BIREME / PAHO / WHO

Latin American and Caribbean Center on Health Sciences Information

CISIS Formatting Language

Version 3.x

São Paulo - 2002-2006

CISIS Formatting Language

Card catalog

BIREME / PAHO / WHO (Brazil)

CISIS Formatting Language. / BIREME (org.). São Paulo : BIREME / PAHO / WHO, 2002-2006.

41 p.

1. User manual. 2. Information access. 3. Information systems. 4. Information management. 5. Public health. 6. Public Health services. I. BIREME II. Title

Warning - Any mention in this document to companies, institutions, persons or products are not an endorsement or recommendation given by BIREME / PAHO / WHO, thus it does not mean a preference to a similar one, cited or not.

BIREME / PAHO / WHO

Latin American and Caribbean Center on Health Sciences Information

Rua Botucatu, 862 - V. Clementino

This document was produced with the Documents Conformation Methodology (NorDoc) developed by BIREME.

Table of contents

Abbreviations used

- **ANSI**. American National Standards Institute.
- **ASCII**. American Standard Code for Information Interchange.
- **BIREME**. Latin American and Caribbean Center on Health Sciences Information.
- **BVS**. Biblioteca Virtual em Saúde (*see* VHL).
- **CGI**. Common Gateway Interface.
- **FST**. Field Selection Table.
- **HTML**. HyperText Markup Language.
- **HTTP**. HyperText Transfer Protocol.
- **ISO**. International Organization for Standardization.
- **MFN**. Master file number.
- **PAHO**. Pan American Health Organization.

- **STW**. STop Word file.
- **UNESCO**. United Nations Educational, Scientific and Cultural Organization.
- **URL**. Universal Resource Locator.

How to use this manual

This manual is organized in alphabetic order of command or function. It has also listed the names associated to each command or function in order to help user in the location of desired topics.

Each command/function has all information about its use described in a table format, containing syntax and samples.

The document is complemented by a glossary, a list of abbreviations and the bibliographic references to relevant documents.

Preface

About BIREME

Year after year, BIREME has been following its mission of being a center dedicated to scientific and technical health information for the region of Latin America and the Caribbean. Founded in Brazil in 1967, under the name of Regional Medicine Library (which the acronym BIREME comes from), it has always met the growing demand for up-to-date scientific literature from the Brazilian health systems and the communities of healthcare researchers, professionals and students. Then, in 1982, its name changed to Latin-American and Caribbean Center on Health Sciences Information so as to better express its dedication to the strengthening and expansion of the flow of scientific and technical health information across the region, but kept the acronym.

Networking, based on decentralization, on the development of local capacities, on sharing information resources, on developing cooperative products and services, on designing common methodologies, has always been the foundation of BIREME's technical cooperation work. It has been like this that the center established itself as an international model that fosters professional education with managerial and technical information with the adoption of information and communication paradigms that best meet local needs.

The main foundations that gave origin and which support the existence of BIREME are following:

- ✓ access to scientific and technical health information is essential for the development of health;
- ✓ the need to develop the capacity of Latin American and Caribbean countries to operate their sources of scientific-technical health information in a cooperative and efficient manner;
- ✓ the need to foster the use and to respond to the demands for scientific-technical health information from governments, health systems, educational and research institutions.

BIREME, as a specialized center of the Pan-American Health Organization (PAHO)/ World Health Organization (WHO), coordinates and conducts technical cooperation activities on the management of scientific information and knowledge with the aim of strengthening and expanding the flow of scientific health information in Brazil and in other Latin American and Caribbean countries as a key condition for the development of health, including its planning, management, promotion, research, education, and care.

The agreement that supports BIREME is renewed every five years by the members of the National Advisory Committee of the institution (PAHO, Brazilian Ministry of Health, Brazilian Ministry of Education and Culture, Secretary of Health of the State of São Paulo, and Federal University of São Paulo – Unifesp). The latter provides the physical infrastructure necessary for the establishment of the institution.

In 2004 the institution took on the responsibility of becoming a knowledge-based institution.

The Virtual Health Library (VHL)

With the rise and consolidation of the internet as the prevailing means of access to information and communication, BIREME's technical cooperation model evolved,

as of 1998, to build and develop the Virtual Health Library (VHL) as a common space for the convergence of the cooperative work of producers, intermediaries, and users of information. The VHL promotes the development of a network of sources of scientific and technical information with universal access on the internet. For the first time there has been a real possibility of equal access to health information.

To BIREME, the Virtual Health Library is a model for the management of information and knowledge, which includes the cooperation and convergence between institutions, systems, networks, and initiatives of producers, intermediaries, and users in the operation of networks of local, national, regional and international information sources favoring open and universal access.

Today, every country in Latin America and the Caribbean (Region) participates either directly or indirectly in the cooperative products and services offered by the VHL, which includes over 1,000 institutions in more than 30 countries.

The VHL is simulated in a virtual space of the internet formed by a collection or network of health information sources in the Region. Users of different levels and locations can interact and navigate in the space of one or many information sources, regardless of where they are. Information sources are generated, updated, stored and operated on the internet by producers, integrators, and intermediaries, in a decentralized manner, following common methodologies for their integration in the VHL.

