall be used to refer to these architectures elsewhere. For instance, the Switchable Embedded Content approach shall be referred to as a 'Level 3 Architecture'.

Of these approaches, the Resource-Based architecture (level 4) is the most powerful, flexible and reliable and provides the most capabilities in terms of language management and maintenance.

However, all four approaches have the potential to produce applications that will satisfy these standards. The trade off is balancing the initial development cost against the ongoing maintenance and management cost when trying to achieve the same level of quality, capability and compliance.

5.6.1 Parallel/Mixed Text (Level 1)

This is the most basic approach. It attempts to include both Welsh and English at the same time throughout the user interface.

This approach suffers from many significant drawbacks and the cost, effort and functional tradeoffs required to fully meet these standards can be prohibitive. In addition, beyond basic web sites (limited functionality and/or static content), design quality is poor.

However, this can be an effective approach where constraints exist (particularly when using third party applications) and to use mixed text is the only possibility to present a bilingual capability. However, when alternatives exist or there is an opportunity to define or influence the design these options should be given serious consideration.

This approach will not be considered any further in this document. That isn't to say that it isn't a valid approach where it is used, but that it isn't a recommended approach. Where this approach is selected it is suggested that potential impacts and costs are carefully considered before committing to it.

5.6.2 Parallel Mirroring (Level 2)

This is the approach typically adopted for web sites and software applications with little functionality and relatively static content. It is the next simplest approach to implement, but costs can quickly escalate if the scope of the system becomes more complex or if ongoing changes are made to content or functionality.

Though suitable for small scale web sites, this approach can lead to substantial quality problems, inconsistencies and high maintenance costs if used for any solution that is required to scale in either complexity or content. Therefore, where this approach is adopted, it is recommended that an upgrade strategy to a more sophisticated architecture is planned to handle any future increase in scope or complexity.

The parallel mirroring approach establishes separate and parallel resource structures that 'mirror' each other with identical content (in the respective language) and functionality.

The primary benefit of this approach is its simplicity and that it can be implemented using only basic technologies. These are also its primary drawbacks since it doesn't scale very well to larger and more complex applications and the lack of more sophisticated technologies leads to a manual and usually costly maintenance activity.

Other drawbacks of this approach are:

- Language selectors need to be embedded throughout the user interface and each needs to point to a different place in the parallel structure. This either requires effort, adds to maintenance costs and is a significant quality exposure or it requires scripting or other functional support;
- It becomes very difficult to ensure that equal quality and content is provided for each language. If these applications are allowed to grow unchecked, they quickly atrophy to significantly poorer quality support for one of the languages (usually Welsh, in a bilingual context);
- Any functional changes need to be made to both structures at the same time. The cost of maintenance is therefore effectively doubled;
- Where parallel structures can work in a bilingual setting, they quickly fail in a multilingual setting where the need to maintain additional structures leads to exponential increases in effort and cost;
- Automated content management and integrity checking across the two structures is prohibitively difficult to implement resulting in ongoing maintenance costs to ensure the required level of quality is maintained;
- When more complex functional issues such as bilingual data management and bilingual data interfaces are faced, the parallel structure cannot resolve these without being re-engineered into a single structure (i.e. effectively a level 3 architecture);

- Maintaining state and data context as the language switch occurs is complex.

It is recognised that there are many successful implementations of solutions using this parallel structure and these standards do not legislate against this approach in any way.

However, it is strongly recommended that the limitations and costs associated with this approach are fully analysed and considered before adopting it. Such an analysis will generally find a level 3 or 4 architecture to be more suitable.

5.6.3 Switchable Embedded Content (Level 3)

This approach resolves many of the challenges of a level 2 architecture by having just a single structure. All user interface elements that are language-sensitive are embedded in the user interface with a primitive logic switch controlling which element is displayed.

This approach improves upon parallel mirroring by eliminating several issues:

- Maintaining language selectors, since all selectors simply refresh the current area of the user interface to display the alternate language;
- Development, deployment and maintenance of parallel structures is eliminated;
- Maintaining state and data context during a switch is greatly simplified since it tends to be just a case of self-reference for the user interface object;
- Providing bilingual data management becomes more feasible. However, providing full functional support often results in this structure resembling the central resource (level 4) approach.

Though this approach is far more effective in addressing a number of problems than the parallel mirroring approach, there are still a number of deficiencies when used for large scale, complex or dynamic data content and functionality. These include:

- With the language-elements embedded within the user interface, it is difficult to provide a language management toolset that assists in maintenance of the languages and that can be used to check integrity, quality, consistency and equal treatment of the languages;
- Any oversight in providing support for both languages within one area of the user interface is difficult to trap as an error or exception since there is no 'meta' language management capability;
- It is difficult to identify opportunities to reuse language components across the application with each use of text or a language-sensitive graphic being separately embedded in the user interface;
- The application will face similar challenges as the parallel mirroring approach if it needs to scale up to multilingual support. Each aspect of the user interface will require modification. However, the effort is linear, not exponential as it is for parallel mirroring.

Again, this discussion of architectural approaches is more concerned with an examination of the differing approaches rather than establishing firm standards. From this perspective, this local resource approach can certainly be recommended as a cost-effective approach to meet the requirements of these standards. There will be challenges in reaching full bilingual functionality and quality which will add to

development and maintenance costs. However, for many applications, these challenges and their costs will be containable.

However, it is recommended that a full analysis is undertaken before adopting this approach to ensure that it does provide sufficient capability to meet all requirements.

5.6.4 Resource-Based (Level 4)

This is the most sophisticated approach with language support becoming a set of requirements that can be addressed by software functionality.

It treats all language and culturally sensitive elements of the user interface as data items in themselves. These 'data items' are then stored in a central data store. When an area of the user interface needs to be rendered, the set of data items for that area of the user interface and for the relevant user language are retrieved from the data store.

This approach builds upon the benefits of the Switchable Embedded Content approach and addresses its drawbacks as follows:

- Integrity and consistency checking can be undertaken both on and off line. Offline checks can take the form of analysis reports that will identify any untranslated language-elements. Online checks can treat the inability to retrieve a language-element in the preferred language as an application error;
- If there are problems in retrieving language-elements in the relevant language, dynamic decisions can be made about which language to render to ensure that the user is still able to complete their functional objectives;
- Language-elements (text, graphics, etc) can be shared across the user interface, thereby reducing the translation and language maintenance efforts;
- The data store of language-elements automatically propagates to any future releases of the software;
- Integration with translation support tools becomes achievable, providing online translation (even content management) capabilities, the ability to mark items requiring translation (linked to error logging of unavailable entries) and data imports/exports that link to translation tools and memories (see the IT Strategy document for further information in this area);
- Where a user interface element is language-neutral (e.g. the name of an individual), only a single copy of it is stored, thereby reducing content management overheads;
- It easily scales to a requirement for a multilingual solution. It simply requires the addition of a new language identified in the central data store and the translation of all text – a translation effort rather than a technical effort;
- Turning all user interface language-elements into data items aligns the user interface language management approach with the management of multilingual data (section 8);
- The combination of the integrity and quality checks, translation tools and other language support functionality results in an application for which the level of language support improves over time, avoiding the entropy challenge of the other approaches;

Of all approaches, this is the most recommended for functional software systems and larger scale content based applications that will benefit from the content management approach.

However, the additional initial development cost associated with this architecture should be carefully weighed against its benefits to ensure that the cost is justified. For smaller software applications (such as basic websites), this level of capability could be overkill.

5.7 Translation and Content Support Tools

Section 5.6 (specifically 5.6.4) makes reference to the benefits to be obtained from having translation support capabilities (tools) built into bilingual software applications.

For larger scale and complex applications, the challenge of maintaining consistent text in both languages becomes substantial. This will either result in increasing costs or increasing quality issues when both languages are not supported equally.

The purpose of such tools is:

- To work with functionality handling text absence (see 5.5.1) to flag language items to be reviewed;
- To enable translators, system administrators and developers to manually flag language-elements that require translation or review;
- To provide data import/export capability, direct integration with translation tools and/or a user interface to assist translators with their activity;
- To analyse and assist in the identification, management and consistent translation of duplicate language-elements;
- Change management and version control capabilities for language-elements to assist in the migration, installation and upgrade of the software application.

5.8 Error and Event-Driven Messages

Display of Messages

All messages intended for the user and originating from the application shall be shown in the user's language of choice, regardless of frequency of use.

Care should be taken to ensure that all messages that have the potential of being displayed to the user are given the same treatment as all other user interface language-elements.

To simplify this requirement, it is recommended that all application errors are 'mapped' onto a limited number of user friendly messages. This not only serves to insulate the end user from confusing technical messages, but also reduces the amount of translation required and the potential for untranslated messages to get through the quality checks.

Some messages will originate from the operating platform and where these are beyond the control of the software application, there is little that can be done to control the language of these messages. However, a software application should make all possible efforts to trap these areas and replace them with user friendly messages in the correct language for the user.

When displaying error and event messages to a user, consideration should be given to the languages understood and used by the individuals providing technical support for

users of the software. Where there is likelihood that these will be different to the language of the user, the following approaches can help to mitigate the impact:

- Inclusion of language-neutral information in the message that the user can relay to the technical support provider to assist in the identification of the problem. For example, an event number or identifier;
- Either logging the problem in a data store accessible by the technical support provider or providing the user with the capability of 'dumping' information regarding the event (and any other contextual information) so that it can be forwarded to the provider of technical support;
- The capability for the provider of technical support to remotely connect to the software application and, specifically, the users session so that they can diagnose the users problem in their own language, independent of the users language.

5.9 Integration with Language Support Tools

Equivalency of Language Tool Support

Where a software application is integrated with an external tool or resource that provides language support, this shall only be done when an equivalent tool can be provided for each supported language.

For example, if an English spellchecker is integrated, then a Welsh language spellchecker should also be included to ensure both languages benefit from the same level of functional support.

6 User Interface

A broad interpretation will be taken of user interfaces, being all means through which a user communicates with a software application. Where a particular type of interface is inferred, the concept being discussed should be taken as being applicable to all types of user interface.

6.1 Language

Equivalent Functionality

On no account shall functionality be diminished for a user as a result of their language preference. The same functionality and behaviour shall be present for both languages. The Welsh and English versions of the user interface shall convey an equal level of information and detail at all times.

