

The  
WWWISIS  
Handbook  
(for Versions 4 and 5)

*Andrew Buxton*

# THE WWWISIS HANDBOOK (FOR VERSIONS 4 AND 5)

---

**Andrew Buxton**

*Information Systems Manager*

*Institute of Development Studies at the University of Sussex*

March 2002

# The WWWISIS Handbook

## CONTENTS

1. Introduction to CDS/ISIS and WWWISIS
  - 1.1 CDS/ISIS
  - 1.2 Software from BIREME
  - 1.3 Versions of WWWISIS
2. Introduction to web publishing
  - 2.1 Client-server architecture
  - 2.2 Uniform resource Locators (URLs)
  - 2.3 HyperText Markup Language (HTML)
  - 2.4 Other markup languages
  - 2.5 The Common Gateway Interface (CGI)
3. Installation of WWWISIS
  - 3.1 IP settings
  - 3.2 Installing the webserver software
  - 3.3 Licensing of WWWISIS
  - 3.4 Installing WWWISIS
  - 3.5 Creating and loading ISIS databases
4. Introduction to IsisScripts
  - 4.1 Extensible Markup Language (XML)
  - 4.2 IsisScript
  - 4.3 Content type
  - 4.4 Tasks
  - 4.5 Parameters
  - 4.6 Loops
  - 4.7 cipar
  - 4.8 gizmo
  - 4.9 Flow control
  - 4.10 Comments
5. CGI variables and environment variables
  - 5.1 Common Gateway Interface (CGI)
  - 5.2 The input form
  - 5.3 Reading the values
  - 5.4 Using the values in an IsisScript
  - 5.5 Scope of variables
  - 5.6 Environment variables
6. Performing a search and displaying the results
  - 6.1 A simple search script
  - 6.2 Enhancing the display of the results
  - 6.3 Enhancing the search form

7. Accessing the inverted file
  - 7.1 A form for displaying part of the index file
  - 7.2 The IsisScript for displaying the index
  
8. Creating and editing records
  - 8.1 The Update task
  - 8.2 Record locking
  - 8.3 A form for creating new records
  - 8.4 The IsisScript for creating new records
  - 8.5 A form for editing an existing record
  - 8.6 The IsisScript for editing a record
  
9. The IAH Interface
  - 9.1 Introduction
  - 9.2 Installation
  - 9.3 Initial configuration
  - 9.4 Searching the sample databases
  - 9.5 Setting up your own database
  - 9.6 Order of records in display
  - 9.7 URL for searching your database

## CHAPTER 1

### INTRODUCTION TO CDS/ISIS AND WWWISIS

#### 1.1 CDS/ISIS

Micro CDS/ISIS is a package for creating and manipulating textual databases. It was released by UNESCO in 1985, and since then over 20,000 licences have been issued by a worldwide network of distributors. It is particularly suited to bibliographical applications and is used for the catalogues of many small and medium-sized libraries. Versions have been produced in Arabic, Chinese, English, French, German, Portuguese, Russian and Spanish amongst other languages. UNESCO makes the software available free for non-commercial purposes (although distributors are allowed to pass on their expenses). This has led to a wide uptake by organizations in the South, where foreign exchange may not easily be available, as well as organizations in the North involved in joint projects.

Versions of Micro CDS/ISIS have been produced for the MS-DOS, Unix, and Windows operating systems. At the time of writing, the latest version is 1.4 for Windows. More details of CDS/ISIS, including the list of distributors, is available on the UNESCO website, <http://www.unesco.org>.

The original CDS/ISIS ran on an IBM mainframe and was designed in the mid 1970s under Mr Giampaolo Del Bigio for UNESCO's Computerized Documentation System. It was based on the internal ISIS (Integrated Set of Information Systems) at the International Labour Office in Geneva. The source code was made available to other not-for-profit organizations and by 1983 there were 80 installations worldwide.

In October 1985 Mr Del Bigio demonstrated a version of CDS/ISIS for mini- and microcomputers programmed in Pascal at a User Meeting for Latin America and the Caribbean and Version 1 was released in December. It ran on an IBM XT with 256K of memory. In 1987 version 2 was produced with much improved speed and security as well as a single menu giving access to the main functions. The official release was version 2.3. Since then, the database structure has remained the same and databases created on one version can be used on another without conversion. Version 3 came out in 1992 and permitted networking as well as introducing some new inversion techniques (which are needed for the IAH interface described in Chapter 9).

At the International Conference in Bogota held in 1995 to mark 10 years of CDS/ISIS, Mr Del Bigio presented the first results of the migration to Microsoft Windows and Abel Packer from the Regional Library of Medicine (BIREME), Sao Paulo, presented CISIS and ISIS\_DLL. ISIS\_DLL, developed jointly with UNESCO, permits applications written in compatible Windows languages, e.g. Visual Basic, to interact with CDS/ISIS databases.

## 1.2 Software from BIREME

CISIS is CDS/ISIS rewritten in the C programming language. For MS DOS and Windows it takes the form of the program **mx.exe**, and permits command-line operation of CDS/ISIS. For example, a search on the CDS database for "water" with the results displayed in the SHORT format can be performed by using the command

```
mx cds bool=water pft=@short.pft
```

It is the C version which lies partly behind the Windows version of CDS/ISIS (which is written in C++) and it also made possible performing CDS/ISIS operations on a web server through the CGI interface. This is the product WWWISIS. The Version 3 program for 32-bit Windows is **wwwi32.exe** and you can perform the same search from a web browser using the URL

```
http://myserver/cgi-bin/wwwi32.exe/[in=ab.in]
```

Here "myserver" is the address of the server and **ab.in** is a parameter file including similar parameters to the example above:

```
db=cds  
bool=water  
pft=@short.pft
```

The database structure in WWWISIS is the same as in CDS/ISIS (DOS and Windows versions) so databases can be built up with those versions and then offered for searching using WWWISIS. If you are passing data between DOS/Windows and Unix, they can be exported from one version as an ISO file and then imported into the other. The inverted file for WWWISIS can be built using the **mx** program if required.

## 1.3 Versions of WWWISIS

The example above shows how WWWISIS Version 3 works. It uses a parameter file whose name has the extension **.in** and is specified like [in=ab.in]. The output that is passed to the client is specified in this file in three parts:

- (a) The *prolog* which could contain a message like "Here are your search results" and set the background for the display
- (b) The *print format* used to display the records retrieved from the database. This is written using a mixture of the CDS/ISIS formatting language and HyperText Markup Language (HTML)
- (c) The *epilog* with could contain a message like "End of display"

The search (bool=....) would not usually be written in the IN file like the example above, as it would always perform the same search. Rather the search is picked up from a box on the screen, which the user fills in, through the Common Gateway Interface (CGI). If you are not already familiar with them, these concepts will be explained later in Chapter 5.

Version 3 is available free from the UNESCO website but it will not be covered further in this Handbook. Version 4, released in 2000, introduced quite a different approach (the Isis Scripting language). It is with versions 4 and 5 that the Handbook is concerned. They are available for downloading from the BIREME website and you will need to purchase a licence from BIREME if you want to use them over a network. (For practice and development work you can run server and client on the same computer without a licence.)

Version 5 was released in 2001 and added several new features. It allowed searching with field qualifiers as in CDS/ISIS, e.g. plants/(24), and with brackets and proximity operators. It also introduced a new tag <isisxml> which can be used like <pft>. This allows the output of ISIS records in three different XML formats, one of which is illustrated below.

```
<cds mfn="1">
  <field tag="44">
    Methodology of plant eco-physiology
  </field>
  <field tag="26">
    <subfield id="a">Paris</subfield>
    <subfield id="b">Unesco</subfield>
    <subfield id="c">1965</subfield>
  </field>
</cds>
```

## CHAPTER 2

### INTRODUCTION TO WEB PUBLISHING

#### 2.1 Client-server architecture

There are three parts to the technology of the World Wide Web (WWW):

- The *server*, which is the computer that holds the information
- The *client*, which is what the user employs to view the information
- The *protocol*, which conveys the information from one to the other

The server may be a large and powerful computer with many gigabytes of disk storage holding a vast web site. However, almost any computer is capable of acting as a webserver and there are several free or inexpensive programs available that will turn a PC into a webserver. Normally the server will have a permanent connection to the Internet, so that users elsewhere can call up information from it at any time they choose. A permanent Internet connection is common in universities and large companies, but might be too expensive for a small organization or an individual. For such websites the best option might be to store the site on the computer of an *Internet Service Provider* (ISP) who may provide some space free as part of their service. However, they may not allow running programs such as WWWISIS on their computers.

The information on the server may consist of *static* pages or *dynamic* pages. Static pages are held as individual files on the server and look the same each time they are called up. Dynamic pages are generated on demand and not held as individual files, for example the results of a search on a database. The kind of information can include text, images, sounds, videos and others. The collection of all the information on the Internet, which is linked together as *hypertext* (see Section 2.2), comprises the World Wide Web.

The client can be almost any computer running web browser software. The best known products are Netscape Navigator and Microsoft's Internet Explorer. Navigator grew out of an earlier program called Mosaic. There is a text-only browser called Lynx and several other graphical packages such as Opera.

When a client makes a request for information to the server, it uses the HyperText Transfer Protocol (HTTP). The server processes the request and again uses HTTP to send the information back to the client. The client interprets the message sent back and displays it on the screen for the user to view.

It is possible, especially for learning and test purposes, for the server and client to be based on the same computer. In this case no network is needed and the client refers to the server as *localhost*. Also some organizations operate a web service over their own internal network only: this is known as an *intranet*.

## 2.2 Uniform Resource Locators (URLs)

The way that a user selects a particular page from the WorldWide Web is by means of its Uniform Resource Locator, e.g.

**<http://www.ids.ac.uk/index.html>**

This requests the server whose address is [www.ids.ac.uk](http://www.ids.ac.uk) to send the file **index.html** using HyperText Transfer Protocol (HTTP). There are other protocols used on the Internet which were used before the invention of the World Wide Web, such as File Transfer Protocol (FTP) and Telnet. However, with WWWISIS we are concerned only with HTTP. Most modern browsers will assume HTTP without you needing to specify it when you enter a URL.

As the name suggests, HTTP is the format used for transferring hypertext, which is what the information on the web consists of. It is pieces of text or other items joined together by hyperlinks (usually just called *links*). In a printed encyclopaedia you might have an entry like

**EDINBURGH.** The capital of Scotland and the home of the Edinburgh Festival.

To read more about Scotland, you would have to turn to that entry. However, if this is on a webpage, the word "Scotland" can form a link to another page (or another place on the same page) where more information is given about Scotland. As well as words, images or parts of images can act as links. For example, there could be a map of Scotland where each county on the map forms a link to more information about that county.

The address **www.ids.ac.uk** signifies a computer with the name **www** at **ids.ac.uk**, a *subdomain* of the Internet located at IDS (the Institute of Development Studies). This in turn is part of the **ac.uk** domain – the academic network in the United Kingdom.

If you enter just a directory name in a URL, e.g.

**<http://www.ids.ac.uk/blds/>**

the server will normally be set to send the page **index.htm** (or **index.html**) from that directory, if such a file exists. If it does not, the server may send a listing of the files in that directory – or the webmaster may have turned off that facility for security reasons.

There may be several layers of directories (or *folders* in Windows terminology) as well as a filename, e.g.

**<http://www.ids.ac.uk/blds/kenya/docdel/introduction.html>**

The files stored in the *root* directory and its subdirectories form the IDS *website*. The root directory is specified by the webmaster when the server is set up.

Each webpage is specified by an HTML file. The filenames may end in **.htm** or **.html**. The original extension was **.html**, which was allowed in Unix, the operating system of the early web servers. DOS and early versions of Windows could handle only three-letter extensions so **.htm** came into use. Now you will see both, but it is as well to pick one or the other and be consistent on your own website.