The VHL organizes information in a structure that integrates and interconnects reference databases, specialist directories, events and institutions, a catalogue of the information resources available on the internet, collections of full texts with a highlight for the SciELO (*Scientific Electronic Library Online*) collection of scientific journals, selective information dissemination services, information sources to support education and decision-making, news, discussion lists, and support to virtual communities. The space of the VHL is, therefore, a dynamic and decentralized network of information sources based on which it is possible to retrieve and extract information and knowledge to support health decision-making processes.

The Virtual Health Library can be visualized as a distributed base of scientific and technical health knowledge that is saved, organized and stored in electronic format in the countries of the Region, universally accessible on the internet and compatible with international databases.

Structure of Reference Listing

<format specification>	**<name>**
Support:	If Standard, means that command/function has the same usage/result both in ISIS and CISIS. If CISIS is specified, means that it is available only in CISIS. Functions that are enhanced in CISIS, are presented with Standard/CISIS notation. (item always present)
Function type:	Displays the data type of a function return value. Possible values are: boolean, string and numeric. (item only applicable to functions)
Syntax:	Formal notation of a command/function usage. (item always present)
Definition:	Explanation of a command/function usage.
Components:	Explains additional features of a command/function.
Notes:	Elucidates particularities, restrictions and/or differences between ISIS and CISIS.
Examples:	1 Lists one or more examples of command/function usage.
See also:	Lists related commands and functions.

Commands and Functions

% " "" || () / /* @

 # unconditional newline

Support: Standard

Syntax: **#**

Definition: Begins a newline unconditionally.

Examples:

```
1   #,("address:"v3(9,9)+|; |#),
2   (|Name: |v1^n,c20,|Surname:|v1^s/),###,
```

See also: **/** [conditional newline]
% [reset blank line]

% reset blank line

Support: Standard

Syntax: **%**

Definition: Resets all previous blank lines, if any.

Examples:

```
1   |Name: |v1^n,c20,|Surname:|v1^s,###,%,/,
2   v10/#,v20/#,v30/#,%#,
```

See also: **/** [conditional newline]
[unconditional newline]

"string" conditional literal

Support: Standard

"string" **conditional literal**

Syntax: "<text>"<field selector>"<text>"
"<text>"<dummy field (selector)>
"<text>"<not present>

Definition: Outputs the text between the quotes only if <field selector>,<dummy field (selector)> or <not present> evaluates to TRUE. Both prefix and suffix literals can be placed together when using <field selector>, and data field content is also output.
If combined with <dummy field (selector)>, output is generated if data field is present. If combined with <not present>, outputs only if data field is absent.

Notes: <text> is produced only once, independing on repeatable fields.

Examples:

```
1  "Author: "v1^a,
2  "this text outputs if data field 10 is present"d10,
3  "this text outputs if data field 10 is absent"n5,
```

See also: **'string'** [unconditional literal]
|string| [repeatable conditional literal]
d [dummy field selector]
n [not present]
v [field selector]

'string' **unconditional literal**

Support: Standard

Syntax: '<text>'

Definition: Unconditionally outputs the text contained between a single quotes pair.

Notes: Unconditional literals may be placed anywhere in a format and can be passed as parameters to functions.

Examples:

```
1  this text will always output',
2  'Name: ',v1/,
```

See also: **"string"** [conditional literal]
|string| [repeatable conditional literal]

|string| **repeatable conditional literal**

Support: Standard

Syntax: |<text>|<+><field selector><+>|<text>|

Definition: Outputs the text between the vertical bars for each occurrence of a repeatable field, only if field selector evaluates to TRUE. Combined with a repeatable field, command behavior can be extended by using the <+> operator.
When <**+**> is present, the first prefix literal and/or the last suffix literal is not output.
e.g.:

```
(|; |+v1), /* prefix-literal */
(v1+|; |), /* suffix-literal */
```

|string| **repeatable conditional literal**

Notes: Prefix and suffix-literals can be used together, including with <+> operator. If a prefix or suffix-literal is used with <+> outside a repeatable group, literal contents may not be output as desired. If field is not repeatable, literal output occurs for the first and only occurrence of data field.

Examples:

1 (|; |+v1^s,|,|v1^n*0.1|.|),

2 (v10|: |, ,v11,| - |v12),

See also: **"string"** [conditional literal]
'string' [unconditional literal]
(format**)** [repeatable group]
v [field selector]

(format) **repeatable group**

Support: Standard

Syntax: **(**<format>**)**

Definition: Applies the format between the parenthesis to all occurrences of repeatable field(s) individually, or only once, if no repeatable field is present.

Notes: Repeatable groups cannot be nested.

Examples:

1 **(**|; |+v2^s/**)**,

2 **(**v1,c15,v2,c35,v3/**)**,

3 **(**if iocc<=3 then f(iocc,1,0),| - |v3/ else '-> more than 3'/, fi**)**,

See also: **|string|** [repeatable conditional literal]
v [field selector]

/ **conditional newline**

Support: Standard

Syntax: **/**

Definition: Begins a new line if not at the beginning of a line.

Notes: Several conditional newline commands (,/,/,/,/,) have the same effect of a single one.

