

BIREME / PAHO / WHO

Latin American and Caribbean Center on Health Sciences Information

IsisScript Language Reference

Version 1.1

São Paulo - 2006

Copyright © 2006 - BIREME / PAHO / WHO

IsisScript Language Reference

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Card catalog

BIREME / PAHO / WHO (Brazil)

IsisScript Language Reference. / BIREME (org.). São Paulo : BIREME / PAHO / WHO, 2006.

69 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

Table of contents	I
Abbreviations used	V
How to use this manual	VII
Preface	1
About BIREME	1
The Virtual Health Library (VHL)	2
IsisScript Reference	5
<IsisScript>	5
<i>IsisScript.name</i>	5
<call>	6
<i>call.name</i>	6
<cgitable>	7
<define>	7
<display>	8
<do>	8
<i>do.task</i>	9
do.task=fullinversion	9
do.task=import	10
do.task=invertedload	10
do.task=keyrange	10
do.task=list	11
do.task=mastersort	12
do.task=mfnrange	12
do.task=search	13
do.task=update	13
<export>	14
<extract>	14
<field>	15
<i>field.action</i>	16
field.action=add	16

field.action=cgi	16
field.action=define	16
field.action=delete	17
field.action=export	17
field.action=hl	18
field.action=import	18
field.action=occ	18
field.action=replace	19
field.action=statusdb	19
field.action=statusfile	19
<i>field.from</i>	20
<i>field.previous</i>	20
field.previous=add	20
field.previous=delete	20
<i>field.split</i>	21
field.split=flddir	21
field.split=occ	21
<i>field.tag</i>	22
field.tag=list	22
<i>field.type</i>	22
field.type=flag	22
<file>	23
<i>file.action</i>	23
file.action=append	23
file.action=close	24
file.action=create	24
file.action=delete	24
file.action=unlock	24
<i>file.type</i>	25
file.type=database	25
file.type=file	25
file.type=inverted	25
file.type=master	25
file.type=output	26
file.type=tempfile	26
<flow>	26
<i>flow.action</i>	27
flow.action=exit	27
flow.action=jump	27
flow.action=skip	28
<function>	28
<i>function.action</i>	28
<i>function.from</i>	29
<i>function.name</i>	29
<i>function.split</i>	29
<i>function.tag</i>	30
<hl>	30
<htmlpft>	30
<i>htmlpft.action</i>	31
htmlpft.action=convert	31
<i>htmlpft.type</i>	31

<label>	32
<list>	32
<i>list.action</i>	33
list.action=delete	33
list.action=load	33
<i>list.type</i>	33
list.type=freq	33
list.type=list	34
list.type=sort	34
<loop>	34
<parm>	35
<i>parm.name</i>	35
parm.name=actab	36
parm.name=bufferize	36
parm.name=cipar	36
parm.name=count	37
parm.name=db	37
parm.name=decod	38
parm.name=delimiter	38
parm.name=expire	38
parm.name=expression	39
parm.name=file	39
parm.name=freqsum	39
parm.name=from	40
parm.name=fst	40
parm.name=gizmo	40
parm.name=indexlist	41
parm.name=key	41
parm.name=keyfield	42
parm.name=keylength	42
parm.name=keys	42
parm.name=keybdb	43
parm.name=lockid	44
parm.name=maxlk	44
parm.name=mfn	44
parm.name=posting	45
parm.name=posttag	45
parm.name=prefix	46
parm.name=reset	46
parm.name=reverse	47
parm.name=sort	47
parm.name=stw	47
parm.name=suffix	48
parm.name=task	48
parm.name=to	48
parm.name=type	49
parm.name=uctab	49
<i>parm.tag</i>	50
<i>parm.type</i>	50
parm.type=check	50
<pft>	51

<i>pft.type</i>	51
<i>pft.type=check</i>	51
<i>pft.type=reload</i>	52
<proc>	52
<return>	52
<i>return.action</i>	53
<i>return.split</i>	53
<i>return.tag</i>	53
<i>return.tag</i>	54
<section>	54
<i>section.name</i>	55
<trace>	55
<update>	55
<write>	56
Bibliographic references	57
Glossary	58

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 one single chapter containing all commands, parameters and options available in the IsisScript language.

The commands are organized in Level 2 heading, and their elements follows the hierarchy until Level 4.

Each command, parameter or option will contain a brief description of its purpose. Additional information such as attributes, syntax, elements contained and where it is applicable are also provided and it completes with a sample code illustrating the command, option or parameter.

Knowledge of formatting language and CDS/ISIS database model and structure are the only requirements to the user.

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.

IsisScript Reference

<IsisScript>

1. It may contain: <function> <section> <trace>
2. Attributes: name
3. Syntax: <IsisScript> ... </IsisScript>

The element **<IsisScript>** is used to indicate a group of instructions IsisScript.

The attribute *name* is used to name and document the group.

Example

```
<IsisScript name>HelloWorld>
<section>
<display>Hello World!</display>
</section>
</IsisScript>
```

IsisScript.name

1. It may be used in: <call> <function> <IsisScript> <section>
2. Syntax: name=...

The attribute **name** is optional, it is used just for the purpose of IsisScript documentation.

Example

```
<IsisScript name>HelloWorld>
<section>
```

```

<display>Hello World!</display>
</section>
</IsisScript>

```

<call>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: name
4. Syntax: <call> ... </call>

The element **<call>** indicates the call of a function. The function to be called is specified by the attribute *name*.

The argument of the element **<call>** is passed as a parameter for the function.

Example

```

<IsisScript>
<function name="First">
<display>FIRST </display>
</function>
<function name="Second">
<display>SECOND </display>
</function>
<function name="ParamTest" action="replace" tag="1" split="occ">
<display><pft>##'ParamTest'</pft></display>
<display><pft>ALL</pft></display>
<return action="replace" tag="9999" split="occ"><pft>(v1/)</pft></return>
</function>
<section>
<call name="First">now</call>
<call name="Second">now</call>
<call name="ParamTest"><pft>'One' / 'Two' </pft></call>
<display><pft>ALL</pft></display>
</section>
</IsisScript>

```

call.name

1. It may be used in: <call> <function> <IsisScript> <parm> <section>
2. Syntax: name=...
Name of the function to be called.

Example

```

<IsisScript>
<function name="First">
<display>FIRST </display>
</function>
<function name="Second">
<display>SECOND </display>

```

```

</function>
<function name="ParamTest" action="replace" tag="1" split="occ">
<display><pft>## 'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action="replace" tag="9999" split="occ"><pft>(v1/)</pft></return>
</function>
<section>
<call name="First">now</call>
<call name="Second">now</call>
<call name="ParamTest"><pft>'One' / 'Two' /</pft></call>
<display><pft>ALL</pft></display>
</section>
</IsisScript>

```

<cgitable>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <cgitable> ... </cgitable>

For each line of the argument specified, you should add to the specified field the CGI content ("value") corresponding to the specified "name". Each argument line is equivalent to the definition <field action="cgi" tag="...">name</field>

Example

```

...
<cgitable>
<pft>'2001 db' /
'2002 from' /
ref(['CONFIG' ]1, (v1^t,x1,v1^n/)) /
</pft>
</cgitable>
...

```

<define>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <define> ... </define>

Defines fields that will be used inside the element <loop>. Each line is equivalent to the definition <field action="define" tag="...">Isis_...</field>

Example

```

...
<define>
<pft>'1001 Isis_Current' /

```

```
'2002 Isis_Total'/
</pft>
</define>
...
```

<display>

1. It may contain: <htmlpft> <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <display> ... </display>
Sends a text to the current output. IsisScript starts having as current output the "standard output" of the operational system.
Output redirection is informed through the element <file>.

