DOWNLOAD

RELEASE 7.21
June 2007

David L. Rotman
Cedarville University
251 North Main Street
Cedarville, OH 45314
rotmand@cedarville.edu

Contents

DOWNLOAD Purpose

DOWNLOAD is a utility program which produces output files in ASCII, WordPerfect, HTML, XML, dBASE, or DIF formats. Output files can be used by service bureaus and government agencies or used directly by standard software packages. DOWNLOAD is easy to use, because it uses syntax like the LIST statement. The examples in the next chapter illustrate the ease with which the program can be used.

This software was written at Cedarville University by Doug Sjoquist and modified by Dave Rotman. You may freely distribute this software, but this software is not to be sold by itself nor as part of any other software package. A current version of the software may be obtained via anonymous ftp from:

ftp.cedarville.edu

This software is made available on an "as-is" basis, with no warranty of any kind.

There is a “low-volume” list serve for folks who wish to share DOWNLOAD questions, suggestions, usage hints, etc. For more information, connect to:

http://mail.cedarville.edu/mailman/listinfo/download-list

Currently, the DOWNLOAD software is being maintained by Dave Rotman. Suggestions for new features or bug fixes should be sent to rotmand@cedarville.edu. Since Dave cannot devote a lot of time to support, you can often get usage questions answered more quickly by posting an item to the download-list or e-mail list serves which focus on Unidata/Universe software or Datatel products.

About Cedarville University

Cedarville University was chartered by the State of Ohio in 1887. The University operates from a conservative theological position but is progressive in its use of technology. Academic programs are offered in over 100 areas. The largest majors are engineering, nursing, business, and education. The University enrolls approximately 3,100 students from 45 states and several foreign countries. The University occupies a 400-acre campus in close proximity to Dayton, Cincinnati, and Columbus. For more information, visit http://www.cedarville.edu.

.

Simple Examples

1. Create a file of customer names and addresses so that the file can be loaded into a spreadsheet or database program.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
    FILE _HOLD_ CUSTOMER.DAT
```

Sample output for two customer records:

```
"Harold Johnson","341 S. Main St.","Buffalo","NY","12533"
"Elsie Gordon","P.O. Box 18","West Plains","OH","45509"
```

2. Use the same customer database, but this time produce a Web page (HTML file):

```
SELECT CUSTOMERS SAMPLE 3
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
    FILE _HOLD_ CUSTOMER.HTM FORMAT HTML
```

Note that an HTML file can also be read by many spreadsheet and word-processing programs. This method might actually give better results than reading the comma-separated-values text file.

3. Use the same customer database, but this time produce a fixed-length file for use by an external service bureau.

```
SELECT CUSTOMERS SAMPLE 3
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
    FILE _HOLD_ CUSTOMER.DAT FORMAT FIXED
```

Sample output for three customer records:

```
Harold Johnson  341 S. Main St.  Buffalo      NY12533
Elsie Gordon    P.O. Box 18      West Plains  OH45509
Linda Falling   3428 Smith St.   Rockford     MA03291
```

4. Use the same customer database, but this time produce an XML file for use by an e-commerce system.

```
SELECT CUSTOMERS SAMPLE 3
```

```
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
      FILE _HOLD_ CUSTOMER.XML FORMAT XML
```

Sample output for three customer records:

```
<?xml version="1.0"?>
<download>
<customers>
  <name>Harold Johnson</name>
  <street>341 S. Main St.</street>
  <city>Buffalo</city>
  <state>NY</state>
  <zip>12533</zip>
</customers>
<customers>
  <name>Elsie Gordon</name>
  <street>P.O. Box 18</street>
  <city>West Plains</city>
  <state>OH</state>
  <zip>45509</zip>
</customers>
<customers>
  <name>Linda Falling</name>
  <street>3428 Smith St.</street>
  <city>Rockford</city>
  <state>MA</state>
  <zip>03291</zip>
</customers>
</download>
```

5. Use the same customer database, but this time produce a "mail merge" file for use by Corel WordPerfect or Microsoft Word.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
      FILE _HOLD_ CUSTOMER.DAT FORMAT WP51
```

6. Use the same customer database, but this time produce a "dbf" file for use as a dBASE or Paradox database.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME STREET CITY STATE ZIP \
      FILE _HOLD_ CUSTOMER.DAT FORMAT DBF
```

7. Produce a file showing customer id numbers, names, and date of first order.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS ID.NO NAME \
      ORDER.DATES \
      FILE _HOLD_ CUSTOMER.DAT
```

Sample output for two customer records:

```
"10485","Harold Johnson","05/18/02"
"30216","Elsie Gordon","12/11/98"
```

(By default, only the first value of a multi-valued field is included. This default can be overridden with the NUM.VALUES phrase as shown in the next example.)

8. Revise the previous example to show all order dates, not just the first one.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS ID.NO NAME \
      ORDER.DATES NUM.VALUES ALL\
      FILE _HOLD_ CUSTOMER.DAT
```

Sample output for two customer records:

```
"10485","Harold Johnson","05/18/02","12/12/02","11/18/03"
"30216","Elsie Gordon","12/11/98","03/12/99"
```

(Harold Johnson has placed a total of three orders and Elsie Gordon has placed a total of two orders.)

9. Produce a file containing customer name, city, and zip code. The first record in the output file should identify the field names.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME CITY ZIP \
      HEADING FIELD.NAMES
      FILE _HOLD_ CUSTOMER.DAT
```

Sample output for two customer records:

```
"NAME","CITY","ZIP"
"Harold Johnson","New York","12533"
"Elsie Gordon","West Plains","45509"
```

A slightly different version of the command will use the field labels (item 4 in the dictionary) rather than the field names:

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME CITY ZIP \
      HEADING FIELD.LABELS
```

```
        FILE _HOLD_ CUSTOMER.DAT
```

Sample output for two customer records:

```
"Customer Name","City","Zip Code"
"Harold Johnson","New York","12533"
"Elsie Gordon","West Plains","45509"
```

10. Repeat the previous example, but produce the output as a Web page.

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS NAME CITY ZIP \
        HEADING FIELD.LABELS \
        FILE _HOLD_ CUSTOMER.HTM \
        FORMAT HTML
```

Sample output for two customer records:

Customer Name	City	Zip Code
Harold Johnson	New York	12533
Elsie Gordon	West Plains	45509

11. Produce a file of invoices showing invoice number, customer name, gross amount, and net amount.

```
GET.LIST INVOICE.LIST
DOWNLOAD INVOICES INV.NO INV.NAME INV.GROSS INV.NET \
        FILE _HOLD_ INVOICE.DAT
```

Sample output for two invoice records:

```
"I10345","Harold Johnson",543.28,495.87
"I20956","Elsie Gordon",125.04,125.04
```

12. Produce a file of invoices showing invoice number, customer name, item number, and item amount for all item numbers beginning with the letter Q.

```
GET.LIST INVOICE.LIST
SELECT INVOICES BY @ID \
        BY.EXP ITEM.NUMBER \
        WHEN ITEM.NUMBER LIKE 'Q...'
```

```
DOWNLOAD INVOICES INV.NO INV.NAME \
    BY.EXP ITEM.NUMBER \
    ITEM.NUMBER ITEM.AMT \
    FILE _HOLD_ INVOICE.DAT
```

Sample output for two invoice records:

```
"I10345","Harold Johnson","Q104",56.75
"I10345","Harold Johnson","Q131",18.56
"I20956","Elsie Gordon","Q117",41.00
```

(Invoice I10345 contained two items beginning with the letter Q and invoice I20956 contained only one item beginning with the letter Q.)