6.1.1 Quality of Language

An equal quality of language shall be present in both languages. The equality can be measured through grammar, spelling and comprehensibility. Section 3.4 provided a general discussion of language quality and these guidelines should be adopted. Where the developer of a system isn't fully competent in a language or qualified to provide the content in that language, then the effort to translate and produce content in that language should be outsourced to a suitable and qualified translator.

Language used should be based upon standard dictionaries and terminology resources and should be clear, readable and aimed at a level of comprehension suitable for the intended user. Useful references include:

- Cymraeg Clir (Canolfan Bedwyr)
- Crystal Mark (Plain English Campaign)

Welsh language resources to assist in meeting these standards can be found in the resources section at the end of this document (section 13.3).

6.1.2 Abbreviations

When abbreviating a word or phrase, care should be taken to ensure that the result is meaningful, accurate and consistent with existing conventions. Some specific considerations are:

- When creating abbreviations, awareness of digraph letters is essential and an abbreviation should include the full digraph, not just the first character;
- Abbreviations are not always formed from the first letter of a word or first word of a phrase.

Section 4.1.3.1 provides an example of both of these points.

Therefore, it is recommended that abbreviations and acronyms are only produced by skilled linguists or translators and through reference to existing terminology repositories and guides.

6.1.3 Policies, Disclaimers and other Legal Text

Equality of language and the requirement to translate all language-elements in a user interface extends to all text, including text that isn't a functional part of the software application. This includes legal disclaimers, terms and conditions, license agreements, conditions of use, etc.

In addition, it is suggested that organisations include a reference or link to their published policy on bilingualism and their Welsh language scheme where applicable.

6.1.4 Simultaneous Deployment

When a software application is deployed, upgraded or when fixed content is modified, care should be taken to ensure that both languages are considered during the deployment activity to ensure that both are equally available at the same time.

The inability to deploy both languages simultaneously should be treated at the same severity level as the inability to deploy any other key functionality or content and any decision to proceed regardless should be documented, approved and subject to future audit.

6.1.5 Monolingual Content

There are some cases due to policy decisions or justifiable necessity where content or functionality might only be available in a single language on either a temporary or permanent basis.

Where this occurs, a clear statement should be present to state this in order that the users of the language not supported understand the reasons for this. This statement should be presented in the language not being displayed to ensure that it is understandable.

6.2 Design & Layout

There are a number of important aspects to the layout and design of the user interface that should be adhered to in order to ensure equal treatment of the languages supported by a software application.

6.2.1 Parity of Design Quality

The quality of the user interface design shall have parity between Welsh and English. Graphics, layout and text arrangements should be of equal quality.

6.2.2 Mixed Language Interface

Mixed Language Interfaces

Where an area of the user interface presents both languages simultaneously, there shall be equal prominence given to both languages. If it isn't possible to provide equal prominence, favour shall be given to the users preferred language.

'Prominence' refers to position, use of fonts, colour, sufficient space, etc.

If both languages appear together, care should be taken to separate the languages such that screen reading utilities (i.e. text to speech) are able to discriminate between the languages.

It is the difficulty in meeting these standards and treating both languages equally when they are placed together that leads to the recommendation that separate interfaces are available with a language selector continually available.

6.2.3 Anchor Components

A user interface will have key areas and components that act as anchors, to provide a user with a reference to the whole interface and from which users will create an intuitive understanding of the functionality of a software application. These include main menus, function bars, language selectors, control boxes, data entry fields, etc. Care should be taken to ensure that these are similarly placed for both languages.

This will not only act to provide equal treatment for both languages, but will also assist learners and those less confident in their language skills to use an application in a language other than their first language.

6.2.4 Prominence of Information with a Language Context

Information that is sensitive to the language of the user should be placed so that prominence, focus and emphasis is either language-neutral or priority is given to information relevant to the preferred language of the user.

6.2.5 Grammar Neutrality

The layout of the user interface should be done with care to ensure that the placement of related components doesn't have a dependence upon the grammatical structure of the language.

This is particularly relevant where functionality allows for the user to construct a phrase from separate user interface components, such as the ordinal issue described in 4.1.3.2.

There are two aspects to be considered here:

- When switching languages, the ordering and placement of user interface components might need to change in order to satisfy the grammatical requirements of a language. This conflicts with the guidelines of 6.2.3 and can lead to a degree of confusion when using the application;
- The order of words sometimes change based upon the context of the words in the phrase. The ordinal example in 4.1.3.2 is a good example of this.

Care should therefore be taken to structure the layout of a user interface such that it doesn't change when a language switch occurs and also that the rules of the grammar are not broken as data selections change. If such a conflict unavoidably occurs then a different design should be used that satisfies both requirements.

6.2.6 Labels & Placeholders

When designing a user interface layout, care should be taken to ensure that all areas of the interface with language-elements can be resized to accommodate differing lengths of text as the user interface is translated.

This is particularly relevant to labels and other language-elements embedded within a user interface. Though larger areas of text will often result in roughly equivalent character counts, single words can vary in length by 200-300% or more and therefore are subject to substantial impact.

Sufficient Space for Text

The layout of user interfaces shall be designed to ensure that there is sufficient space allowed for the display of text in either language without truncation or overlaying occurring.

This should not only be a design consideration, but also a key aspect of the testing process. Once all of the text in an application has been translated and before release, a test activity should take place to verify not only linguistic quality, accuracy and consistency, but also to identify any layout problems.

A recommended approach (though only possible with level 4 architectures – see 5.6) is to store the size of each element of the user interface in a data store. A report can then easily compare the size of each element against the space required for each language and highlight potential problems.

6.2.7 Images, Graphics and Icons

Images should be culturally neutral wherever possible.

The use of flags to denote languages is an example of the inappropriate use of images, particularly where a country is bilingual or a language is spoken in more than one country. For instance, an English (St. George) flag doesn't correlate to the English language for English speaking Americans, Australians, Welsh, etc. Therefore, it is strongly recommended that national flags are not used as a reference for language choice.

Care should be taken not to use images that hinge on idioms in either language. Idioms rarely translate exactly, if at all.

Further discussion on use of images, particularly the use of text within images occurs in section 6.4.

6.2.8 Other User Interfaces

As the support for the Welsh language in other (i.e. non-graphical) user interface technologies increases, this area of the standards will extend to include these technologies. In the absence of specific standards, software in these areas should make the best efforts to make a language switch as intuitive and accessible as possible for a user without the loss of state or data context.

6.3 Accessibility

There are already standards and best practices in place for accessibility. This section is intended to complement and not replace these existing standards and practices.

In fulfilling the standards in this document, it is important for a software application to also meet any relevant and available accessibility standards for both languages. Further, meeting these bilingual standards should result in a more accessible and not a less accessible system.

For instance, whilst meeting the requirement of a suitable font that fully supports the diacritic marks found in Welsh and the other requirements of section 4.3.5, it is essential that this font also meets the requirement of any accessibility standards.

Accessibility

Where accessibility support and functionality is included in an application, an equal level of support shall be provided for each language. For example, alt tags in HTML, Tool tips for desktop applications, etc.

Further, when a mixed or parallel text layout is used for graphical user interfaces, the two languages should be clearly distinguishable (or separated) to facilitate the use of 'screen-reading' and text-to-speech utilities.

As accessibility standards evolve and as new technologies bring solutions to address accessibility issues, all relevant bilingual standards shall remain applicable to ensure equal treatment of both languages.

6.4 Graphical User Interface (GUI)

Throughout the rest of this document, care has been taken to express standards and guidelines such that they can be applied to as broad a variety of architectures, technologies and user interface modes as possible.

However, there remain some standards that are specific to a graphical user interface. These are:

- Any logo or branding should be appropriate for the context of the language currently being displayed. If the brand is inherently bilingual or language-neutral then it can be used for both languages. If the brand is language or culturally specific then two versions should be available with the appropriate version displayed depending on the language;
- Where text occurs within graphics (i.e. for buttons, links or just as part of the overall design), these graphics should either be bilingual, or a copy of each graphic should be available, for each language with the relevant one displayed depending upon the user interface language. Ideally, the use of text embedded in graphics should be avoided to both simplify this issue and also improve the accessibility of the application;
- If bilingual text is used within a graphic, care should be taken to ensure that the text for the alternate language (i.e. the language other than that being displayed) is appropriate and inoffensive (politically, culturally and linguistically);
- Where links are provided to other applications, websites and other resources, the link should be relevant to the current language. For instance a link in a Welsh language web page should lead to the Welsh version of a web site whilst the same link in the English version of the page should lead to the English language version of the web site.



Note: the use of text within graphics should be subject to careful consideration. Not only does it complicate a bilingual system, it can also create accessibility problems.

6.5 User Assistance

User assistance is the general term for help, user assistance and user support for a software application. This includes help files, tool tips, automated assistants, reference documentation, FAQs, etc.

User Assistance

Where user assistance is delivered through the same user interface as the software application, then it shall be supported by the same bilingual capabilities such as ever present language selectors, bilingual searching, etc.

When translating user assistance files, there are several areas where inconsistencies can occur and that merit special attention:

- As software changes and upgrades and fixes take place, it is important to ensure that user assistance is updated to reflect these changes. These updates should occur for both languages to ensure consistency and equality;
- The translation of terminology in user assistance files should be done with close reference to the terminology of the software application being described to ensure that consistency is maintained. It is easy to translate the same term in slightly different ways for the user interface and for user assistance leading to user confusion;
- Where language-sensitive information (screenshots, error messages, commands, etc) are used as examples, these should be checked to ensure that the language used in these examples is consistent with the user assistance language.

It is often helpful to include a bilingual glossary with help documentation to assist users in understanding new terminology and for those users less confident in their language skills.

6.6 Installers

Software installers are themselves software applications and should therefore comply with all relevant standards. These include:

- Providing the user with the option of selecting their preferred language;
- Using a resource file for language-sensitive user interface elements (text, graphics, etc);
- Using terminology that is consistent with public standardised resources;
- Legal agreements, disclaimers, terms and conditions and license agreements should be available in the users preferred language;
- The language selected by the user performing the installation should not be assumed to be the preferred language for the end user.