Example

```
<IsisScript>
<section>
<display><pft>'Content-type: text/html' /#</pft></display>
<display>Hello World!<br></display>
<field action=cgi tag=100>^n^v</field>
<display><pft>( |100=|v100|<br>|/)</pft></display>
</section>
</IsisScript>
```

<do>

1. It may contain: <call> <cgitable> <define> <display> <do> <export> <extract> <field> <file> <flow> <hl> <label> <list> <loop> <parm> <proc> <return> <trace> <update>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: task
4. Syntax: <do> ... </do>
The element <do> informs the beginning of an IsisScript task. Possible tasks include: record interval, key interval, search, repetition, list, records import, database sorting, inverted file loading and complete inverted file generation. The task of the list can be frequency, sorting or simple.
The attribute *task* is used to inform the type of task to be executed.
The parameters necessary for task execution are informed through the element <parm>.
The element <loop> informs the block of instructions to be executed for each task item.

Example

```
...
```

```

<do task="search">
<parm name="db">CDS</parm>
<parm name="expression">plants * water</parm>
<loop>
<display><pft>mfN/</pft></display>
</loop>
</do>
...

```

do.task

1. **Options:** fullinversion import invertedload keyrange list mastersort mfnrange search update
2. **It may be used in:** <do>
3. **Syntax:** task=...

Informs the generating task about records that will be used by the element <loop>.

The absence of the attribute *task* indicates a repetition, regardless of the task.

Example

```

...
<do>
<parm name=to>500</field>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1002>Isis_Total</field>
<loop>
<display>
<pft>newline(' <br>'),v1001, '/', v1002/</pft>
</display>
</loop>
</do>
...

```

do.task=fullinversion

Makes the complete generation of the inverted file.

The database is informed by the element <parm name=db>.

The FST is informed by the element <parm name=fst>.

Example

```

...
<do task=fullinversion>
<parm name=db><pft>v2001</pft></parm>
<parm name=fst><pft>v2061</pft></parm>
<field action=define tag=1102>Isis_Status</field>
<display><pft>'Full inversion: ',v2001/</pft></display>
<loop></loop>
<display><pft>'Finished. '/</pft></display>
<display><pft>'Lock status = 'v1102/</pft></display>
</do>

```

...

do.task=import

Accesses data from a text file and loads it in the form of database records.

The file name is informed through the element *<parm name=file>*.

The element *<parm name=type>* may be used to inform the type of text file; text file *ISO2709* is assumed if the type is not informed.

Example

```
...
<do task=import>
<parm name=file><pft>v2041</pft></parm>
<parm name=type><pft>v2042</pft></parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...
```

do.task=invertedload

Makes the inverted file load through a database that has the sorted keys.

The database to be loaded is informed by the element *<parm name=db>*.

The database that has the sorted keys is informed by the element *<parm name=keyfdb>*.

Example

```
...
<do task=invertedload>
<parm name=db>TEST</parm>
<parm name=keyfdb>TESTKEYS</parm>
<field action=define tag=1>Isis_Posting</field>
<field action=define tag=2>Isis_Key</field>
<field action=define tag=1102>Isis_Status</field>
<display><pft>'Inverted load ...'</pft></display>
<loop></loop>
<display><pft>'Lock status = 'v1102</pft></display>
<flow action=exit>
<pft>if val(v1102) <> 0 then v1102 fi</pft>
</flow>
<display><pft>'Inverted load: TEST loaded.'</pft></display>
</do>
...
```

do.task=keyrange

Runs a list of inverted file keys of a database.

The database is informed by the element *<parm name=db>*.

The initial key is informed by the element `<parm name=from>`.

The final key is informed by the element `<parm name=to>`.

The quantity of keys may be limited by the element `<parm name=count>`.

The element `<parm name=reverse>` may be used to run the keys in reverse order.

The element `<parm name=posting>` may be also used to run the posting list of each key.

The element `<parm name=posttag>` may be used to run the posting lists of each key only for the specified tag.

Example

```

...
<do task=keyrange>
<parm name=db><pft>v2001</pft></parm>
<parm name=from>
<pft>v2002,if a(v2002) then v2101 fi</pft>
</parm>
<parm name=count><pft>v2003,"20"n2003</pft></parm>
<parm name=reverse><pft>v2004</pft></parm>
<parm name=to><pft>v2006</pft></parm>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1>Isis_Key</field>
<field action=define tag=2>Isis_Postings</field>
<display><pft>'##) POSTINGS',c15,'KEY'/#</pft></display>
<loop>
<display>
<pft>f(val(v1001),2,0),' ' ,v2,c15,v1</pft>
</display>
<field action=export tag=1031>
<pft>if val(v1001) = 1 then '1' fi</pft>
</field>
<field action=export tag=1032>1</field>
</loop>
<display>
<pft>'*****'/,
f(val(v1001),2,0),' ' ,v1031,' / ' ,v1032/
</pft>
</display>
</do>
...

```

do.task=list

Runs the list of items previously loaded through the element `<list action=load type=...>`.

The list may be sorted through the element `<parm name=sort>`.

The initial item is informed by the element `<parm name=from>`.

The final item is informed by the element `<parm name=to>`.

The quantity of records may be limited by the element `<parm name=count>`.

The element `<parm name=reverse>` may be used to run the list in reverse order.

Example

```

...
<list action=load type=sort><pft>'1'/','2'/','3'/'</pft></list>
<list action=load type=sort><pft>'9'/','8'/',</pft></list>
<list action=load type=sort><pft>'F'/','Z'/','A'/'</pft></list>
<do task=list>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1002>Isis_Items</field>
<field action=define tag=1>Isis_Item</field>
<loop>
<display>
<pft>v1001,',',v1002,c10,v1/'</pft>
</display>
</loop>
</do>
...

```

do.task=mastersort

Sorts the records of a database.

The database is informed by the element *<parm name=db>*.

The sorting key is informed by the element *<parm name=key>*.

The key length is informed by the element *<parm name=keylength>*.

Example

```

...
<do task=mastersort>
<parm name=db>CDS</parm>
<parm name=key><pft>v24</pft></parm>
<parm name=keylength>100</parm>
<field action=define tag=1102>Isis_Status</field>
<display><pft>'Key sort ...'</pft></display>
<loop></loop>
<display><pft>'Lock status = 'v1102</pft></display>
<flow action=exit>
<pft>if val(v1102) <> 0 then v1102 fi</pft>
</flow>
<display><pft>'Key sort: CDS sorted.'</pft></display>
</do>
...

```

do.task=mfnrange

Runs a list of records of a database.

The database is informed by the element *<parm name=db>*.

The initial MFN is informed by the element *<parm name=from>*.

The final MFN is informed by the element *<parm name=to>*.

The quantity of MFNs may be limited by the element *<parm name=count>*.

The element *<parm name=reverse>* may be used to run the MFNs in reverse order.

Example

```

...
<do task=mfnrange>
<parm name=db>CDS</parm>
<parm name=from>25</parm>
<parm name=count>10</parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...

```

do.task=search

Surveys a database and runs a list of records found.

The database is informed by the element *<parm name=db>*.

The search expression is informed by the element *<parm name=expression>*.

The initial record is informed by the element *<parm name=from>*.

The final record is informed by the element *<parm name=to>*.

The quantity of records may be limited by the element *<parm name=count>*.

The element *<parm name=reverse>* may be used to run the records in reverse order.

Example

```

...
<do task="search">
<parm name="db">CDS</parm>
<parm name="expression">plants * water</parm>
<loop>
<display><pft>mfn</pft></display>
</loop>
</do>
...

```

do.task=update

Updates or adds a record to the database.

The database is informed through the element *<parm name=db>*.

The element *<parm name=mfn>* is used to inform the MFN to be updated or if a new record is to be added.

The element *<parm name=lockid>* is used to identify the "owner" of the record.

The element *<parm name=expire>* may be used to inform that the record "owner" had its time to remain with the record blocked, expired.

Example

```

...
<do task=update>

```

```

<parm name=db>CDS</parm>
<parm name=mfn>New</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=replace tag=20>
<pft>date</pft>
</field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
</update>
</do>
...

```

<export>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <export> ... </export>

Adds the current record in a text file.

The file name must have been previously informed through the element *<parm file=...>*.

The type of standard file is the ISO 2709; to generate another type of export file inform previously with the element *<parm type=...>*. The types of export file are: *ISO2709*, *HLine* and *VLine*.

Example

```

...
<do task=mfnrange>
<parm name=db>CDS</parm>
<parm name=file>CDS.ISO</parm>
<loop>
<export>this</export>
</loop>
</do>
...