Examples:

1 v1**/**,

2 v1**/**,v3**/**,v10**/**,mfn**/**,

3 s(v1,v3,v10)**/**,

See also: **#** [unconditional newline]
% [reset blank line]

/*string*/ **comment**

Support: CISIS

Syntax: **/*** <comment> ***/**

Definition: Encapsulates a comment.

Notes: Comments can span over several lines.

/*string*/ **comment**

Examples:

1 **/*** this is a single line comment ***/**,

2 **/*** this comment begins here
and ends here ***/**,

3 if a(v10) **/***and p(v20) ***/** then v20/ fi,

@ **include format file**

Support: CISIS

Syntax: **@**<filename>

Definition: Includes a format specification stored in a file in the current format.

Notes: <filename> may include drive and path information. Syntax of commands from the included file is evaluated when current format is executed. It is mandatory to separate <filename> with commas.

Examples:

1 **@**test.pft,v20,

2 s(**@**c:\temp\test.pft,v3),

3 if v1='L' then **@**large.pft, fi,

A

a(field selector) **absence check**

Support: Standard

Function type: Boolean

Syntax: **a(**<field selector>**)**

Definition: Returns TRUE if data field is absent, otherwise returns FALSE.

Notes: All components of field selector may be used, except indent.

Examples:

1 if **a(**v12**)** then v13 else v12, fi,

2 if **a(**v20^b**)** and p(v30) then v40/, fi,

See also: **p** function
v [field selector]

average value of expression

See: ravr function

B

break **conditional branching/quitting**

Support: CISIS

Syntax: **break**

Definition: Breaks the execution of a repeatable group format. When outside a repeatable group, quits the execution of the current format.

Notes: The execution resumes after the end of the repeatable group. When used inside a **ref function**, execution continues with the format after the function.

Examples:

```
1  (if iocc > 10 then '10+ occurrences'/,break else v5^n|-|,v5^s,/, fi,),
```

See also: **(**format**)** [repeatable group]

C

c **column**

Support: Standard

Syntax: **c**<int>

Definition: Skips to the specified column in the current or next line.

Examples:

```
1  'Name: ',c10,v1^n/,
2  if p(v1^s) then c10,v1^s/, fi,
```

See also: **x** [spacing]

cat(file) **dump file**

Support: CISIS

Function type: String

Syntax: **cat(**<format>)

Definition: Outputs the contents of a file whose name is generated by <format>.

Examples:

```
1  mfn,cat('myfile.html'),
2  cat('current document'/, ,if v10='c' then 'firstdoc.txt' else 'default.doc' fi),
3  cat(v101),
```

See also: **s(**expression**)**

comment

See: /*string*/

conditional literal

See: "string"

conditional newline

See: /

continue **repeatable conditional branching**

Support: CISIS

Syntax: **continue**

Definition: Executes the next occurrence of a repeatable group if at least one data field has a subsequent occurrence.

Notes:

Examples:

1 (if iocc = 1 then **continue** else v10/ fi),

2 (f(iocc,1,0),'=',v70,continue/),

See also: **(**format**)** [repeatable group]

current date

See: date, date function

D

d **dummy field selector**

Support: Standard

Syntax: **d**<field tag><subfield>

Definition: Outputs the conditional literal prefix if a data field or subfield exists. Used in conjunction with conditional literals.

Notes: Dummy field (selector) does not return a value.
Known bugs: When in a repeatable group, <subfield> is evaluated only for the first occurrence of data field.

Examples:

1 "this text outputs if data field 10 exists"**d**10,

2 "Name: "v20(5,5)/, ,"Name: "n20,v21(5,5)/,

See also: **"string" [conditional literal]**
n [not present]
v [field selector]

date date(keyword) **current date**

Support: CISIS

Function type: String

Syntax: **date**
date(<keyword>)

date date(keyword) **current date**

Definition: Outputs current system date. Used without parameters, returns:
yyyymmdd hhmmss w nnn
where:
aaaa = year
mm = month
dd = day
hh = hour
mm = minute
ss = second
w = day of the week (0-6)
nnn = number of elapsed days from Jan 1st.

Components: keywords **DATETIME** and **DATEONLY**

Notes: **DATETIME** shows system date in european format plus current time (dd/mm/yy hh:mm:ss) while **DATEONLY** displays the same output without time information.

Examples:

1 'Today is ',**date**,

2 'Current date: ',**date(DATEONLY)**/, ,'Current time: ',mid(**date(DATETIME)**,10,8)/,

display file

See: cat function

dummy field (selector)

See: d

E

environment variable set

See: putenv function

F

f(num expr,length,decimals) **format value**

Support: Standard

Function type: String

Syntax: **f(**<format>,<expr-1>,<expr-2>**)**

f(num expr,length,decimals) format value

Definition: Converts a numeric value to string. <format> is the numeric expression to be converted. <expr-1> and <expr-2> are optional and determine the minimum output length and the number of decimal places respectively.

Notes: If <format> is not a valid numeric expression, an error is issued. If <expr-2> is set, <expr-1> must also be placed or a syntax error is issued. If only <expr-1> is defined, the result is output in scientific exponent notation. If the number of characters required to represent <format> is greater than <expr-1>, additional positions are provided automatically. If <expr-1> is missing, a width of 16 characters is assumed.