## 2.3 HyperText Markup Language

Hypertext pages are written in a language called HyperText Markup Language (HTML). It consists of the text that the user sees on the screen plus codes called *tags* which add special effects to the text, e.g.

```
<H2>How to order documents</H2>.
```

Here the `<H2>` tag specifies that what follows is to be displayed as a level two heading and the `</H2>` tag turns off this specification. Exactly how a level two heading should be displayed is at the discretion of the browser (or controlled by a stylesheet). The original intention was for HTML to specify the logical divisions of the text and leave its appearance to the browser, but there are now settings for point size, colours, fonts, etc., so the intention is partly thwarted.

Most tags are used in pairs like the above example, specifying the start and end of an *element*. Elements are nested inside each other. The whole page is written between the tags `<HTML>` and `</HTML>`. The main division of the page is into the Head (which contains the Title and other information) and the Body, e.g.

```
<HTML>

<HEAD>
<TITLE>My first web page </TITLE>
</HEAD>

<BODY>
Hello world!<BR>
This is your Webmaster speaking.
</BODY>

</HTML>
```

Spaces and blank lines have no significance in HTML, so all the above could be written on one line. It was laid out as it was just for clarity to the (human) reader. An example of a tag not used in pairs is `<BR>`, which inserts a line break. HTML tags are not case-sensitive but upper-case makes them easier to distinguish from the text. In this Handbook we use upper-case partly to distinguish HTML tags from IsisScript tags.

Tags often have one or more *attributes* to define the properties of the element that follows, for example

```
<FONT FACE="Arial" COLOR="red" SIZE="2">Some text</FONT>
```

The attributes may be placed in any order. The double inverted commas are strictly necessary only if the value contains a space or special character.

It is not a function of this Handbook to teach you all you need to know about HTML. There are plenty of other books available.

## 2.4 Other markup languages

A markup language is a way of placing codes within text in a way that they can be distinguished from the text (and a specification of what those codes are and mean). HTML is one example of a markup language – the codes can be distinguished because they are enclosed in `<...>`. A different (non-standard) way of coding text would be to use special symbols, e.g. \* for "put the next word in italics", and % for "put the next word in bold", so we could have:

```
Here is a *fictional %markup %language
```

The Standard Generalized Markup Language (SGML) was defined by the International Standards Organization in 1986 as ISO 8879. It provides a structure for markup languages, e.g. how the codes are distinguished from the text, but it doesn't specify what they are. If you are familiar with ISO 2709 and the MARC formats used in library cataloguing, this is a similar kind of relationship. ISO 2709 specifies the structure and the MARC formats specify what the data elements are within that structure. HTML is one application of SGML.

Another language, XML (Extensible Markup Language) has appeared, derived from SGML. It is a simple version of SGML that can be used to write programs. Again it is a "template" for creating languages rather than a language itself. One well-known application of XML is Channel Definition Format (CDF) which was used to create channels in Internet Explorer 4. IsisScripts which are used in WWWISIS version 4 onwards are another application.

## 2.5 The Common Gateway Interface (CGI)

We mentioned in Section 2.2 above that static web pages are held on the server in the root directory or in subdirectories of that directory. Typically the root directory is called **wwwroot**. Any files to be sent to the client, whether text pages, images, sound files, etc. need to be stored there. When the webserver software is configured, the root directory has to be specified and that determines what the website will consist of.

The webserver also recognises another directory, usually called **cgi-bin** (for CGI binary files). This directory contains program files and the programs can be run on the server by specifying them in the URL, e.g.

```
http://www.buxton.org/cgi-bin/myprog.exe
```

Normally the program will produce some output in HTML and this will be passed back by the server to the client as a *dynamic* page. Next time the program is run it

may produce different output because different values of input variables may have been passed to it through the Common Gateway Interface.

The gateway programs may be of various types. They can be compiled programs written in any language (e.g. C++ or Visual Basic) or executable scripts written in special scripting languages like Perl, Cold Fusion, or the Isis scripting language.

When the program runs, we normally want to control it by specifying some parameters, e.g. "Find me books about RAILWAYS and display the results in the DETAILED format." The user can enter these parameters on a form, which is a special kind of HTML page, and the values are passed through the CGI interface. They are sent as *name-value pairs*, separated by & signs, e.g.

```
searchterm=railways&format=detailed
```

The program may need some other parameters (e.g. "Which database does this person want to search?"). These can be built into the form but *hidden* from the user, so that the user is not aware that they are being sent along with the values he/she has entered on the form.

We will be dealing further with forms and passing values through CGI in Chapter 5.

## CHAPTER 3

### INSTALLATION OF WWWISIS

The installation of WWWISIS itself is very easy, but there are some preliminaries that you need to deal with first. This chapter considers installation of the Windows version only.

#### 3.1 IP Settings

As explained in Section 2.1, you can use a single computer as both server and client perhaps for testing purposes or while you learn about IsisScripts. In that case you will not need to worry about the network settings.

However, the general reason for running WWWISIS is to give other people access to your database(s) over an intranet or over the Internet. In this case you will need to make sure that the computer you are using as the server is connected to the network when users want to access the database. Normally this implies a permanent network connection. Each computer on the Internet or on an intranet that operates on Internet Protocol (IP) is identified by an IP number, such as 139.184.17.25. Sometimes the IP number is assigned only when the computer makes a network connection. This *dynamic* allocation has the advantage of saving on numbers, since the same number can be used for different computers provided they are not connected simultaneously. However, it means that the number of the computer may change from one day to the next, which is most unhelpful for a server! So you should make sure that the computer you are using as server has a permanently-assigned (or "hard coded") IP number. (Common dynamic systems are BOOTP and DHCP, though use of these does not necessarily mean that your IP address changes each time you connect.)

Corresponding to the IP number is an IP address, e.g. **www.ids.ac.uk** (or, in fact, more than one address can be allocated to the same IP number). This is the form in which users will normally call your webserver, although they could use the number if they know it. The address is preferable because (a) it is easier to remember and gives an indication of what and where the site is, and (b) if the server computer breaks down or proves inadequate you can use another computer and make the name map to the IP number of the new computer.

Before proceeding with loading the server software, you will need to know some IP details of the computer you are planning to use. If it is running Windows NT, you can enter the command:

```
ipconfig /all
```

and this will give you all the information. If it is running another operating system there will be a similar command: if necessary ask your network manager for assistance.

## 3.2 Installing the webserver software

WWWISIS will work with various webserver software. However, it does not work with the O'Reilly WebSite server which is now very dated.

### 3.2.1 Omni HTTPd

OmniHTTPd is produced by Omnicron Technologies Corporation. There are two versions available by downloading from their website,

**<http://omnicron.ab.ca/httpd/download.html>**

- The freeware version runs under Windows 95 only. It is dated 18 June 1997 and is not being updated. This may be useful for learning or development, but Windows 95 has only rudimentary security and should not be used to offer web services on the internet.
- The Professional version runs under Windows 98/ME and Windows NT/2000.

After the installation program has been downloaded it can be run. By default, it will install into the folder **c:\httpd**. The actual program file is **ohttpd.exe**. If you are running it under Windows 95, you can put a line in your **autoexec.bat** file to run the program whenever your computer is started:

```
C:\HTTPD\OHTTPD.EXE
```

In Windows NT or 2000 it is probably most convenient to run it as a service. When the program is running, you can right-click on its icon on the taskbar and select Properties to get to the configuration details. Although there are eleven dialog boxes, there are only two that need attention to get started. In the Server dialog box you need the following settings:

*Site address:* Enter here the full IP address of the computer as explained in Section 3.1 above, e.g. **www.mycompany.org**

If you are only going to operate it as localhost, use the IP number 127.0.0.1 here.

*Server root:* This is explained in Section 2.2. All the HTML files which constitute your site must be placed in this folder or in subfolders of it. By default it will be set to **C:\httpd\HTDOCS**

*Default index:* This is the page which you want to display if a user specifies as URL your site address with no filename. Typically it will be set to **index.htm** or **index.html**.

*Admin email:* This should be the email address of the administrator of the website, to be used if there are any important messages, e.g.

```
Andrew@mycompany.org
```

The other dialog box to check is the Standard CGI box (not to be confused with Windows CGI). In this, there is a setting:

/CGI-BIN                    c:\httpd\CGI-BIN

This is the directory used for CGI programs, as explained in Section 2.5.

### 3.2.2 Apache

The Apache software is the package used on courses at BIREME. It is available for a wide variety of platforms from the Apache website, <http://www.apache.org> or from a mirror site. The version for Microsoft Windows is located in the directory /**dist/httpd/binaries/win32** and at the time of writing the latest version is **apache\_1.3.20\_win32\_no\_src-r2.msi**. After version 1.3.17 it has been available as a Microsoft Installer file (extension .MSI). This needs the Microsoft Installer program to install it. Windows 2000 and ME come with Installer Support: otherwise you need to download it.

When you run the Installer, the domain and name of your server and an email address are picked up automatically. (You may need to correct your email address.) If you are installing under Windows NT, you can choose whether to run Apache as a service or to start it up manually.

By default, it installs into a directory /**Program files/apache Group/Apache**. The configuration file **httpd.conf** is in a subdirectory **conf** and there are other subdirectories **cgi-bin** and **htdocs**.

### 3.2.3 Internet Information Server (IIS)

IIS version 3 comes with the Windows NT Server operating system. Version 4 can be used with the NT 4 option pack and version 5 comes with Windows 2000 Server. We would not recommend setting up IIS specially for WWWISIS, but if you have it already it can be used for with WWWISIS.

When you click the Internet Service Manager icon it brings up the "Microsoft Management Console". This shows the sites managed by your server. (There can be several). Highlight "Default Web Site" and click the right mouse button. This will bring up a set of dialog boxes and one is for specifying the Home Directory for the site. No CGI directory is created by default and you can create a directory or directories as you wish. However, you do need to right click on the directory name and set Execute permission.

## 3.3 Licensing of WWWISIS

WWWISIS Versions 4 and 5 can be downloaded free from BIREME's website. They will work if you have the client (i.e. the web browser) and the server (i.e. WWWISIS) on the same computer and call up **localhost** on the browser. However, if you want to use WWWISIS over a network, whether an intranet or the Internet, you will need a licence from BIREME. This licenses the program on one server for one year and covers updates released during the year. The program will in fact continue to work

and this is legal, but any new functions added during the year will no longer work. The licence is tied to the IP address of the server so you need to quote the IP address when applying.

The licence takes the form of a file **wxis.lic** and this needs to be copied into the same directory as **wxis.exe**. If you try to access WWWISIS over a network without this file being present you will receive an error message.

### 3.4 Installing WWWISIS

This is very easy.

Download WWWISIS Version 4 or Version 5 from BIREME's website:

**<http://www.bireme.br/wwwisis/I/download.htm>**

The capital I in the directory name specifies the English-language version of the downloading page. At the time of writing, the file size of Version 4 is 295 Kbytes.

Choose to *save* the program (**wxis.exe**) to disk, not to run it, and save it into the folder that you have configured for CGI programs.

If you have purchased a licence for WWWISIS, copy the **wxis.lic** file which you have been sent by BIREME into the same directory as **wxis.exe**.

### 3.5 Creating and loading databases

As explained in Chapter 1, the database structure for WWWISIS is compatible with that of CDS/ISIS. You can therefore create a database in any version of CDS/ISIS to use with WWWISIS. This is probably the most convenient way to set up a database when you are first learning to use WWWISIS. The databases can be held in any directory on the server, provided that:

- (a) you have a *cipar* parameter in your IsisScript to tell **wxis** where to find the database (see Section 4.7) and
- (b) the directory has suitable permissions set for the database actions you intend (e.g. read or write).

If your data contain special characters, such as diacritics, you will need to set up a gizmo file (see Section 4.8) to make them display correctly in a web browser.

The **mx** utility (see Section 1.2) can be used to create a database from an ISO file, including the index files. This provides a way of transferring data between a DOS or Windows system and a Linux or Unix system - you will need to export the database from the CDS/ISIS database as an ISO file and import it into the WWWISIS system. You can also create a database for use with WWWISIS from an ISO file produced by other library software. Note that the ISO file must be in the format produced by the DOS version of CDS/ISIS, i.e. with a carriage return after every 80 characters. The end-of-field and end-of-record markers may be DOS version default values, i.e. hash signs (#), or the Windows version default values (ASCII characters 30 and 29 respectively).