13. Produce a file showing each term that a student attended our institution. Each term should be on a line by itself. The student's name should appear on each line.

```
SELECT STUDENTS \
    BY NAME \
    BY.EXP REG.TERMS
DOWNLOAD STUDENTS \
    BY.EXP REG.TERMS \
    NAME \
    REG.TERMS \
    FORMAT FIXED
```

Output would look like this:

```
Anthony, Susan        1995FA
Anthony, Susan        1997WI
Anthony, Susan        2000FA
Anthony, Susan        2003SP
Lincoln, Abraham      1996FA
Washington, George    1997FA
Washington, George    1998WI
Washington, George    2002FA
```

Help-file Printout

```
DOWNLOAD 7.21                                                June 2007


This is a brief introduction, intended to help you get started with DOWNLOAD.
For more extensive help, consult the documentation file DOWNLOAD.PDF which
came with the DOWNLOAD software.


Basic command line syntax

   DOWNLOAD FileName field.name1 field.name2 ... field.nameN


More complete syntax

   DOWNLOAD [BEGIN] [DICT] FileName
      [[field.prefix] field.names [field.qualifiers]]
      [SUBR("sname"[,arg1,etc]) [field.qualifiers]]
      [LITERAL "value" [field.qualifiers]]
      [EVAL "expression" [field.qualifiers]]
      [@variables [field.qualifiers]]
      [Record.IDs]
      [Phrase from Dict]
      [Phrase from VOC]
      [Options]


To view this help file, enter:
   DOWNLOAD HELP

To see the limits on the number of data fields, EVAL expressions,
etc., enter:
   DOWNLOAD HELP MAX



Specifying data to include on the output

   - Reference a data field or I-descriptor described in the current
        dictionary.
   - Reference a data field or I-descriptor described in the VOC file.
   - Use the SECONDARY.FILE option to relate another file to this one.
        This is like doing a TRANS (translate) to another file, but
        without creating an i-descriptor.
   - Use a literal value specified with the "LITERAL 'value'" clause.
   - Use an EVAL expression.
   - Return a value from a subroutine specified with the
        "SUBR('sname')" clause.
   - Use an "at" variable to return a system-defined value.

   All of these items may also include qualifiers or prefixes which
   further define how the value is to be downloaded.  There are also
   command options that may be used to change the default behavior
   of DOWNLOAD.
```

```
Valid Field Clauses

   DataFieldName [Field Qualifier(s)]
      You may use any data field from the data file(s) specified
      on your command line. You may also use a phrase from the
      dictionary or use '@RECORD' to process all data fields from
      the input file.

   I-descriptorName [Field Qualifier(s)]
      The dictionary item for a data field or an I-descriptor can come
      from either the current dictionary (which can be changed with the
      USING DICT option,) or the VOC file.

   LITERAL "constant value" [Field Qualifier(s)]
      The default format and length for this type of value is the actual
      length of the data, left-justified.  LIT may be used in place of
      the keyword LITERAL.

   EVAL "expression" [Field Qualifier(s)]
      The default format and length for this type of value is the actual
      length of the data, left-justified.

   SUBR("subroutine.name" [, argument2]) [Field Qualifier(s)]
      This clause can have from 1 to 10 arguments and functions similarly
      to the SUBR usage in I-descriptors.  The subroutine should return
      the value to be downloaded in the first argument.  If the value
      being returned is multi-valued, then the field qualifier
      MULTI.VALUE should be added since the default is single-value.
      The default format and length for this type of value is "30L".

   @variable [Field Qualifier(s)]
      You may select from any of the following variables:
      @ACCOUNT              host operating system path
      @DATE                 system date (internal format) that
                               DOWNLOAD began running
      @DAY                  day of the month that DOWNLOAD began running
      @LOGNAME              user's login name
      @MONTH                month of the year (numeric) that DOWNLOAD
                               began running
      @SYSTEM.RETURN.CODE   system return code at the start of execution
                               (usually, the number of records in the
                               active select list)
      @TIME                 system time (internal format) that
                               DOWNLOAD began running
      @YEAR                 year (four digit) that DOWNLOAD began running



Record Layout Command Options and Default Values
   ***** Option ****************************   ***** Default Value *****
   BY.EXP MVField1                             none
   COMMA.CHAR comma.char                       ,
   DEFAULT field.qualifier new.default.value    none
   DETAIL ...detail layout options....         default is DETAIL
   EOR.CHAR end.of.record.char                 (null)
   FIELD.GAP #blanksbetweencolumns             zero (FIXED) or 2 (XML)
   FOOTING ...record layout options...         no report footing record
```

```
   FORMAT
      COMMA|DBF|DIF|FIXED|HTML|                 COMMA
      QUOTE|TAB|WP50|WP51|XML
   HEADING ...record layout options...          no report heading record
   NO.LINEFEED                                  Off (LF between records)
   RECORD.LENGTH fixed.size                     none (only valid with FIXED)
   RECORD.ORIENTATION HORIZONTAL | VERTICAL     HORIZONTAL
   REMOVE.PUNCTUATION                           Off (leave punctuation)
   QUOTE.CHAR quote.char                        " (regular double-quote)
   UPCASE                                       Off (do not change case)
   WHEN MVField2 oper Field | Value(s)          none


Execution Environment Command Options and Default Values
   ***** Option ****************************   ***** Default Value *****
   [NO.]DISPLAY.COUNT                           DISPLAY.COUNT
   [NO.]PRINT.ERRORS                            PRINT.ERRORS
   LPTR                                         Off (errors/layout on screen)
   NO.PAGE                                      Off (pause at end of screen)
   PRINT.LAYOUT                                 Off (do not print layout)
   PROGRESS.INTERVAL                            10
   WRITE.INTERVAL                               10 0 (sleep 0 seconds)



Records and Files to Process Command Options and Default Values
   ***** Option ****************************   ***** Default Value *****
   APPEND                                       Off (do not append)
   FILE Type1File RecordName                    Off (display on screen)