Examples:

1 **f(**val(v1),2**)**,

2 **f(**((3+5)/2)+1,4,2**)**,

3 **f(**v2**)**,

See also: **val** function

field selector

See: v

field update

See: proc function

find string

See: instr function

format value

See: f function

G

getenv(expression) get environment variable

Support: CISIS

Function type: String

Syntax: **getenv(**<format>**)**

Definition: Returns the value of an environment variable.

Notes: If <format> does not generate a valid environment variable name, no value is returned.

Examples:

1 'Current path: ',**getenv(**'PATH'**)**,

2 (v1|=|,**getenv(**v1**)**/),

See also: **putenv** function

I

if ... then ... else ... fi **conditional flow control**

Support: Standard

Syntax: **if** <bool expr> **then** <format-1> [**else** <format-2>] **fi**

Definition: Executes a block of formatting language specifications (<format-1>) if <bool expr> evaluates to TRUE. Allows the execution of another block of formatting language specifications (<format-2>) by using the clause **else** when <bool expr> evaluates to FALSE.

Notes: Clause **then** precedes the first block of formatting language specifications. **else** is optional and if present in the code, must be followed by a format. The **fi** clause must always end the command scope and if missing, a syntax error is issued. The **if...fi** command can span over several lines, therefore it is recommended the use of indentation.

Examples:

1 **,if**instr(v5,'ab')>0 **then** ,v5/, **fi,**

2 **,if** p(v10) **then** ,|Title: |v3, **else** ,|Alternate title: |v4, **,fi,**

include format file

See: @

instr(string1,string2) **find string**

Support: CISIS

Function type: Numeric

Syntax: **instr(**<format-1>,<format-2>**)**

Definition: Returns a number specifying the starting position of the string generated by <format-2>, found in string settled by <format-1>. If there is no match, return value is zero.

Notes: Both <format-1> and <format-2> must generate strings, otherwise a syntax error occurs. The use of **s function** may help in cases where a complex string is required as parameter.

Examples:

1 if **instr(**v5,'ab'**)**>0 then v5/, fi,

2 if **instr(**s(|'|v1|'|)**,**v5**)**>0 then v1, fi,

3 left(v18,**instr(**v18,'.'**)**-1),

iocc **occurrence index**

Support: CISIS

Function type: Numeric

Syntax: **iocc**

iocc **occurrence index**

Definition: Returns the occurrence index number (starting from 1), otherwise returns zero.

Examples:

```
1  ("Author: "v1/, ,if iocc > 3 then 'et all',break, fi),
2  (f(iocc,3,0),|.|v10/),
```

See also: **nocc** function

K

key lookup

See: l function

key postings

See: npost function

keyword

See: v, date / date function

L

l(key) l([inverted file],key) **key lookup**

Support: Standard/CISIS

Function type: Numeric

Syntax: **l(**<format key>**)**
l([<format ifname>]<format key>**)**

Definition: Returns the **MFN** of the first posting (if any) by using the key generated by <format key> to search the current inverted file. It can also seek in a specific inverted file defined by <format ifname>.

Notes: Keys are converted to upper case before the expression is seek. Default display mode used by lookup is mpl. If a different mode is specified in the FST file, key must also use the appropriate mode. If key is not found, function returns zero. The <format ifname> parameter must evaluate to a string having a valid inverted file name, otherwise a syntax error is issued. This is often used in conjunction with ref function to allow output of data fields from a different record.

Examples:

```
1  if l(v15)<> 0 then |Term: |v15, fi,
2  ref(l(['books']v1,'-',v2),v10/),
```

l(key) l([inverted file],key) **key lookup**

See also: **ref** function

left(string,length) **left substring**

Support: CISIS

Function type: String

Syntax: **left(**<format-1>,<format-2>**)**

Definition: Returns a new string, containing the leftmost characters from of the original string generated by <format-1>. <format-2> specifies the actual number of characters to be read from <format-1> starting from left to right.

Notes: If the string generated by <format-2> is greater than the size defined by <format-1>, function returns the <format-1> string. If <format-2> is zero or is set to a negative number, returns a NULL string.

Examples:

```
1  if left(v1^n,2)='Ma' then v1^n/, fi,
2  left(v1,instr(v1,'.')-1),
```

See also: **right** function
mid function

lw(number) **set line width**

Support: CISIS

Function type: Numeric

Syntax: **lw(**<int>**)**

Definition: Sets the output line width to <int> characters.

Notes: Output line width default depends on the application program

Examples:

```
1  if size(v10) > 76 then lw(254), fi,
2  lw(70),v20/,lw(10),v30/,
```

M

master file name

See: mstname function

maximum value of expression

See: rmax function

mdl, mdu, mhl, mhu, mpl, mpu **mode**

Support: Standard

Syntax: **m**<mode><conv>

Definition: Sets a new displaying mode for the current output.

mdl, mdu, mhl, mhu, mpl, mpu **mode**

Notes: Default mode is mpl. <mode> represents the desired mode to be set. <conv> specifies whether upper case conversion is set. Mode may appear several times in a format and its formatting effect is active until a new mode is set.

<mode> can be as follows:
p = proof: fields are displayed as they are stored in records.
h = heading: control characters and descriptor delimiters are ignored, except subfield descriptor, which are translated to punctuation.
d = data: similar to heading mode, appends a full stop at the end of data fields, followed by two spaces.