A batch file **loadiso.bat** is provided which needs two parameters, the name of the ISO file and the name of the database to be created, e.g.

```
loadiso books jan.iso
```

No Field Definition Table is needed for a WWWISIS database. However, you will need to create a Field Select Table (FST) to allow searching and one or more display formats. These may be written in a text editor, if you do not have CDS/ISIS, and the inverted file can be built with another utility based on **mx.exe**, **fullinv.bat**. This needs three parameters, the name of the database, the name of the FST and the name of the inverted file. (The last two have the same name as the database in CDS/ISIS.) For example,

```
fullinv books books.fst books
```

## CHAPTER 4

### INTRODUCTION TO IsisScripts

For clarity, tags are shown in **bold** in this chapter and Chapter 5.

IsisScripts are text files and may be written in any text editor or in a word processor provided that you save the file as text. You could also use an editor intended for writing HTML or another markup language, especially if it lets you define your own customised tags.

#### 4.1 Extensible Markup Language (XML)

As explained in Section 2.4, the IsisScript language is an application of Extensible Markup Language (XML). It resembles HTML in that elements are specified by means of tags which are written in angle brackets. For example, in HTML a level 2 heading is specified like this:

```
<H2>Sources of further information</H2>
```

In IsisScript, this text could be displayed using

```
<display>Sources of further information</display>
```

IsisScript tags often occur in pairs, e.g. **<display>** and **</display>** indicating the start and close of that element. Tags may specify one or more *attributes*, each set to some *value* with an equals sign, e.g.

```
<field action=cgi tag=3020>
```

Tags can also be nested, i.e. one element can form part of another, e.g.

```
<display><pft>'Database name is ',v10</pft></display>
```

Here, what is to be displayed is specified by means of an ISIS print format and that is enclosed by the tags **<pft>** and **</pft>**. If this was not written as a print format, the characters v,1 and 0 would be displayed. Note in this example that no text is allowed between **<display>** and **<pft>** or between the **</pft>** and **</display>** tags. That is why 'Database name is' is written as a literal in the print format.

#### 4.2 IsisScript

An HTML page is an element beginning with the tag **<HTML>** and ending with **</HTML>**. IsisScripts begin with the tag **<IsisScript>** and end with **</IsisScript>**. Note that unlike HTML, IsisScripts are case sensitive – you must use a capital I and capital S in the tags.

You can include a name in the IsisScript tag, e.g.

```
<IsisScript name=MyFirstScript>
```

### 4.3 Content type

Any CGI program must tell the server what kind of output it is trying to send back to the client. Normally this will be an HTML document, i.e. text/html (though it could be e.g. text/plain). This information is sent as a header specifying the "Content-type" which **must** be followed by a blank line:

```
<display><pft>'Content-type: text/html'##</pft></display>
```

This statement must be written in the ISIS formatting language using ## (or /#) to produce the blank line. You cannot use the HTML <br> element here because the server has not yet been told to expect HTML!

You can now put together a complete script:

```
<IsisScript name=Hello>
  <display><pft>'Content-type: text/html'##</pft></display>
  <display>Hello World!</display>
</IsisScript>
```

If you save this in a file called **hello.xis** in your /cgi-bin directory, you can try it out using the following web page:

```
<HTML>
<BODY>
<A HREF=http://localhost/cgi-bin/wxis.exe/?IsisScript=hello.xis>Run Hello
script</A>
</BODY>
</HTML>
```

### 4.4 Tasks

The script can perform one or more tasks. Each task starts with a <do> tag and ends with a </do> tag. The following types of task are available:

(a) <do task=mfnrange> this goes through a database record by record within the range of MFNs specified

(b) <do task=search> this performs a search on a database for a specified search expression

(c) <do task=keyrange> this allows access to the inverted file. (The terms in the inverted file are called "keys".)

(d) <do task=update> this updates the database by adding, deleting or modifying records

(e) `<do task=list>` this operates on a list (e.g. fields extracted from a database) rather than on a normal ISIS database

## 4.5 Parameters

Within the task, you need to specify some *parameters* to tell WXIS the details of the task. This is done using the tags `<parm>` and `</parm>`. Each parameter has a name attribute and a value specified between the tags, e.g.

```
<parm name=from>1</parm>
```

A crucial parameter for most tasks is the name of the database to be used, e.g.

```
<parm name=db>cds</parm>
```

The WWWISIS *Reference Manual* tells you what parameters can be used in what tasks.

## 4.6 Loops

Usually in a task relating to a database or list you want to do the same thing to each one of a set of records, e.g. display each record retrieved by a search or each term in a section of the inverted file. For the tasks *mfrange*, *search* and *keyrange* this is done using a *loop* – the loop starts again for each record until the last record in the set has been processed. The loop is specified with the tags `<loop>` and `</loop>`. For the task *update*, the tags `<update>` and `</update>` are used.

You can use a loop in the `hello.xis` script to display the message several times (specified by the parameter `to`). This does not need a task specifying as it is not operating on a database. The loop part is indented in this example to emphasize where the loop starts and ends.

```
<IsisScript name=Hello>
<display><pft>'Content-type: text/html'##</pft></display>
<do>
  <parm name = to>10</parm>
  <loop>
    <display>Hello world!<br></display>
  </loop>
</do>
</IsisScript>
```

A simple script to display the MFN and title of the first ten records in the CDS database could be written like this.

```
<IsisScript name=ShowTen>
<display><pft>'Content-type: text/html'##</pft></display>
<do task = mfrange>
  <parm name=db>cds</parm>
  <parm name=from>1</parm>
  <parm name=to>10</parm>
  <loop>
```

```

        <display><pft>mfñ,' - ',√24,'<br>'</pft></display>
</loop>
</do>
</IsisScript>

```

## 4.7 cipar

The example above assumes that the CDS database is held in the same directory as the Isis script. Usually this is not desirable – you want to keep the database(s) in one directory and the scripts in the /cgi-bin directory. So you need to tell WXIS where to find the database. This is done using a parameter called **cipar**. It is convenient to place it at the beginning of the script before the tasks. An example would be:

```
<parm name=cipar> cds.*=c:\bases\cds\cds.*</parm>
```

This indicates that when files with the name cds and any extension are referred to in the script they are to be found as files with the same name in the c:\bases\cds directory. (In technical terms, you are saying that the logical name cds.\* is equivalent to the physical name c:\bases\cds\cds.\*) Any other files not in the cgi-bin directory should be specified in the same way. To specify more than one equivalence you can use the **<pft>** tag and make each one a literal, e.g.

```

    <parm name=cipar>
<pft>
'cds.*=c:\bases\cds\cds.*'/
'short.pft=c:\bases\cds\short.pft'/
'error.htm=c:\httpd\docs\error.htm'/
    </pft>
</parm>

```

## 4.8 gizmo

Another useful parameter is **gizmo**. It makes a reference to a gizmo database. For example, if you have a gizmo database called GIZ1, the line would read:

```
<parm name=gizmo>giz1</parm>
```

A gizmo database is used to convert a character or string of characters in the main database into another character or string of characters for display purposes. Most often, it is used to convert "special characters" such as é, ç or Ñ into text representations which will display correctly in HTML (*&eacute;*; *&ccedil;*; and *&Ntilde;*). The text versions can be found in books on HTML.

The gizmo database is a normal CDS/ISIS database with these two fields defined:

- 1 Input value
- 2 Output value

The data entry worksheet and display format can be very basic and you do not need a Field Select Table. Each special character and its text version are then entered in a separate record in the database, e.g.

MFN=1 1: é 2: &eacute;
MFN=2 1: ç 2: &ccedil;

Note that the location of the gizmo file must be specified with the **cipar** parameter, e.g.

```
<parm name=cipar> giz1.*=c:\bases\giz1.*</parm>
```

#### 4.9 Flow control

It is a frequent requirement in IsisScripts, as in other computer programs, to take different actions depending on the outcome of some test (a "condition"). For instance, we may want to give the user a choice of two methods of searching (a) by entering a Boolean expression or (b) by selecting from the index file. If (a) is chosen we present one search form (**search.htm**): if (b) is chosen we present another form (**index.htm**). The user chooses by clicking a radio button which gives the variable named "choice" the value BOOL or the value INDEX. This value is put by the IsisScript into the field 5001 using the statement

```
<field action=cgi tag=5001>choice</field>
```

We can now jump to the appropriate part of the script using a flow element with the action attribute set to "jump", i.e.

```
<flow action=jump><pft>v5001</pft></flow>
```

This causes a jump to the label whose name is in field 5001, i.e. to BOOL or to INDEX. The labels are indicated in the script as *label* elements:

```
<label>BOOL</label>  
(statement to display SEARCH.HTM)  
<label>INDEX</label>  
(statement to display INDEX.HTM)
```

The statements to display the forms are slightly complicated and are not the point here. However, we need to make sure that after SEARCH.HTM is displayed WXIS does not go on processing the script and also display INDEX.HTM. This is achieved by another flow element, this time with the action *exit*, which ends the execution of the IsisScript. So we have the structure:

```
<flow action=jump><pft>v5001</pft></flow>  
<label>BOOL</label>  
(statement to display search.htm)
```

```
<flow action=exit>1</flow>
<label>INDEX</label>
(statement to display index.htm)
<flow action=exit>2</flow>
```

(The argument of the *flow* element, 1 or 2 in this example, is the return code that is passed to the operating system.)

#### 4.10 Comments

It is a very good idea when writing IsisScripts to include explanations of what is happening. This helps anyone else trying to understand your scripts – and you when you come back to a script you wrote six months ago. Comments are written within the tag `<!-- -->`, for example:

```
<!-- Write the document header -->
<display><pft>'Content-type: text/html'##</pft></display>
```

Such comments are ignored by WWWISIS – they are only for human consumption!

## CHAPTER 5

### CGI VARIABLES AND ENVIRONMENT VARIABLES

#### 5.1 Common Gateway Interface (CGI)

If you want the script always to operate on the same data and produce the same results, you can specify the values of any parameters needed in the script itself. The example in Section 4.6 always operates on the cds database, always displays records 1 to 10, and always uses the display format mfn, dash and title (field 24).

More usually, you want the user to have some say in what is done, e.g. which records to display and perhaps which format to use. To do this, you need:

- (1) to present the user with a form written in HTML with some textboxes or other controls
- (2) to read the values which he or she chooses, and
- (3) to use them as parameters in the IsisScript. The third operation is achieved by means of the CGI (Common Gateway Interface) environment.

#### 5.2 The input form

The form which appears as a page (or part of a page) in the web browser will be written in HTML. It begins with the **FORM** element which has the attributes **ACTION** and **METHOD** and ends with **</FORM>**. The **ACTION** attribute states what you want to happen when the form is submitted – i.e. run wxis.exe with the relevant IsisScript. The **METHOD** specifies how the form data are sent to the server. The most common values are POST and GET: POST is now the preferred method and the use of GET is deprecated.

To allow input by the user, the form will contain one or more of the following controls.

##### (a) Text box

This is a box in which the user can type some text, e.g.

Enter your search:

In HTML this would be coded:

Enter your search:  
<**INPUT** TYPE=text NAME=expression>

Whatever the user types in the box is the value that will be assigned to CGI variable called “expression”.

You can specify the width of the textbox (in characters), e.g.

```
<INPUT TYPE=text NAME=db SIZE=10>
```

## (b) Text area

A textbox allows the user to enter only a single line of data: pressing the Enter key moves on to the next control. If you want to allow several lines of text, such as occurrences of a repeating field, you need a textarea. For example:

Field 2:

This would be coded:

```
Field 2: <TEXTAREA rows=3 cols=30 name=field2></TEXTAREA>
```

The text which the user types in the textarea will be assigned to the variable “field2”. (The code needed to separate the occurrences of the field is quite straight-forward.) Setting rows=3 defines the height of the textarea but the text scrolls so more than 3 rows can be entered. Setting cols to 30 makes the area thirty characters wide.

## (c) Option list

This is a drop-down list where the user can select from a predefined set of values, e.g.

