WIS IMPLEMENTATION NOTE No.1

Regular Expression in WIS DAR Metadata

TOYODA Eizi
2010-05-03
Last update: 2010-05-18

Remark: This discussion paper is provided to facilitate discussion in WMO/CBS/IPET-MDI. Future standardization progress in WMO may obsolete part or entirety of discussion here.

Executive Summary

Regular Expression is powerful but maybe too powerful to use safely. The portability and security issues are discussed, considering the use in "MX_DataFile/fileName" of WIS DAR Metadata for globally-distributed data, which is currently GTS data. The author makes recommendations as follows:

1. Metadata creator or manager should provide regular expression that is understandable for all GISCs. Recommended syntax is chosen from common subset of various implementations (cf. §3.1.1 for detail; syntax letters are **? * + {,} (|) ^ $ [] [^] [***a***-***z***] \d \w . \. \+**).
2. GISC Cache should support at least syntax recommended here. It may also support other metacharacter for further development, but it must not alter recommended syntax and its semantics. It is advised to check regular expression provided in metadata before matching.
3. Although WMO Filename Convention is case-insensitive, metadata creators should give regular expression in correct case of letters. For example, location identifier should be described as **[A-Z]{4}**, not **[a-z]{4}**.
4. GISC Cache should use "ignore-case" option of regular expression library in matching metadata and dataset fragment.
5. Above recommendations should be reviewed when use of "MX_DataFile/fileName" is extended beyond current GTS files and bulletins, or WIS global data is extended beyond current GTS.

1. Introduction

The first meeting of WMO/CBS/IPET-MDI (27-29 April 2010) agreed to use regular expression as a mechanism of metadata-fragment matching for information for global distribution. Regular expression was chosen because of greater expressional power than wildcard; practically there are many filename patterns that is well described in regular expression but confused in wildcards.

However there are significant incompatibilities in regular expression syntax among implementations. And unfortunately, security consideration is necessary for some environment. Thus it is considered safer way to limit syntax to portable subset of implementations.

This document provides overview the issues and practical guidelines, designed by following principles:

- Shorthand notation is welcome within below two principles.
- WIS Standard should not be too restrictive on details of implementation, such as programming language or operating system.
- WIS Centre must be robust to security threats, such as injection of unexpected regular expression by mistake or by malicious intent.

2. Study of Current Technologies

Note: skip this section if you feel tired reading. The recommendation (§3) should be understandable by itself.

2.1 Brief History of Regular Expression

The regular expression has its roots in linguistics and computer sciences in 1950s. In theory, only three operators are necessary: zero-or-more repetition *****, choice **|**, and grouping **( )**.

The theory comes into widespread practical use by Unix operating systems. Commonly-referenced standard is Single Unix Specification (SUS) [SUSv2] which is developed in 1990s, and is superset of POSIX also known as IEEE Std 1003 or ISO/IEC 9945. It is still often used when writing programs in C[1].

These Unix standard defines two versions of syntax: basic tools like sed(1), vi(1) or grep(1) uses basic regular expression (BRE), while awk(1) and egrep(1) uses extended regular expression (ERE). Only ERE is considered of contemporal value, since BRE has different syntax (ex. **\{ \}** for **{ }**) and is not real regular expression since it lacks choice operator (**|**)~~: by the way comma-separated list of regular expressions (**/***re1***/, /***re2***/,** ...) is outmoded habit of that time~~.

Contemporary programing languages implement regular expression with significant extension from SUS. The Perl programming language is a leader and virtual reference, but there is still difference among them.

2.2 Observations from Comparison

Appendix A compares regular expression syntaxes of SUSv2 BRE, Perl, Java, XPath, and JavaScript. The last two might sound exotic for GISC cache implementation but are chosen as typical environments in which it is not easy for users to replace regular expression library.

- Greedy quantifiers (**?**, *****, **+**, and **{***m***,***n***}** without suffix **+** or **?**) are compatible
- Non-greedy (reluctant or possessive) quantifiers cause portability problem
- Anchors (**^** and **$**) are well portable. XSD does not have anchors, but it is just because the purpose is clearly limited to match for entire name. XPath has anchor support.