<conv> can be as follows:
u: converts data to upper case
l: leaves data unchanged

Examples:

```
1  mpl,"First author: "v10[1]/,
2  mpu,"Second author: "v10[2]/,
3  mdl,"Third author: "v10[3]/,
```

mfn mfn(length) **record number**

Support: Standard

Function type: String or numeric

Syntax: **mfn**
mfn(<int>**)**

Definition: Returns the master file number of a record.

Notes: An integer value can be passed as parameter to set the return value size. **mfn** returns the appropriate return type according to the format requirements.

Examples:

```
1  'Record: ',mfn(3)/,
2  if mfn > 2 then mfn/, fi,
3  ref(mfn-1,v2/),
```

See also: **ref** function
l function

mid(string, start, length) **substring**

Support: CISIS

Function type: String

Syntax: **mid(**<format-1>,<format-2>,<format-3>**)**

Definition: Returns a new string, containing a specified number of characters from the original string (<format-1>). <format-3> gives the actual number of characters to be read from <format-1> and <format-2> gives the start position in string where extraction begins.

mid(string, start, length) **substring**

Notes: If <format-2> is greater than <format-1> size, function returns a NULL string. If <format-2> is zero or is set to a negative number, default is 1.

Examples:
1 **mid(**v2,2,80**)**,
2 **mid(**v1,instr(v1,'key'),size(v1)**)/**,

See also: **right** function
left function

minimum value of expression

See: rmin function

mode

See: m...

mstname **master file name**

Support: CISIS
Function type: String
Syntax: **mstname**
Definition: Returns the current master filename.
Examples:
1 'Current database: ',**mstname**/,
2 ref(['names']l(['names']'X39BJ'), ,'Database now is ',**mstname**/),

N

n **not present (dummy field selector)**

Support: Standard
Syntax: **n**<field tag>
Definition: Tests the absence of a data field. Used in conjunction with conditional literals.
Notes: As a dummy (field) selector, it does not return a value.
Examples:
1 "this text outputs if data field 10 is absent"**n**10,
2 "Author:"v10/, ,"Author: "**n**10,v20/,

See also: **"string" [conditional literal]**
d [dummy field selector]
v [field selector]

newline(string) **set newline**

Support: CISIS
Function type: String

newline(string) **set newline**

Syntax: **newline(**<format>**)**

Definition: Sets and/or resets default CR/LF pair with character(s) from resulting <format>.

Notes: <format> may even contain reserved escape sequences such as:
\r - carriage return
\n - line feed
Subsequent \ commands will be automatically replaced by resulting format until a newline sets a new character or string.

Examples:
```
1  newline(if v151='unix' then '\n' else '\r\n' fi,
2  newline(v301),
3  newline('<BR>'),
```

See also: **/** [conditional newline]
[unconditional newline]

nocc(field) **number of occurrences**

Support: CISIS

Function type: Numeric

Syntax: **nocc(**<field selector>**)**

Definition: Returns the number of occurrences of a data field or subfield defined by <field selector>.

Notes: <field selector> means only field and subfield in this function scope. All other field selector components, if used, issue a syntax error.

Examples:
```
1  if nocc(v3)> 10 then 'Too many occurrences.'/, fi,
2  'There are ',f(nocc(v20),2,0),' authors.'/,
```

See also: **iocc** function
v [field selector]

not present

See: n

npost(key) npost([inverted file],key) **key postings**

Support: CISIS

Function type: Numeric

Syntax: **npost(**<format key>**)**
npost([<format>],<format key>**)**

Definition: Returns the total postings for a key (given by <format key>) in an inverted file. <format> if defined, must generate a string containing the inverted file name. <format key> settles the key to be searched in the inverted file.

Examples:
```
1  if npost(v1)> 1 then 'duplicate key ',v1,' found'/, fi,
2  'There are ',f(npost(v20),3,0),'keys for ',v20,'. '/,
```

See also: **l** function

number of occurrences

See: nocc function

O

occurrence index

See: iocc function

P

p(field selector) **presence check**

Support: Standard

Function type: Boolean

Syntax: **p(**<field selector>**)**

Definition: Returns TRUE if data field is present, otherwise returns FALSE.

Notes: All field selector components can be used, except indent.

Examples:

1 if **p(**v12**)** then v12 else v13, fi,

2 if **p(**v50^a**)** and **p(**v50^b**)** then v50^a/,v50^b/, fi,

See also: **a** function
v [field selector]

proc(field update format) **field update**

Support: CISIS

Function type: String

Syntax: **proc(**<fldupd format>**)**

Definition: Appends or replaces data fields in the current record. <fldupd format> is a format that generates the field update specific commands to be executed by the function.

proc(field update format) **field update**

Notes: A CISIS field update specification is a string of **d** (delete) and **a**(add) and **h** (add) commands to be applied in the current record. All **d** (delete) commands must preceed the add commands.
The following commands are available:
d* - deletes all present fields
d<field tag> - deletes all occurrences of <field tag>
d<field tag>**/**<occ> - deletes occurrence <occ> of <field tag>
a<field tag>**#<**string>**#** - adds <string> as a new occurrence of <field tag>
h<field tag> <n> <string> - adds <string>, <n> bytes long, as a new occurrence of <field tag>
The **#** delimiter may be any non-numeric character.
A space must be provided between the <field tag>, <n> and <string> parameters of the **h** command.