Select your display format:

 ▼

When the user clicks on the arrow, the drop-down list opens:

Medium	▼
Short	
Medium	
Long	

In HTML this would be written:

Select your display format:

```
<SELECT NAME=format>  
  <OPTION value=1> Short  
  <OPTION value=2 SELECTED> Medium  
  <OPTION value=3> Long  
</SELECT>
```

When the user selects one of the options, the CGI variable “format” will be set to 1, 2 or 3 corresponding to the name selected. The one shown before the user chooses is set by the attribute SELECTED.

By including the attribute MULTIPLE in the SELECT element, you can allow the user to choose more than one value (not appropriate here!).

### (d) Check box

This is a box that the user can “check” (i.e. click in) to make a yes/no or true/false choice, e.g.

Search titles only

In HTML it is coded:

```
<INPUT TYPE=checkbox NAME=titles>Search titles only
```

### (e) Radio button

This allows the user to select only one of a predefined list of options. Because the form has to show all the all the options all the time, it is suitable only where the possibilities are few. For example:

Boolean search  
Index search

In HTML this appears as:

```
<INPUT TYPE=radio NAME=searchtype VALUE=BOOL CHECKED>
```

Boolean search

<br>

```
<INPUT TYPE=radio NAME=searchtype VALUE=INDEX> Index  
search
```

The CGI variable “searchtype” will be given the value `BOOL` or `INDEX` depending on which option is set. The attribute “CHECKED” means that this button is selected by default.

### (f) Hidden variables

The `INPUT` element with `TYPE=hidden` defines an invisible input field whose value will be sent along with the other form values when the form is submitted. It can be used to send certain values that `wxis.exe` needs for execution (e.g. the name of the IsisScript):

```
<INPUT TYPE=hidden NAME=IsisScript VALUE=mysearch.xis>
```

Another use is to pass information from one form to another, without the user having to retype it.

## 5.3 Reading the values

The form always needs to contain another control, the Submit button, to send the form data back to the server and carry out the action specified in the form element. The text on the button can be whatever you like, e.g.

Proceed

It is coded in HTML:

```
<INPUT TYPE=submit VALUE="proceed">
```

## 5.4 Using the values in an IsisScript

To make use of the form values in an IsisScript, they need to be read from the CGI environment and stored in the WWWISIS “virtual record”. This is a record held in memory which is not actually part of the database(s) being accessed and which can contain whatever fields you choose. Each value is read in using a **field** element with the **action** attribute `cgi` and the **tag** attribute set to the chosen field number, e.g.

```
<field action=cgi tag=3010> expression </field>
<field action=cgi tag=3020>format</field>
```

This means that whatever expression the user entered will be stored in field 3010 and the format that he/she chose from the option list will be stored in field 3020. You cannot use the values from CGI without first reading them into the virtual record. CGI variables can be read anywhere in an IsisScript – you do not need to read them all at the beginning.

Once they have been read, you can use the values in a **parm** element by using an ISIS print format between the tags `<pft>` and `</pft>`, e.g.

```
<parm name=expression><pft>v3010</pft></parm>
```

So to summarize, the way that a value entered on a web form affects the operation of WWWISIS is:

Text on form -> CGI variable -> field in virtual record -> IsisScript parameter

## 5.5 Scope of variables

Typically you will have read values from the CGI environment into fields of the virtual record outside of the loop. Within the loop you are working through real database records and you cannot access the virtual record. You therefore need to *import* the variables you wish to access within the loop, using the `action=import` attribute of the field element, e.g.

```
<loop>
<field action=import tag=3020>3020</field>
...
</loop>
```

Note that in this statement the `v` must be omitted in specifying field 3020.

The value in field 3020 can then be used within the loop to control the display format, e.g.

```
<loop>
<field action=import tag=3020>3020</field>
<display><pft>if v3020='1' then @short.pft
```

```

                else if v3020='2' then @medium.pft
                fi fi
        </pft></display>
</loop>

```

(For this example to work, the locations of short.pft and medium.pft must have been specified in a cipar parameter.)

## 5.6 Environment variables

As well as reading in values from the HTML form, you can read in certain “environment variables” to which WXIS gives values while it is processing. Some examples are:

**(a) Isis\_Current.** This is the current index of the processing loop, i.e. the first time through the loop it is set to 1, the second time to 2, and so on. If the task is *mfnrange* it will be equal to the MFN of the record being processed, but if the task is *search* it will be 1 for the first record retrieved, 2 for the second, etc.

**(b) Isis\_Total.** If the task is *mfnrange*, this will be the number of records in the database. If the task is *search*, it will be the total number of records retrieved. You can use Isis\_Current and this variable to display the results as “Record 1 of 6”, “Record 2 of 6”, etc. as shown in Section 6.2e. You can also test for zero records retrieved and display an appropriate message.

**(c) Isis\_Status.** This variable provides a way of checking that an operation with WXIS has been completed successfully. It should be tested after the end of the loop. The value 0 indicates success: a value greater than 0 indicates an error.

Like the CGI variables, Isis environment variables need to be copied into numbered fields. This is done by means of a **field** element with the action attribute set to define, e.g.

```

<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1002>Isis_Total</field>
<field action=define tag=1020>Isis_Status</field>

```

In WWWISIS version 3, environment variables were assigned to numbered fields automatically. In version 4 you need to assign them as above. There is nothing special about using tags 1001, 1002 and 1020 – just that they were used for these values in version 3!

We shall see more about how to use these environment variables in Chapter 6.

## CHAPTER 6

### PERFORMING A SEARCH AND DISPLAYING THE RESULTS

Probably the most common use of WWWISIS is to provide an online catalogue for users to search over the WorldWideWeb. We will look at a simple script for carrying out a search and then at how some refinements can be made.

#### 6.1 A simple search script

We have already seen in Section 5.2 how to construct a form in HTML. For searching, there will be at least a textbox for the search expression and a submit button. The search page could be written something like this:

```
<HTML>
<HEAD>
<TITLE>My search form</TITLE>
</HEAD>

<BODY>
<H2>Simple search example</H2>
Please enter your search.....
<BR>
<FORM ACTION="http://localhost/cgi-bin/wxis.exe/" METHOD=post>
<INPUT TYPE=hidden NAME=IsisScript VALUE="mysearch.xis">
<INPUT TYPE=text NAME=searchspec>
<INPUT TYPE=submit NAME=mysubmit VALUE="Do search">
</FORM>
</BODY>
</HTML>
```

The body contains a form with the action set to run wxis.exe and the method of exchanging data set to post. The value of the IsisScript parameter, specifying the name of the script, is sent as a hidden value. Then there is a textbox and whatever search term(s) the user types into the box will be assigned to the CGI variable called “searchspec”. Finally there is a submit button which will carry the caption “Do search”.

The IsisScript can be written in Notepad or any text editor.

```
<IsisScript name=mysearch>
<display><pft>'Content-type: text/html'##</pft></display>
<parm name=cipar> cds*=c:/bases/cds/cds.*</parm>
<field action=cgi tag=3020>searchspec</field>
```

These statements have all been explained above: beginning the script, sending the HTML header, specifying where the database is, and copying the value of the CGI variable “searchspec” into field 3020 of the virtual record. Now we start the search task.

```
<do task=search>
  <parm name=db>cds</parm>
  <parm name=expression><pft>v3020</pft></parm>
```

Two parameters need to be specified: *db* must contain the name of the database and *expression* must contain the search expression (read from field 3020 in the virtual record).

Now we want a loop to display each record retrieved using the CDS print format:

```
<loop>
  <display><pft>@CDS.PFT</pft></display>
</loop>
```

In the ISIS formatting language, the name of a format is specified by prefixing it with an “@” sign. WXIS.EXE must be able to find this file by means of the *cipar* parameter.

We can now close the script:

```
</do>
</IsisScript>
```

The complete script can be saved in a file called *mysearch.xis* in the *cgi* directory defined for your server. Provided that you have the CDS database in the directory **c:\bases\cds** with a print format *CDS.PFT*, you should be able to use the form to do a search and to display the results in a browser. Make sure that you have your web server started and that you use **http://localhost** in the URL, not **file://c.**

## 6.2 Enhancing the display of the results

If you have used the *CDS.PFT* (which dates from the days of the DOS version of CDS/ISIS) you will find that the results are not very attractive. You can improve the format for use in a web browser by incorporating some HTML tags as literals.

### (a) Spacing

To produce new lines or blank lines, you can use these HTML tags:

```
<BR> to produce a new line
<P> to produce a new paragraph
<HR> to produce a horizontal line
```

### (b) Character formatting

You can put text into bold or italics using these tags:

```
<B> and </B> to display one or more fields or literals in bold
<I> and </I> to display one or more fields or literals in italics
```

You can also use attributes of the FONT element to change the appearance of the text, e.g.

```
<FONT FACE="arial" COLOR="#00CC00">
```

This will change to Ariel font and blue text colour. (Note the American spelling of color.) So we can make the CDS.PFT format look a bit more interesting by changing it to this:

```
'<B>',MFN(4),'</B>' - ',V12,'<FONT FACE = arial COLOR=red>', V24,'</FONT>', (|
(|V76^Z|: |,V76^*|) |),V70+|; |,V25,V26,V30,(|V44|),V50,/|// |V71/|// |V72/|// |V74/
"<BR><I>KEYWORDS: ",V69,'</I><HR>'
```

Now we have the MFN displayed in bold, the title in red and in Ariel font (which is cancelled after displaying field 24), a new line before the keywords (which are shown with the name KEYWORDS in italic) and a horizontal line between records. The result in black-and-white is like this:

**0005** - Anti-transpirants as a research tool for the study of the effects of water stress on plant behaviour. Gale, J.; Poljakoff-Mayber, A. 1965. p. 269-274, illus.  
(Methodology of plant eco-physiology: proceedings of the Montpellier Symposium)  
Incl. bibl.

*KEYWORDS: plant physiology; soil moisture; plant transpiration; evapotranspiration; measurement and instruments.*

If you want to change to Times New Roman font you would normally write its name (which is more than one word) in inverted commas:

```
<FONT face="Times New Roman">
```

This works all right within an unconditional literal (enclosed in single inverted commas). However, double inverted commas are also used to indicate a conditional literal in the CDS/ISIS formatting language, and it may be better just to use </FONT> to revert to the default font.

### (c) Background

You can change the background colour used in the display using the BGCOLOR attribute of the BODY element. This could be done immediately after the HTML header in the IsisScript, e.g.

```
<IsisScript name=mysearch>
```

```
<display><pft>'Content-type: text/html'###</pft></display>
```

```
<display><BODY BGCOLOR="FFFFCC"></display>
```

### (d) Prolog and Epilog

Version 3 of WWWISIS had the facility to specify a format to display something before the results (the Prolog) and after the results (the Epilog). In an IsisScript you can use the DISPLAY element to display text at any point, so you could produce a prolog and epilog like this:

```
<IsisScript name=mysearch>
<display><pft>'Content-type: text/html'####</pft></display>
<field action=cgi tag=3020>searchspec</field>
<parm name=cipar>cds.*=c:\isis\data\cds.*</parm>
<display><H2>Start of results</H2></display>
      <do task=search>
      ....
<display><H2>End of results</H2></display>
</IsisScript>
```

### (e) Numbering of results

If you wish, you can number the records retrieved as “Record 1 of 6”, “Record 2 of 6”, etc. To do this you can use two of the Environment Variables referred to in Section 5.6. Isis\_Total is the total number of records retrieved and Isis\_Current is the current record being processed in the loop. These variables must first be copied into numbered fields:

```
<do task=search>
...
<field action=define tag=1001>Isis_Current</field>
<field action=define tag=1002>Isis_Total</field>
```

They can then be displayed for each record in the loop:

```
<loop>
  <display><pft>'Record ',v1001,' of ', v1002 </pft></display>
```

Note that you need <pft>...</pft> within the display element in order to get the *contents* of fields 1001 and 1002.