---

1. It is a good question whether C is still used, but the Author considers it is unsafe to ignore top-ranked language in TIOBE index <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

- Assertions (**(?=***re***)**, **\b**, etc.) is not supported in XSD/XPath. They are unpopular for theoreticians because of deviation from formal language theory.
- Choice operator (**|**) is fairly portable.
- Mark and reference within the pattern is not portable for at least SUSv2 ERE and XPath.
- Traditional character set expression (**.**, **[***list***]**, **[^***list***]**) is fairly portable.
- Behavior of range depends largely on environment. SUSv2 defines it as range in collating sequence, while modern standards (such as Java or XPath) tend to simply refer to Unicode character code.
- Well-known character set expressions (**\d**, **\w**, **\s**) seems to share same meaning for US-ASCII block, while behavior differs significantly for overall Unicode characters.
- Other character set expressions (**[[:***alpha***:]]**, **\p{***Space***}**, etc.) are less portable.
- Set features such as collating element (**[[.***ch***.]]**), equivalence class (**[[=***a***=]]**), or character set algebra (intersection & subtraction) should be avoided to be portable.
- Limited non-printable expressions (**\n**, **\r**, **\t**) are portable. Hexadecimal single-octet (**\x***FF*) is almost portable except for XPath, while there is no portable way to specify all Unicode characters. For example, **\x{1000}** denotes MYANMAR LETTER KA in Perl, but it may be understood as "thousand times of lowercase x" by other implementations.

One notable fact is that all implementation checked here has "ignore-case" option no matter how the regular expression is written.

2.3 Security Consideration

Perl has some syntax that allows data providers are able to execute any code in regular expression (ex. **(?{***code***})** and **(??{***code***})**). This kind of feature can be security problem if a metadata processor is implemented in Perl, since GISC Cache processes regular expression in metadata created by external centres.

Although this syntax is enabled only when pragma `re` is specified, it is unrealistic to build non-trivial system since it should be now programmer's responsibility to guarantee that the pragma is never specified in the entire system to pass security audit. Command-line option -T inspects insecure operation, but it is often forgotten for non-CGI programs.

Perl regular expression has another suspicious feature **\Q***re***\E** in which variables $var or @var are interpolated. Although this does not directly invoke code, regular expression suppliers could investigate variables in programs which is otherwise not revealed to them.

3. Recommendations

This section consists of blue boxes (recommendation for IPET-MDI report, WIS WIKI or future Manual on WIS) and rationale or notes to that.

3.1 Syntax of Regular Expression

Argument in this section assumes that regular expression is only used for matching filename that is already known to compliant to WMO Filename Convention (FNC). FNC limits filename to a subset of US-ASCII (shown in Appendix B), which simplifies the argument.

Recommendation 1 (for metadata creator/manager)

Metadata creator or manager should provide regular expression that is understandable for all GISCs. Recommended syntax consists of:

*x***?**	zero or one occurrence of *x*
*x******	zero or more occurrence of *x*
*x***+**	one or more occurrence of *x*
*x***{***m***}**	exactly *m* times of occurrence of *x*
*x***{***m***,}**	at least *m* times of occurrence of *x*
*x***{***m***,***n***}**	at least *m* times but not more than *n* times of occurrence of *x*
(*re***)**	grouping (*re* must be valid expression hence may not begin with **?**)
^*re*	string that begins with *re*
*re***$**	string that ends with *re*
*re***|***alt*	choice: either *re* or *alt* (usually used within grouping)
[*abc***]**	positive bracket: single character either *a*, *b*, or *c*
[^*abc***]**	negative bracket: any single character neither *a*, *b* nor *c*
[*a-z***]**	range: shorthand for **[***abcdefghijklmnopqrstuvwxyz***]**
\d	shorthand for **[0-9]**
\w	shorthand for **[0-9A-Z_a-z]**
.	any single character
\.	full stop character
\+	plus sign character
pqr123_-,	alphanumeric, underline, hyphen-minus, and comma denote character itself

where

- *m* and *n* is nonnegative integer (ex. 4 or 12);
- *x and y may be single character, grouping* or *bracket* (hence it cannot begin with metacharacters like question mark (**?**));
- *re* and *alt* is regular expression derived from above syntax; and
- *abc* is non-empty sequence of ITU-T IA5 graphic characters[2] in which
  - right square bracket (**]**) must be the first;
  - left square bracket (**[**) is recommended to be at the end; and
  - hyphen-minus (**-**) must be the first or the last (to match hyphen-minus itself) or middle of range (i.e. "**i-o-u**" is illegal).

It is worth noting that:

- non-standard syntax will result implementation-dependent behavior or error; for example it is advised to avoid escaping (prefixing backslash) characters not described above (such as **\e**);
- the metacharacters (such as **.** or **+**) lose its special meaning in bracket, thus **[\.]** would match either backslash or full stop.

Rationale:

- Recommended syntax is indicated, instead of listing constructs to be avoided. That is because unrecommended syntax is mostly environment-dependent and it is difficult to have comprehensive list. Description is carefully written to avoid such nonstandard expression intentionally or by mistake.