7 Outputs

Outputs from a software application are those objects containing information that are produced by the application and which are intended for use independently from the application itself.

In many ways these are just another aspect of the user interface. However, since the full language management functionality of the application may not be available or not relevant, there are specific considerations for bilingual support.

There are a number of types of such objects. This section will discuss some general issues but will focus on two common types: reports and emails.

7.1 General Guidance

System outputs are slightly different to other types of user interface for a number of reasons:

- They tend not to contain as much contextual help as other user interface modes since they are usually non-functional and text-intensive;
- They tend to contain longer passages of text and more formal text than a graphical user interface;
- It is more difficult to make bilingual emails and reports visually appealing and to discriminate between two languages when they are in a mixed format (i.e. language 1 / language 2);
- Recipients of emails and reports can easily forward them to others who might have different language preferences and abilities;
- Given the lack of contextual clues and the more formal text, it is essential that a user is fully capable in the language used. This is particularly important since outputs often contain vital and essential information. It is important to realise that a user's preferred user interface language isn't always their most proficient language.

7.2 Monolingual or Bilingual

The best approach is to always produce bilingual output (structures and options for doing so are discussed in 7.3). Though there might be cases when a single language is more appropriate, the justifications for this should be carefully evaluated to ensure that there is no risk of limiting the comprehension of the recipient and that such a decision still means that both languages are being treated equally.

Some cases where a single language output is appropriate are:

- When it is known that there is a single recipient, the language of that recipient is definitely known and the information is such that the recipient is unlikely to forward or distribute the information further. An example is a password email;
- Where there is a closed list of recipients (or any recipient will, by definition, be competent in the language), it is known that all recipients are sufficiently proficient in the language and it isn't possible for the information to be forwarded to anyone insufficiently competent;
- Personal communications where the language used is at the discretion of the individuals involved;

- Screen prints and other outputs requested by a specific user and that user has the ability (and hence responsibility where appropriate) to obtain the equivalent outputs in the alternate language. In this case, the option for monolingual or bilingual output might be considered as a value-add item of functionality to aid the user.

Cases where bilingual outputs are appropriate include:

- Where an email is sent to a group or a report is produced where the language preference of every recipient isn't absolutely known, where it is known that the recipient group is mixed-language or where the object can be forwarded to an individual of unknown language preference and competence;
- When an email is sent to an open group/list or a report is produced for public usage.

When producing a bilingual output, it will usually be necessary to identify which language has prominence (i.e. appears first, on the left, etc). This judgement might be based upon the nature of the software application, any known demographics of the recipient group and/or the subject matter of the content.

7.3 Layout and Format

The layout and format of bilingual emails and reports should adhere to all relevant standards for user interfaces. Given the amount of textual information that is usually involved, the layout should be chosen with care to ensure maximum legibility whilst ensuring that both languages are given equal treatment.

Further guidelines can be found in the Welsh Language Board document 'A Guide to Bilingual Design'.

7.3.1 Mixed/Parallel Text

This approach involves the placement of both languages together by placing them in parallel or by mixing words, phrases or paragraphs of text.

Though straightforward, care should be taken to ensure the result is legible and readable and is useable by text-to-speech utilities.

The election to use this approach instead of those following is at the discretion of the designer who should follow existing guidelines.

7.3.2 Sequential Text

Sequential text is particularly relevant to text emails and other media where there is minimal control over format. It simply involves two passages of text, one in each language. The languages are arranged sequentially with all content text for one language ending before the other commences.

Sequential Text

Where sequential text is displayed and the application is aware of the users preferred language, this shall be displayed first with the alternate language second. There shall be a clear separation between the two languages.

Where the amount of text is more than a single paragraph, it is likely that the text for the alternate language will not be visible when the email or report is first opened or previewed. In this situation a single line should appear at the top of the output, in the alternate language, to advise the user that text in this language follows the text in the primary language.

7.3.3 Alternating Pages

This format is well suited to reports and documents where there is good control over formatting and page breaks.

It is particularly suited to double sided printing where each language can appear on either side of a sheet. Examples of this are standard letters, statements, invoices and bills.

It is also well suited to booklet-type approaches where each language can appear on opposite pages.

It simply involves producing a single page of information in one language and then repeating the same information in the alternate language on the next page. Care should be taken to account for differing lengths of textual information and to ensure that page breaks occur at the same point in the information for each language. Also, page numbers should only be incremented every alternate page.

7.3.4 Separate Objects

This approach results in two separate objects being produced containing the same data and text for each language.

This is ideal for objects that will be distributed to mixed language groups of users and allows each recipient to use the format that best suits their language preference and ability.

In many cases, this is also a simpler approach since it allows for two separate monolingual outputs and eliminates the design considerations and limitations of addressing the needs of both languages simultaneously. However, care should be taken to ensure that each object has complete language integrity with Welsh files having a Welsh language file name, title, etc.

Examples of situations where this is an ideal approach include:

- The production of data objects (documents, spreadsheets) onto a bilingual website where the relevant object can be placed on the appropriate language page;
- Sending emails to a list of recipients where the correct language can be sent to those for whom the language preference is known and two emails to those for whom it is uncertain;
- Producing objects containing meta-data where the language used to describe the object can be in the same language as the object (also see 10.2).

7.4 Mail Merge

Mail Merge

A mail merge shall not be performed in a way that produces poor quality text in either language.

Where the data merged into the template is language-sensitive, it shall be in the same language as the surrounding text in the template (i.e. English words not embedded in the Welsh text in the template and vice versa).

The structure of the resulting output shall comply with all other standards in this section. Further, the template used should be sufficiently flexible to support differing sentence structures.

Where the preferred language of the recipient is known, prominence shall be given to that language.

7.5 Character Set

As for the discussion in section 4.3, the character set used for outputs shall be one that supports the full Welsh alphabet, including all diacritics.

This requirement is particularly appropriate for applications that are responsible for the transmission, forwarding, storage and display of email messages and data objects.

As a consequence of many email systems not currently being fully compliant with this standard, it is advisable to restrict the use of diacritic characters in emails to just those in the ASCII character set (i.e. to limit use of diacritics, particularly for the 'w' and 'y' characters). If these characters are used, there is a risk that they will not only be corrupted themselves, but some email systems will also corrupt the entire message.

Where it is known and proven that a recipient can receive these characters without any corruption occurring, their use is encouraged. Where this isn't known then either an equivalent representation should be used (e.g. w[^], y[^] - see 4.3.4.2) or the text should be moved to a file attachment where the integrity of the character set can be ensured.

7.6 Reports

The term 'Reports' is a general name given to all static objects produced by a software application that contain information laid out in a user-friendly format.

These can include:

- Analysis and management reports containing data, graphs and statistics;
- A 'print-friendly' version of information displayed upon a screen;
- Information regarding a transaction (such as an invoice) that is produced in document form;
- Standard letters produced by a software application;
- Statements, bills, invoices sent to customers;
- Etc.

The common feature of reports is that they contain information that is presented in a human-readable format that is intended for use outside of the scope of the application itself.



The standards and guidelines relevant to reports have been fully covered elsewhere in this section and document.

7.7 Emails

These standards and guidelines apply to emails that are generated by a software application and automatically sent to either a closed or open community of users. They are not intended to apply to emails sent between individual users or personal emails, though some of the discussion might prove useful as guidance in these situations.

However, these standards do apply to software applications that transmit, forward, store or display email messages.

7.7.1 Subject Line

Email Subject Line

The subject line for a bilingual email shall be bilingual and represented as either English/Welsh or Welsh/English. The language order will be the same as the primary/alternate languages selected for the email body.

For monolingual emails, the subject line shall be consistent with the language of the email.

7.7.2 Subject Line Modifiers

Email Subject Line Modifiers

Where emails are replied to or forwarded, it is typical (in an English context) to modify the subject line with 'Re:' or 'Fw:' text. Where the user initiating the reply or forward has expressed a preference for the Welsh language, the Welsh abbreviations 'Atb:' and 'Yml:' shall be used instead. If the email content is bilingual, then both abbreviations shall be used, i.e. Re/Atb: or Atb/Re:.

7.7.3 Disclaimers and other text

Disclaimers

Where email systems automatically append text to emails such as signatures or disclaimers, this text shall be bilingual.

This is particularly important for disclaimers where the recipient cannot be relied upon to have understood the disclaimer unless they are addressed in their preferred language.

7.7.4 Attachments

Email Attachments

Where attachments are included with emails, they shall either:

- Be language-neutral, in which case no standards apply;
- Contain bilingual content that must meet the standards in this document;
- Be replicated with one attachment per language in the case where the

attachment is monolingual. In this case, there must be equal content and quality of language as required by these standards.

7.7.5 Email Address Domain Names

Where domain names are available in both languages, individual users email addresses should be configured to receive emails on both domain names (e.g. joe.bloggs@bwrdd-yr-iaith.org.uk and joe.bloggs@welsh-language-board.org.uk).

Functional email addresses (e.g. 'info', 'support', 'enquiries', etc) shall be set up in the language consistent with the domain name.

Where a single or language-neutral domain name is used, functional names should be set up in each language (e.g. contact@draig.co.uk and cyswllt@draig.co.uk).

7.7.6 Reply Name and Address

Since it is only possible to have a single 'reply to' address for an email, the following rules shall apply:

For automated emails, where the reply name is a functional description (i.e. 'support', 'enquiries', etc) as opposed to an individual's name:

- Where the preferred language of the user is known, the reply to address shall have the name (text before the '@') in the same language as the user. Where a domain name exists for each language, the relevant domain name shall also be used;
- For group emails or where the preferred language isn't known, the language should be consistent with the primary language selected for the rest of the email.

For automated emails where the reply name is that of an individual and where a domain name exists for each language, the domain name relevant to the preferred language of the user shall be used.

For individual emails, no standards shall apply and the user shall be enabled to nominate their preferred domain name subject to its availability and being configured as described in section 7.7.5.

8 Data Management

The true value of a software application is its ability to store and process data and present this to users. The challenge is how the language and cultural specific aspects of this data should be managed to fulfil the objective of allowing a user to interact with the system in their preferred language.