### (f) Displaying the number of hits

Rather than displaying “Record 1 of 6” etc. you may want to show the total number of hits before displaying any records. Since this number is not available before the loop is started, you need to show it as part of the display of the first record only. Assuming that you have defined tags 1001 and 1002 as above, you can test for the first record by looking at the value of v1001:

```
<loop>
  <display><pft>if val(v1001) = 1 then
    'Records found: ',v1002,'<br>' fi </pft></display>
  <display><pft>@cds.pft</pft></display>
</loop>
```

The value in field 1002 is available after completing the loop, so if you want to display the number of hits after all the records you can put the display element after the loop.

### **(g) Displaying a message if no records are retrieved**

If your search retrieves no records, nothing will display and there will be no indication of what the problem is. So after the loop ends, you may want to have a statement:

```
<display><pft>if val(v1002)=0 then  
'No records found' fi</pft></display>
```

If you really want to be helpful, you could have a whole screen full of information about why this might have happened (e.g. the user has mis-spelled the search term, or used the wrong format of author name, etc.) You could compose an HTML page to explain all this and call it NOHITS.HTM. Then you can display it instead of the rather terse "No records found".

The way to include an HTML page within an IsisScript is to use the **cat** keyword. (This derives from a command in Unix which is used to concatenate text files.)

```
<display><pft>if val(v1002)=0 then cat(NOHITS.HTM) fi </pft> </display>
```

Remember to mention NOHITS.HTM in the cipar parameter or WXIS will not be able to find it.

### **(h) Displaying the results in a table**

A neat way to display the results of a search might be like this:

MFN:	3
Author:	Jones, John
Title:	Balliol College: a history
Publisher:	Oxford University Press

You can do this by using a table in your display format. In the above example, each table row contains two data items – the literal which is the name of the field and the field contents. At its simplest, the display format can be written like this:

```
'<TABLE>  
<TR> <TD> MFN: </TD> <TD>' mfn, '</TD> </TR>  
<TR> <TD> Author: </TD> <TD>' v70, '</TD> </TR>  
<TR> <TD> Title: </TD> <TD>' v24, '</TD> </TR>  
<TR> <TD> Publisher: </TD> <TD>' v26, '</TD> </TR>  
</TABLE>'
```

Note that everything apart from the variables indicating the data fields is part of a literal. The table is enclosed in <TABLE>...</TABLE>, each row is enclosed in



```

<HTML>
<BODY>
<FORM action="http://localhost/cgi-bin/wxis.exe/" method=post>
  <INPUT type=hidden name=IsisScript value="mysearch2.xis">

```

Term 1:<INPUT type = text NAME = term1>

```

<SELECT NAME=operator>
<OPTION VALUE=* SELECTED>AND
<OPTION VALUE=+>OR
<OPTION VALUE=^>AND NOT
</SELECT>

```

Term 2:<INPUT type = text NAME = term2>

```

<INPUT TYPE=SUBMIT VALUE = "Search">
</FORM>
</BODY>
</HTML>

```

Three values chosen by the user are passed to the CGI environment: **term1**, the first search term, **term2**, the second search term (if there is one), and **operator**, the operator selected from the drop-down list. In the HTML code for this list, VALUE is the value passed to CGI and the text after the tag is what is displayed on the screen.

The wxis.exe program is called with the IsisScript mysearch2.xis. This is as follows:

```

<IsisScript name=mysearch2>

<display><pft>'Content-type: text/html'####</pft></display>

<field action = cgi tag = 3020>term1</field>
<field action = cgi tag = 3021>operator</field>
<field action = cgi tag = 3022>term2</field>

<parm name=cipar>cds.*=c:\isis\data\cds.*</parm>

<display><H2>Start of results</H2></display>

<do task=search>

  <parm name = db>cds</parm>
  <parm name = expression><pft>v3020,if v3022<>" then ',v3021,'
  ',v3022 fi</pft></parm>

</loop>

```

```

<display><pft>'mfn=',mfn,'<BR><B>',v24'<HR><B>'</pft></display>

</loop>

</do>

<display><H2>End of results</H2></display>

</IsisScript>

```

The values of three CGI variables are put into the fields 3020, 3021 and 3022. These three are then used to form the search expression, with spaces before and after the operator. In case term2 is not present, an “if” condition is used to add the operator and term2 only if term2 is not empty.

Another enhancement to the search form would be to have different textboxes for different fields, e.g.

Author	<input type="text"/>
Title	<input type="text"/>

You might want to make provision for different operators to be used to link these terms, but for now we will assume that they are to be linked only with AND. The contents of the boxes will be assigned to variables which we will call “author”, and “title”. In the CDS database these correspond to fields 70 and 24.

In the IsisScript we copy them into fields 3070 and 3024:

```

<field action = cgi tag = 3070>author</field>
<field action = cgi tag = 3024>title</field>

```

In constructing the search expression, we need to check whether each textbox has been filled in. We only want the AND operator if both fields are present. We can do this as follows:

```

<parm name=expression><pft>
if v3020<>" then v3020,'/(70)' fi,
if v3020<>" and v3024<>" then '*' fi
if v3024<>" then v3024,'/(24)'
</pft></parm>

```

Note that field qualifiers such as /(70) are not implemented in WWWISIS Version 4.

## CHAPTER 7

### ACCESSING THE INVERTED FILE

This chapter describes a simple script for displaying part of the inverted (or index) file, allowing the user to choose where in the file to start and how many terms to display. With a more complicated script you could allow terms in the display to be used to perform a search.

#### 7.1 A form for displaying part of the index file

This is a simple form to allow the user to type in:

- (a) where to start the display (assigned to the variable startterm )
- (b) how many terms to display (assigned to the variable howmany)

When the form is submitted, it calls wxis.exe with the script myindex.xis

```
<HTML>

<HEAD>
  <TITLE>My index form</TITLE>
</HEAD>

<BODY>
  <H2>Show part of index</H2>
  <FORM ACTION="http://localhost/cgi-bin/wxis.exe" METHOD=post>
  <INPUT TYPE=hidden NAME=IsisScript VALUE="myindex.xis">
  Where to start:
  <input type=text name=startterm>
  <p>
  How many terms to display:
  <input type=text name=howmany>
  <p>
  <INPUT TYPE=submit value="Show index">
  </FORM>
</BODY>

</HTML>
```

#### 7.2 The IsisScript for displaying the index

The script starts in the usual way. The values of startterm and howmany are read from the CGI environment into the fields 4001 and 4002. Then a heading is displayed.

```
IsisScript name=myindex>
<display><pft>'Content-type: text/html'/#</pft></display>
```

```

<parm name=cipar><pft>'cds.*=c:\isis\data\cds.*' </pft></parm>
<field action=cgi tag=4001>startterm</field>
<field action=cgi tag=4002>howmany</field>
<display><h2>Part of index file</h2></display>

```

The task for working on the index file is "keyrange".

```

<do task=keyrange>

```

Values are assigned to the parameters for this task - db, from and count:

```

<parm name=db>cds</parm>
<parm name=from><pft>v4001</pft></parm>
<parm name=count><pft>v4002</pft></parm>

```

The key itself (i.e. the term in the index) is assigned to field 1 and the number of postings to field 2:

```

<field action=define tag=1>Isis_Key</field>
<field action=define tag=2>Isis_Postings</field>

```

To align the numbers of postings, the display is done in the form of a table:

```

<display><table></display>

```

Now we loop through the specified keys and for each one display a line in the table (between <tr> and </tr>) with two cells (between <td> and </td>). The first cell contains the value of field 1 and the second the value of field 2 – as defined above, not as in the database records.

```

<loop>
<display><pft>'<tr><td>',v1,'</td><td>',v2,'</td></tr>'</pft></display>
</loop>

```

This completes the table – and the script:

```

<display></table><h2>End of display</h2></display>
</do>
</IsisScript>

```

Here is an example of the output:

### **Part of index file**

PACIFIC ISLANDS	1
PAKISTAN	8
PAR	2
PARASITES	2
PART	1
PART-TIME COURSES	1

PATEL, A.M.		1
PAYS	3	
PECS	1	
PERDU	1	

**End of display**

## CHAPTER 8

### CREATING AND EDITING RECORDS

As well as searching a database through a WWW interface, WWWISIS allows users to add new records or edit existing ones. Adding new bibliographic records may be useful if you have a co-operative cataloguing system, or it could be used for web users to add their names and addresses to a database for placing orders or onto a mailing list. Editing records may be useful if you want libraries to be able to add their holdings to a periodicals list, or add comments or reviews to document records. If users are allowed to add or change bibliographic records, some kind of password protection will probably be needed.

#### 8.1 The Update task

Record creation and editing are achieved in WWWISIS by means of the Update task. This starts with `<do task=update>` and then three parameters must be specified: the database name, the Field Select Table (FST) name for updating the inverted (index) file, and the MFN of the record. For a new record, which does not have an MFN allocated yet, the value is "New". In CDS/ISIS the FST must have the same name as the database, but specifying it here ensures that the new or changed record gets inverted. An example for creating a new record is shown below:

```
<do task=update>
<parm name=db>cds</parm>
<parm name=fst><pft>cat('cds.fst')</pft></parm>
<parm name=mfnn>New</parm>
```

The update commands for this record are written between the tags `<update>` and `</update>` (rather than `<loop>` and `</loop>` since no looping is involved).

#### 8.2 Record locking

If you have a multi-user database, it is important to make sure that when one user is making changes to a record another user cannot make contradictory changes. The first user needs to *lock* the record while he or she is making changes so that other users are excluded. When the changes are complete, the record needs to be *unlocked* again. If it is a new record, no lock command is needed but an unlock is required. Locking is achieved by the `<write>` tag:

<code>&lt;write&gt;Lock&lt;/write&gt;</code>	get ownership of the record
<code>&lt;write&gt;Unlock&lt;/write&gt;</code>	release the record
<code>&lt;write&gt;NoUnlock&lt;/write&gt;</code>	don't remove the lock, if it exists – leave the lock information as it is
<code>&lt;write&gt;Delete&lt;/write&gt;</code>	delete the record

This `<write>` command is essential even with a single-user system. No change is made to the database unless you include it.

The lock information is usually written into field 1101 of the record, but this can be changed. Field 1101 is not a repeatable field and only one value can be held. You can also control for how many seconds the lock is applied:

```
<parm name=expire>3600</parm>
```

You can ask the user to login on the update form and then use his or her userID as the lock, or you can get the value from the environment variable REMOTE\_ADDR:

```
<parm name=lockid><pft>getenv('REMOTE_ADDR')</pft></parm>
```

The locking mechanism is different from that used in CDS/ISIS or Winisis, so you should not use those packages to update the database at the same time as using WWWISIS.

### 8.3 A form for creating new records

We shall take a simple example where the user is allowed to create a record with two fields only – 24 for Title and 70 for Author (repeatable) – but more fields could easily be added.

Here is the form:

```
<HTML>
<HEAD>
<TITLE>My data entry form</TITLE>
</HEAD>

<BODY>
<H2>Enter a new record - </H2>
<FORM ACTION="http://localhost/cgi-bin/wxis.exe" METHOD=post>
<INPUT TYPE=hidden NAME=IsisScript VALUE="dentry.xis">
Title:
<INPUT TYPE=text NAME=field24 SIZE=40>
<p>
Author(s):
<TEXTAREA ROWS=3 COLS=40 NAME=field70> </TEXTAREA>
<p>
<INPUT TYPE=submit value="Create record">
</FORM>
</BODY>
</HTML>
```

This form is not very different from the search form in Section 6.1. It contains an input box for the user to enter the Title and a textarea three rows high for entering the Authors. (More than three authors can be entered because the contents of the box scroll upwards.) The appearance of the form can be improved by putting the prompts and boxes in a table so that they line up nicely. The choice of the variable names "field24" AND "field70" is arbitrary – they could have been "fred" and "Jean", but the

names chosen remind you what fields the data go into. As in the search form, the name of the IsisScript is sent as a hidden value.