   FROM SelectList#                             0 (default select list)
   OVERWRITING                                  Off (do not overwrite)
   SAMPLE [SampleSize]                          Entire list or file
   USING [DICT] AlternateInfoFile               DICT DBMSFileName


HTML Command Options and Default Values
   ***** Option ****************************   ***** Default Value *****
   HTML.TITLE "HTMLTitleToUse"                  empty
   HTML.TITLE field.name
   HTML.TITLE EVAL "expression"
   HTML.BODY "HTMLBodyToUse"                    empty (white background)
   HTML.BODY field.name
   HTML.BODY EVAL "expression"
   HTML.TOP "HTMLTopToUse"                      empty
   HTML.TOP field.name
   HTML.TOP EVAL "expression"
   HTML.TABLE "HTMLTableToUse"                  BORDER="1" (small visible)
   HTML.TABLE field.name
   HTML.TABLE EVAL "expression"
   HTML.BOTTOM "BottomHTMLTextToUse"            empty
   HTML.BOTTOM field.name
   HTML.BOTTOM EVAL "expression"


XML Command Options and Default Values
   ***** Option ****************************   ***** Default Value *****
   XML.ALLOW.PERIODS                            off (remove periods)
   XML.FILE.ATTRIBUTE "XMLFileAttributeToUse"   none
   XML.FILE.ATTRIBUTE field.name
```

```
   XML.FILE.ATTRIBUTE EVAL "expression"
   XML.FILE.NAME "XMLRootFileNameToUse"         VOC name for primary file
   XML.FILE.NAME field.name
   XML.FILE.NAME EVAL "expression"
   XML.GROUP.ATTRIBUTE "XMLGroupAttributeToUse" none
   XML.ROOT.ATTRIBUTE "XMLRootAttributeToUse"   none
   XML.ROOT.ATTRIBUTE field.name
   XML.ROOT.ATTRIBUTE EVAL "expression"
   XML.ROOT.NAME "XMLRootNameToUse"             "download"
   XML.ROOT.NAME field.name
   XML.ROOT.NAME EVAL "expression"
   XML.ROOT.NAME NONE
   XML.UPCASE                                   off (elements in lowercase)
   XML.VERSION "XMLVersionToUse"                <?xml version="1.0"?>
   XML.VERSION NONE


Valid operators for WHEN option:

   EQ, NE, GE, GT, LE, LT, LIKE, UNLIKE


Valid Field Prefixes:

  BREAK.ON field.name
  TOTAL field.name
  AVERAGE field.name
  MIN field.name
  MAX field.name


ValidField Qualifiers:    ------- Valid for these formats -------   Default
                          FIXED COMMA HTML  WP50  DIF   DBF   XML   Value
                                QUOTE       WP51
                                TAB
   * layout qualifiers
   BEG.COL                Yes                                      1
   END.COL                Yes                                      n/a
   COLUMNS                Yes                                      n/a
   LENGTH                 Yes   Yes   Yes   Yes   Yes   Yes   Yes   none
   LINE                   Yes   Yes                                1

   * data formatting qualifiers
   COL.HDG                Yes   Yes   Yes   Yes   Yes   Yes   Yes   dict
   CONV                   Yes   Yes   Yes   Yes   Yes   Yes   Yes   dict
   NO.NULLS               Yes   Yes   Yes   Yes   Yes   Yes   Yes   false
   DEFAULT.VALUE          Yes   Yes   Yes   Yes   Yes   Yes   Yes   null
   FMT                    Yes   Yes   Yes   Yes   Yes   Yes   Yes   dict

   * HTML qualifiers
   HTML.CELL                    Yes                                none
   HTML.START                   Yes                                none
   HTML.END                     Yes                                none
   HTML.ROW                     Yes                                none

   * XML qualifiers
   XML.ALLOW.CHARACTERS                                      Yes   remove
                                                                   illegal
                                                                   chars
```

```
   XML.ASSOC.NAME                                                Yes   dict name
   XML.ATTRIBUTE                                                 Yes   off
   XML.ELEMENT                                                   Yes   on
   XML.GROUP.NAME                                                Yes   off
   XML.NAME                                                      Yes   dict name
   XML.SUBASSOC.NAME                                             Yes   none

   * multi-value qualifiers
   MV.ORIENTATION         Yes   Yes   Yes                              horizontal
   NUM.VALUES             Yes   Yes   Yes   Yes   Yes   Yes   Yes   1
   NUM.VALUES ALL         Y/N   Yes   Yes   Yes               Yes
      (valid with FORMAT FIXED & vertical orientation)
   SINGLE.VALUE           Yes   Yes   Yes   Yes   Yes   Yes   Yes   single
      | MULTI.VALUE
   SUBVALUE.SEPARATOR           Yes   Yes   Yes               Yes   varies
   VALUE.SEPARATOR              Yes   Yes   Yes                     varies


Valid control heading and footing record layouts:

   HEADING.ON break.field [...record layout...]
   HEADING.ON break.field NONE
   FOOTING.ON FINAL [...record layout...]
   FOOTING.ON FINAL NONE
   FOOTING.ON break.field [...record layout...]
   FOOTING.ON break.field NONE
   DET.SUP     (show only heading/footing lines)

   For each break field (each use of BREAK.ON), a default footing record
   with the same layout as the detail record will be setup, as well as a
   final control footing record (different from the report footing record).
   This default may be disabled with the optional keyword NONE following
   the FOOTING.ON phrase.



SECONDARY.FILE option
  ***** Option **********************  ***** Default Value *****
  SECONDARY.FILE filename
    [KEY primaryFileFieldName]         @ID
    [ALIAS text]                       filename



Examples


1.  To create a "comma-quote" file of id numbers and names, use
    statements like the following:
       GET.LIST MAJOR.DONORS
       DOWNLOAD PEOPLE ID.NO NAME FILE _HOLD_ DONOR.DAT

    Sample output for a single record from statement above:
          "1031567","Carnegie, Andrew"


2.  To change the above to a WordPerfect (or Word) merge file:
       GET.LIST MAJOR.DONORS
```

```
DOWNLOAD PEOPLE ID.NO NAME FILE _HOLD_ DONOR.DAT FORMAT WP51
```

3. To change the above to a Web page (HTML) with column headings:

```
GET.LIST MAJOR.DONORS
DOWNLOAD PEOPLE ID.NO NAME \
   HEADING FIELD.NAMES \
   FILE__HOLD_ DONOR.HTM FORMAT HTML
```

4. The following example creates a data file named GRAD2002.DAT in the directory named &HOLD&. The file is in comma format with the student's name, 2 lines (always) of address, city, state, zip, first and second major, and all terms that have been transcripted or registered.

```
GET.LIST GRADS
DOWNLOAD STUDENTS \
   NAME ADDRESS NUM.VALUES 2 CITY ST ZIP \
   MAJOR NUM.VALUES 2 \
   REG.TERMS WHEN REG.STATUS = 'T''R' NUM.VALUES ALL \
   FORMAT COMMA FILE &HOLD& GRAD2002.DAT
```