```

<extract>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <extract> ... </extract>

Adds the keys extracted from the current record in the key database.

The FST should be previously informed through the element *<parm name=fst>*.

The database that will contain the keys should be previously informed through the element `<parm name=keysdb>`.

The field that will contain the keys should be previously informed through the element `<field action=define tag=...>Isis_Key</field>`.

The field that will contain the data on the posting should be previously informed through the element `<field action=define tag=...>Isis_Posting</field>`.

Example

```
...
<do task=mfnrange>
<parm name=fst>1 0 v1</parm>
<parm name=keysdb>tmpl</parm>
<field action=define tag=1>Isis_Posting</field>
<field action=define tag=2>Isis_Key</field>
<loop>
<extract>this</extract>
</loop>
</do>
...
```

<field>

1. It may contain: `<pft>`
2. It may be used in: `<do>` `<function>` `<hl>` `<loop>` `<section>` `<update>`
3. Attributes: action from previous split tag type
4. Syntax: `<field>` ... `</field>`

The element **<field>** is used to add, modify, delete, import, export and define fields.

The attribute *action* informs the use of the element.

The attribute *tag* informs the field to be affected.

The attribute *split* may be used to inform that each line will be a new field occurrence.

The attribute *previous* informs whether the previous content will be eliminated or not.

The attribute *type* informs the type of field to be accessed.

The attribute *from* informs the number of the field to be accessed.

Example

```
...
<field action=replace tag=2><pft>v400^b</pft></field>
...
```

field.action

1. Options: add cgi define delete export hl import occ replace statusdb statusfile
2. It may be used in: <field> <file> <flow> <function> <htmlpft> <list> <return>
3. Syntax: action=...

Indicates the action that the element *<field>* will execute.

Example

```
...
<field action=replace tag=2><pft>v400^b</pft></field>
...
```

field.action=add

Adds a new occurrence to the field specified by the attribute **tag**.
The argument of the element contains the data to be added.

Example

```
...
<field action=add tag=2>A</field>
<field action=add tag=2>B</field>
<field action=add tag=2>C</field>
...
```

field.action=cgi

Adds the content of the field specified by the attribute **tag** with the corresponding CGI value.
The argument of the element *<field>* indicates which is the CGI item whose value will be added.

Example

```
...
<field action=cgi tag=2>phone</field>
...
```

field.action=define

Defines the number of the field specified by the attribute **tag** as being for automatic IsisScript update.
The argument of the element *<field>* indicates what type of information will be stored in the field.

The following arguments can be defined:

Isis_Current – Index of current *<loop>* execution

Isis_Total - Total of times possible for the *<loop>*

Isis_From - Parameter *from* of the *<loop>*

Isis_To - Parameter *to* of the *<loop>*
Isis_Lock - Field of storage of record lock control
Isis_Status – Storage of task execution status
Isis_Item – List item
Isis_Value - Quantity of items calculated by frequency counting
Isis_Key – Current key
Isis_Posting – Data on current posting
Isis_Postings - Total postings of current key
Isis_Items - Total list items
Isis_ErrorInfo – Error pointer in search
Isis_Keys - Keys to highlight text
Isis_MFN - Number of the MFN storage field to export or import records
Isis_RecordStatus - Number of the field that will contain record status

Example

```
... <field action=define tag=1001>Isis_Current</field>...
```

field.action=delete

Eliminates one or all occurrences of the field specified by the attribute **tag**.
 The argument of the element *<field>* indicates the occurrence to be eliminated,
 use *ALL* in the argument to indicate all occurrences.

Example

```
...
<field action=delete tag=5>ALL</field>
<field action=delete tag=6>1</field>
...
```

field.action=export

Modifies the content of one or more fields in the previous IsisScript scope.
 The field is specified by the attribute **tag**.
 Use *tag=list* to indicate that the argument contains a list of fields to be
 exported.

Example

```
...
<field action=export tag=2205>5</field>
<field action=export tag=list>6,7,21/29</field>
...
```

field.action=hl

Substitutes the content of the field specified by the attribute **tag** with a new value using a text highlight.

Example

```
...
<parm name=prefix><b></parm>
<parm name=suffix></b></parm>
<parm name=keys><pft>(v1022/)</pft></parm>
<field action=hl tag=70><pft>(v70/)</pft></field>
...
```

field.action=import

Modifies the content of one or more fields in IsisScript current scope by copying the fields of the previous scope.

The field é specified by the attribute **tag**.

Use *tag=list* to indicate that the argument contains a list of fields to be imported.

Example

```
...
<field action=import tag=2070>70</field>
<field action=import tag=list>201,206,301/314,321</field>
...
```

field.action=occ

Modifies the content of the field specified by the attribute *tag* with the content of the field specified by the attribute *from*, only with the occurrence specified by the argument of the element.

Example

```
...
<do>
<parm name=to><pft>f(nocc(v70),1,0)</pft></parm>
<field action=define tag=1001>Isis_Current</field>
<loop>
<field action=import tag=70>70</field>
<field action=occ tag=1 from=70><pft>v1001</pft></field>
...
</loop>
</do>
...
```

field.action=replace

Substitutes the content of the field specified by the attribute **tag**.
The argument contains the new content.

Example

```
...
<field action=replace tag=2><pft>v400^b</pft></field>
...
```

field.action=statusdb

Replaces the content of the field specified by the attribute **tag** with the database status informed by the argument of the element *<field>*.

If there is the master or the inverted file or both, the field will be created containing the subfield **s** (^s). The subfield **s** will contain the character *m*, if the master exists, and the character *i*, if the inverted exists.

If the master exists, the field will also contain the subfield **n** (^n) with the total of records in the database plus one, the subfield **d** (^d) with the number of data entry lock (Data Entry Lock), and the subfield **e** (^e) with the exclusive lock number (Exclusive Write Lock).

Example

```
...
<field action=statusdb tag=1091><pft>v2001</pft></field>
<flow action=jump>
<pft>if v1091^s: 'm' then 'LABEL_OK' fi</pft>
</flow>
...
```

field.action=statusfile

Replaces the content of the field specified by the attribute **tag** with the status of the file informed by the argument of the element *<field>*.

After execution, the field will contain the subfield **s** (^s) with the character "e" if the file exists, otherwise, the field will be absent.

Example

```
...
<field action=statusfile tag=1091>C:\AUTOEXEC.BAT</field>
<flow action=jump>
<pft>if v1091^s: 'e' then 'LABEL_OK' fi</pft>
</flow>
...
```

field.from

1. It may be used in: <field>
2. Syntax: from=...
Specifies the number of the field that will be accessed by the attribute *action=occ*.

Example

```
...
<do>
  <parm name=to><pft>f(nocc(v70),1,0)</pft></parm>
  <field action=define tag=1001>Isis_Current</field>
  <loop>
    <field action=import tag=70>70</field>
    <field action=occ tag=1 from=70><pft>v1001</pft></field>
  ...
</loop>
</do>
...
```

field.previous

1. Options: add delete
2. It may be used in: <field>
3. Syntax: previous=...
Specifies whether the field being imported or exported will have its previous content eliminated or if new occurrences will be added.
Absence of this attribute means that the previous content will be eliminated.

Example

```
...
<field action=import tag=1 previous=add>200</field>
...
```

field.previous=add

Adds new occurrences.

Example

```
...
<field action=export tag=200 previous=add>1</field>
...
```

field.previous=delete

Informs that previous occurrences should be eliminated.

Example

```
...
<field action=export tag=4001 previous=delete>1</field>
...
```

field.split

1. Options: flddir occ
2. It may be used in: <field>
3. Syntax: split=...
Indicates how the data will be stored.

Example

```
...
<field action=replace tag=1 split=occ><pft>(v200/)</pft></field>
...
```

field.split=flddir

The text to be stored in the field specified by the attribute **tag** is the record fields directory with the respective contents.

Each line contains the number of the field (5 digits), a blank space and the content of the field.