Examples:

1 **proc(**'d70',|a10#|v70|#|**)**,

2 **proc(**if v24*0.4 = 'Tech' then 'd*', fi**)**,

putenv(expression) **environment variable set**

Support: CISIS

Function type: String

Syntax: **putenv(**<format>**)**

Definition: Sets an environment variable at the operating system level with its corresponding value.

Notes: The variable is available only within the scope of current process.

Examples:

1 **putenv(**'TEST=test'**)**,getenv('TEST'),

2
```
set CIPAR=somefile
set
mx null "pft=putenv('CIPAR=another'),getenv('CIPAR')/"
set
```

See also: **getenv** function

R

ravr(string) **average value of expression**

Support: Standard

Function type: Numeric

Syntax: **ravr(**<format>**)**

Definition: Returns the average value of a given format. <format> must generate a string expression.

Notes: Can be used to compute the average of numeric values in repeatable fields.

Examples:

1 f(**ravr(**s(v8,x1,v1)**)**,3,0),

ravr(string) **average value of expression**

```
2  f(ravr(v1,x1,v2),5,2),
3  f(ravr('8;15;16.73'),3,2),
4  if ravr(v20|;|)>=5 then 'pass'/ else 'fail'/, fi,
```

See also: **rmin** function
rmax function
rsum function

record number

See: mfn, mfn function

ref(mfn, format) ref([master file]mfn, format) **record reference link**

Support: Standard/CISIS

Function type: String

Syntax: **ref(**<expr>,<format>**)**
ref([<format dbname>]<expr>,<format>**)**

Definition: Executes <format> with the record selected by <expr>. If <format dbname> is set, another (or the same) database can be referenced and a different record is selected.

Notes: <expr> can be any format returning the MFN of a record. **l** function can be used to perform a seek and return the MFN.

Examples:

```
1  ref(l(v3),v1/,v2/,v3/),
2  if ref(['account']l(['user']v2),v4)='active' then |Name: |v10/, fi,
3  (if p(v99) then ref([v99]1,v30/), fi),
```

See also: **l** function

repeatable conditional literal

See: |string|

repeatable group

See: (format)

replace(string1, string2, string3) **replace**

Support: CISIS

Function type: String

Syntax: **replace(**<format-1>,<format-2>, <format3**)**

Definition: Returns a new string, after replacing <format-2> with <format-3>.

Notes: If <format-2> is a null string or is not found in <format-1), function returns <format-1> string.
If <format-3> is null, <format-2> string will be excluded from <format-1>.
replace is case sensitive for both search string (<format-2>) and replace string (<format-3>).

replace(string1, string2, string3) **replace**

Examples:

```
1  replace('Mary And John','And','and')/,
2  if replace(v1^a,'01x','01X')= '894501X' then v1^n/, fi,
3  replace(s(v304,v333),',',', ')/,
4  replace(s(if v415='spanish' then v299 else 'none' fi),v1,v759)/,
```

reset blank line

See: %

right(string, length) **right substring**

Support: CISIS

Function type: String

Syntax: **right(**<format-1>,<format-2>**)**

Definition: Returns a new string, containing the rightmost characters of the original string (<format-1>). <format-2> gives the actual number of characters to be read from <format-1> starting from right to left.

Notes: If <format-2> is greater than <format-1> size, function returns <format-1> string. If <format-2> is zero or is set to a negative number, returns nothing.

Examples:

```
1  if right(v1^n,1) = 'r' then v1^n/, fi,
2  right(v65,4)/,
```

rmax(string) **maximum value of expression**

Support: Standard

Function type: Numeric

Syntax: **rmax(**<format>**)**

Definition: Returns the maximum value of a given format. <format> must generate a string expression.

Notes: Can be used to compute the maximum of numeric values in a repeatable field.

Examples:

```
1  f(rmax('72,54,2'),2,0),
2  f(rmax(v1,x1,v4,x1,(v8|,|)),5,2),
3  if rmax(v40|;|)>val(v41) then 'Limit of ',v41,'exceeded.'/, fi,
```

See also: **rmin** function
ravr function
rsum function

rmin(string) **minimum value of expression**

Support: Standard

Function type: Numeric

Syntax: **rmin(**<format>**)**

rmin(string)	**minimum value of expression**
Definition:	Returns the minimum value of a given format. <format> must generate a string expression.
Notes:	Similar to **rmax function**, rmin can compute the minimum of numeric values in a repeatable field.
Examples:	1 f(**rmin(**'10;2;5;4;-2'**)**,2,0), 2 f(**rmin(**v1,x1,v2,x1,'44'**)**,4,2), 3 if **rmin(**v80||,v90| |,v100| |**)** < 1990 then 'Wrong decade.'/, fi,
See also:	**rmax** function **ravr** function **rsum** function

rsum(string)	**sum of expression**
Support:	Standard
Function type:	Numeric
Syntax:	**rsum(**<format>**)**
Definition:	Returns the sum of a given format. <format> must generate a string expression.
Notes:	Similar to **rmax** and **rmin functions**, rsum computes the sum of numeric values in a repeatable field.
Examples:	1 f(**rsum(**'102,45,-37'**)**,2,0), 2 f(**rsum(**v1,x1,v3,x1,f(val(v8)+2)**)**,4,2), 3 if **rsum(**v20^d**)**>1000 then 'Aborted.'/ else 'OK'/, fi,
See also:	**rmax** function **ravr** function **rmin** function