Sample output for a single record from statement above:

```
"Smith, John Q","250 North Main","","Columbus","OH","44444","ENG",
   "BUS","1999FA","2000WI","2000SP","2000FA","2001WI","2001SP"
```

5. The following example calls a local subroutine to determine the mailing name and address.

```
GET.LIST GRADS
DOWNLOAD STUDENTS \
   SUBR('S.GET.ADDRESS','AR',5) MULTI.VALUE NUM.VALUES 5 \
   MAJOR NUM.VALUES 2 \
   REG.TERMS WHEN REG.STATUS = 'T''R' NUM.VALUES ALL \
   FORMAT COMMA FILE &HOLD& GRADUATES.2002.DATA
```

Sample output for a single record from statement above:

```
"Mr. John Smith","250 North Main","Columbus, OH 44444","","","ENG",
   "BUS","199FA","2000WI","2000SP","2000FA","2001WI","2001SP"
```

6. The following example uses a LITERAL phrase and an EVAL phrase.

```
GET.LIST GRADS
DOWNLOAD STUDENTS \
   @ID \
   LITERAL 'XYZ' \
   STU.CLASS \
   EVAL "STU.CLASS:'ABC'" \
   FORMAT COMMA FILE &HOLD& GRADUATES.2002.DATA
```

Sample output for two records from statement above:

```
"12345","XYZ","FR","FRABC"
"10463","XYZ","JR","JRABC"
```

7. To create an XML file of id numbers and names, use statements like the following:

```
   GET.LIST MAJOR.DONORS
   DOWNLOAD PEOPLE ID.NO NAME FILE _HOLD_ DONOR.XML \
      FORMAT XML
```

Sample output for a single record from statement above:

```
<?xml version="1.0"?>
<download>
<people>
  <idno>1031567</idno>
  <name>Carnegie, Andrew</name>
</people>
</download>
```

8. Examples using the SECONDARY.FILE option

The SECONDARY.FILE option lets you reference fields from other files without creating a lot of i-descriptors.

```
DOWNLOAD PEOPLE SECONDARY.FILE STUDENTS KEY @ID
     LAST.NAME FIRST.NAME STUDENTS->CLASS
```

references fields LAST.NAME and FIRST.NAME from the PEOPLE file and the field CLASS from the students file (the same record key is used for both files).

```
DOWNLOAD STUDENTS SECONDARY.FILE STUD.SCHEDS KEY LAST.SS.KEY \
   NAME STUD.SCHEDS->COURSE NUM.VALUES ALL
```

references field NAME from the STUDENTS file and field COURSE from the STUD.SCHEDS file. The record key for STUD.SCHEDS is computed in field LAST.SS.KEY of the STUDENTS file.

The SECONDARY FILE can also be used to fetch related information from a single data file:

```
DOWNLOAD PEOPLE \
   SECONDARY.FILE PEOPLE KEY PARENT.ID ALIAS PGS \
   SECONDARY.FILE PEOPLE KEY SPOUSE ALIAS SP \
   NAME PARENT.ID PGS->NAME SPOUSE SP->NAME \
   FORMAT FIXED FIELD.GAP 2
```

retrieves data from the PEOPLE file:

```
   NAME is the person's name
   PARENT.ID is the id number of the person's parent
   PGS->NAME is the name of the parent (accessed via PARENT.ID)
   SPOUSE is the id number of the person's spouse
   SP->NAME is the name of the spouse (accessed via SPOUSE)
```


```
* This software was written at Cedarville University by Doug Sjoquist
* and modified by Dave Rotman.  You may freely distribute this
* software, but this software is not to be sold by itself nor as
* part of any other software package.  A current version of the
* software may be obtained via anonymous ftp from:
*                    ftp.cedarville.edu
* This software is made available on an "as-is" basis, with no
* warranty of any kind.
*
*
* For more information about Cedarville University, visit:
*     www.cedarville.edu
```

Available File Formats

DOWNLOAD can produce output in a variety of formats as described below. Which format you use will likely be dictated by the application that will be reading your output file (spreadsheet, database, word processor, external service bureau, etc.). Here are the available formats:

COMMA

This is the default output format for DOWNLOAD files. Non-numeric fields are surrounded by quotation marks on the output. All fields are separated by commas. This file format can be used by many programs, including word processors and spreadsheets and is the traditional layout for importing data. (See also QUOTE, TAB, HTML, and XML formats.)

This is the basic approach:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT COMMA \
        FILE DOCS PEOPLE.DAT
"George Washington","Mt. Vernon",38.56,"Geo",49
"Abraham Lincoln","Gettysburg",123.12,"Abe",36
"William Clinton","Little Rock",108.00,”Bill”,57
```

If you want the field names to appear in your output file, try this:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT COMMA \
        FILE DOCS PEOPLE.DAT \
        HEADING FIELD.NAMES
"NAME","CITY","RATING","NICNAME","AGE"
"George Washington","Mt. Vernon",38.56,"Geo",49
"Abraham Lincoln","Gettysburg",123.12,"Abe",36
"William Clinton","Little Rock",108.00,"Bill",57
```

You may change the character used to separate fields using the COMMA.CHAR option and you may change the quotation character via the QUOTE.CHAR option.

If you want a character like a tab instead of a comma to separate the fields, you can specify FORMAT TAB instead of FORMAT COMMA. Note that FORMAT TAB omits the quotation marks surrounding data fields.

DBF

The DBF layout creates a file in native format for the database program dBASE. This layout can also be read by many other database packages (Paradox, FoxPro, etc.) and spreadsheet programs. You should use ".DBF" as part of your file name so that the database program can access the file.

```
DOWNLOAD CUSTOMERS \
        NAME ZIP \
        FORMAT DBF FILE DOCS CUST.DBF
```

The field names from the Unidata/Universe file will become the database field names for the dBASE file. Note that dBASE field names are limited to 10 characters. Longer names will be truncated by DOWNLOAD. If the truncation results in a duplicate field name, DOWNLOAD will adjust field names until each one is unique.

Supported dBASE field types include character, numeric, and date. Fields which are not dates and not numeric will be stored as character fields.

dBASE files are limited to about 65,000 records per file.

DIF

DIF is an acronym for "data interchange format". This format was developed many years ago to facilitate exchange of data between different software packages. It is (was?) considered a "native" format for Excel and QuattroPro, so output files in DIF format could be read directly as spreadsheet files (no importing is necessary). With later versions of spreadsheets, you may get better results using TAB, HTML, or XML formats.