---

2. Graphic character is space and printable character, defined in ITU-T Rec. T.40 Section 4.

- Character set and escape are limited to be meaningful for FNC. For example, ***** or **\s** are portable but cannot be used in this purpose.
- Warning on environment-dependency of range is not indicated, because modern character encoding scheme is ASCII-compatible for FNC character set, and EBCDIC is not expected to be in use.
- Following features are excluded not only because of incompatibility with XPath:
    - Backtrack assertion such as **(?=***re***)** or **(?<=***re***)** is used for limiting context of matching string, which is irrelevant for filename matching.
    - Word boundary assertion **\b** or **\B** has incompatible or undocumented definition in some environment. It is irrelevant for FNC because it does not detect most-often used delimiter underline (_).
    - Reference **\1**, **\2**, ... **\9** is used in detection of repeated but variable string (ex. **(\w+)\1** matches "wikiwiki" but rejects "funiculifucnicula"), which is not expected for the time being.
- After all, these restrictions might look too strict, but it is safer to begin with portable subset of syntax.

Recommendation 2 (for GISC implementers)

GISC Cache should support at least syntax shown in recommendation 1. It may also support other metacharacter for further development, but it must not alter recommended syntax and its semantics.

It is advised to check regular expression provided in metadata before matching. Some expression may increase security risk. WIS center may detect and disable syntactically correct but semantically inappropriate pattern of regular expressions at its decision.

Notes:

- The first paragraph does not specify implementation detail of GISC Cache. It is possible to develop a software even in C with SUSv2 regular expression library; in that case there has to be a wrapper routine to convert **\d** to **[0-9]** before compiling regular expression.
- It is expected that the recommended syntax of regular expression is extended in the future. Here extension is allowed for phased transition of operational systems without breaking compliance to this recommendation.
- Example of inappropriate pattern in terms of security is **(?{***code***})** in Perl.
- The concept of "inappropriate" includes too resource-consuming regular expression. Currently there is only general and less-likely reservation because more experience is needed to have specific guideline.

3.2 Case Sensitivity

Recommendation 3 (for metadata creator/manager)

Although WMO Filename Convention is case-insensitive, metadata creators should give regular expression in correct case of letters. For example, location identifier should be described as **[A-Z]{4}**, not **[a-z]{4}**.

Rationale:

- Even with following recommendation 4, it is misleading for human operator/ maintainer to allow regular expressions that fails in case-sensitive match.
- It is considered easy for metadata creator to specify correct case of letters.

- Assuming case-sensitivity might be problematic in the future if the MX_DataFile/ fileName construct is extended to allow other case-sensitive namespaces.

Recommendation 4 (for implementers)

GISC Cache should use "ignore-case" option of regular expression library in matching metadata and dataset fragment.

Rationale:

- Even with previous recommendation 3, it is safer to perform case-insensitive match.

4. Considerations for non-ASCII Filename

There was an opinion in IPET-MDI-1 meeting that the "information for global distribution" of WIS may include non-ASCII filename in the future. In that case discovery metadata will have to contain "environmentDescription" or "MX_DataFile/fileName" with regular expression for such filename.

Such consideration seems to be beyond of IPET-MDI ToR, but following considerations are necessary with regard of metadata compatibility.

4.1 Range Definition and Character Sets

Should expression **[a-z]** match letter **à** (a with accent grave)? Similar questions can be made for **\w** or **\d**.

The answer should be NO for WIS perspective. Now GTS metadata is being created, and it should not be desirable if there is possibility that a regular expression will match something not intended now. It is welcome tendency that Java and XSD does *not* let **[a-z]** match letter **à**. Implementers are advised to suppress locale-dependent feature if working with Perl or C, typically by explicitly specifying "C" locale.

4.2 Case Sensitivity

When filename is extended beyond FNC, could we assume case-insensitivity? The answer is probably negative. Thus we have to review recommendation 4 above.

4.3 Summary

Recommendation 5 (for CBS/IPET-MDI)

Above recommendations should be reviewed when

- use of "MX_DataFile/fileName" is extended beyond current GTS files and bulletins, or
- WIS global data is extended beyond current GTS.