## 8.4 The IsisScript for creating new records

The script starts with some fairly standard stuff:

```
<IsisScript name=dentry>
<display><pft>'Content-type: text/html'/#</pft></display>
<parm name=cipar><pft>'cds.*=c:\isis\data\cds.*',/
'htm.pft=c:\isis\data\web.pft'</pft></parm>
```

Then we have the update task:

```
<do task=update>
  <parm name=db>cds</parm>
  <parm name=fst><pft>cat('cds.fst')</pft></parm>
  <parm name=mfn>New</parm>
  <field action=define tag=1102>Isis_Status</field>
```

The database name is assigned to the parameter "db". The "fst" parameter is required to specify what Field Select Table is to be used to invert the new record. The "mfn" parameter is set to New – to create a new record. Then a field with tag 1102 is defined to contain the environment variable Isis\_Status (see Section 5.6).

Now we can specify the update actions to take place on the new record:

```
<update>
<field action=cgi tag=24>field24</field>
<field action=cgi tag=70>field70</field>
<field action=replace tag=70 split=occ><pft>(v70/)</pft></field>
<write>Unlock</write>
```

The data for fields 24 and 70 are obtained from the CGI environment (using the same names as on the data entry field!). There is a complication with field 70 in that it might contain several lines. The action=replace splits the data into separate occurrences using the CDS/ISIS concept of a repeating group (v70 followed by new line). Finally we release the new record by using the <write> tag to unlock it. If we do not do this it will not be added to the database.

Now we check the value of field 1002, i.e. the status of the operation, and if successful we write the message "Created!". (We should also write an error message in case it is not successful.)

```
<display>
<pft>if val(v1102) = 0 then '<b>Created!</b><hr>' fi </pft>
</display>
```

Now we will display the new record with its MFN for confirmation. We will separate repeats of field 70 with a semi-colon and space.

```

<display><pft>'<font face=arial>
MFN: ',MFN,'<br>
Author: ',v70+|; |,'<br>
Title: ',v24,'</font></pft></display>

```

This completes the update and we add the closing tags:

```

</update>
</do>
</IsisScript>

```

To be used by the form in Section 8.1 above, this script must be saved in the **/cgi-bin** directory with the name **dentry.xis**

### 8.5 A form for editing an existing record

Again we will take a simple example to show what is involved. This form first asks for a userID, which is used in the record lock. (It is not checked against a list of allowed IDs in this example.) It allows the user to select which record to edit by entering its MFN. Then it allows him or her to enter some data in field 900, replacing anything that may be there already.

```

<HTML>
<HEAD>
<TITLE>My editing form</TITLE>
</HEAD>

<BODY>
<H2>Select record to be edited - </H2>
<FORM ACTION="http://localhost/cgi-bin/wxis.exe" METHOD=post>
<INPUT TYPE=hidden NAME=IsisScript VALUE="recredit.xis">

Please enter your userID:
<INPUT TYPE=text NAME=userid>
<p>
Please enter the MFN:
<INPUT TYPE=text NAME=whichmfn>
<p>
Please enter data for field 900:
<INPUT TYPE=text NAME=field900>
<p>
<INPUT TYPE=submit value="Add new data">

</FORM>
</BODY>
</HTML>

```

## 8.6 The IsisScript for editing a record

The script starts the same way as the one in Section 8.4:

```
<IsisScript name=recredit>
<display><pft>'Content-type: text/html'/#</pft></display>
<parm name=cipar><pft>'cds.*=c:\isis\data\cds.*',/
'htm.pft=c:\isis\data\web.pft'</pft></parm>
```

Next we need to read the values of whichmfn, field900 and userid from the CGI interface:

```
<field action=cgi tag=3001>whichmfn</field>
<field action=cgi tag=900>field900</field>
<field action=cgi tag=2002>userid</field>
```

Now we can start the update task. We use the userid held in field 2002 as the lockid.

```
<do task=update>
<parm name=db>cds</parm>
<parm name=lockid><pft>v2002</pft></parm>
<parm name=mfn><pft>v3001</pft></parm>
<parm name=fst><pft>cat('cds.fst')</pft></parm>
<field action=define tag=1101>Isis_Lock</field>
<field action=define tag=1102>Isis_Status</field>
```

The actual update operation is quite simple. We lock the record, delete all existing occurrences of field 900, write the new data in, and then unlock the record again:

```
<update>
<write>Lock</write>
<field action=delete tag=900>all</field>
<field action=cgi tag=900>field900</field>
<write>Unlock</write>
```

(You could delete just the first occurrence of field 900 by specifying 1 instead of all.) Now to check the record, we will display it using the HTM.PFT format with the new field 900 added and a horizontal rule:

```
<display><pft>@htm.pft,'New data = ,v900,'<hr>'</pft></display>
```

We will also check the value of field 1102 (Isis\_Status) and print a confirmation if its value is zero:

```
<display><pft>if val(v1102)=0 then
'<b>Update successful!</b>' fi</pft></display>
```

This completes the update and we add the closing tags:

```
</update>  
</do>  
</IsisScript>
```

## CHAPTER 9

### THE IAH INTERFACE

#### 9.1 Introduction

The IAH interface (Interface Alan Hopkinson) is an application developed in IsisScript by BIREME to permit searching of CDS/ISIS databases over an intranet or the Internet. Retrieval can be done either by typing in search terms or by selecting terms from the inverted file. It uses WWWISIS to do the searching but it provides a ready-made script with an attractive interface, saving you having to learn the scripting language and write scripts yourself. As delivered, it operates in the languages English, Portuguese and Spanish.

An example of the search form in English is shown in Figure 9.1 below.

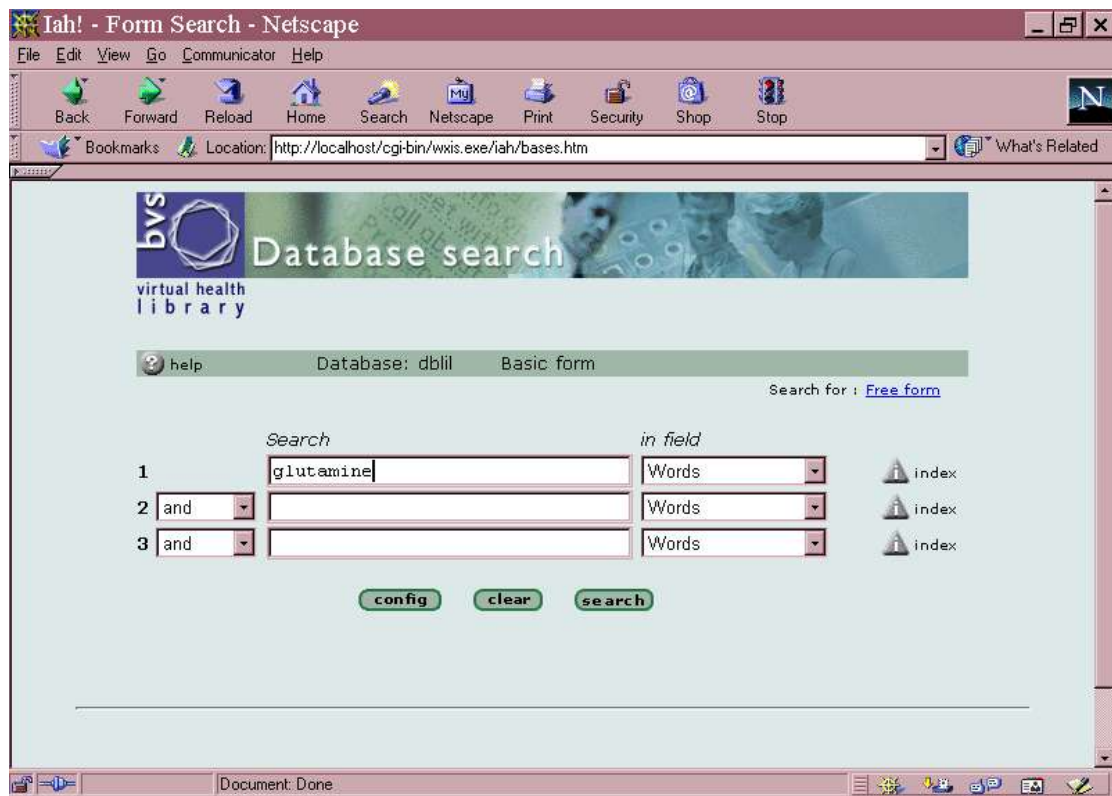


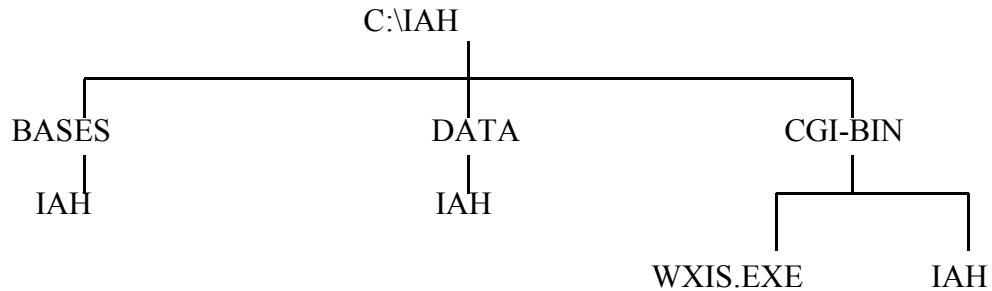
Figure 9.1 Basic search form in English

BIREME charges US\$500 for the package or \$2,500 with customisation. Because it uses WWWISIS it needs a licence file based on the server IP number to operate over a network.

Two sample databases are included: LILACS and SeCS.

## 9.2 Installation

The package is supplied on two floppy disks. Place the first disk in your disk drive and run the program `a:\iah.exe`. The suggested folder into which files will be installed is `c:\iah`. If you accept this, a folder structure will be set up as shown below:



The files which are in this structure need to be copied into your website folder structure. If you are using the Omni webserver, it will by default be installed in a folder `C:\HTTPD` with subfolders `HTDOCS` for the HTML document files and `CGI-BIN` for the CGI programs. You should proceed as follows:

- (a) create a folder `BASES` below `C:\HTTPD` and copy the `IAH` folder from `C:\IAH\BASES` to `C:\HTTPD\BASES`. This contains the sample databases.
- (b) copy the `C:\DATA\IAH` folder to `C:\HTTPD`
- (c) copy the `IAH` folder and the `WXIS.EXE` file from `C:\IAH\CGI-BIN` to the server `CGI-BIN` folder (`C:\HTTPD\CGI-BIN`)

## 9.3 Initial configuration

Configuration is done through the file `iah.def`, which will have been copied into the folder `C:\HTTPD\HTDOCS\IAH\` folder if you are using the Omni server (or the `IAH` subfolder of whatever folder you use for HTML documents). It is a text file and can be edited with a text editor such as Notepad. An example is shown below:

[PATH]

```
PATH_DATA=/iah/
PATH_CGI-BIN=/httpd/cgi-bin/iah/
PATH_DATABASE=/httpd/bases/iah/
```

[APPEARANCE]

```
BODY BACKGROUND COLOR=#E0EBEB
BODY BACKGROUND IMAGE=
BODY TEXT COLOR=black
BODY LINK COLOR=blue
BODY VLINK COLOR=blue
```

```
BAR BACKGROUND COLOR=#A5BDAD
BAR TEXT COLOR=black
ERROR TEXT COLOR=
WARNING TEXT COLOR=
```

[HEADER]

```
LOGO IMAGE=^pbvsp.gif^ebvse.gif^ibvsi.gif
LOGO URL=www.bireme.br
HEADER IMAGE=^ponlinep.gif^eonlinee.gif^ionlinei.gif
```

[IAH]

```
MANAGER E-MAIL=scielos@bireme.br
MAINTENANCE=OFF
REVERSE MODE=OFF
MULTI-LANGUAGE=ON
```

The important part for getting started is the [PATH] section. PATH\_DATA must show the path of the folder where the static HTML pages are found relative to the root of the webserver (so /iah/ is correct if installed as above). PATH\_CGI-BIN must show the complete (absolute) path of the CGI-BIN folder (in which WXIS.EXE is installed). PATH\_DATABASE must show the complete path of the folder in which the IAH databases are located.