This is an example of creating a DIF file:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT DIF FILE DOCS PEOPLE.DIF
```

If you want the field names to be the first row in the spreadsheet, use this variation (with the HEADING option):

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT DIF FILE DOCS PEOPLE.DIF \
        HEADING FIELD.NAMES
```

FIXED

This layout forces the values of fields in every record to be formatted to the same length. The actual length of each field is controlled by the dictionary FMT option, or you

can override the FMT when you run DOWNLOAD. The FIXED layout is often used by service bureaus. This layout is also useful if the data will be read by older languages like COBOL and FORTRAN. Some people like to use this layout for spreadsheet importing, though doing so requires a "parse" statement in the spreadsheet. (The COMMA, DIF, HTML, TAB, and XML formats are much easier to use with spreadsheets.)

```
DOWNLOAD PEOPLE \
      NAME CITY RATING NICKNAME AGE \
      FORMAT FIXED \
      FILE DOCS PEOPLE.DAT
George Washington   Mt. Vernon    38.56Geo       49
Abraham Lincoln     Gettysburg   123.12Abe       36
William Clinton     Little Rock  108.00Bill      57

DOWNLOAD PEOPLE \
      NAME CITY RATING NICKNAME AGE \
      FORMAT FIXED \
      FILE DOCS PEOPLE.DAT \
      HEADING FIELD.NAMES
NAME                CITY       RATING  NICKNAME AGE
George Washington   Mt. Vernon    38.56Geo       49
Abraham Lincoln     Gettysburg   123.12Abe       36
William Clinton     Little Rock  108.00Bill      57
```

HTML

This layout produces an HTML text file used by browsers such as Netscape and Internet Explorer. The output file would typically be copied to a Web server (or written directly there, if the user has a VOC pointer to an appropriate Web-server directory). Here is an example of generating an HTML file (note that many Web sites require that all file names use lowercase characters):

```
DOWNLOAD PEOPLE \
      NAME CITY RATING NICKNAME AGE \
      FORMAT HTML \
      FILE DOCS people.htm
```

Related keywords include HTML.TITLE, HTML.BODY, HTML.TOP, , HTML.TABLE, and HTML.BOTTOM. Field qualifiers that can be used include HTML.CELL, HTML.START, HTML.END, and HTML.ROW.

The HTML format can also be an efficient mechanism for transferring data to a spreadsheet. In Excel, for example, you can directly read an HTML file containing a table and the cell values will be placed appropriately in the spreadsheet.

QUOTE

This is similar to the COMMA format, except that all fields are surrounded by quotation marks on the output (as opposed to just the non-numeric fields). All fields are separated by commas. This file format can be used by many programs, including word processors and spreadsheets. (See also COMMA and TAB formats.)

This is the basic approach:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT QUOTE \
        FILE DOCS PEOPLE.DAT
"George Washington","Mt. Vernon",”38.56","Geo",”49"
"Abraham Lincoln","Gettysburg",”123.12","Abe",”36"
"William Clinton","Little Rock",”108.00","Bill",”57"
```

If you want the field names to appear in your output file, try this:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT COMMA \
        FILE DOCS PEOPLE.DAT \
        HEADING FIELD.NAMES
"NAME","CITY","RATING","NICNAME","AGE"
"George Washington","Mt. Vernon",”38.56","Geo",”49"
"Abraham Lincoln","Gettysburg",”123.12","Abe",”36"
"William Clinton","Little Rock",”108.00","Bill",”57"
```

TAB

This is similar to the COMMA format, except that the fields are separated by TAB characters. Quotation marks are omitted from all fields. This file format can be used by many programs, including word processors and spreadsheets.

This is the basic approach:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT TAB \
        FILE DOCS PEOPLE.DAT
George Washington<tab>Mt. Vernon<tab>38.56<tab>Geo<tab>49
Abraham Lincoln<tab>Gettysburg<tab>123.12<tab>Abe<tab>36
William Clinton<tab>Little Rock<tab>108.00<tab>Bill<tab>57
```

WP50

This layout produces a "merge" file used by WordPerfect version 5.0. The output file can be used to generate letters, envelopes, etc. using the WordPerfect 5.0 merge

operations.

Here is an example of generating a WP50 merge file:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT WP50 \
        FILE DOCS PEOPLE.DAT
```

WP51

This layout produces a "merge" file used by WordPerfect version 5.1 and later versions (including Windows versions). The output file can be used to generate letters, envelopes, etc. using the WordPerfect merge operations. This format is also suitable for use by Microsoft Word.

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT WP51 \
        FILE DOCS PEOPLE.DAT
```

XML

This layout produces an XML file used by software applications for data exchange, creating Web-pages, doing e-commerce, etc. Here is an example of generating an XML file:

```
DOWNLOAD PEOPLE \
        NAME CITY RATING NICKNAME AGE \
        FORMAT XML \
        FILE DOCS people.xml
```

Related keywords include FIELD.GAP, XML.ALLOW.PERIODS, XML.ROOT.NAME, XML.FILE.NAME, and XML.UPCASE. Field qualifiers that can be used include XML.ATTRIBUTE, XML.NAME, XML.GROUP.NAME, XML.ASSOC.NAME, and XML.SUBASSOC.NAME.

The XML format can also be an efficient mechanism for transferring data to a spreadsheet. In Excel, for example, you can directly read an XML file and the data values will be placed appropriately in the spreadsheet.

The XML capabilities of DOWNLOAD are designed to be easy to use but not necessarily as comprehensive as commercial XML products. XML files can be produced by DOWNLOAD with a simple command-line syntax, without resorting to template files or other special programs.

Defining Output Data

Overview

Output data for DOWNLOAD can be obtained by:

- Using a data field or I-descriptor (virtual field) described in the dictionary of the file being processed
- Using a literal value
- Using dictionary entries in the VOC file or in some other data file
- Using an additional data file which has a logical relationship to the file being processed (without having to create TRANS virtual fields)
- Calling a subroutine
- Using an “EVAL” expression
- Using a pre-defined "@" variable

All of these items may also include qualifiers or prefixes which further define how the value is to be downloaded. There are also command options that may be used to change the default behavior of DOWNLOAD. This chapter illustrates field definition. The next chapter explains each of the available qualifiers, modifiers, and command options.

Using Data Fields and Virtual Fields

You may use any data field from the data file(s) specified on your command line. In the example below, "NAME" and "HOME.PHONE" are fields in file CUSTOMERS.

```
SELECT CUSTOMERS SAMPLE
DOWNLOAD CUSTOMERS NAME HOME.PHONE
```

Fields may be specified via a dictionary phrase. For example, if the following dictionary item ‘BALANCES’ is present in the dictionary for CUSTOMERS:

```
001: PH
002: CUR.BAL OVER.30.BAL OVER.60.BAL
```

then these two DOWNLOAD statements are equivalent:

```
DOWNLOAD CUSTOMERS CUR.BAL OVER.30.BAL OVER.60.BAL
DOWNLOAD CUSTOMERS BALANCES
```

If you wish to fetch all data fields in a file, you may use the ‘@RECORD’ notation:

```
DOWNLOAD CUSTOMERS @RECORD
```

Ordinarily, you would also want to specify that all values will be processed:

```
DOWNLOAD CUSTOMERS @RECORD DEFAULT NUM.VALUES ALL
```

The ‘@RECORD’ notation may also be used with secondary files. The following statement will produce a file of customers (id number and name) and all information

about the salesperson assigned to each customer.