Example

```
...
<do task="mfncrange">
<parm name="db">CDS</parm>
<parm name="count">5</parm>
<loop>
<field action="replace" tag="1" split="flddir">ALL</field>
<display><pft>ALL</pft></display>
</loop>
</do>
...
```

field.split=occ

Indicates that each line of the argument of the element *<field>* will be stored in a new field occurrence specified by the attribute *tag*.

Example

```
...
<field action=replace tag=1 split=occ><pft>(v200/)</pft></field>
...
```

field.tag

1. Options: list
2. It may be used in: <field> <parm>
3. Syntax: tag=...

The attribute **tag** is used to specify the number of the field.

Use **tag=list** to inform that the list of fields will be passed by the argument of the element <field>.

Example

```
...
<field action=replace tag=2><pft>v400^b</pft></field>
...
```

field.tag=list

Informs that the list of fields will be passed by the argument of the element <field>.

Use the comma "," to separate the list of fields. Use the slash "/" to indicate a field interval. Use open brackets "[" to inform that the field will be stored with the field number specified after the character colon ":" and before closing brackets "]".

Example

```
...
<field action=import tag=list>1,2,3,11/19,[30:20]</field>
...
```

field.type

1. Options: flag
2. It may be used in: <field> <file> <htmlpft> <list> <pft>
3. Syntax: type=...

Informs the type of data that is being accessed.

Example

```
...
<field action=cgi tag=2011 type=flag>trace</field>
...
```

field.type=flag

Informs that the field that is being accessed from CGI is the type present / absent.

If it is present and empty, the content of the field will be stored with *On*, otherwise the field will be stored with the value passed.

This attribute is only effective with the attribute *action=cgi*.

Example

```
...
<field action=cgi tag=2011 type=flag>trace</field>
...
```

<file>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: action type
4. Syntax: <file> ... </file>

The element **<file>** may be used to create, unlock and close databases, create temporary files, delete files and change IsisScript standard output.

The attribute *action* informs the action.

The attribute *type* informs the type of file that will suffer the action.

Example

```
...
<file action=create type=database>TESTX</file>
...
```

file.action

1. Options: append close create delete unlock
2. It may be used in: <field> <file> <flow> <function> <htmlpft> <list> <return>
3. Syntax: action=...

Informs the action to be executed by the element *<file>*.

Example

```
...
<file action=create type=database>TESTX</file>
...
```

file.action=append

Opens an output file to add the texts in the end.

Example

```
...
<file action=append type=output>TEST.LOG</file>
...
```

file.action=close

Closes the output file when the attribute *type=output* or a database when the attribute *type=database*.

Example

```
...
<file action=close type=output>TEST.LOG</file>
...
```

file.action=create

Combined with the attribute *type=output*, the action is to create a new output file. Combined with the attribute *type=database*, the action is to initialize a database.

Example

```
...
<file action=create type=database>TESTX</file>
...
```

file.action=delete

Deletes a file.

Example

```
...
<file action=delete type=file>TESTX</file>
...
```

file.action=unlock

Unlocks a database.

The data entry lock values (Data Entry Lock) and the exclusive lock (Exclusive Write Lock) are zeroed.



The records remain unchanged.

Example

```
...
<file action=unlock type=database>CDS</file>
...
```

file.type

1. Options: database file inverted master output tempfile
2. It may be used in: <field> <file> <htmlpft> <list> <pft>
3. Syntax: type=...
Informs the type of file that is the target of the action.

Example

```
...
<file action=create type=database>TESTX</file>
...
```

file.type=database

Informs that the action applies to a database.

Example

```
...
<file action=create type=database>TESTX</file>
...
```

file.type=file

The action applies to a file.
Valid only for the action *action=delete*.

Example

```
...
<file action=delete type=file>TEST.LOG</file>
...
```

file.type=inverted

Informs that the action applies to the inverted file of a database.
Valid only for the action *action=create*.

Example

```
...
<file action=create type=inverted>TESTX</file>
...
```

file.type=master

Informs that the action applies to the master file of a database.
Valid only for the action *action=create*.

Example

```
...
<file action=create type=master>TESTX</file>
...
```

file.type=output

Informs that the action applies to the output file.

Example

```
...
<file action=create type=output>TEST.LOG</file>
...
```

file.type=tempfile

Informs that the action applies to a temporary file.

Valid only for the action *action=create*.

The argument informs the number of the field in which the single temporary file name returned by the operational system will be stored.

Example

```
...
<file action=create type=tempfile>4001</file>
...
```

<flow>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: action
4. Syntax: <flow> ... </flow>

The element **<flow>** is used to deviate the sequence of execution of IsisScript instructions.

The attribute *action* informs the action.

Example

```
...
<flow action=jump><pft>if p(v1) then 'GO' fi</pft></flow>
<display>Field 1 absent</display>
<flow action=exit>1</flow>
<label>GO</label>
<display>Field 1 present, continue</display>
...
...
```

flow.action

1. Options: exit jump skip
2. It may be used in: <field> <field> <flow> <function> <htmlpft> <list> <return>
3. Syntax: action=...

Informs the action that the element *<flow>* should follow.

Example

```
...
<flow action=jump><pft>if p(v1) then 'GO' fi</pft></flow>
<display>Field 1 absent</display>
<flow action=exit>1</flow>
<label>GO</label>
<display>Field 1 present, continue</display>
...
...
```

flow.action=exit

Terminates the current IsisScript execution.

The argument of the element *<flow>* informs the code to be returned to the operational system.

Example

```
...
<flow action=jump><pft>if p(v1) then 'GO' fi</pft></flow>
<display>Field 1 absent</display>
<flow action=exit>1</flow>
<label>GO</label>
<display>Field 1 present, continue</display>
...
...
```

flow.action=jump

Deviate IsisScript execution to the corresponding *<label>* element.

The argument of the instruction *<flow>* informs the destination.

Example

```
...
<flow action=jump><pft>if p(v1) then 'GO' fi</pft></flow>
<display>Field 1 absent</display>
<flow action=exit>1</flow>
<label>GO</label>
<display>Field 1 present, continue</display>
...
...
```

flow.action=skip

Deviates the IsisScript execution to the beginning of the *<loop>* of the current scope or abandons the current scope and returns to the previous scope.

The argument of the element *<flow>* should be *Next* or *Quit* respectively.

Example

```
...
<do>
  <parm name=db>CDS</db>
  <loop>
    <flow action=skip>
    <pft>if a(v24) then 'Next'
    else if val(v26^c) > 1989 then 'Quit' fi
    fi
  </pft>
  </flow>
  <display><pft>@CDS.PFT</pft></display>
  </loop>
</do>
...
```

<function>

1. It may contain: *<call>* *<cgitable>* *<define>* *<display>* *<do>* *<export>* *<extract>* *<field>* *<file>* *<flow>* *<hl>* *<label>* *<list>* *<parm>* *<proc>* *<return>* *<trace>*
2. It may be used in: *<IsisScript>*
3. Attributes: action from name split tag
4. Syntax: *<function>* ... *</function>*

The element **<function>** starts a block of statement of a function.

Use the attributes *action*, *tag* and *split* to receive parameters as described for the element *<field>*.

Example

```
...
<function name="Test" action="replace" tag="1">
  <display>Inside Test function<br></display>
  <display><pft>'Field 1 = ',v1</pft></display>
</function>
...
```

function.action

See: *<field action=...>*

Passes parameters to the function.

Has the same functionality of the attribute *action* of the element *<field>*.

Example

```
...
<function name="Test" action="replace" tag="1">
<display>Inside Test function<br></display>
<display><pft>'Field 1 = ',v1</pft></display>
</function>
...
```

function.from

See: <field from=...>

Passage of parameters to the function.

Has the same functionality of the attribute *from* of the element <field>.

Example

```
...
<function name="Test" action="replace" tag="1">
<display>Inside Test function<br></display>
<display><pft>'Field 1 = ',v1</pft></display>
</function>
...
```

function.name

1. It may be used in: <call> <function> <IsisScript> <section>

2. Syntax: name=...

The attribute **name** identifies the function that is being stated.

This name will be used by the element <call> to call the function.

Example

```
...
<function name="Test" action="replace" tag="1">
<display>Inside Test function<br></display>
<display><pft>'Field 1 = ',v1</pft></display>
</function>
...
```

function.split

See: <field split=...>

Passage of parameters to the function.