The other parameters do not affect whether the interface will start and can be modified later. However, it may be worth setting MANAGER E-MAIL in the [IAH] section to your own email address so that users can contact you in the case of any errors.

## 9.4 Searching the sample databases

Provided that you have the webserver running, you should now be able to open the initial page of the Setup module using the URL

**<http://servername/iah/setup.htm>**

where servername is the address of your server. If you are using a browser on the same computer as the server it will be:

**<http://localhost/iah/setup.htm>**

The setup screen shown in Figure 9.2 should appear. Choose the language in which you wish to work, leave the button "Setup del sistema" set, and click on "aplicar". After a while you should receive a list of files found, databases created, and the message "System setup finished successfully". If the setup was not successful check that you have copied all the files into the right places and that you have the file **iah.def** properly configured.

You can now go back to the first setup page and setup the sample databases, LILACS and SeCS. Click the button against "Setup del sistema" and then "aplicar". On the next form, check that "Test data base ( LILACS format )" is set and click "aplicar". It will be some time before you get a reply because the indexes and other files have to be generated but you should eventually be told "Data base setup finished successfully". You can then go back and repeat the process for the second test database, LILACS.

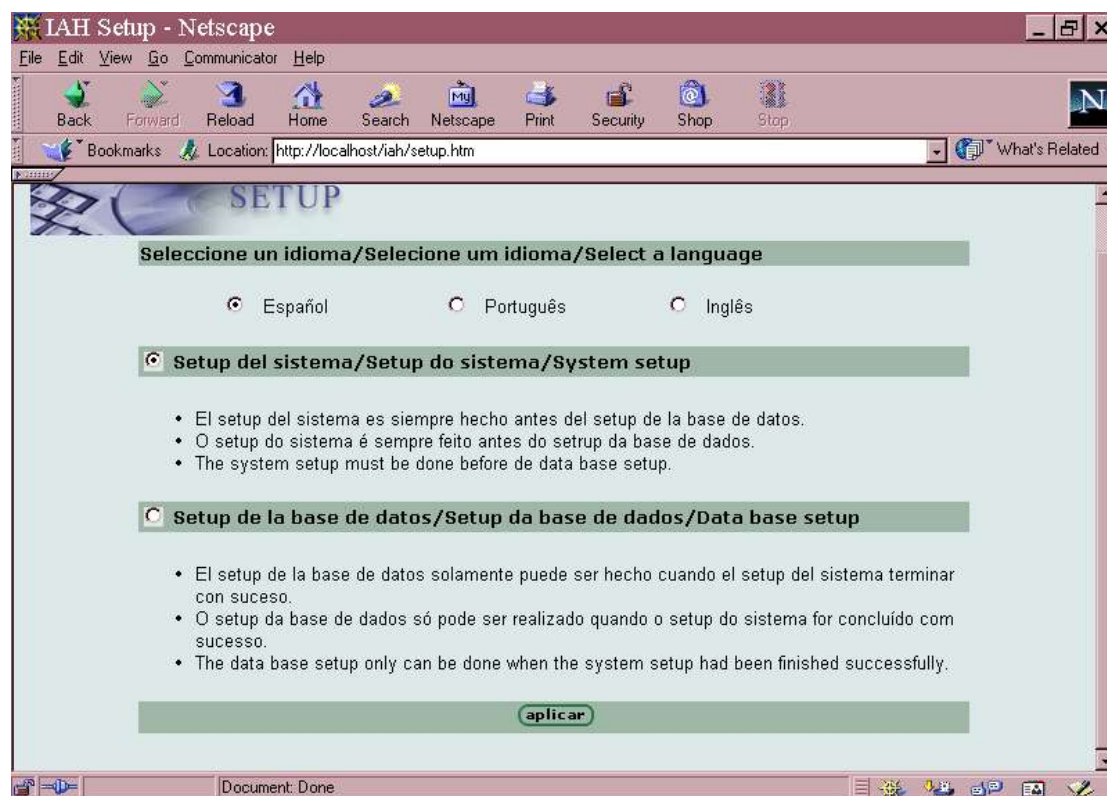


Figure 9.2 First setup screen

Now you can try accessing the test databases. Open the page

**<http://servername/iah/bases.htm>**  
(e.g. <http://localhost/iah/bases.htm>)

This page allows you to select one of the two databases you have set up. Click one of the names and you will get the search screen. To change to English, click the "config" button. The search configuration screen is shown in Figure 9.3.

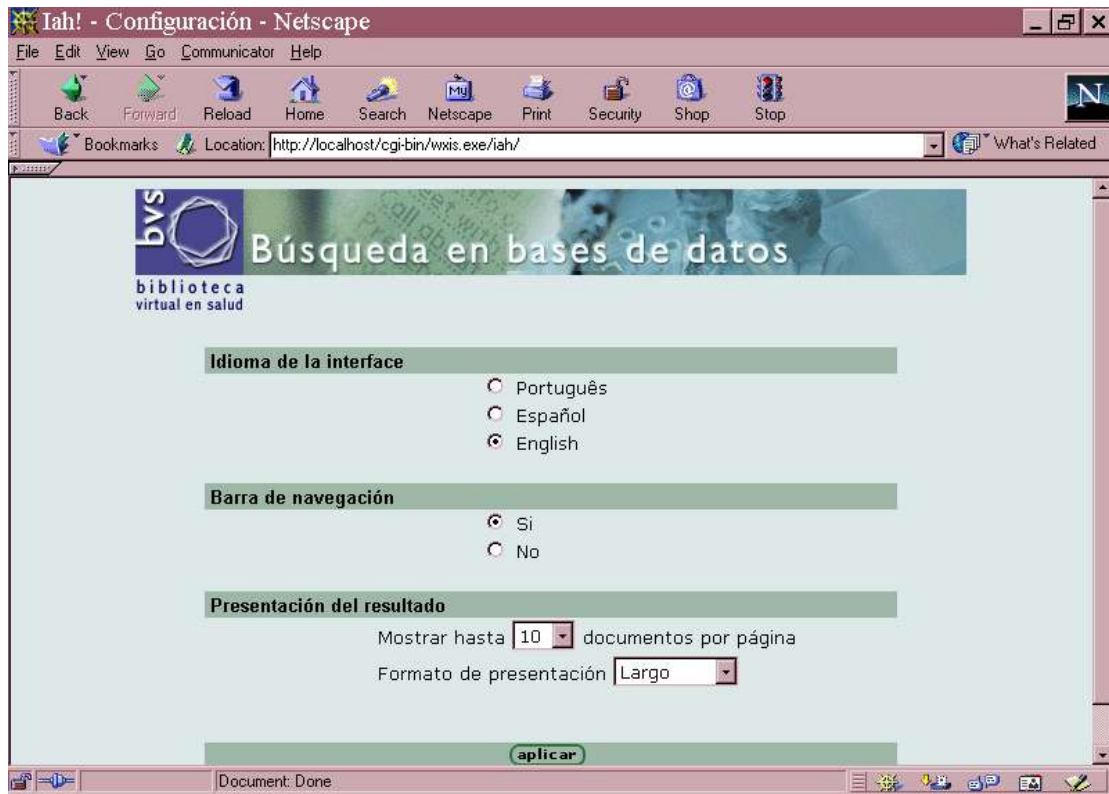


Figure 9.3 Search configuration screen

You can choose which language to use, whether or not the navigation bar appears, how many documents to show on a (web) page and which format is used to display the records. Click "aplicar" and you should be taken back to the search screen in your selected language, e.g. Figure 9.4 where a search is shown for "heart disease". Selecting "All words" will search for HEART AND DISEASE: selecting "Any word" will search for HEART OR DISEASE.

Click "search" and the search results should display. You can find out what some of the icons stand for in the display by positioning the cursor on them (assuming that your browser is configured for JavaScript).

There are actually two search forms, *basic form* and *free form*. The basic form has a single textbox to enter your search term(s). The free form has three boxes for search terms and boxes that allow you to choose the logical operators to connect them and the field(s) to be searched. You can also choose to display part of the index ("terms dictionary" in CDS/ISIS) *for the specified field only* (or all types of index terms) and select search terms from that.

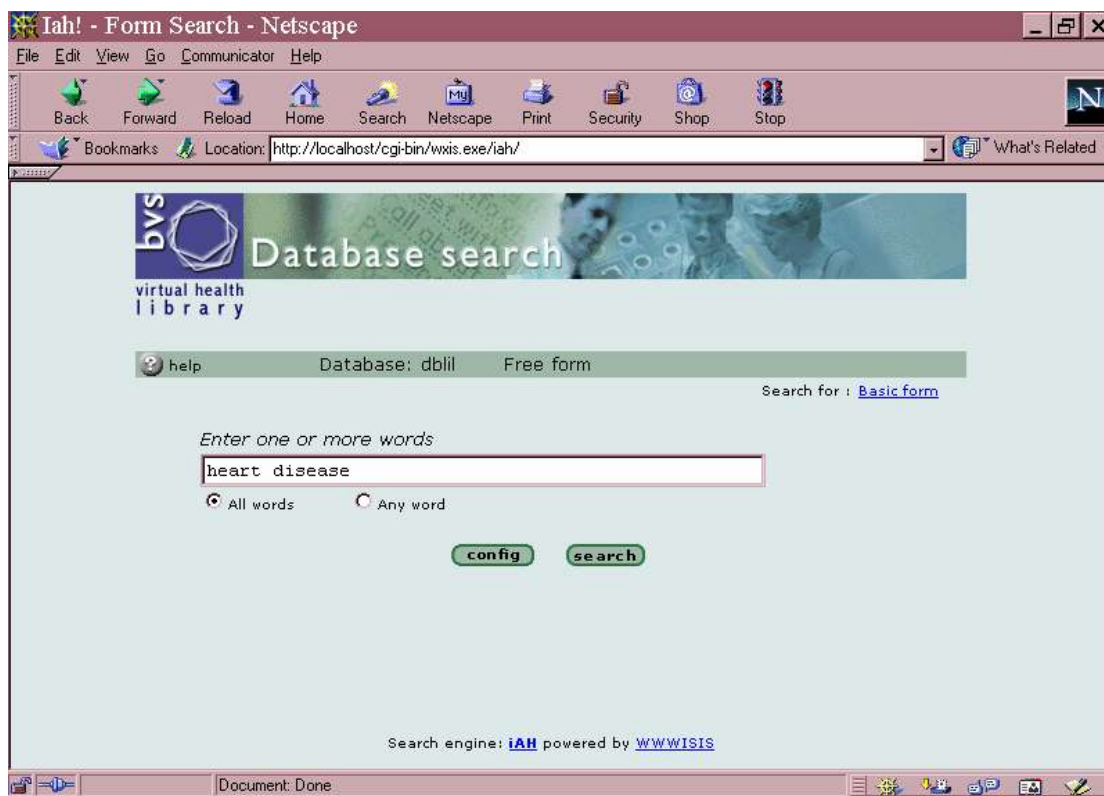


Figure 9.4 Example search in LILACS

## 9.5 Setting up your own database

Assuming that the sample databases are working correctly, you will probably now want to configure your own database(s) that you have created in CDS/ISIS. This requires a good knowledge of CDS/ISIS and if you do not have this it may be better to use BIREME's customisation service.

Although you will have an index and display formats for your database in CDS/ISIS, they are unlikely to be suitable for IAH and you will have to create new ones as described below. You will also need to know how to write the URL that leads to the search page for your database.

### 9.5.1 Transferring the database records

The database records themselves are loaded into IAH as an ISO file. You need to export your records from CDS/ISIS as an ISO file and save it in the directory **bases\iah\isos**.

### 9.5.2 Writing the Field Select Table

The Field Select Table (FST) specifies what fields from the record go onto the index, and in what form, to allow for direct searching. To allow for searching on index terms from a specified field only in IAH, you must set up an FST which puts a *prefix* in front of each search term indicating what field it came from. This can be done using indexing techniques 5,6,7 and 8. These techniques are described in the documentation

for later versions of CDS/ISIS for DOS (not in the *Reference Manual* for the DOS version 2.3).