Data will usually originate from a source outside of the scope of control of the software application, and potentially from outside the scope of control or influence of the organisation that owns the application. In defining the standards for a software solution, we therefore need to accept and accommodate this reality.

The standards in this section cover the ability to enter, process, store and present bilingual data. There will also be some clarification and discussion of how to handle third party data that isn't provided bilingually (8.4). Also provided are guidelines and recommendations on methods that can improve the quality and efficiency of the management of bilingual data.

8.1 General Guidance

In order to allow users to communicate with a software application using their preferred language, a bilingual software application must:

- Ensure that software can fully support and manage bilingual content;
- Treat both languages equally, both in data management and data entry/provision requirements for users;
- Provide functionality to encourage and facilitate the provider of the data to provide it in a bilingual manner (and enforce this where relevant);
- Help in maintaining consistency between data held in both languages;
- Allow for integrity reporting and statistical analysis of the availability and consistency of bilingual data;
- Ensure that the user has equal access to data if it is only available in a single language, even if that isn't the preferred language of the user;
- Provide an explanation to the user when data isn't available in their preferred language;
- Make sure that system functionality and capability isn't compromised by this additional functional requirement of bilingual management – to turn it into a functional asset rather than a liability – and to ensure that the level of service and access to functionality by the user isn't impaired;

There are three further issues to be aware of relating to the entry of data by a user:

1. Firstly, an individual user is entitled to communicate in their language of choice and therefore, in many situations the entry of data in both languages cannot be a mandatory requirement;
2. Secondly, to ensure equal treatment of the languages it isn't acceptable to assume or require a user of one of the languages to enter data in the other language if this isn't done for all languages. i.e. if it isn't obligatory for an English language user to enter Welsh data, then it cannot be obligatory for a Welsh language user to enter English data;
3. No assumption should be made about which is the preferred or mandatory language unless the user's language preference is already known.

8.2 Identifying Bilingual Data

Before we progress with this section, it is worth defining what we mean by 'bilingual data'. The terms 'language-sensitive' and 'language-neutral' were defined in section 1.6 and can be applied to data. Therefore, bilingual data is defined as any item of data that contains language-sensitive information, irrespective of its form.

The most obvious form of this data is text. However, it is equally important that we consider other forms of data such as graphics, photographs, video clips, sound files, etc (see 8.3 for guidance on managing this type of data).

When dealing with textual data, there are some cases where the data is clearly language dependent (i.e. free text) and cases where the data is language dependent but well structured (e.g. county names, salutations, etc).

However, there are other cases where the language dependency isn't so obvious and care should be taken to identify where these might exist and to implement the capability to manage these data items bilingually. Examples of these types of data include text embedded in graphics, Internet domain names and email addresses amongst others.

8.3 Managing Bilingual Data

Depending on the nature and functionality of the software application, data might be in a format that is meaningful to the application or might be contained in objects and files that the application manages without knowledge of their content.

Where the application has the capability to discern the content, then these standards define how this shall be undertaken in a bilingual manner. Where this isn't the case, then the application should have the ability to manage two copies of the object and identify each copy with the relevant language (or as a shared resource where the object content is language-neutral).

Managing Data Objects and Files

When managing data objects or files that potentially contain language-sensitive data, a software application must have the capability to manage one copy for each language. When an object or file is identified as being language-neutral, the application should be able to identify it as being shared.

8.4 Handling Third Party Data

It has been stated that a bilingual software application must allow for data to be entered, processed, stored and presented bilingually and with equal regard for each language.

However, this doesn't mean that the owner of the software application is responsible for ensuring that data is provided to the user in their preferred language when this data has only been provided to the system in a single language.

Provision of Bilingual Data

It is always the obligation of the data provider to provide data that is, at a minimum,

consistent with their language schemes, statutory obligations and commercial policies.

The role of the software application is to support, facilitate and encourage the entry of bilingual data, be capable of storing and processing this data without preference to either language and make the best efforts to deliver data in the preferred language of the user, providing advice where the data isn't available.

Where a user or third party provider of data supplies data in a single language, it is recommended that an explanation for this is provided by the third party so that this can be passed on to the user as a disclaimer if the data isn't presented in the preferred language for the user.

8.5 Enumerated Data Items

Enumerated items simply substitute a number or other unique identifier for an item of text or other language-sensitive data item. For example, a list of counties can be represented internally by numbers:

Identifier	Language	County
1	EN	Anglesey
1	CY	Ynys Môn
2	EN	Gwynedd
2	CY	Gwynedd
3	EN	Flintshire
3	CY	Sir y Fflint
4	EN	Cardiff
4	CY	Caerdydd
...

Wherever possible, it is recommended that this practice is adopted since it provides a number of substantial benefits to bilingual software applications (indeed, to the management of data in general), including:

- The simplification of storage of bilingual data items, requiring just the identifier to be stored;
- The ability to display the equivalent text (or other data) for each data item in the preferred language for the user;
- Reduction of the need for bilingual data entry since the user just selects the relevant enumerated item(s) displayed in their own preferred language;
- Ensured, automated and guaranteed quality of translation between languages;
- Enhanced abilities to perform statistical analysis;
- Improved opportunities to reuse standard interface components and resources across the application;
- Predictable operation and ensured data validation;
- Delegation of language management to the presentation layer, enabling the application layer to operate in a language-neutral manner.

Additionally, the use of enumerated data items enables the re-use of existing translations and standardised terminologies. To encourage and support this activity, the Welsh Language Board have published a number of standardised terminology lists on the publications list of their website.

8.6 **Mandatory, Optional or Inappropriate**

There will be circumstances where the nature of the functionality or the user community makes bilingual data entry a mandatory requirement. When this is the case, the absence of data in one of the languages should be treated in the same manner as the absence of any other mandatory data item.

Alternatively, there are cases where the nature of the functionality means that data will either never be displayed to other users or where the language used is irrelevant to the application and entirely at the discretion of the user (i.e. personal emails, personal data, etc). In these cases, the requirement for bilingual data entry is mitigated, but other aspects such as data storage remain equally important.

The general case requires bilingual data entry and display to occur wherever possible and to be supported as outlined in 8.4 and throughout the rest of this section.

The approach taken for each application, for each data set and for each user community should be determined and agreed during the requirements definition process.

8.7 **Data Capture**

Data Capture

Bilingual software must fully support and facilitate the entry of data from users and collection of data from other systems and sources in both English and Welsh.

The data held, managed and processed by a software application is usually derived from inputs external to the application. Where this input data is language-sensitive, the application must provide the ability to receive the data in both languages without showing a preference for either language.

8.7.1 **Automated Data Entry/Procurement**

Data submitted or collected electronically should adhere to all relevant standards for the data interface. If these standards do not provide for bilingual data then an agreed and documented deviation from that standard should be adopted to allow for the submission of bilingual data. This is covered in more detail in section 8.10 where data transmission and data interfaces are discussed.

Where the e-Government standards are relevant, the means of handling bilingual data is discussed in section 10.

Automated Data Procurement

Where the interface to the data provider allows for the selection of data in multiple languages and the data is available in both English and Welsh, then the data shall be obtained in both languages.

8.7.2 Manual Data Entry

Manual Data Entry

Where a manual user interface is used to enter data, this interface should allow for the entry of all language-sensitive items in both languages.

(Before discussing this standard, a reference to section 8.6 is essential as it isn't always appropriate to require the entry of bilingual data. Where this is the case, this standard and the following discussion isn't relevant.)

To achieve this, data entry fields can either be placed in parallel or sequentially on the same area of the interface or they can be arranged in different places, but linked to the same interface state (e.g. use of tabs on a graphical interface to provide a data entry page for each language).

Placing the data entry fields for each language on the same page is more straightforward for the user, making the requirement to enter both languages more explicit. However, this approach is less scalable if the language support for the application needs to be extended to a multilingual situation.

Where the data entry fields for each language are placed separately, it is important that they occur within the same user 'state' where the user should be expected to enter data for both languages before progressing to the next step in their interaction with the application. When the user does make this progression, if data is saved, then data for both languages should be saved, irrespective of the current language page being displayed.

Both of these approaches result in one language being given prominence over the other, either through data entry fields being placed on the left of or above the other language (when on the same page) or being on the first page displayed to the user (tabbed and more scalable approaches). The more prominent language should be the preferred language for the user.

8.7.3 Encouraging Bilingual Data Entry

Encouraging Bilingual Data

Bilingual software applications shall employ whatever mechanisms are possible to ensure equal treatment of all languages and encourage entry of data for all supported languages.

Since data isn't always available in both languages, it is important for a software application to allow users to enter data in a single language. However, there is the potential for users to neglect to enter data in the alternate language, even when it is available and the software application should discourage this.

Some examples of how to do achieve this follow.

8.7.3.1 Use of indicators

Bilingual Data Entry Indicators

Where the layout of the interface doesn't make it completely clear and evident that there is the capability to enter both languages, indicators shall be located alongside those fields that require bilingual entry.

Ideally, this indicator can also provide a link to where the equivalent data entry field in the alternate language is located.

8.7.3.2 Reminder Messages

Encouragement can be provided through a reminder message when the user first attempts to submit the data, drawing attention to the ability to enter the alternate language. This message should be placed so that the user can easily access the data entry fields not completed in order to rectify the situation.

This technique can also be employed when data is updated by detecting an attempt to update data in one language without updating the field in the other language. When this occurs, a similar message can be displayed with a slightly different emphasis recognising that the change might have only been relevant to one of the languages. However, if the change resulted in the data for one of the languages being removed, then it should be handled in the same manner as if the data had only been provided in one language initially.

8.7.3.3 Explanations/Disclaimers

Where data is provided in only one language, it is good practice to ask the user to provide an explanation for this. For instance, it might be that the data is only available in one language, that the viewer of the data should make manual contact to obtain the translation, that the data is currently awaiting translation, etc.

This approach provides two benefits:

- Firstly, it provides the software application with an explanation that can be used when displaying the data as to why it is only available in one language;
- Secondly, it reminds, encourages and reinforces the desired outcome of bilingual data entry by making the entry of data in a single language less efficient.

An alternate approach that makes the data entry activity more efficient but does lower the prominence of the alternate language is to build a generic 'disclaimer' into the Terms & Conditions that a user agrees to when using the software application.