S

s(expression)	**string**
Support:	Standard
Function type:	String
Syntax:	**s(**<format>**)**[command component]
Definition:	Returns the concatenation of string expressions generated by <format>.
Components:	extraction
extraction:	Extracts partial content of the resulting string. <offset int> is the first position to start extraction, while <length int> determines how many characters will be extracted. If <length int> is omitted or is greater than the resulting string, the default is the end of the resulting string.
Notes:	Can be passed to functions that require a string expression as parameter.

s(expression) **string**

Examples:

1 if **s(**v1,v2,v3**)**:'ABCDE' then **s(**v1,v2,v3**)***0.50, fi,

2 if **s(**|*|v5|*|**)**:**s(**'*E*'**)**then 'English'/, fi,

See also: **v** [field selector]

select ... case ... elsecase ... endsel **conditional branch control**

Support: CISIS

Syntax:
select <format expr>
case <option-1>: <format-1>
case <option-2>: <format-2>
case <option-n>: <format-n>
[**elsecase** <format-0>]
endsel

Definition: Evaluates <format expr> and compares the result to each **case** option (<option-1>, <option-2>...<option-n>). If an option matches <format expr>, the appropriate block of formatting language specifications is executed (<format-1>, <format-2>...<format-n>), otherwise **elsecase**clause (if defined) is executed (format-0).

Notes: <format expr> must generate a string or numeric value. If <format expr> evaluates to string, all option values in **case** clauses must be of string type, otherwise, if <format expr> is numeric, option values must be also numeric.

Examples:

1
```
select s(v5)
case '1': ,f(val(v5)/2,2,2)/,
case '2': ,v5/,
case '3': ,v6,'-',v1/,
elsecase ,|Error in field v5 = |v5/,
endsel,
```

2
```
select nocc(v7)
case 0: 'absent'/,
case 1: 'one occurrence'/,
case 2: 'two occurrences'/,
elsecase 'more than 2 occurrences'/,
endsel,
```

See also: **if ... then ... else ... fi**

set line width

See: lw function

size(string) **string size**

Support: CISIS

Function type: Numeric

Syntax: **size(**<format>**)**

Definition: Returns the string size.

Notes: <format> must return a string or a syntax error occurs.

Examples:

1 if **size(**v10**)**> 76 then lw(254), fi,

size(string) **string size**

2 f(**size(**v10,v20),1,0**)**,

spacing

See: x

string

See: s function

string size

See: size function

string to value

See: val function

string type

See: type function

substring

See: mid function

sum of expression

See: rsum function

system(expression) **system call**

Support: CISIS

Function type: String

Syntax: **system(**<format>**)**

Definition: Executes the argument produced by <format> as an operating system command.

Notes: <format> must generate a string containing the code to be executed. The possible output from the command is sent directly to the standard output.

Examples:

1 **system(**'dir'**)**,

2 if p(v2) then **system(**'type ',v2**)**, fi,

T

type(string) **string type**

Support: CISIS

type(string) **string type**

Function type: String

Syntax: **type(<format>)**

Definition: Returns the string type as follows:
A - if string contains only alphabetic characters (according to a default alphabetic character table, like ISISAC.TAB) or space
N - if string contains only numeric characters (0-9)
X - for all other cases

Notes: Format must generate a string or a syntax error is issued.

Examples:

1 if **type(**v1**)**='N' then f(val(v1),3,2)/ else v1/, fi,

2 if s(**type(**v1**)**,**type(**v2**)**,**type(**v3**))**<>'AAA' then 'Invalid character type detected'/, fi,

U

unconditional literal

See: 'string'

unconditional newline

See: #

V

v **field selector**

Support: Standard

Syntax: **v**<field tag>[command components]

Definition: Outputs data field contents. Content can be selected, restricted, narrowed, extracted or indented by using command components (see below). **v** stands for variable length field.

Components: subfield, occurence, extraction and indent

syntactic order: **^**<subfield id> **[**<index>**[..**<upper index>**]] ***<offset int>**.**<length int> **(**<first line int>**,**<next line int>**)**

subfield: Restricts the output to the contents of a subfield. If data field exists but subfield is not present, no output is generated.