```
DOWNLOAD CUSTOMERS \
        CUST.ID \
        CUST.NAME \
        SECONDARY.FILE SALESFORCE \
                KEY CUST.SALESPERSON \
                ALIAS SF \
        SF->@RECORD \
        FILE _HOLD_ CUSTOMER.DAT
```

Using Literal Values

Literal values can be used wherever a regular data field would be expected. The default format and length for this type of value is the actual length of the data, left-justified. (The formatting can be overridden; see "CONV" and "FMT" options in the next chapter.)

These commands would produce a file to be used in generating labels or letters to parents of students:

```
SELECT STUDENTS SAMPLE 3
DOWNLOAD STUDENTS LITERAL "To the parents of:" \
        NAME STREET CITY ZIP \
        FILE _HOLD_ STU.DAT
```

Output records would look like this:

```
"To the parents of:","Adam Warren","58 Scott St","Aurora","IN","46509"
"To the parents of:","Susan Van Til","P.O. Box 17","Silas","MN","50187"
"To the parents of:","Mary Smith","4238 Main","Boktaw","WY","80261"
```

“LIT” may be used in place of “LITERAL”:

```
SELECT STUDENTS SAMPLE 3
DOWNLOAD STUDENTS LIT "To the parents of:" \
        NAME STREET CITY ZIP \
        FILE _HOLD_ STU.DAT
```

Using VOC Items

DOWNLOAD can retrieve data based on dictionary entries which have been placed in the VOC. To use this feature, create the item in DICT VOC, compile it, and then copy it to the VOC file. For example, suppose that you want to be able to include the length of the first data field in each output record. Create the DICT entry shown below and then follow the commands illustrated:

```
DICT VOC LEN.FIELD1
001: I
002: LEN(FIELD(@RECORD,@FM,1,1))
```

```
003:
004: LEN.FIELD1
005: 4R
006: S

CD VOC LEN.FIELD1
COPY FROM DICT VOC TO VOC 'LEN.FIELD1'
GET.LIST MYLIST
DOWNLOAD CUSTOMERS NAME LEN.FIELD1
```

In this example, NAME is a field on the CUSTOMERS file and LEN.FIELD1 is an I-descriptor defined in the VOC file. Because LEN.FIELD1 is defined in the VOC, it can be used when DOWNLOADing **any** file.

Using an Alternate Dictionary

You can DOWNLOAD using data from one file and control the DOWNLOAD using a dictionary of another file. See USING in the chapter "Syntax Guide" for more information.

Using an Additional Data File

DOWNLOAD has the capability of obtaining data from more than one file during processing. This is especially useful when a field in the first file is actually the key to a second file. See SECONDARY.FILE in the chapter "Syntax Guide" for more information.

Using Subroutines

Subroutines are invoked in a fashion similar to use of SUBR in I-descriptors. The subroutine should return the value to be downloaded in the first argument. If the value being returned is multi-valued, then the field qualifier MULTI.VALUE should be added since the default is single-value. The default format and length for this type of value is "30L".

Suppose that letters are to be generated to customers, indicating the balance due, the due date, and the rate of interest. The rate of interest depends on the customer rating and the size of the balance due. A subroutine named "GET.RATE" has been written to determine the interest rate. "GET.RATE" has the following definition:

```
SUBROUTINE GET.RATE(OUT.INT.RATE,IN.CUST.RATING,IN.BAL.DUE)
```

This subroutine can be called directly by DOWNLOAD without having to create a virtual field:

```
SELECT CUSTOMERS WITH BAL.DUE GT 0.00
DOWNLOAD CUSTOMERS \
        BAL.DUE DUE.DATE \
        SUBR('GET.RATE',CUSTOMER.RATING,BAL.DUE) \
```

```
        CONV "MD2" FMT "6R" \
    NAME STREET CITY STATE ZIP \
    FILE DOCS CUSTOMER.DAT
```

Subroutines used by DOWNLOAD can have up to ten arguments.

Using EVAL Expressions

EVAL expressions can be used wherever a regular data field would be expected. The default format and length for this type of value is the actual length of the data, left-justified. (The formatting can be overridden; see "CONV" and "FMT" options in the next chapter.) The EVAL operation in DOWNLOAD invokes the native database processing for EVAL, so there is 100% compatibility with the native EVAL. Note, however, that there is a significant performance penalty (sometimes exceeding 400%) for using an EVAL expression compared with a virtual field containing the same logic. This performance penalty is insignificant when used for heading records, HTML title information, etc. but could be significant when used for detail lines.

These commands would produce a file to be used in generating labels or letters to students encouraging them to set a target for improving their grade-point average:

```
SELECT STUDENTS SAMPLE 3
DOWNLOAD STUDENTS \
        CUM.GPA \
        EVAL "CUM.GPA+50" CONV “MD2" \
        NAME STREET CITY ZIP \
        FILE _HOLD_ STU.DAT
```

Output records would look like this:

```
1.83,2.33,"Adam Warren","58 Scott St","Aurora","IN","46509"
3.24,3.74,"Susan Van Til","P.O. Box 17","Silas","MN","50187"
2.76,3.26,"Mary Smith","4238 Main","Boktaw","WY","80261"
```

Note that EVAL operates on the record key (@ID) and record (@ID) from the primary file. It is not possible to use EVAL directly against a secondary file (although the EVAL expression can do a translate to the secondary file).

Using "@" Variables

"@" variables (pronounced "at variables") give you access to predefined system values as shown below. These variables can be used anywhere a regular dictionary item can be used. Here is a typical example using the "@DATE" variable:

```
SELECT CUSTOMERS SAMPLE 5
DOWNLOAD CUSTOMERS \
        NAME BUS.PHONE @DATE CONV "D4/" \
        FILE DOCS CUSTOMER.DAT
```

Output would look like this:

```
"Richards Paint Shop","937-555-4040","03/15/2002"
"Sarah's Sewing Center","888-555-4321","03/15/2002"
"JemCenter","402-404-4060","03/15/2002"
"Supreme Fire Stoppers, Inc.","800-555-5555","03/15/2002"
"Abas Abacus Company","616-444-1212","03/15/2002"
```

The "@" variables can be used to generate a heading record required by some service bureaus. Suppose that the header must contain a record count and the date that the data file was created. The following commands could be used:

```
GET.LIST MY.LIST
DOWNLOAD CUSTOMERS \
    DETAIL NAME BUS.PHONE \
    HEADING @YEAR @MONTH @DAY \
        @SYSTEM.RETURN.CODE FMT "6'0'R" \
    FILE DOCS PHONE.DAT FORMAT FIXED
```

The first record in the output file would look like this (assuming that the DOWNLOAD occurred on 04/17/2002 and the select list contained 156 records):

```
20020417000156
```

Detail records (after the heading record) would look like this:

```
Harrison, William E.          513-777-9812
Johnson, Wilma                937-444-1212
Kennedy, Marla Ima            800-412-9812
```

Here is an example using the "@COUNTER" variable:

```
SELECT CUSTOMERS SAMPLE 5
DOWNLOAD CUSTOMERS \
    @COUNTER NAME BUS.PHONE \
    FILE DOCS CUSTOMER.DAT
```

Output would look like this:

```
1,"Richards Paint Shop","937-555-4040"
2,"Sarah's Sewing Center","888-555-4321"
3."JemCenter","402-404-4060"
4."Supreme Fire Stoppers, Inc.","800-555-5555"
5,"Abas Abacus Company","616-444-1212"
```

At variables may also be used in output filenames. See the “File” section in the “Syntax Guide”.

Each of the "@" variables is described in the table below.

Description of "@" Variables		
Variable	Definition and typical usage	Sample output
@ACCOUNT	host operating system path where DOWNLOAD is being run DOWNLOAD MYFILE @ACCOUNT	/user3/livedir
@COUNTER	display a running counter (first record will be 1, second record will be 2, etc.)	38
@DATE	system date (internal format) that DOWNLOAD began running DOWNLOAD MYFILE @DATE DOWNLOAD MYFILE @DATE CONV "MD4/"	12860 03/17/2003
@DAY	day of the month that DOWNLOAD began running DOWNLOAD MYFILE @DAY	17 (run on 03/17/03)
@LOGNAME	login name for person running DOWNLOAD DOWNLOAD MYFILE @LOGNAME	harryg
@MONTH	month of the year that DOWNLOAD began running DOWNLOAD MYFILE @MONTH	03 (run on 03/17/03)
@PATH	host operating system path where DOWNLOAD is being run DOWNLOAD MYFILE @PATH	/user3/livedir
@SYSTEM.RETURN.CODE	system return code at the start of DOWNLOAD execution (This is usually the number of records in active select list.) GET.LIST MY.LIST 48 records retrieved to list 0. DOWNLOAD MYFILE @SYSTEM.RETURN.CODE	48
@TIME	system time (internal format) that DOWNLOAD began running DOWNLOAD MYFILE @TIME DOWNLOAD MYFILE @TIME CONV "MTH"	37197 10:19AM
@YEAR	four-digit year that DOWNLOAD began running DOWNLOAD MYFILE @YEAR	2003 (run on 03/17/03)

Getting Help

To see a current copy of DOWNLOAD's online help file, just enter:

DOWNLOAD HELP

at the database prompt. More complete help is available in the document 'download.pdf' provided with the software.

To see the current maximum values for number of data fields, literals, etc. that DOWNLOAD can process in a single run, enter:

DOWNLOAD HELP MAX

at the database prompt.

There is an electronic "listserv" for people who wish to share questions, hints, etc. regarding the use of DOWNLOAD. For more information, connect to:

http://mail.cedarville.edu/mailman/listinfo/download-list

Syntax Guide

@-VARIABLES

"@" variables give you access to predefined system values such as the date and time. See the chapter "Defining Output Data" for details.

ALIAS

Some commands which use a SECONDARY.FILE may become quite long and hard to read. The ALIAS option will let you shorten the command. The command:

```
DOWNLOAD STUDENTS CLASS MAJOR \
        SECONDARY FILE PEOPLE KEY @ID \
        PEOPLE->LASTNAME PEOPLE->FIRSTNAME \
        PEOPLE->PHONE
```

could be written as:

```
DOWNLOAD STUDENTS CLASS MAJOR \
        SECONDARY FILE PEOPLE KEY @ID ALIAS PEO \
        PEO->LASTNAME PEO->FIRSTNAME \
        PEO->PHONE
```

APPEND

If the DOWNLOAD command specifies an output file which already exists, the default behavior is to terminate with an error message. If you want the output of the current session to be added to an existing file, use the APPEND option. The APPEND option is only appropriate for ASCII-style output (FIXED, COMMA, HTML, QUOTE, TAB, and XML).

```
DOWNLOAD CUSTOMERS NAME ZIP \
        FILE DOCS CUSTOMER.DAT APPEND
```

If the output file format is HTML, DOWNLOAD will remove the closing ‘</BODY>’ and ‘</HTML>’ tags from the existing file prior to adding to the file. The additions will not include ‘<HTML>’, ‘<TITLE>’, or ‘<BODY>’ tags since these should already be present in the existing file. Here is an example which will produce two different HTML tables in a single output file:

```
GET.LIST BIGCUST
DOWNLOAD CUSTOMERS \
        NAME ZIP \
        FILE DOCS CUSTOMER.DAT \
        FORMAT HTML
```

```
GET.LIST BIGINV
DOWNLOAD INVOICES \
      INV.NUM INV.DESC INV.AMOUNT \
      FILE DOCS CUSTOMER.DAT APPEND \
      FORMAT HTML
```

If the output file format is XML, DOWNLOAD will suppress the XML version tag when the APPEND option is used.

See also OVERWRITING.

AVERAGE

To include the average of a numeric field in a control-break line, use the AVERAGE modifier. (See also BREAK.ON and FOOTING.ON.)

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE \
      FOOTING.ON STATE STATE AVERAGE BAL.DUE
```

```
"FL",600.00
"FL",300.00
"FL",450.00                    {this is the detail break line}
"OH",180.00
"OH",                          {note the null value)
"OH",60.00
"OH",80.00                     {this is the detail break line}
"TN",900.00
"TN",900.00                    {this is the detail break line}
"TN",340.00                    {this is the final break line}
```

Adding the NO.NULLS qualifier will exclude null values from calculation of the average.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE \
      FOOTING.ON STATE STATE AVERAGE BAL.DUE NO.NULLS
```

```
"FL",600.00
"FL",300.00
"FL",450.00                    {this is the detail break line}
"OH",180.00
"OH",                          {note the null value}
"OH",60.00
"OH",120.00                    {this is the detail break line}
"TN",900.00
"TN",900.00                    {this is the detail break line}
"TN",408.00                    {this is the final break line}
```

BEGIN

For readability, you may wish to modify how you store paragraphs which execute DOWNLOAD. DOWNLOAD command lines can be as long as your operating environment will allow. You can split the lines using the command-continuation character for your environment (typically, a backslash "\" or underscore "_"), or you can use the BEGIN/END built into DOWNLOAD. The following three paragraphs are equivalent.