Has the same functionality of the attribute *split* of the element <field>.

Example

```
...
<function name="Test" action="replace" tag="1">
<display>Inside Test function<br></display>
```

```

<display><pft>'Field 1 = ',v1</pft></display>
</function>
...

```

function.tag

See: <field tag=...>

Passage of parameters to the function.

Has the same functionality of the attribute *tag* of the element <field>.

Example

```

...
<function name="Test" action="replace" tag="1">
<display>Inside Test function<br></display>
<display><pft>'Field 1 = ',v1</pft></display>
</function>
...

```

<hl>

1. It may contain: <call> <cgitable> <define> <display> <do> <export> <extract> <field> <file> <flow> <label> <list> <parm> <proc> <trace>
2. It may be used in: <do> <function> <loop> <section> <update>
3. Syntax: <loop> ... </loop>
The element <hl> starts a block of instructions to highlight the text.

Example

```

...
<hl>
<parm name="prefix"><b></b></parm>
<parm name="suffix"></b></parm>
<parm name="keys"><pft>(v1022/)</pft></parm>
<field action="hl" tag="18"><pft>v18</pft></field>
<display><pft>ALL</pft></display>
</hl>
...

```

<htmlpft>

1. It may contain: <pft>
2. It may be used in: <display>
3. Attributes: action type
4. Syntax: <htmlpft> ... </htmlpft>

Interprets and formats an HTML file that contains instructions about the format language.

Example

```
...
<display>
<htmlpft><pft>cat('Test.htm')</pft></htmlpft>
</display>
...
```

htmlpft.action

1. Options: convert
2. It may be used in: <field> <file> <flow> <function> <htmlpft> <list> <return>
3. Syntax: action=...

Specifies an action different from the standard action of the element *<htmlpft>*.

Example

```
...
<file action=create type=output>TEST.PFT</file>
<display>
<htmlpft action=convert>
<pft>cat('TEST.HTML')</pft>
</htmlpft>
</display>
<file action=close type=output>now</file>
...
```

htmlpft.action=convert

Converts an HTML with format specifications to a format, however, it does not interpret the format converted, it shows the specification generated.

Example

```
...
<file action=create type=output>TEST.PFT</file>
<display>
<htmlpft action=convert>
<pft>cat('TEST.HTML')</pft>
</htmlpft>
</display>
<file action=close type=output>now</file>
...
```

htmlpft.type

See: <pft type=...>

Type of action to be taken for format execution.

Example

```
...
<display>
<htmlpft type=reload>
<pft>cat('Test.htm')</pft>
</htmlpft>
</display>
...
```

<label>

1. It may be used in: <do> <function> <hl> <loop> <section> <update>
2. Syntax: <label> ... </label>

The element **<label>** indicates a point to which IsisScript may deviate the sequence of instruction execution through the element *<flow action=jump>*.

Example

```
...
<field action=cgi tag=2001>db</field>
<flow action=jump><pft>if p(v2001) then 'OK' fi</flow>
<display>db parameter absent, must exit</display>
<flow action=exit>0</exit>
<label>OK</label>
<display>db parameter present, continue</display>
...
```

<list>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: action type
4. Syntax: <list> ... </list>

The element **<list>** changes IsisScript internal list. The options are: add items to the list or eliminate all items from the list.

The attribute *action* indicates the option.

The attribute *type* indicates the type of list.

Example

```
...
<list action=load type=freq><pft>(v66/)</pft></list>
...
```

list.action

1. Options: delete load
2. It may be used in: <field> <file> <flow> <function> <htmlpft> <list> <return>
3. Syntax: action=...

Informs the action to be executed in the IsisScript list.

Example

```
...
<list action=load type=freq><pft>(v66/)</pft></list>
...
```

list.action=delete

Eliminates all items of the list.

Example

```
...
<list action=delete>now</list>
...
```

list.action=load

Adds new items to the list.

Each line informed by the argument will be a new item in the list.

Example

```
...
<list action=load type=freq><pft>(v66/)</pft></list>
...
```

list.type

1. Options: freq list sort
 2. It may be used in: <field> <file> <htmlpft> <list> <pft>
- Informs the type of storage in the list.

Example

```
...
<list action=load type=freq><pft>(v66/)</pft></list>
...
```

list.type=freq

Lists the counting of items frequency.

A repeated item is not inserted in the list, it is added in the total occurrences of that item.

Example

```
...
<list action=load type=freq><pft>(v66/)</pft></list>
...
```

list.type=list

The list of items without sorting.

Example

```
...
<list action=load type=list><pft>(v66/)</pft></list>
...
```

list.type=sort

The list is sorted by item content.

Example

```
...
<list action=load type=sort><pft>(v66/)</pft></list>
...
```

<loop>

1. It may contain: <call> <cgitable> <define> <display> <do> <export> <extract> <field> <file> <flow> <hl> <label> <list> <parm> <proc> <return> <trace>
2. It may be used in: <do>
3. Syntax: <loop> ... </loop>

The element **<loop>** indicates a group of instructions that will be repeated for all the data found according to the type of task specified in the corresponding element *<do>*.

Example

```
...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<loop>
<display><pft>mfN</pft></display>
</loop>
</do>
...
```

<parm>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Attributes: name tag type
4. Syntax: <parm> ... </parm>

The element **<parm>** informs a parameter for the block of instructions it belongs to.

Example

```
...
<do task=search>
  <parm name=db> <pft>v2001</pft></parm>
  <parm name=expression><pft>v2005</pft></parm>
  <loop>
    <display><pft>mf/ </pft></display>
  </loop>
</do>
...
```

parm.name

1. Options: actab buffersize cipar count db decod delimiter expire expression file freqsum from fst gizmo indexlist key keyfield keylength keys keysdb lockid maxlk mfn posting posttag prefix reset reverse sort stw suffix task to type uctab
2. It may be used in: <call> <function> <IsisScript> <parm> <section>
3. Syntax: name=...
Name of the parameter.

Example

```
<IsisScript>
<section>
  <parm name=actab><pft>cat(' ISISAC.TAB')</pft></parm>
  <parm name=uctab><pft>cat(' ISISUC.TAB')</pft></parm>
  <parm cipar><pft>'CDS.*=/bases/cds/cds.*'/,
  'ACTAB=/isis/menu/isisac.tab'</pft>
  'UCTAB=/isis/menu/isisuc.tab'</pft>
</parm>
  <do task=search>
    <parm name=db>CDS</parm>
    <parm name=expression>plants*water</parm>
    <parm name=to>10</parm>
  <loop>
    ...
  </loop>
</do>
...
</section>
</IsisScript>
```

parm.name=actab

Changes IsisScript table of alphabetic characters during the current section. The table of alphabetic characters informs, for update of the inverted file and key extraction, what characters are considered alphabetic. The characters that are not in the table are considered delimiters. In a section without the option `<parm name=actab>` IsisScript assumes the ANSI table.

Example

```
<IsisScript>
<section>
<parm cipar><pft>'CDS.*=/bases/cds/cds.*' / ,
'ACTAB=/isis/menu/isisac.tab' /</pft>
'UCTAB=/isis/menu/isisuc.tab' /</pft>
</parm>
<parm name=actab><pft>cat('ACTAB')</pft></parm>
<parm name=uctab><pft>cat('UCTAB')</pft></parm>
<do task=search>
<parm name=db>CDS</parm>
<parm name=expression>plants*water</parm>
<parm name=to>10</parm>
<loop>
<display><pft>mpu,v24,mpl</pft></display>
</loop>
</do>
...
</section>
</IsisScript>
```

parm.name=bufferize

Enables changing the size of the WXIS internal buffer (in bytes) that is used to store the formatting result.

Example

```
...
<parm name="bufferize">90000</parm>
...
```

parm.name=cipar

Activates a table of assignments of logical names with physical file names for the current section.

Each line contains an assignment, to the left of the character = (equal) is the logical name and to the right, the physical file name.

The character * (asterisk) indicates that the assignment is valid for any database file.