Each line in an FST consists of three parameters with spaces between them:

The *ID* which identifies that type of index term (usually the same as the tag of the field from which it is selected)

The *Indexing Technique*. For example technique 0 means extract the whole field as one term, whereas technique 4 means extract each word ("keyword") individually to go onto the index

The *Extraction Format* which is written in the CDS/ISIS formatting language and which specifies the field, fields, or parts of fields from which the terms are to be taken and possibly adding other data to the data extracted from the record. So the required prefix can be added to the terms extracted from a given field.

A simple example of a line in an FST is:

```
70 0 v70
```

This extracts the whole of field 70 from the record and puts it on the index with the ID 70.

The line

```
70 0 'AU_' v70
```

would put the prefix AU\_ in front of the data extracted from field 70. If field 70 is a repeatable field, you can generate a prefix for each repeat using the line

```
70 0 MHU, ('AU_' v70/)
```

A more complicated example is

```
24 8 '*TI_*' v24
```

This extracts data from field 24. It uses technique 8, which is like technique 4 in that it extracts each word individually, but it puts a prefix in front of each term. (If you used technique 4 the prefix would appear with the first term.) The prefix is the literal TI. The inverted commas enclose a literal in CDS/ISIS, but techniques 5,6,7 and 8 require another delimiter round the repeatable prefix. Here the asterisk is used as that delimiter. So the entries in the index for a title "Principles of organic synthesis", assuming that OF is a stopword, will be:

```
TI_ORGANIC  
TI_PRINCIPLES  
TI_SYNTHESIS
```

This first converts field 70 to Heading Mode Upper-case, in case there are subfield markers or data inside <...>. It makes a repeatable group of the prefix AU\_, an

occurrence of field 70, and a new line. Each occurrence of field 70 will generate an index term with the prefix AU\_.

So you need to write an FST which extracts data from the fields that you want to make searchable in IAH, in the form that you want them to appear (whole fields, each subfield, text between <...>, or each word), and with a prefix. Unlike in CDS/ISIS, the FST for generating the inverted file need not have the same name as the database, but it should still have the extension .FST. You could write it in a text editor rather than in CDS/ISIS if you wish. The FST needs to be saved in the directory **bases\iah\fst**s.

### 9.5.3 Writing the display format(s)

You need one or more display formats for viewing the records in IAH. These are files with the extension .PFT. Some discussion of writing display formats for use on the web, which involves the use of the CDS/ISIS formatting language and HTML, has been given in Section 6.2.

The format file(s) should be saved in the folder **bases\iah\pfts\i** for formats using Inglês (English). There are similar folders **e** for Español (Spanish) and **p** for Português (Portuguese).

### 9.5.4 Editing the configuration file

You now need to make a copy of the file **dblii.def** (in the subfolder IAH below the website root) and give it the name of your database with the extension .DEF. For the sake of example we will call the database JANE so this file will be **jane.def**. You then need to edit it (using a text editor) for your own database. It is quite a complicated file and we will examine it section by section.

(a) [FILE\_LOCATION]

```
FILE BASE.*=%path_base%jane.*  
FILE INVERTED.*=%path_base%jane.*
```

```
FILE short.pft=%path_base%pfts%lang%janep.pft  
FILE long.pft=%path_base%pfts%lang%janed.pft  
FILE title.pft=%path_base%pfts%lang%janet.pft
```

The lines FILE BASE= and FILE INVERTED= specify the locations of the database and the inverted file. They are shown above using the variable %path\_base% which has been set in the IAH.DEF file, or you could write e.g.

```
FILE_BASE.*=/bases/iah/jane.*
```

The next three lines specify *logical* names (which are referred to later) for the three print formats defined for this database. The actual filenames are **janep.pft**, **janed.pft** and **janet.pft**. The parameter %lang% can take the values P (Portuguese), E (Spanish) or I (English) – see above – and you can have a display format in each

language. Although three formats are shown in the example you could have more or fewer.

**(b) [INDEX DEFINITION]**

```
INDEX Tw=iKeywordsePalabraspPalavrasd*xTW
yINVERTEDuTW_mTW_fB
INDEX Ti=iTitle wordsePalabras de titulopPalavras do tituloxTI yINVERTED
uTW_mTW_fB
INDEX Au=iAuthoreAutorpAutorxAU yINVERTEDuAU_mAU_fB
```

This contains definitions of each index available for searching (as shown in the *free* search form). There can be a separate FST for each index or a single FST with prefixes for each kind of term. The subfields in the definition are as follows:

- <sup>i</sup> Name of the index in English
- <sup>e</sup> Name of the index in Spanish
- <sup>p</sup> Name of the index in Portuguese
- <sup>d</sup> The value \* specifies that this is the default index
- <sup>x</sup> Prefix of this index in the FST
- <sup>y</sup> Logical name of the inverted file (defined in the FILE\_LOCATION section)
- <sup>u</sup> Prefix of the index in the FST with punctuation
- <sup>m</sup> Prefix of the index in the FST with punctuation (reserved for future use)
- <sup>f</sup> Indicates if the index will be available in the basic form
- <sup>t</sup> Type of the index. The value "short" produces a short index, i.e. all terms are displayed without the user having to choose an initial letter

**(c) [APPLY\_GIZMO]**

```
GIZMO=ASC2HTML
```

If you are using data containing "special characters" such as à, ñ or ü, you will need a *gizmo* database to convert them to codes when they sent to a web browser. Gizmo databases are explained in Section 4.8. The database ASC2HTML is supplied with the IAH package and converts characters in extended ASCII to their text versions for HTML. You also need to specify the location of the gizmo files in the section [FILE\_LOCATION].

**(d) [FORMAT\_NAME]**

```
FORMAT short.pft=iCitationeCitationpCitação
FORMAT long.pft=iDetailedeDetalladopDetallhado
FORMAT title.pft=iTitleeTitulopTitulo
FORMAT DEFAULT=short.pft
```

This section indicates the logical names of the display formats and their names in each of the three languages: <sup>i</sup> for the English name, <sup>e</sup> for the Spanish name and <sup>p</sup> for the Portuguese name. The default format is specified in FORMAT DEFAULT.

**(e) [HELP\_FORM]**

HELP FORM FREE=<sup>^</sup>ihelpformi.htm<sup>^</sup>ehelpforme.htm<sup>^</sup>phelpformp.htm  
HELP FORM BASIC=<sup>^</sup>ihelpformi.htm<sup>^</sup>ehelpforme.htm<sup>^</sup>phelpformp.htm  
HELP FORM ADVANCED=<sup>^</sup>ihelpformi.htm<sup>^</sup>ehelpforme.htm<sup>^</sup>phelpformp.htm

This section specifies the help files for formulating searches – FREE for the Free search form, BASIC for the Basic search form and ADVANCED for the Advanced search form (assuming that all three are defined – see (f) below).. The files should be written in HTML and saved in the subfolder **\iah\hlps** of the server root. The same file can be specified for each type of searching – as in the example above. Files can be specified for each language as elsewhere.

**(f)** [DISPLAY\_FORM]  
AVAILABLE FORMS=F,B

This section contains the parameter AVAILABLE FORMS which indicates the search forms available. The possible values are F (Free form), B (Basic form) and A (Advanced form). The first one listed will be the default form.

## 9.6 Order of records in display

When you have your database operating you may wish to make some refinements. If your database consists of records which have been entered over a period of time, you may like to display them with the most recent first. To do this, edit the penultimate line of the **iah.def** file to read REVERSE=ON. This will apply to all databases using IAH on your server so if any databases have been sorted alphabetically do not make this change – they will look very strange in reverse alphabetical order!

## 9.7 URL for searching your database

The URL for searching the database JANE will be:

**<http://servername/cgi-bin/wxis.exe/iah/?IsisScript=iah/iah.xis&base=jane&lang=e>**

You should replace "servername" with the name of your server and set the "lang" parameter to the language of the search form which you want to display. On the page which contains the link, it will be something like:

**<A HREF="/cgi-bin/wxis.exe/iah?IsisScript=iah/iah.xis&base=jane&lang=e">  
JANE</a> The Jordan and Near East database**

## INDEX

- Apache web server 3.2.2
- attribute 2.3
  
- background colour 6.2c
- BIREME 1.1, 1.2
  - website 3.4
- body tag 2.3
- BR tag 2.3
  
- CDS/ISIS *see* Micro CDS/ISIS
- CGI *see* Common Gateway Interface
- character formatting 6.2b
- check box 5.2d
- cipar tag 4.7
- CISIS 1.2
- client 2.1
- comments 4.10
- Common Gateway Interface 2.5, 5.1
- content-type 4.3
- creating records 8.3, 8.4
  
- deleting records 8.2
- directory 2.2
- display formats 6.2
- display tag 4.1
- do tag 4.4
- domain 2.2
  
- editing records 8.5, 8.6
- environment variables 5.6
- Extensible markup Language 2.4, 4.1
  
- field select table
  - for IAH 9.5.2
- field tag 5.4
- flow tag 4.9
- font tag 2.3
- form tag 5.2
- forms 5.2
- fullinv utility 3.5
  
- gizmo database 4.8, 9.5.4c
  
- head tag 2.3
- heading tag 2.3
- hidden variables 5.2f
- hits
  - number of 6.2f
  
- HTML *see* Hypertext Markup Language
- HTML tag 2.3
- HTTP *see* Hypertext Transfer Protocol
- hyperlink 2.2
- hypertext 2.1
- Hypertext Markup Language 2.3
- Hypertext Transfer Protocol 2.1, 2.2
  
- IAH interface 9
  - configuration 9.3
  - installation 9.2
- IIS *see* Internet Information Server
- IN files (Version 3) 1.3
- index file *see* inverted file
- indexing technique 9.5.2
- input tag 5.2
- installation
  - of IAH 9.2
  - of WWWISIS 3
- Internet Information Server 3.2.3
- Internet Protocol 3.1
- Internet Service Provider 2.1
- Inverted file
  - displaying 7
- IP *see* Internet Protocol
- ISIS 1.1
- Isis\_Current 5.6
- ISIS\_DLL 1.1
- Isis\_Status 5.6, 8.6
- Isis\_Total 5.6
- IsisScript 4.2
- ISISXML tag 1.3
- ISO file 3.5, 9.5.1
  
- label tag 4.10
- licence for WWWISIS 3.3
- LILACS database 9.4
- loadiso utility 3.5
- localhost 2.1, 3.3
- loop tag 4.6, 5.5
  
- markup language 2.4
- Micro CDS/ISIS 1.1
- mx.exe 1.2, 3.5
  
- name-value pair 2.5
- new record 8.3, 8.4
- no hits message 6.2g
- numbering of results 6.2e

Omni HTTPd 3.2.1  
 option list 5.2c  
 order of records  
   in IAH 9.6  
  
 parm tag 4.5  
 pft tag 4.1, 5.4  
 prefix in inversion 9.5.2  
 protocol 2.1  
  
 radio button 5.2e  
 record locking 8.2  
 REVERSE MODE  
   in IAH 9.6  
 root directory 2.2, 2.5  
  
 scope of variables 5.5  
 search form 6.1, 6.3  
 searching 6  
 SeCS database 9.4  
 server 2.1  
 SGML *see* Standard Generalised  
   Markup Language 2.4  
 spacing of results 6.2a  
 Standard Generalised Markup  
   Language 2.4  
 subdomain 2.2  
  
 table tag 6.2h  
 tabular display 6.2h  
 tag 2.3  
 task=keyrange 7.2  
 task=mfnrange 4.6  
 task=search 6.1  
 task=update 8.1  
 tasks 4.4  
 text area 5.2b  
 text box 5.2a  
 title tag 2.3  
  
 UNESCO 1.1  
 Uniform Resource Locator 2.2  
 Unix  
   transferring databases to/from 3.5  
 unlocking records 8.2, 8.4  
 update tag 4.6, 8.4  
 URL *see* Uniform Resource Locator  
  
 virtual record 5.4  
  
 webservice software 3.2  
 write tag 8.2  
  
 WWWISIS  
   versions 1.3  
 wxis.exe 3.4  
 XML *see* Extensible Markup  
   Language