8.7.3.4 Consistency of Mandatory Data

When data is entered into a software application, it is typical to define mandatory and optional fields. In a bilingual application this becomes more difficult to manage since users are generally able to enter data in either language.

Consistent Set of Mandatory Data

It is conceivable that a user will enter data into some mandatory fields in one language and other mandatory fields in the alternate language. This situation should be identified and handled by requiring that the full set of mandatory fields is entered in at least one language.

8.7.4 Providing Data Objects

Section 8.3 discussed data objects that contain language-sensitive information where the application manages the object itself and not the data contained within it.

Where functionality allows for the upload and storage of such an object and bilingual data entry is required (see 8.6), the ability to provide one copy for each supported language is important.

Where a copy of an object isn't provided for a language, the data provider should either be able to indicate that an object is either language-neutral or identify which language is covered. Where a required language isn't covered, the user should be asked to provide an explanation which can be used as a disclaimer in the same manner as for textual data (see 8.7.3.3).

8.8 Data Storage

Having supported the entry of bilingual data, it is important to ensure that the data is stored correctly so as not to compromise the integrity of the data and to ensure that it can be accessed effectively.

8.8.1 Character Sets

Storage Platform Character Set

The storage platform shall support the full Welsh and English alphabets, including all diacritics.

A full discussion of character sets is provided in section 4.3.

8.8.2 Sort Orders

The sorting of bilingual information is covered in section 4.2. This is generally applicable to the management and display of the user interface.

However, there are many situations where the sort order of text data is relevant within the application and particularly at the data storage layer.

Most database packages allow for the selection of a pre-defined sort order or collation. Where it is possible to use such functionality to return results sorted according to the English or Welsh alphabet, that functionality should be used.

These standards are not complete in this respect and notes for further consultation occur in section 11.1.

8.9 Data Display

With the availability of bilingual data, it should be possible to present a user interface that is completely in the preferred language of the user. In achieving this, the relevant standards apply to the presentation of the data (e.g. sorting, bilingual data objects, switching language at the request of the user, etc.).

Where the data isn't language-sensitive or bilingual data management isn't appropriate (see 8.6), the presentation of the data to the user is very much the same as for a single language interface.

The challenge arises when data isn't available for display in the preferred language of the user but is available in the alternate language. In this situation, the following standard shall apply.

Unavailable Data

When data isn't available for display in the preferred language of the user but is available in the alternate language, a bilingual software application shall:

- Display the data for the alternate language. On no account shall data be withheld from a user because it isn't available in their preferred language;
- Indicate to the user that the situation is acknowledged and display any disclaimer or explanation supplied by the data provider;
- Log the occurrence to enable future analysis to take place;
- Where appropriate, log the occurrence with the same severity and priority as other non-fatal application errors.

These standards require two forms of logging to take place. The first is simply to allow for analysis to take place in the future either on a system wide basis or specific to a particular data provider.

The second form of logging applies to the situation where it has been defined as a functional requirement of the software application that the data is available in the preferred language of the user and this isn't available. In such a situation, the occurrence shall be treated as a defect and should be handled and logged in the same manner, at the same priority and with the same severity as any other non-fatal functional error.

8.10 Data Transmission and Interfaces

Provision of Data

Where the application provides a data interface to other systems and third parties, this interface shall:

- Use a character set that supports all characters for each language and ensure that no characters in the data are removed or corrupted;
- Be capable of providing data in either and/or both languages according to the format and protocol agreed;
- Be compliant with the bilingual requirements of all applicable standards and conventions.

This standard applies to all applications that manage, process or handle bilingual data. For instance, email, network applications and other data transmission systems are particularly subject to the need to fully support the character sets used.

This issue is covered further in section 10 where e-Government standards and guidelines are discussed.

8.11 Meta-Data

Meta-data (or meta-information) describes the content and purpose of other objects. It can be used in a number of manners; common applications are to aid searching, indexing and cross-referencing. Some examples of meta-data are:

- HTML meta-tags used by search applications and engines;
- Document properties (title, author, etc);
- Image descriptions to aid indexing and searching (e.g. for clipart);
- Library information used to categorise, organise, and index objects (e.g. MP3 files, documents, etc);
- e-Government category lists (see 10.2).

However, meta-data can be as diverse and as prolific as the data it describes. More importantly, it often has a more substantial impact on the operation of a software application and the ability for an application to operate in a bilingual manner.

8.11.1 Encoded Meta-Data

As for the data itself (see 8.5), when meta-data is encoded or enumerated in a language-neutral format, it is far easier to manage in a bilingual situation.

When unencoded meta-data (i.e. free text descriptions) is used, it is often necessary to include it in both languages. Using encoded meta-data eliminates this need and also simplifies search and maintenance functionality.

An example of encoded meta-data is the Integrated Public Sector Vocabulary (IPSV) which is further described in 10.2.

8.11.2 Creating Meta-Data

Creating Meta-Data

Whenever meta-data is used to describe an object which is bilingual, language-neutral or only available in a single language, it shall be created in both languages. Where the provision of meta-data in both languages isn't supported by the standards defined for the object type being described or the software interface used to enter the meta-data, mixed text shall be used.

Where the object being described is language-specific and versions exist for each language, then the meta-data shall be in the same language as the object which it is describing.

Where multiple items of meta-data are used and the sequential order of this information is relevant, care shall be taken to ensure equal precedence for both languages.

This is an essential standard, since many existing forms of meta-data are relatively unstructured and equivalence in how both languages are used is easy to attain.

For instance, when using HTML meta-tags in web pages, the HTML standards allow for any text to be used within the tags, thereby allowing equivalent tags to be used in each language. Further, since the order of HTML tags is important, with many search engines only using the first group of tags, it is important that Welsh and English tags are alternated to ensure that each language has equal placement in search engines.

Web sites that have static content with separate pages for Welsh and English are a good example of the case where single language meta-tags can be used since corresponding tags can be used in the corresponding pages. This is only appropriate where there is a corresponding page/object in the alternate language. When just a single page exists, then the bilingual approach should be used, irrespective of the language for the page.

Identifying Language

When defining metadata and how metadata fields are to be used and interpreted, a field should be assigned that will describe the language(s) of the object being described.

A further discussion of meta-data in HTML and XML can be found in the e-Government section on e-GMS (10.1).

8.11.3 Managing Meta-Data

The structure of meta-data varies across the many purposes it fulfils and also in how its definition is controlled. Some forms are controlled by universally agreed standards whilst others are defined for a specific software application and are not used outside of that application.

Managing Meta Data

Where a software application has the ability to define its own meta-data format, that format shall support both languages by providing separate data fields (or meta-data records) for each language.

Where an organisation is involved in the definition of meta-data standards or the agreement of a meta-data interface, that standard shall provide multilingual support through separate data fields (or meta-data records) for each language.

Where a software application provides an interface to maintain or edit meta-data, it shall support and exploit any bilingual or multilingual capabilities of the meta-data format. Where these capabilities do not exist, it shall make best efforts to emulate these through mixed text and insulate the user from this compromise by presenting a bilingual user interface.

All meta-data definitions and software applications managing this meta-data shall adhere to all other standards in this document, particularly the use of a character set supporting all diacritic characters.



8.11.4 Using Meta-Data

Using Meta-Data

When a software application uses meta-data and the meta-data provides separate fields (or records) for each language, this capability shall be exploited to utilise the data specific to the language preferred by the user when a user context is present and to utilise both languages otherwise.

When the meta-data isn't structured or separated into each language, the application shall assume that mixed text is used and handle the meta-data appropriately.

I.e. if meta-data is presented to a user, that user's language preference is known and the meta-data is available in both languages, then the meta-data relevant to the users language shall be used.

If the meta-data doesn't occur in both languages, the user preference isn't known, or the meta-data is only present in the alternate language, then whatever meta-data is available shall be used.

8.12 Searching

A bilingual information system will contain and manage data in both English and Welsh. Ideally, all data will be held in both languages however the reality is that some data might be available in one language and not the other.

The design of the data search capability should be such that the user is provided with the maximum flexibility in how to search, what language to use for the search and what data to search. The following functionality should therefore be implemented for search functions:

- The user should be able to select whether to search English only data, Welsh only data or both languages;
- The default selection should be to search in the current user interface language;
- If a search result item is available in both languages, then only the item in the same language as the current user interface language shall be returned;
- If search result items are only available in the alternate language, then they shall either be shown or their existence otherwise indicated to ensure that a search in either language returns the same results irrespective of the available languages for the items;
- When search results are returned, there should be an indicator of the language of the data;
- A text search should support diacritic equivalence by default, but provide the user with the option of disabling this capability. Diacritic equivalence is where all vowels in the search criteria will match irrespective of diacritic marks (also, the removal of duplicate entries where the duplication criteria is a result of diacritic equivalence);
- Where meta-data is used to enable the search, guidance on this topic can be found in 8.11;
- Issues specific to address searching are covered in more detail in section 9.

8.13 Language Preference

Individuals, organisations (and possibly other entities) will have a preferred communication language. This will generally be the language preferred for all communications, but could also be a range of preferences for different modes of communication types (i.e. verbal, written, visual user interface, etc).

8.13.1 Storing the Language Preference

Storing Language Preference

When storing information about an entity that will have a language preference (i.e. individuals, organisations, etc), it is essential that the data structure used shall allow for the language preference to also be recorded.

This was mentioned in 4.3 from the perspective of a software application being able to determine the language preference of a user. However, for software applications that manage data 'about' individuals and organisation (such as a CRM system), there must be the ability to store and maintain the preferred language of that entity.

A recommended set of values to be used is: 'English Only', 'English Preferred', 'Welsh Preferred', 'Welsh Only'.

This impacts upon the data schema defined for an application, functionality to enable this language preference to be maintained and interfaces between applications where the language preference attribute must be transferred along with any other biographical details.

8.13.2 Using the Language Preference

Whenever a communication occurs with an individual or organisation (or any other language sensitive entity), the language preference should be used to determine with the communication is to be bilingual or multilingual and if bilingual, which is the preferred language (i.e. the language that has precedence).

8.13.3 Bilingual vs. Multilingual

Storing and using language preferences highlights a key difference between a bilingual and multilingual situation. A multilingual approach will tend to have a primary (base) language and one or more secondary (alternate) languages.