Example

```

...
<parm name=cipar>
<pft>
'CDS.ISO=/bases/cds/cds.iso' /
'CDS.*=/bases/cds/cds.*' /
'TEST.PFT=/bases/cds/test.pft' /
</pft>
</parm>
...

```

parm.name=count

Limits the number of times that the group of instructions of the element *<loop>* will be executed.

Example

```

...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<parm name=from> <pft>v2002</pft></parm>
<parm name=count>10</parm>
<loop>
<display><pft>mf</pft></display>
</loop>
</do>
...

```

parm.name=db

Specifies the database to be accessed in the following tasks:

```

task=mnrange
task=keyrange
task=search
task=update
task=fullinversion
task=mastersort
task=invertedload

```

Example

```

...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<loop>
<display><pft>mf</pft></display>
</loop>
</do>
...

```

parm.name=decode

Inform the database of expansion parameters of the decoded fields.

Example

```

...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=decode> <pft>v2101</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<loop>
<display><pft>mfN</pft></display>
</loop>
</do>
...

```

parm.name=delimiter

Inform the field separator to import records in the option *RLine*. In the absence of this parameter, the character | (pipe) is assumed.

Example

```

...
<do task="import">
<parm name="file"><pft>v2041</pft></parm>
<parm name="type"><pft>v2042</pft></parm>
<parm name="delimiter"><pft>v2043</pft></parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...

```

parm.name=expire

Inform the maximum time that the record will remain locked. After this period, the record may be locked by another user (another identification).

Example

```

...
<do task=update>
<field action=cgi tag=2001>db</field>
<field action=cgi tag=2002>mfN</field>
<parm name=db><pft>v2001</pft></parm>
<parm name=mfN><pft>v2002</pft></parm>
<parm name=expire>14400</parm>
<field action=define tag=1101>Isis_Lock</field>
<field action=define tag=1102>Isis_Status</field>
<update>
<write>Unlock</write>

```

```

<display><pft>ALL</pft></display>
<display><pft>'*** LOCK STATUS: 'v1102</pft></display>
</update>
</do>
...

```

parm.name=expression

Informs the search expression.

Example

```

...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<loop>
<display><pft>mf</pft></display>
</loop>
</do>
...

```

parm.name=file

Informs the file name that will be imported or exported.

Example

```

...
<do task=import>
<parm name=file><pft>v2041</pft></parm>
<parm name=type><pft>v2042</pft></parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...

```

parm.name=freqsum

Informs the value to be added in the addition of new items to the frequency list.

Example

```

...
<loop>
<!-- field 1 = Product
field 2 = Quantity -->
<parm name=freqsum><pft>v2</pft></parm>
<list action=load type=freq><pft>v1</pft></list>
</loop>
...

```

parm.name=from

Informs what is the first item to be accessed by the group of the element *<loop>*.

Example

```

...
<do task=search>
<parm name=db> <pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<parm name=from> <pft>v2002</pft></parm>
<parm name=count> <pft>v2003</pft></parm>
<loop>
<display><pft>mf</pft></display>
</loop>
</do>
...

```

parm.name=fst

Informs the FST that will be used to update the inverted file or to extract keys.

Example

```

...
<do task=update>
<parm name=db><pft>v2001</pft></parm>
<parm name=mf<=mf>New</parm>
<parm name=fst>1 0 v1</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=cgi tag=1>Name</field>
<field action=cgi tag=2>Phone</field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
</update>
</do>
...

```

parm.name=gizmo

Informs the database of content conversion parameters.

Example

```

...
<file action=create type=database>GIZMO_DIAC</file>
<do task=update>
<parm name=db>GIZMO_DIAC</parm>
<parm name=mf<=mf>New</parm>
<field action=define tag=1101>Isis_Status</field>
<update>
<field action=replace tag=1>á</field>

```

```

<field action=replace tag=2>á</field>
<write>Unlock</write>
</update>
</do>
<do task=mfnrange>
<parm name=db>CDS</parm>
<parm name=gizmo>GIZMO_DIAC</parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...

```

parm.name=indexlist

Informs the index list for database search.

Example

```

...
<do task=search>
<parm name=db>cds</parm>
<parm name=indexlist><pft>
'^p*^ycds^m** '/,
'^pAU ^ycdsaut^mAU '/,
'^pTI ^ycdstit^mTI '/
</pft></parm>
<parm name=expression>
Au Mag$ or ([Ti] plants AND water)
</parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...

```

parm.name=key

Key to sort the database.

Example

```

...
<do task=mastersort>
<parm name=db>CDS</parm>
<parm name=key><pft>v24</pft></parm>
<parm name=keylength>100</parm>
<field action=define tag=1102>Isis_Status</field>
<display><pft>'Sort ...'</pft></display>
<loop></loop>
<display><pft>'Lock status = 'v1102</pft></display>
<flow action=exit>
<pft>if val(v1102) <> 0 then v1102 fi</pft>
</flow>
<display><pft>'Sort: CDS sorted.'</pft></display>

```

```

</do>
...

```

parm.name=keyfield

Specifies the field that is the key to sort the database.

Example

```

...
<do task="mastersort">
  <parm name="db">CDS</parm>
  <parm name="keyfield">24</parm>
  <parm name="keylength">200</parm>
  <field action="define" tag="l102">Isis_Status</field>
  <display><pft>'Key sort ...'</pft></display>
  <loop></loop>
  <display>
  <pft>'Lock status = 'v1102</pft>
  </display>
  <flow action="exit">
  <pft>if val(v1102) <> 0 then v1102 fi</pft>
  </flow>
  <display>
  <pft>'Key sort: ',v2001,' sorted.'</pft>
  </display>
</do>
...

```

parm.name=keylength

Length of the key to sort the database.

Example

```

...
<do task=mastersort>
  <parm name=db>CDS</parm>
  <parm name=key><pft>v24</pft></parm>
  <parm name=keylength>100</parm>
  <field action=define tag=l102>Isis_Status</field>
  <display><pft>'Sort ...'</pft></display>
  <loop></loop>
  <display><pft>'Lock status = 'v1102</pft></display>
  <flow action=exit>
  <pft>if val(v1102) <> 0 then v1102 fi</pft>
  </flow>
  <display><pft>'Sort: CDS sorted.'</pft></display>
</do>
...

```

parm.name=keys

Informs the list of keys to highlight the text.

Example

```

...
<do task=search>
  <parm name=db>cds</parm>
  <parm name=expression><pft>v2005</pft></parm>
  <parm name=from><pft>v2002,"1"n2002</pft></parm>
  <parm name=count>10</parm>
  <field action=define tag=1001>Isis_Current</field>
  <field action=define tag=1002>Isis_Total</field>
  <field action=define tag=1022>Isis_Keys</field>
  <loop>
  <hl>
    <parm name=prefix><b></parm>
    <parm name=suffix></b></parm>
    <parm name=keys><pft>(v1022/)</pft></parm>
    <field action=hl tag=24><pft>v24</pft></field>
    <field action=hl tag=70 split=occ><pft>(v70/)</pft></field>
  <display><pft>ALL</pft></display>
  </hl>
  </loop>
  <display><pft>
  if val(v1002) = 0 then 'No record found!' fi
  </pft></display>
</do>
...

```

parm.name=keybdb

Database that will contain the keys that will be extracted, if used as parameter for the element *<extract>*.

Database with the keys sorted for inverted file load if used as parameter of the element *<do task=invertedload>*.

Example

```

...
<do task=invertedload>
  <parm name=db><pft>v2001</pft></parm>
  <parm name=keybdb><pft>v2064</pft></parm>
  <field action=define tag=1>Isis_Posting</field>
  <field action=define tag=2>Isis_Key</field>
  <field action=define tag=1102>Isis_Status</field>
  <display><pft>'Inverted load ...'</pft></display>
  <loop></loop>
  <display><pft>'Lock status = 'v1102</pft></display>
  <flow action=exit><pft>
  if val(v1102) <> 0 then v1102 fi
  </pft></flow>
  <display><pft>
  'Inverted load: ',v2001,' loaded.'/
  </pft></display>
</do>
...

```

parm.name=lockid

Record lock identifier.