References

[Java] Sun Microsystems Inc., 2004: JavaTM 2 Platform Standard Ed. 5.0, java.util.regex Class Pattern. http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html

[JavaScript] Mozilla Developer Center, last update 2010-04-30: Regular Expressions. https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Regular_Expressions

[SUSv2] The Open Group, 1997: The Single UNIX ® Specification, Version 2, Regular Expressions. http://www.opengroup.org/onlinepubs/007908799/xbd/re.html

[XSD] W3C, 2004: XML Schema Part 2: Datatypes Second Edition. Appendix F: Regular Expressions. http://www.w3.org/TR/xmlschema-2/#regexs

Appendix A: Comparison of Regular Expression Dialects

Disclaimers:

- The purpose of this table is to demonstrate incompatibility among implementations, and to give suggestion on relatively portable syntax of regular expression.
- Mapping of this table may be incomplete: it should not be used as transcription of regular expressions between languages.
- This table does not mean the author recommends specific software and/or version: it is simply recent versions with good documentation of regular expression.
- Indications like N/A in this table does not necessarily mean a software lacks some feature: it may be just undocumented. Also there may be functionality not in this listing except for SUSv2 ERE and JavaScript, for which documentation kindly states it is complete list.

Functions	SUSv2 ERE (egrep)	XSD and XPath2	Perl 5.8.6	Java 5.0	Java-Script 1.5
Overall Syntax: all non-alpha-numerics followed by backslash are literal	undefined but yes in most imple-mentations	undefined and probably error	YES	YES	YES
[a-z] matches **à**	Yes in some locales	NO	NO	NO	*(unknown)*
there is "ignore case" option out of regular expression syntax	YES	YES	YES	YES	YES
Quantifier: zero or more	*****	*****	*****	*****	*****
zero or one	**?**	**?**	**?**	**?**	**?**
one or more	**+**	**+**	**+**	**+**	**+**
exactly *m* times	**{***m***}**	**{***m***}**	**{***m***}**	**{***m***}**	**{***m***}**
at least *m* times	**{***m***,}**	**{***m***,}**	**{***m***,}**	**{***m***,}**	**{***m***,}**
at least *m* times but not more than *n* times	**{***m***,***n***}**	**{***m***,***n***}**	**{***m***,***n***}**	**{***m***,***n***}**	**{***m***,***n***}**
reluctant quantifier, i.e. as few as possible	N/A	N/A (XPath2 has ***?** etc.)	***?** **??** ... **{***m***,***n***}?**	***?** **??** ... **{***m***,***n***}?**	N/A
possessive quantifier, i.e. as many as locally possible even if overall match would fail	N/A	N/A	**(?>***re****)** **(?>***re***?)** ... **(?>***re* **{***m***,***n***})**	***+** **?+** ... **{***m***,***n***}+** or same as Perl	N/A
Anchor: beginning of string	**^**	unused (XPath2 has **^**)	**^**	**^**	**^**

ending of string	**$**	unused (XPath2 has **$**)	**$**	**$**	**$**
Assertions: pattern *a* followed by pattern *b* (that is not a part of match result)	N/A	N/A	*a***(?=***b***)**	*a***(?=***b***)**	*a***(?=***b***)**
pattern *a* not followed by pattern *b*	N/A	N/A	*a***(?!***b***)**	*a***(?!***b***)**	*a***(?!***b***)**
pattern *a* following pattern *b*	N/A	N/A	**(?<=***b***)***a*	**(?<=***b***)***a*	N/A
pattern *a* not following pattern *b*	N/A	N/A	**(?<!***b***)***a*	**(?<!***b***)***a*	N/A
word boundary	N/A	N/A	**\b**	**\b**	**\b**
non-word boundary	N/A	N/A	**\B**	**\B**	**\B**
Choice: either *a* or *b*	*a***|***b*	*a***|***b*	*a***|***b*	*a***|***b*	*a***|***b*
Parenthesis: mark for reference of *1*st mark within the pattern	N/A	N/A	**(***re***)** ... *****1*	**(***re***)** ... *****1*	**(***re***)** ... *****1*
grouping: ex. *a* or *b* followed by *c*	**(***a***|***b***)***c*	**(***a***|***b***)***c*	**(***a***|***b***)***c*	**(***a***|***b***)***c*	**(***a***|***b***)***c*
alternate syntax of grouping to suppress marking	N/A	N/A	**(?:***a***|***b***)***c*	**(?:***a***|***b***)***c*	**(?:***a***|***b***)***c*
code execution and evaluation	N/A	N/A	**(?{***code***})** ‡ **(??{***code***})** ‡	N/A	N/A
preprocessing operations: case changes	N/A	N/A	**\l**, **\u**, **\L**, **\U**	N/A	N/A
disable metacharacters in *re*	N/A	N/A	**\Q***re***\E** (here **$** and **@** are interpolated) ‡	**\Q***re***\E**	N/A
Single Character: any one character	**.**	**.**	**.**	**.**	**.**
any character in a list	**[***list***]**	**[***list***]**	**[***list***]**	**[***list***]**	**[***list***]**
any character out of a list	**[^***list***]**	**[^***list***]**	**[^***list***]**	**[^***list***]**	**[^***list***]**
collating element: ex. digraph *ch* in Czech locale	**[[.***ch***.]]**	/A	N/A	N/A	N/A