The bilingual situation in Wales means that there are two primary languages (English and Welsh) and if a multilingual approach is used, other languages are secondary after, and not substituting for, these.

Therefore, if the software application is to be multilingual, it should account for both the language preference (English/Welsh) and also record the language preference for the secondary language.

9 Addresses and Geographical Information

Address databases are a subset of Geographic Information (GI). Although the focus of this document is on standardising and facilitating bilingual software within Wales, the use of GI datasets requires attention to both cultural and linguistic detail.

The availability of GI resources is generally increasing, is constantly in flux and datasets employ a variety of different schemas. It is therefore only possible to provide brief overview of the GI sector and provide a few guidelines on the use of GI data in this document.

9.1 Geographic Information (GI)

It is increasingly common for software to integrate with Geographic Information (GI) datasets. These can include (but aren't limited to):

- Address databases;
- Maps of regions, towns, streets, etc;
- Overlays mapping geographic reference points to real-world entities - cash machines, restaurants, museums, libraries, council buildings, etc;
- Statistical data applicable to a geographical area – census data, health characteristics, other demographic data, etc.

Every effort should be made to produce GI datasets that contain bilingual data.

9.2 Address Databases

It is common practice for software to integrate with third party address databases. Reasons for this can include:

- To find addresses quickly – by house number and postcode, for example;
- To ensure valid delivery addresses are used;
- In order to use standard address formats;
- To allow reliable interfacing with third party agencies (delivery companies, credit checking, etc);
- Cleaning existing customer address databases;
- To cross reference other GI datasets – electoral roll, crime rates, Local Education Authorities, etc.

9.2.1 Public Sector Resources

There are several national public sector databases of postal and property-level addresses that are held and maintained by a number of different organisations. These include:

- The Postcode Address File (PAF) maintained by Royal Mail;
- Ordnance Survey GB Address-Point;
- The HM Land Registry (HMLR) property database;
- The Valuation Office Agency (VOA) databases used for council tax;
- The National Land and Property Gazetteer (NLPG);
- The Local Land and Property Gazetteers (LLPG), constructed by Local Authorities and are linked to the NLPG.

The NLPG is the primary initiative to consolidate address data, in addition to maintenance work performed by Ordnance Survey, Royal Mail and the VOA.

9.2.2 Quality of Data

Despite the NLPG initiative and the many potential sources of addresses in the UK, there is still no definitive source of addresses. The situation worsens when the availability and currency of Welsh address data in a consistent and standardised manner is considered.

PAF-based databases do not always return accurate county information for Wales by default. For example, a search for an Anglesey postcode typically returns Gwynedd as the county, whereas a postcode in Conwy typically returns Clwyd.

Note that the Welsh PAF isn't usually provided together with the English PAF, but is often available upon request.

9.3 Relevant Standards

To facilitate the creation of Local Land and Property Gazetteers, a British Standard (BS7666 Spatial datasets for geographical referencing) was created. The standard comprises four parts covering Street Gazetteers, Land and Property Gazetteers, Addresses and Rights of Way.

This addressing standard is now being adopted throughout e-Government as part of the e-GIF standards and is increasingly used by the private sector.

BS7666 Parts One and Two specify a standard for the creation of a gazetteer for holding details on every property, piece of land and street within a defined area. Part Three specifies a model and basic guidelines for the structuring of address-based information.

As far as properties are concerned, the standard is based on the concept of a land parcel unit known as a Basic Land and Property Unit (BLPU). Each BLPU has a unique reference number (UPRN), a spatial reference (grid co-ordinate) and one or more Lang and Property Identifiers (LPI).

The LPI is basically the address of the BLPU in a standard format that uniquely identifies the BLPU in relation to a street as defined and held in the National Street Gazetteer (NSG).

BS7666 (2000) allows for an alternate LPI to be recorded. This should only be used for alternative language address attributes (such as the Welsh content for an address). This alternate type of LPI has equal standing within BS7666 as the main LPI. However, there can be only one alternative LPI for each BLPU which satisfies the bilingual requirement, but could cause problems if further languages are to be used .

There are ongoing developments in this field and this section will evolve accordingly.

9.4 Use of Addresses

Wherever possible, a software application should allow users to enter and manage their addresses in whichever language (or a combination thereof) they prefer. Where

an address has not been provided directly by the user, best efforts should be made to use an address consistent with their preferred language and where this isn't possible, then bilingual addressing should take place.

Where postcode mapping and/or other automated address management solutions are used, it is essential that provision is made for Welsh language addresses to ensure that addresses in both languages are supported and treated equally.

In the absence of a postcode mapping database, at a minimum, a software application should use enumerated lists for countries and counties.

9.4.1 User Entry of an Address

A software application should allow users to enter and manage addresses:

- In a preferred language; or
- In the most appropriate language for the context; or
- Bilingually.

A user interface can be designed in several different ways to capture address information.

A common method of entering addresses is by entering a house name or number and a postcode. These details can then be cross-referenced with an address database to provide a unique ID. The lines of the address can then be populated into text boxes and the user can customise these to their liking.

Note that this method may not work appropriately if searching against a Welsh PAF – building names are only stored in one language, not both.

If such an automated approach is adopted, it is important to still provide a user with the capability to enter their own custom address for the following reasons:

- The data retrieved in a particular language from the address database may not be geographically or linguistically accurate;
- Many people use mixed-language addresses – using an English street name and a Welsh town name, for example;
- New building developments may not be included in the address database.

Wherever enumerated lists are used (e.g. for counties) then these shall be displayed in the preferred language of the user.

Any user interface choices regarding the data entry screens will be dependant on the schema used for address storage. For example, BS7666 stipulates that the building number should be stored separately from the street name. Therefore, these details should be collected separately also.

9.4.2 Using the Appropriate Address

When selecting the address to be used to address a recipient:

- If a user has entered the address themselves, then this should be used as the sole address to communicate with them, irrespective of its language;

- If the user hasn't entered the address themselves, it is known in both languages and the user has expressed an explicit preference for a language, then the address for that language should be used in addressing them;
- If the user hasn't provided their address, the address is known in both languages but the user hasn't expressed a language preference, then a bilingual address should be used with the languages mixed (i.e. English/Welsh or Welsh/English on each line of the address);
- If the address is only known in one language, then there is no alternative other than to use that address. However, as the support for Welsh language addresses increases in databases, it is hoped that this will become a less frequent situation.

9.4.3 Delivery

It is recommended that correspondence is addressed in the recipient's preferred language (see 9.4.2), however the delivery capabilities of the courier should also be considered.

9.5 Storage and Analysis of Address Data

In order to handle address data in a structured way and allow reporting, it is necessary to tie addresses to geographic locations with a minimal reliance upon free-text. This can be achieved in several ways, including:

- Using the full postcode or parts thereof;
- Using enumerated country or county lists;
- Using geographic coordinates.

In using these means it can be ensured that both English and Welsh language addresses are handled correctly.

The storage medium should allow storage of addresses in conformance with any relevant standards, such as BS7666.

10 e-Government Interoperability Framework (e-GIF)

The e-Government Interoperability Framework (e-GIF) sets out the government's technical policies and specifications for achieving interoperability of computer systems across the public sector.

The framework comprises the e-Government Metadata Standard (e-GMS), the Integrated Public Sector Vocabulary (IPSV), Government Data Standards Catalogue, XML Schemas and the Technical Standards Catalogue (TSC).

The standards follow international guidelines and standards by default (such as the Dublin Core), drawing upon European and British standards in other areas.

The purpose of this chapter is to address language-related issues within e-GIF, not to itemise all the standards with which the Public Sector in Wales must comply.

10.1 e-Government Metadata Standard (e-GMS)

The role of the e-GMS standard is to provide high-level data about data content. This could be details of ownership, authorship, subject, title or date of publication, for example. The focus of the standard is very much upon the web (i.e. HTML content) and XML-based content.

e-GMS provides two means to signify the language of content:

- It allows the specification of the language of the document's content;
- It allows the specification of the language of the metadata.

10.1.1 Document Content Language

If a document's content were written in English, the following mark-up can be used:

<code><meta "DC.language" scheme="ISO 939-2/T" content="eng" /></code>	HTML
<code><dc:language scheme="ISO 939-2/T">eng</dc:language></code>	XML

If a document's content were written in Welsh, the following mark-up can be used:

<code><meta "DC.language" scheme="ISO 939-2/T" content="cym" /></code>	HTML
<code><dc:language scheme="ISO 939-2/T">cym</dc:language></code>	XML

If a document is bilingual, the language information appears twice, once for each language:

<code><meta "DC.language" scheme="ISO 939-2/T" content="eng" /></code>	HTML
<code><meta "DC.language" scheme="ISO 939-2/T" content="cym" /></code>	
<code><dc:language scheme="ISO 939-2/T">eng</dc:language></code>	XML
<code><dc:language scheme="ISO 939-2/T">cym</dc:language></code>	

10.1.2 Metadata Language

To specify metadata in a particular language, the `xml:lang` attribute should be used in both HTML and XML:

```
<meta name="DC.title" lang="en"  
      content="Bilingual Software Standards" />  
<meta name="DC.title" lang="cy"  
      content="Canllawiau a Safonau Meddalwedd Dwyieithog" />  
  
<dc:title xml:lang="en">  
  Bilingual Software Standards and Guidelines  
</dc:title>  
<dc:title xml:lang="cy">  
  Canllawiau a Safonau Meddalwedd Dwyieithog  
</dc:title>
```

It is also possible to link data to another resource with a Uniform Resource Identifier (URI) in HTML:

```
<link rel="DC.rights.copyright" hreflang="en"  
      content="http://abc/copyright.html" />  
  
<link rel="DC.rights.copyright" hreflang="cy"  
      content="http://abc/hawlfraint.html" />
```

10.2 Category Lists

Whilst e-GMS provides the means of presenting meta-information about data content, lists determine how data is described and how resources can be categorised uniformly across government.

At national and local government level, the Integrated Public Sector Vocabulary (IPSV) can be used to categorise resources.

At local government level, many other lists are available, including the following:

- Agency Type List;
- Audience List;
- Business Category List;
- Category List;
- Channel List;
- Classification Scheme;
- Directory List;
- Interaction List;
- Navigation List;
- Service List;
- Type List.