Example

```

...
<do task=update>
<parm name=db><pft>v2023</pft></parm>
<parm name=mfn><pft>mfn</pft></parm>
<field action=define tag=1101>Isis_Lock</field>
<parm name=lockid>
<pft>getenv( 'REMOTE_ADDR' ),x1,s(date).8</pft>
</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=cgi tag=1>phone</field>
<field action=replace tag=1><pft>v1</pft></field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
<display>
<pft>'*** LOCK STATUS: 'v1102/</pft>
</display>
</update>
</do>
...

```

parm.name=maxlk

Maximum number of keys (per record) in extraction via FST.

In the absence of this parameter, IsisScript assumes 1024.

Example

```

...
<do task=fullinversion>
<parm name=db><pft>v2001</pft></parm>
<parm name=fst><pft>v2061</pft></parm>
<parm name=maxlk>5000</parm>
<field action=define tag=1102>Isis_Status</field>
<display><pft>'Full inversion: ',v2001/</pft></display>
<loop></loop>
<display><pft>'Finished.'/</pft></display>
<display><pft>'Lock status = 'v1102/</pft></display>
</do>
...

```

parm.name=mfn

Number of the record to be updated, or *New* to indicate that it will be a new database record, or *GetNew* to indicate that it will be a new database record containing the fields imported from the record of the previous IsisScript scope.

Example

```

...
<do task=update>
<parm name=db><pft>v2023</pft></parm>
<parm name=mfn><pft>mfn</pft></parm>
<field action=define tag=1101>Isis_Lock</field>
<parm name=lockid>
<pft>getenv( 'REMOTE_ADDR' ),x1,s(date).8</pft>
</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=cgi tag=1>phone</field>
<field action=replace tag=1><pft>v1</pft></field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
<display>
<pft>'*** LOCK STATUS: 'v1102/</pft>
</display>
</update>
</do>
...

```

parm.name=posting

Quantity of postings for each key.

Use *All* to indicate "all postings".

Example

```

...
<do task=keyrange>
<parm name=db>CDS</parm>
<parm name=from>PLANTS</parm>
<parm name=count>20</parm>
<parm name=posting>All</parm>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1>Isis_Key</field>
<field action=define tag=2>Isis_Postings</field>
<field action=define tag=3>Isis_Posting</field>
<display><pft>
' POSTINGS',c15,'KEY',c46,'POSTING DETAIL' /#
</pft></display>
<loop>
<display><pft>
f(val(v1001),2,0),' ' ,v2,c15,v1,c46,v3/
</pft></display>
</loop>
</do>
...

```

parm.name=posttag

Informs the field number of the posting to be accessed in the key interval.

Example

```

...
<do task=keyrange>
<parm name=db>CDS</parm>
<parm name=from>B</parm>
<parm name=count>20</parm>
<parm name=posttag>70</parm>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1>Isis_Key</field>
<field action=define tag=2>Isis_Postings</field>
<field action=define tag=3>Isis_Posting</field>
<display><pft>
' POSTINGS',c15,'KEY',c46,'POSTING DETAIL'/#
</pft></display>
<loop>
<display><pft>
f(val(v1001),2,0),') ',v2,c15,v1,c46,v3/
</pft></display>
</loop>
</do>
...

```

parm.name=prefix

Prefix to be inserted in the text highlight, or prefix to be used by the element *<htmlpft>*.

Example

```

...
<parm name=prefix>[pft]</prefix>
<parm name=suffix>[/pft]</suffix>
<htmlpft><pft>cat('TEST.HTM')</pft></htmlpft>
...

```

parm.name=reset

Enables the inverted file update not to change information of the master file record with pending inversion. Applicable to databases with multiple inverted files.

Example

```

...
<do task="fullinversion">
<parm name="db">CDS</parm>
<parm name="fst">CDS.FST</parm>
<parm name="reset">Off</parm>
<field action="define" tag="1102">Isis_Status</field>
<display><pft>'Full inversion: CDS'</pft></display>
<loop></loop>
<display><pft>'Finished.'</pft></display>
<display><pft>'Lock status = 'v1102/</pft></display>
</do>

```

...

parm.name=reverse

Informs that the records resulting from the task specified in the element `<do>` will be accessed in reverse order.

Example

```
...
<do task=search>
<parm name=db><pft>v2001</pft></parm>
<parm name=expression><pft>v2005</pft></parm>
<parm name=reverse>On</parm>
<loop>
<display><pft>mf</pft></display>
</loop>
</do>
...
```

parm.name=sort

Informs the sorting format of IsisScript internal list.

Example

```
...
<list action=load type=list>
<pft>'5.00','1.50','10.00','8','3.75','14.20','0.40'</pft></list>
<do task=list>
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1002>Isis_Items</field>
<field action=define tag=1>Isis_Item</field>
<parm name=sort><pft>f(val(v1),10,2)</pft></parm>
<loop>
<display><pft>v1001,'/',v1002,c10,v1</pft></display>
</loop>
</do>
...
```

parm.name=stw

Informs the STW file with the table of words that should not be included in the inverted file that will be used to update the inverted file or extract keys.

Example

```
...
<do task=fullinversion>
<parm name=db><pft>v2001</pft></parm>
<parm name=fst><pft>v2061</pft></parm>
<parm name=stw><pft>v2001,'.stw'</pft></parm>
<field action=define tag=1102>Isis_Status</field>
```

```

<display><pft>'Full inversion: ',v2001/</pft></display>
<loop></loop>
<display><pft>'Finished.'/</pft></display>
<display><pft>'Lock status = 'v1102/</pft></display>
</do>
...

```

parm.name=suffix

Suffix to be inserted in the text highlight, or suffix to be used by the element `<htmlpft>`.

Example

```

...
<parm name=prefix>[pft]</prefix>
<parm name=suffix>[/pft]</suffix>
<htmlpft><pft>cat('TEST.HTM')</pft></htmlpft>
...

```

parm.name=task

Indicates the type of task that will be used by the element `<loop>`. It is effective if the attribute `task` of the element `<do>` is not specified.

Example

```

...
<do>
<field action="cgi" tag="2081">dotask</field>
<parm name="task"><pft>v2081</pft>
<loop>
...
</loop>
</do>
...

```

parm.name=to

Informs what is the last item to be accessed by the group of the element `<loop>`.

Example

```

...
<do task=keyrange>
<parm name=db> <pft>v2001</pft></parm>
<parm name=from><pft>v2002</pft></parm>
<parm name=to> <pft>v2002,'ZZZZZZZZZ'</pft></parm>
<loop>
<display><pft>v1/</pft></display>
</loop>
</do>
...

```

parm.name=type

Informs the type of file to export or import.

Possible types include: *ISO2709*, *HLine*, *RLine* and *VLine*.

ISO2709 is an ISO standard (International Standards Organization), but limits the field identification number to 3 digits.

HLine is more efficient, uses the command H of the element *<proc>*.

RLine is only used for data import, where each line of a sequence file corresponds to a record.

VLine is recommended to enable modification via text editor.

Example

```
...
<do task=import>
<parm name=file><pft>v2041</pft></parm>
<parm name=type><pft>v2042</pft></parm>
<loop>
<display><pft>ALL</pft></display>
</loop>
</do>
...
```

parm.name=uctab

Changes IsisScript character conversion table into capital letters during the current section.

This table informs, for inverted file updating, keys extraction and format language mode option, all the characters corresponding to low case or capital letters with accentuation into the corresponding capital letter without accents.

In a section without the element *<parm name=uctab>*, IsisScript assumes the ANSI table.

Example

```
<IsisScript>
<section>
<parm cipar><pft>'CDS.*=/bases/cds/cds.*' / ,
'ACTAB=/isis/menu/isisac.tab' </pft>
'UCTAB=/isis/menu/isisuc.tab' </pft>
</parm>
<parm name=actab><pft>cat('ACTAB')</pft></parm>
<parm name=uctab><pft>cat('UCTAB')</pft></parm>
<do task=search>
<parm name=db>CDS</parm>
<parm name=expression>plants*water</parm>
<parm name=to>10</parm>
<loop>
<display><pft>mpu,v24,mpl</pft></display>
</loop>
</do>
...
```

```

</section>
</IsisScript>

```

parm.tag

1. It may be used in: <field> <parm>
2. Syntax: tag=...

The attribute **tag** is used to specify the number of the field.