equivalence class: ex. *a* with or without accent in French locale	**[[=*a*=]]**	N/A	N/A	N/A	N/A
intersection: *a* to *z* and *i* to *n*	N/A	N/A	N/A	**[***a-z***&&[***i-n***]]**	N/A
subtraction: *a* to *z* and not in *i* to *n*	N/A	**[***a-z***]-[***i-n***]**	N/A	**[***a-z***&&[^***i-n***]]**	N/A
Character Class: alphanumeric	**[[:alnum:]]** §	**[A-Za-z0-9]**	**[A-Za-z0-9]**	**\p{Alnum} [A-Za-z0-9]**	**[A-Za-z0-9]**
word (alphanumeric or underscore)	**[[:alnum:]_]** §	**\w** ¶	**\w** §	**\w**	**\w**
non-word	**[^[:alnum:]_]** §	**\W** ¶	**\W** §	**\W**	**\W**
(latin) alphabet	**[[:alpha:]]** §	**[A-Za-z]**	**\p{Alpha}** § **[[:alpha:]]** §	**\p{Alpha}**	**[A-Za-z]**
decimal digit	**[[:digit:]]**	**\d** ¶ **[0-9]**	**\d** § **\p{Digit}**§	**\d** **\p{Digit}**	**\d**
non-decimal digit	**[^[:digit:]]**	**\D** ¶ **[0-9]**	**\D** § **\p{Digit}**§	**\D** **\P{Digit}**	**\D**
whitespace (space, tab, newline, carriage return)	**[[:space:]]**[3]	**\s**	**\s**[4] **\p{Space}**	**\s**[5] **\p{Space}**	**\s**[6]
non-whitespace	**[^[:space:]]**	**\S**	**\S** **\P{Space}**	**\S** **\P{Space}**	**\S**
Non-printable: control character: ex. control-*B*	N/A	N/A	**\c***B*	**\c***B*	**\c***B*
backspace	N/A	**[#x08]**	**[\b]**	**\x08**	**[\b]**
alert (bell)	N/A	**[#x07]**	**\a**	**\a**	**\x07**
escape character	N/A	**[#x1B]**	**\e**	**\e**	**\x1B**
form feed	N/A	**[#x0C]**	**\f**	**\f**	**\f**
line feed	N/A	**\n**	**\n**	**\n**	**\n**
carriage return	N/A	**\r**	**\r**	**\r**	**\r**
horizontal tab	N/A	**\t**	**\t**	**\t**	**\t**
vertical tab	N/A	**[#x0B]**	**\v**	**\v**	**\v**
NUL character	N/A	**[#x00]**	**\x00**	**\x00**	**\0**

---

3. POSIX space includes "Form Feed" and "Vertical Tab".
4. Perl \s includes form feed but not vertical tab. It also includes \x{85} (suspicious), \x{2028}, and \x{2029} if Unicode is in effect.
5. Java \s or \p{Space} is same as \s in POSIX.
6. JavaScript \s is same as \s in Perl with Unicode enabled.

single byte of hexadecimal *AB*	N/A	**[#x***AB***]**	**\x***AB*	**\x***AB*	**\x***AB*
single byte of octal code *377*	N/A	N/A	**\0***377*	**\0***377*	N/A
Unicode character of code *ABCD*	N/A	**[#x***ABCD***]**	**\x{***ABCD***}**	**\u***ABCD*	**\u***ABCD*

Notes

- Expressions marked ‡ cause security issue that (meta)data supplier can control or reveal unintended internal structure of pattern-matching software.
- Character sets marked ¶ includes non-latin letters and/or numbers in Unicode.
- Character sets marked § includes non-latin letters if under some locale and/or software configurations.

Appendix B: WMO FNC Characters at a Glance

Table 1: characters explicitly allowed (**bold, blue background**) and explictly prohibited (underline, orange background). Source: Manual on the GTS, Attachment II-15.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL	SOH	STX	ETX				BEL	BS	HT	LF	VT	FF	CR		
10												ESC				
20	SP	!	"	#	$	%	&	'	(	)	*	**+**	**,**	**-**	**.**	/
30	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	:	;	<	=	>	?
40	@	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
50	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	[	\	]	^	_
60	`	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**
70	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**	{	|	}	~	DEL