Further details can be obtained at: <http://www.eds.org.uk/standards/>

10.2.1 Use of Category Lists for English and Welsh Data

The metadata used to describe resources can be used in differing ways. For example, it can be used by other computer systems to automate discovery and knowledge sharing. It can also be used by users to drill-down through categories in portal sites.

Each category list enumerates categories and subcategories. For example, 5755 is the ID for the 'Languages' category in IPSV. However, note that when this information is used in HTML or XML, the English category names are used (since these are effectively just tokens):

```
<meta name="eGMS.subject.category" scheme="IPSV" content="Languages" />
```

Since the English text is just a token, there is no problem in using it in this manner. Indeed, using a Welsh translation in addition is redundant, ineffective and can be confusing.

However, these lists are dual-purpose and can also be used to enable a user to drill down through categories. This is clearly not appropriate in a Welsh language interface and therefore a parallel list of Welsh terms should be maintained and indexed across to the standard category list in this situation.

Note that the following IPSV categories can be used for language related content:

- Language – 5755;
- Language Policy – 3814;
- Welsh Language – 5776;
- English Language – 5761.

The viewer at: <http://www.esd.org.uk/standards/ipsv/viewer/viewer.aspx> provides more details.

10.3 XSD Schemas

The government provides schemas to ensure that data is stored and exchanged using known structured formats.

Many use the LanguageType simple type from the Govtalk core schema. This allows the specification of a language in ISO 639:1988 2 character format – “en”, “cy”, etc.

An example is the PersonalDetailsTypes schema which provides a CitizenDetailsStructure. This allows the specification of a PreferredLanguages element, of the LanguageType simple type. This PersonalDetailsTypes schema is employed by many of the other schemas in which citizen-based information is a component.

Note that the pattern matching statements given in some schemas do not allow the entry of accented characters. These typically state that content should come from the character range A-Z, which doesn't include accented characters.

The following lists other schemas with known language elements.

10.3.1 Archives and Records Management

ERMS 2 allows the specification of a language element in the Metadata element.

10.3.2 Election Markup Language (EML)

EML 2.1 UK allows the specification of a language for two entities:

- A PreferredLanguage in VoterInformationStructure;
- A Language and default Language for an ElectionEvent.

10.3.3 National Public Transport Access Nodes (NaPTAN)

The NaPTAN schema allows the provision of aliased free text, allowing stop points that can have a common name and alternative names. This can be used for bilingualism.

Common names and aliases must specify an `xml:lang` attribute to signify the language of the content. If none is specified, it is assumed that the language is English. This structure allows the common name to be in Welsh and the Alias to be in English, or vice-versa.

A summary of elements that provide aliases is provided in Section 13 of the NaPTAN Schema Guidelines 2.0.

In addition, translations for structured text terms are available (e.g. “Stop”, “Locality”, “Principal Timing Point”, etc). The use of Welsh or English structured names must be accompanied by an `xml:lang` attribute. If none is specified, it is assumed that the language is English.

11 Further Consultation

Though we have attempted to define a comprehensive, correct and implementable set of standards and guidelines, there remain areas where we haven't fully achieved these objectives for several reasons:

- Software (and IT in general) is in a continuous state of change and any fixed set of standards will be difficult to define such that they are entirely 'future-proof';
- Software support for the Welsh language is an area that is rapidly maturing, but there are still a number of areas where further work is required before effective standards can be defined.

For these reasons, whilst we have proposed an effective approach in the main body of the document, we are also keen to continue the consultation process. These areas are outlined in this section and we encourage and will welcome any feedback.

11.1 Sort Orders

Section 4.2 discusses the issues relating to alphabetic sort orders for a user interface. Section 8.8.2 briefly mentions database sort orders.

11.1.1 User Interface Sort Order

The discussion of the user interface sort order is fairly comprehensive and is a good balance between what is achievable and what is ideal. However, in reaching this balance, some compromise has been required, specifically:

- The acceptance that an English language user will likely not be aware of the use of digraphs in the Welsh alphabet, or indeed, that some names and terms are Welsh in origin. The sensible approach is therefore to utilise an English alphabet sort order in this circumstance;
- At present, no algorithm exists (or at least can be conceived by the Authors) that will sort all words containing Welsh digraphs correctly without the use of a lookup dictionary. However, such an approach isn't always feasible or practical.

The first issue is tough and requires a linguistic rather than technological solution and is therefore outside the scope of this document. However, the Board is keen to receive any input and comments on how the approach proposed in 4.2.2 can be refined or improved.

Regarding the digraph sorting issue, assuming that a dictionary lookup will remain beyond the bounds of feasibility in many situations (at least with current technology levels), the only ideal solution is offered by the 'Grapheme Joining Character'. However, it is unrealistic to expect all text to be annotated with this character to ensure that it can be used to provide 100% accurate sorting (though, these standards require it is supported in order to extend it's usage).

Therefore, we have defined a compromise approach that is achievable and will provide the best possible outcome. This is based upon the approach utilised by the current leading developers and localisers of Welsh software solutions. However, as it is a less than perfect approach, we remain keen to solicit any input or suggestions on this subject and welcome any debate on the most appropriate and achievable approach.

11.1.2 Database Sort Order

The architectural approach advocated within this document is to handle language specific functionality at the user interface wherever possible and, hence, make the remainder of an application effectively language-neutral.

This is generally a successful approach, except for the situations where the sort order used will impact upon the functionality of the application. This is particularly apparent at the database layer, especially since database management systems tend to have a default alphabetical sort order.

E.g. The results for a 'top 10' query will differ for an English and Welsh sort order (as defined in 4.2.2) where 'London' and 'Llanberis' are the 10th and 11th entries on the list.

Though the sort order can be changed relatively easily at the user interface layer, this is generally not possible at the database layer. Even when the collation can be specified (e.g. SQL implementations supporting a COLLATE clause), there isn't usually a Welsh collation available.

Therefore, these standards have avoided the issue of defining the database sort order that should be used. This is partly a matter of practicality, since the majority of current database products do not provide a Welsh sort order option (or the ability to define a custom sort order), but mostly reflects the challenge outlined above. Whereas an English only application should use the English sort order and a Welsh only application should use the Welsh order, there doesn't appear to be a generic elegant solution for the bilingual situation.

There are potential solutions through the implementation of functionality integrated with the database layer (i.e. stored procedures and their equivalent), but this capability is implementation-specific and not a sufficiently generic solution at present.

Therefore, we again encourage any feedback that can help to clarify and address this situation and/or help us to refine the approach recommended in this document.

11.2 Addresses & Geographical Information

The issues surrounding bilingual address management and geographical information are diverse and complex. The shortcomings of current bilingual data resources and the relative inexperience of implementing bilingual addressing and GIS solutions are both additional complicating factors.

In this document we have outlined some of the background and basic issues in this area. We believe that there is further information and standards relevant to this area and welcome the input, guidance and suggestions from any individuals or organisations that have experience, in this area.

11.3 eGovernment

Many of the areas addressed by the eGovernment standards have little relevance to language issues and are therefore outside the scope of this document. Section 10 identifies several areas that do have language considerations and recommends suitable approaches.

The eGovernment standards have grown quickly and continue to evolve. Also, though there is some experience in implementing solutions that adhere to these standards and are also bilingual, this is still fairly fragmented.

Though we consider the coverage in section 10 to be a useful starting point, we welcome and encourage feedback and suggestions in this area from those involved in the development and implementation of bilingual eGovernment software solutions.

11.4 Other Standards

Existing and relevant standards have been identified throughout the document. We have attempted to ensure that there is a minimal overlap with existing standards to minimise the potential for contradiction and also to ensure that this document remains relevant, even if those standards change.

However, the world of standards is large and complex and it is possible that we have omitted important standards, that we have unknowingly 'overlapped' with an existing standard that we are unaware of and/or that we have contradicted existing standards.

In order to improve future versions of this document and also to ensure that these standards integrate effectively and with minimal conflict with other standards, we will appreciate any feedback, suggestions and information that will assist with this.

11.5 New Technologies

As mentioned in 1.3, software and technology is continuously advancing. As such, though we have defined standards that are relevant and appropriate at the time of writing, we are aware that new technologies might require extensions and modifications to these standards.

Wherever possible we have taken a fairly generic approach that should withstand many of these changes and also be applicable across a range of terminologies for the same software capabilities. Where these standards already provide relevant guidance, they should be interpreted in a suitable manner.

However, as new technologies emerge and new modes of interfaces between humans and software systems evolve, new issues relating to the handling of language and, specifically, bilingualism will arise. As for all aspects, we ask anyone working with technologies and issues not sufficiently or suitably covered to provide input on the issues faced and appropriate modifications to enable standards to be relevant and provide meaningful guidance.

11.6 Specific Application Areas

As for technologies, the range of software applications is continuously growing and evolving. Again, though it is hoped that the standards are sufficiently generic to ensure relevance across all applications, there will undoubtedly be specific applications and functional capabilities that required coverage but have not been addressed.

Examples of these might include SMS, Instant Messaging, Wearable computers, Speech processing, etc.

Identification of any such omissions and suggestions for suitable standards would be appreciated.



12 Guidance for Web Designers

The preceding standards have been written to be applicable to the widest range of software systems possible. The purpose of this section is to provide clarification on certain areas of the standards for web designers.

This section will only address a subset of the standards with which a web site must comply. It is therefore necessary for web designers to use sections 3 through 10 as their primary reference.

It isn't possible for this section to discuss specific implementation details as the range of web technologies (such as ASP.NET, PHP, JSP, CGI, etc) is vast.

All references to HTML are equally applicable to XHTML.