Example

```

...
<parm name="fst" type="check" tag="1">
<pft>cat(v2065)</pft>
</parm>
...

```

parm.type

1. Options: check
2. It may be used in: <do>
3. Syntax: type=check

Specifies the type of parameter.

Example

```

...
<parm name="fst" type="check" tag="1">
<pft>cat(v2065)</pft>
</parm>
...

```

parm.type=check

Enables syntax verification of an FST. Updates the field specified by the attribute tag with the error code (5 digits), a blank space and the point in which the syntax error was detected, or 00000, if there is not a syntax error.

Example

```

...
<field action="cgi" tag="2065">fst</field>
<parm name="fst" type="check" tag="1"><pft>cat(v2065)</pft></parm>
<display><pft>ALL</pft></display>
...

```

<pft>

1. It may contain: <pft>
2. It may be used in: <call> <cgitable> <define> <display> <export> <extract> <field> <file> <flow> <htmlpft> <label> <list> <parm> <pft> <proc> <return> <trace> <write>
3. Attributes: type
4. Syntax: <pft> ... </pft>
Formats the current record.

Example

```
...
<display><pft>("Author: "v70+|; |)</pft></display>
<display><pft>@CDS.PFT</pft></display>
<display><pft>cat('C:\AUTOEXEC.BAT')</pft></display>
<display><pft>ref(['CONFIG']1,v500/)</pft></display>
...
```

pft.type

1. Options: check reload
2. It may be used in: <field> <file> <htmlpft> <list> <pft>
3. Syntax: type=...
Type of action to be taken for format execution.

Example

```
...
<do>
<parm name=to>10</parm>
<loop>
<display>
<pft type=reload>
<pft>ref(['CONFIG']val(v1),v500/)</pft>
</pft>
</display>
</loop>
</do>
...
```

pft.type=check

Enables verification of the syntax of a format. Returns the error code (5 digits), a space and the point in which the syntax error was detected, or if there is no syntax error.

Example

```
...
<field action="cgi" tag="2065">pft</field>
```

```

<display>
<pft type="check">
<pft>v2065</pft>
</pft>
</display>
...

```

pft.type=reload

Use this option to inform IsisScript that the format will have to be recompiled each time this instruction is executed.

Example

```

...
<display>
<pft type=reload>
<pft>ref(['CONFIG' ]1,v500/)</pft>
</pft>
</display>
...

```

<proc>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <loop> <section> <update>
3. Syntax: <proc> ... </proc>
Modifies the content of the current record.

Example

```

...
<proc><pft>'a',v2024,'~',v2027,'~'</pft></proc>
...

```

<return>

1. It may contain: <pft>
2. It may be used in: <function>
3. Attributes: action split tag
4. Syntax: <return> ... </return>
Exits the current function.

Example

```

...
<function name=ParamTest action=replace tag=1 split=occ>

```

```

<display><pft>##'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action=replace tag=9999 split=occ>
<pft>(v1/)</pft>
</return>
<display>Parameter field 1 absent!</display>
</function>
...

```

return.action

See: <field action=...>

Returns the parameters to the one that called the function. Has the same functionality of the attribute *action* of the element <field>.

Example

```

...
<function name=ParamTest action=replace tag=1 split=occ>
<display><pft>##'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action=replace tag=9999 split=occ>
<pft>(v1/)</pft>
</return>
<display>Parameter field 1 absent!</display>
</function>
...

```

return.split

See: <field split=...>

Returns the parameters to the one that called the function. Has the same functionality of the attribute **split** of the instruction <field>.

Example

```

...
<function name=ParamTest action=replace tag=1 split=occ>
<display><pft>##'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action=replace tag=9999 split=occ>
<pft>(v1/)</pft>
</return>
<display>Parameter field 1 absent!</display>
</function>
...

```

return.tag

See: <field tag=...>

Returns the parameters to the one that called the function. Has the same functionality of the attribute *tag* of the element *<field>*.

Example

```
...
<function name=ParamTest action=replace tag=1 split=occ>
<display><pft>##'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action=replace tag=9999 split=occ>
<pft>(v1/)</pft>
</return>
<display>Parameter field 1 absent!</display>
</function>
...
```

return.tag

See: *<field tag=...>*

Returns the parameters to the one that called the function. Has the same functionality of the attribute *tag* of the instruction *<field>*.

Example

```
...
<function name=ParamTest action=replace tag=1 split=occ>
<display><pft>##'ParamTest' /</pft></display>
<display><pft>ALL</pft></display>
<return action=replace tag=9999 split=occ>
<pft>(v1/)</pft>
</return>
<display>Parameter field 1 absent!</display>
</function>
...
```

<section>

1. It may contain: *<call>* *<cgitable>* *<define>* *<display>* *<do>* *<export>* *<extract>* *<field>* *<file>* *<flow>* *<hl>* *<label>* *<list>* *<parm>* *<proc>* *<trace>*
2. It may be used in: *<IsisScript>*
3. Attributes: *name*
4. Syntax: *<section>* ... *</section>*

The element **<section>** is used to start a sequence of instructions that access common fields and make use of common tables.

The attribute *name* may be used for identification.

Example

```
<IsisScript name=Test>
<section name=TestFirst>
```

```

<display><pft>mpu, 'Test' </pft></display>
</section>
</IsisScript>

```

section.name

1. It may be used in: <call> <function> <IsisScript> <section>
2. Syntax: name=...
The attribute **name** is optional; it is used for section identification.

Example

```

<IsisScript name=Test>
<section name=TestFirst>
<display><pft>mpu, 'Test' </pft></display>
</section>
</IsisScript>

```

<trace>

1. It may contain: <pft>
2. It may be used in: <do> <function> <hl> <IsisScript> <loop> <section> <update>
3. Syntax: <trace> ... </trace>
Activates or disables the display of the instruction that is being executed.
Possible modes are normal, line to line or table, respectively: *On*, *BR* and *Table*.

Example

```

...
<trace>On</trace>
...

```

<update>

1. It may contain: <call> <cgitable> <define> <display> <do> <export> <extract> <field> <file> <flow> <hl> <label> <list> <parm> <proc> <return> <trace> <write>
2. It may be used in: <do>
3. Syntax: <update> ... </update>
Starts a block of instructions to modify or add a record.

Example

```

...
<do task=update>

```

```

<parm name=db>CDS</parm>
<parm name=mfn>New</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=append tag=1>One more</field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
</update>
</do>
...

```

<write>

1. It may contain: <pft>
2. It may be used in: <update>
3. Syntax: <write> ... </write>

Element that records record modification.

If the element <*parm name=mfn*> indicates *New* or *GetNew*, then it adds a new record, otherwise it updates the mfn passed as argument.

If the argument of the element <**write**> is *Unlock*, the record will be unlocked after being recorded, if it is *Lock*, the record will be recorded and locked, if it is *NoUnlock*, the record will remain locked and the lock information will remain the same; if it is *Delete*, the record will be deleted.

Example

```

...
<do task=update>
<parm name=db>CDS</parm>
<parm name=mfn>New</parm>
<field action=define tag=1102>Isis_Status</field>
<update>
<field action=add tag=1>plus one </field>
<write>Unlock</write>
<display><pft>ALL</pft></display>
</update>
</do>
...

```

Bibliographic references

1. PACKER, Abel Laerte et al. WWWISIS : el camino hacia Internet. INFOISIS, Buenos Aires, v. 2, n. 3-4, p. 7-21, 1996.
2. SANTOS, Gildenir Carolino, PIETROSANTO, Ademir Giacomo. O acesso em base de dados em economia e educação, pela Internet através da ferramenta WWWISIS. Presented in Seminário Nacional de Bibliotecas Universitárias, 10., 1998, Fortaleza. (electronic version: diskette).
3. JAYAKANTH, F. Implementing WWWISIS for providing Web access to bibliographic databases. INSPEL, [Germany], v. 35, n.1, p. 42-54, 2001.

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 tiniest 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.