v	**field selector**
occurrence:	Narrows the output to one or a range of occurrences of a repeatable field. <index> and <upper index> refer to the first (or unique) and last occurrences, respectively. If the specified <index> is greater than the actual number of occurrences, no output is generated. The same occurs if data field is not repeatable and <index> is set to a number equal or greater than 2. However, if <index> is set to 1 and it is used in a non-repeatable field, content is normally output. This component must be used outside a repeatable group; otherwise, <upper index> is ignored. If double dot (**..**) is used and <upper index> is missing **LAST** is assumed. The **LAST** keyword is set with the value of total occurrences of a data field.
extraction:	Extracts partial content of a data field, subfield or occurrence. <offset int> is the first position to start extraction, while <length int> determines how many characters will be extracted. If <length int> is omitted or is greater than field length, the default is the end of data field.
indent:	Aligns the output of data field, subfield, occurrence or extracted content, according to <first line int> (alignment for the first line) and <next line int> (alignment for successive lines). Both values are numeric constants. If current line position differs from zero, indentantion is disabled.
Notes:	The behavior of the field command depends on the component(s) used. No output is generated when data field is absent or when component performs a restriction or extraction that is out of data boundary.
Examples:	1 **v**2/,**v**3^a| - |,**v**1/, 2 **v**1^n*0.3, 3 (|; |+**v**3^s)/, 4 **v**20**[**4**]**, 5 **v**10**[**2..7**]**/, 6 **v**5**[**3**..]**/,/* equals to ,**v**5**[**3**..LAST]**, */ 7 **v**1[**LAST**]*****2**.**7/, 8 **v**1**(**5,5**)**/, 9 |Title: |**v**1^n**(**5,5**)**/,
See also:	**"string"** [conditional literal] **d** [dummy field selector] **n** [not present] **|string|** [repeatable conditional literal] **(**format**)** [repeatable group]

val(string)	**string to value**
Support:	Standard
Function type:	Numeric
Syntax:	**val(**<format>**)**
Definition:	Returns the numeric value of the argument generated by <format>.

val(string)	**string to value**
Notes:	If <format> produces only non-numeric characters, function returns zero. If more than one numeric value is encountered, only the first one is returned.
Examples:	1 if **val(**v2**)**>5 then 'Error'/ else 'OK'/, fi, 2 f(**val(**v2**)**/3,4,2),

X

x	**spacing**
Support:	Standard
Syntax:	**x**<int>
Definition:	Inserts the number of spaces defined by <int> before the next block of formatting language specifications is output.
Notes:	If <int> is greater than the available space in the current line, it skips to the next line.
Examples:	1 'Name: ',**x5**,v1^n/, 2 (v1,**x3**,v2,**x8**,v3/),
See also:	**c** [column]

Bibliographic references

1. UNESCO. *Mini-micro CDS/ISIS:* Reference manual (version 2.3). Organized by Giampaolo Del Bigio. Paris: United Nations Educational, Scientific and Cultural Organization, 1989. 286 p. ISBN 92-3-102-605-5.

2. BUXTON, Andrew, HOPKINSON, Alan. *The CDS/ISIS for Windows Handbook* [online]. Paris: United Nations Educational, Scientific and Cultural Organization, 2001 [cited 30 August 2006]. 164 p. Available from internet: <http://bvsmodelo.bvs.br/download/winisis/winisis-handbook-en.pdf>.

3. SUTER, Tito. "Prehistoria" e historia del MicroISIS [online]. In: *Manual para instructores de Winisis*. Buenos Aires: Centro Atómico Constituyentes (CAC), Comisión Nacional de Energía Atómica (CNEA), 1999 [citado el 30 Agosto 2006]. p. 21-26. Disponible en internet: <http://www.cnea.gov.ar/cac/ci/isis/isidams.htm>.

Glossary

- **Application**. Program used to execute tasks in connection with an application, such as the creation or edition of texts, drawings, animations, layout, etc. E.g.: text processor, database manager, Internet browser, etc.
- **Backup**. Procedure used to duplicate one or more files and/or directories in another storing device (tape or disc), thus producing a backup copy that may be restored in the event of accidental deletion or physical damage to the original data.
- **Bibliographic Database**. Electronic version of a catalog or bibliographic index.
- **Browser**. Internet page navigator, such as Internet Explorer and Netscape Navigator.
- **CDS/ISIS – MicroISIS**. Software program developed and maintained by UNESCO to manage bibliographic data.

- **CGI**. The Common Gateway Interface is a standard for interfacing external applications with information servers, such as HTTP or Web servers.

- **Database**. Collection of data that are structured to be easily accessed and handled. It is formed by units called records whose attributes are represented by fields. For example, in a file called "customer base", each customer is a record, with several fields such as "NAME", "CUSTOMER CODE", "TELEPHONE" etc.

- **Field**. Record element that provides storage of specific information. See Database.

- **File**. In computing, a set of data that may be saved into some type of storing device. The data files are created by applications, such as a text processor for example.

- **ISO Format (of files)**. Standard established by the ISO to allow the exchange of data between institutions, networks and users.

- **LILACS Format**. A bibliographic description format established by BIREME, based on the UNISIST Reference Manual for Machine-readable Bibliographic Descriptions.

- **Posting**. It is the address of a key extracted from the master file.

- **Presentation format**. Set of commands that defines the data output of an ISIS database.

- **Record**. Set of structured data aimed to store a specific subject. *See Database*.

- **Subfield**. Element that contains the tyniest piece of information in a field, whose meaning may be unclear if it is not analysed outside the scope of a group of elements.

- **UNISIST**. Intergovernmental program designed to foster cooperation in the field of scientific and technological knowledge.

- **URL**. Standard defined for the addressing of data contents via the TCP/IP protocol. Internet browsers use the URL to access Web pages.