12.1 Character Sets

Topic:	Use of character sets within HTML documents
Section(s):	4.3.1
Guidance:	<p>The character set shall always be expressed using a meta tag in HTML.</p> <pre><meta http-equiv="Content-Type" content="text/html; charset="utf-8" /></pre> <p>Note that the actual encoding of the file should always match that which is given in the meta tag. Unicode is the recommended encoding.</p>
Reference(s):	http://www.w3.org/International/O-charset.html

Topic:	Use of Unicode characters in other character sets
Section(s):	4.3.3, 4.3
Guidance:	<p>Where it isn't possible to use the Unicode character set, it is possible to still use individual Unicode characters by encoding them:</p> <ul style="list-style-type: none"> • &#373; gives 'ŵ'; • &#375; gives 'ŷ'; • Etc. <p>The decimal numbers to be used after "&#" are given in the table at the start of section 4.3.</p>

12.2 Initial Implicit Language Assumption

Topic:	Using a browser's locale to identify a language preference
Section(s):	5.1.1
Guidance:	It is possible to retrieve the browser's preferred languages from the ACCEPT_LANGUAGE HTTP header.
Reference(s):	<p>http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.4</p> <p>The method of retrieving this header is platform-specific. A search for "ACCEPT_LANGUAGE" using a web search</p>

	engine is recommended in order to find out the correct approach for your platform.
--	--

Topic:	Using a domain name to make a language assumption
Section(s):	5.1.1
Guidance:	It is possible to retrieve the domain name from the HTTP_HOST environment variable.
Reference(s):	The method of retrieving this variable is platform-specific. A search for “HTTP_HOST” using a web search engine is recommended in order to find out the correct approach for your platform.

Topic:	Using a querystring element to identify a language preference
Section(s):	5.1.1
Guidance:	For example: <code>http://mywebsite.co.uk/mypage.asp?lang=cy-gb</code> The ‘lang’ parameter’s value can then be retrieved from the request object.
Reference:	The method of retrieving this value is platform-specific. A search using a web search engine is recommended in order to find out the correct implementation approach for your platform.

Topic:	Using a referring URL to make a language assumption
Section(s):	5.1.1
Guidance:	It is sometimes possible to retrieve a referring URL from the HTTP_REFERER environment variable. If this could be identified as being a known Welsh language domain name, the initial language could be assumed to be Welsh, for example. Note that this cannot be relied upon as a means of making an initial language assumption, as some browsers (or Internet security products) do not allow browsers to give a referring URL when requesting a page.
Reference(s):	The method of retrieving this variable is platform-specific. A search for “HTTP_REFERER” using a web search engine is recommended in order to find out the correct approach for your platform.

12.3 Ubiquitous Language Selector

Topic:	Location of Language Selector
Section(s):	5.1.3
Guidance:	The emerging standard for the location of language selectors is in the top right-hand corner of a web page. This is known as the “sweet spot”.
Reference(s):	http://www.corante.com/goingglocal/

12.4 Persistence of Language Selection

Topic:	Persistence of Language Selection
Section(s):	5.2
Guidance:	<p>In order that a language decision can be reversed by clicking on a “back” button, it will be necessary to store the language preference as part of a querystring, rather than associating a server-side language preference with a session identifier or similar.</p> <p>For example:</p> <p>http://mywebsite.co.uk/mypage.php?lang=en-gb</p> <p>Rather than:</p> <p>http://mywebsite.co.uk/mypage.php?session=123456</p>

12.5 Storing Language Preference

Topic:	Application Level Profile
Section(s):	5.3.1
Guidance:	<p>In order to store a language setting at this level, it will be necessary to write a cookie to the user’s browser cache. Note that browser security settings may limit the success of this method.</p>
Reference(s):	http://www.cookiecentral.com/

12.6 Domain Names

Topic:	Internationalised Domain Names (IDNs)
Section(s):	5.4
Guidance:	<p>At the time of going to print, IDNs are not supported on domain names ending in .com, .net, .org, .edu or .uk.</p> <p>The UK domain registry (Nominet) is currently in consultation on whether IDNs are desirable and viable for the UK market.</p>
Reference(s):	https://lists.nominet.org.uk/mailman/listinfo/nom-pwg-idn

12.7 Error and Event-Driven Messages

Topic:	Display of Messages
Section(s):	5.8
Guidance:	<p>It should be remembered that any message displayed may also need to be accessible. As such, options such as Javascript alert boxes may not be a viable approach to displaying messages.</p>
Reference(s):	http://www.w3.org/WAI/

12.8 Mixed Language Content

Topic:	Mixed Language Content
Section(s):	6.2.2, 6.3
Guidance:	<p>Whenever both English and Welsh are used on the same page, the language of each passage of text should be given.</p> <p>It is possible to place a “lang” attribute on most HTML tags (the exceptions being base, br, frame, frameset, hr, iframe, param and script according to the XHTML standard).</p> <p>For example:</p> <pre><p lang="en">English</p> <p lang="cy">Cymraeg</p></pre>
Reference(s):	http://www.w3schools.com/tags/ref_standardattributes.asp

12.9 Meta data

Topic:	Multilingual meta-tags
Section(s):	8.11, 10
Guidance:	<p>A meta-tag can be used to state the primary language of an HTML document:</p> <pre><meta http-equiv="Content-Language" content="en-gb"> <meta http-equiv="Content-Language" content="cy-gb"></pre> <p>Meta-tags can be used multilingually (i.e. one meta-tag for English, another for Welsh), as exemplified in section 10.1.2:</p> <pre><meta name="DC.title" lang="en" content="Bilingual Software Standards" /> <meta name="DC.title" lang="cy" content="Canllawiau a Safonau Meddalwedd Dwyieithog" /></pre>
Reference(s):	http://www.html-reference.com/META_httpequiv_language.htm

13 Resources & Bibliography

13.1 Character Sets

Unicode:

<http://www.unicode.org/>

Code charts:

<http://www.unicode.org/charts/>

Ascitable.com:

<http://www.asciitable.com/>

W3.org document on character sets:

<http://www.w3.org/TR/1999/REC-html401-19991224/charset.html - code-position>

ISO 8859-14:

<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=29630>

13.2 Keyboard Key Sequences

Microsoft (Windows XP service pack 2 & later)

<http://www.microsoft.com/globaldev/handson/user/welsh.msp>

To Bach Schema

<http://www.draig.co.uk/tobach> & <http://www.meddal.com/>

13.3 Language Resources

Welsh Language Act:

http://www.legislation.hms.gov.uk/acts/acts1993/Ukpga_19930038_en_1.htm

Plain English:

<http://www.plainenglish.co.uk/>

Association of Welsh Translators (CCC):

<http://www.welshtranslators.org.uk/>

Canolfan Bedwyr:

<http://www.bangor.ac.uk/ar/cb/>

Cymraeg Clir (Plain Welsh):

http://www.bangor.ac.uk/ar/cb/cymraeg_clir.php

Cysgliad (includes Cysill and Cysgeir):

<http://www.bangor.ac.uk/ar/cb/meddalwedd.php>

13.4 Welsh Language Board

Main website:

<http://www.welsh-language-board.org.uk/>, containing:

- Snapshot survey (2001) – Websites of organisations complying with statutory language schemes. Lingua Cambria Cyf
- Snapshot survey (2003) – Websites of organisations complying with statutory language schemes. Cwmni Cymad
- A guide to bilingual design, 2001 edition.
- Windows XP & Office 2003 Available in Welsh:
- Information Technology Resources
- Information Technology and the Welsh Language: A Strategy Document

13.5 Localisation & Multilingual Computing

Websites:

IBIS (Interfaces to Bilingual Information Systems:
<http://weblife.bangor.ac.uk/ibis/default.html>

Bilingual Web Development Resources
<http://www.comp.glam.ac.uk/%7EDaniel.Cunliffe/bilingual/>

Microsoft GlobalDev: Locales & Languages :
<http://www.microsoft.com/globaldev/DrIntl/faqs/Locales.mspix>

List of Locale ID (LCID) Values Assigned by Microsoft
<http://www.microsoft.com/globaldev/reference/lcid-all.mspix>

ISO 639 (Wikipedia)
http://en.wikipedia.org/wiki/ISO_639

ISO 3166-1 alpha-2 (Wikipedia)
http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

Books:

Developing International Software, second edition (2002), Dr International, Microsoft Press, ISBN 0-7356-1583-7

Internationalization and Localization Using Microsoft .Net, 2002, Nick Symmonds, Apress, ISBN 1-59059-002-3

Periodicals:

Multilingual Computing & Technology, <http://www.multilingual.com/>, +(1) 208 263 8178

13.6 Address Management

Addressing for Britain is not as Simple As Going from A to B, Gavin Keith, Robin McLaren, AGI, 2003.

<http://www.knowledgeconsortium.co.uk/documents/Doc47 - Addressing for Britain.PDF>

BS7666 for Beginners, The National Land & Property Gazetteer, April 2003.

<http://www.nlpg.org.uk/public/download/BS7666forBeginners.pdf>

NLPG e-zine, The National Land & Property Gazetteer, March 2004.

<http://www.nlpg.org.uk/ezone/March2004e-zine.htm>

Review of Royal Mail's Licence Condition 20 - Postcode Address File Code of Practice, A Decision Document, Postcomm, March 2004.

<http://www.postcomm.gov.uk/documents/licensing/PAFCodeofPracticeReviewFinalDoc.pdf>

Geographic Information Strategy for Wales, AGI Cymru with sponsorship from the Welsh Assembly Government, 2003.

<http://www.cymruarlein.wales.gov.uk/fe/default.asp?n1=1&n2=7&n3=223>

13.7 e-Government Standards & Guidelines:

Office of the e-Envoy: <http://www.govtalk.gov.uk/>

e-Government Interoperability Framework Version 6.1, e-Government Unit, Cabinet Office, 18 March 2005.

http://www.govtalk.gov.uk/schemasstandards/egif_document.asp?docnum=949

What Are All These Lists?, Issue 0.02 – Draft – 3 September 2004.

<http://www.esd-toolkit.org/forums/download.php?id=445>

Expressing Dublin Core in HTML/XHTML meta and link elements, Dublin Core Metadata Initiative

<http://dublincore.org/documents/dcq-html/index.html>

GovTalk Schemas, e-Government Unit, Cabinet Office

<http://www.govtalk.gov.uk/schemasstandards/xmlschema.asp>

13.8 Other

e-Gymraeg Discussion List:

<http://www.jiscmail.ac.uk/lists/e-gymraeg.html>

Nominet consultation on Internationalised Domain Names (IDNs)

<http://www.nominet.org.uk/Pab/PabConsultationPapers/IdnConsultation/>