```
001: PA
002: GET.LIST <<I2,LIST TO GET>>
003: DOWNLOAD CUSTOMERS NAME CITY ZIP
```

```
001: PA
002: GET.LIST <<I2,LIST TO GET>>
003: DOWNLOAD CUSTOMERS \
004:    NAME \
005:    CITY \
006:    ZIP
```

```
001: PA
002: GET.LIST <<I2,LIST TO GET>>
003: DOWNLOAD CUSTOMERS BEGIN
004: DATA NAME
005: DATA CITY
006: DATA ZIP
007: DATA END
```

BEG.COL

One method of setting up layouts for fixed-length records is to specify the beginning column for each field. The command:

```
DOWNLOAD CUSTOMERS NAME BEG.COL 1 \
        STATE BEG.COL 40 \
        ZIP BEG.COL 55 \
        FORMAT FIXED
```

would produce an output file where the NAME would be in positions 1-39, the STATE in positions 40-54, and the ZIP starting in position 55.

BREAK.ON

The BREAK.ON clause works much like the BREAK.ON clause for the LIST statement. BREAK.ON allows you to generate a total line when the value of the specified field changes. By default, the break line contains the same fields as the detail lines. The layout of the total line can be changed by using the

FOOTING.ON clause.

Example using the default break line:

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE
```

```
"FL",552.87
"FL",300.00
"FL",300.00                    {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"OH",50.00                     {this is the detail break line}
"TN",985.12
"TN",985.12                    {this is the detail break line}
"TN",985.12                    {this is the final break line}
```

Note that the break lines are exact duplicates of the preceding detail lines. The example below uses explicit break line specifications:

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
      FOOTING.ON STATE \
      LITERAL "SUBTOT" TOTAL BAL.DUE
      FOOTING.ON FINAL \
      LITERAL "GRANDTOT" TOTAL BAL.DUE
```

```
"FL",552.87
"FL",300.00
"SUBTOT",852.87                {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"SUBTOT",425.00                {this is the detail break line}
"TN",985.12
"SUBTOT",985.12                {this is the detail break line}
"GRANDTOT",2289.99             {this is the final break line}
```

BREAK.SUP

The BREAK.SUP option causes a control-break to occur without causing the field to appear on output lines. This option is typically used when you do not want the the break field to appear on the detail lines.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.SUP STATE \
      NAME BAL.DUE \
      FOOTING.ON STATE \
      STATE LITERAL "SUBTOT" TOTAL BAL.DUE
      FOOTING.ON FINAL \
```

LITERAL "GRAND" LITERAL "TOT" TOTAL BAL.DUE

```
"Adams, William",552.87
"Cooker, Anne",300.00
"FL","SUBTOT",852.87          {this is the detail break line}
"Smith, Betty",250.00
"Billings, Thomas",125.00
"Cowder, Mary Ann",50.00
"OH","SUBTOT",425.00          {this is the detail break line}
"Belgia, Torin",985.12
"TN","SUBTOT",985.12          {this is the detail break line}
"GRAND","TOTAL",2289.99       {this is the final break line}
```

BY.EXP

The BY.EXP option tells DOWNLOAD to process the active select list as an exploded list and to assume that the explosion was done on the field specified. Other fields which have the same association (field 7 of the dictionary) as the exploded field will be handled accordingly. The commands:

```
SELECT STUDENTS BY.EXP REG.TERMS \
    WHEN REG.TERMS LIKE '...FA...'
DOWNLOAD STUDENTS NAME BY.EXP REG.TERMS \
    REG.TERMS REG.STATUS
```

will generate a DOWNLOAD where the student name appears on each output record, the registration term will appear only if it contains the string "FA", and the registration status corresponding to the "FA" terms will appear.

If a STUDENTS record looks like this:

```
001: 96/FA}97/WI}97/SP}97/FA}98/WI  {REG.TERMS}
002: F}P}P}W}F                      (REG.STATUS}
003: Washington                     {NAME}
```

the output would look like this:

```
"Washington","96/FA","F"            {the first value}
"Washington","97/FA","W"            {the fourth value}
```

Note that the BY.EXP clause assumes that the active select list is an exploded list. DOWNLOAD does not do sorting/exploding. This is one syntax difference between LIST and DOWNLOAD.

COL.HDG

There may be occasions when you want to override the heading for a dictionary item or set the heading for an EVAL expression. This is especially important in producing XML output.

```
SELECT CUSTOMERS SAMPLE
DOWNLOAD CUSTOMERS \
    @ID \
```

CUST.NAME \
CUST.BAL \
HEADING FIELD.NAMES

would produce:

```
"@ID","CUST.NAME","CUST.BAL"        {heading}
"408","ABC Power",58.17             {detail}
"156","Henry Carpets",158.50        {detail}
```

SELECT CUSTOMERS SAMPLE
DOWNLOAD CUSTOMERS \
@ID \
CUST.NAME COL.HDG "Name" \
CUST.BAL \
HEADING FIELD.NAMES

would produce:

```
"@ID","Name","CUST.BAL"             {heading}
"408","ABC Power",58.17             {detail}
"156","Henry Carpets",158.50        {detail}
```

SELECT CUSTOMERS SAMPLE
DOWNLOAD CUSTOMERS \
@ID \
CUST.NAME \
CUST.BAL \
EVAL "IF CUST.BAL GT 10000 THEN 'HIGH' ELSE ''"
FORMAT XML

would produce:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>408</id>
  <custname>ABC Power</custname>
  <custbal>58.17</custbal>

<evalifcustbalgt10000thenhighelse></evalifcustbalgt10000thenhighel
se>
</customers>
<customers>
  <id>156</id>
  <custname>Henry Carpets</custname>
  <custbal>158.50</custbal>

<evalifcustbalgt10000thenhighelse>HIGH</evalifcustbalgt10000thenhi
ghelse>
</customers>
</download>
```

```
SELECT CUSTOMERS SAMPLE
DOWNLOAD CUSTOMERS \
        @ID \
        CUST.NAME \
        CUST.BAL \
        EVAL "IF CUST.BAL GT 10000 THEN 'HIGH' ELSE ''" \
                COL.HDG "warning" \
        FORMAT XML
```

would produce:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>408</id>
  <custname>ABC Power</custname>
  <custbal>58.17</custbal>
  <warning></warning>
</customers>
<customers>
  <id>156</id>
  <custname>Henry Carpets</custname>
  <custbal>158.50</custbal>
  <warning>HIGH</warning>
</customers>
</download>
```

COLUMNS

You can specify the starting and ending columns for a field using the COLUMNS option. Note that this option is only appropriate for a FIXED format output.

```
DOWNLOAD CUSTOMERS NAME PHONE \
        COLUMNS 51 65 ZIP FORMAT FIXED
```

will place the PHONE number in columns 51 through 65 of the output. The ZIP code will start in column 66.

COMMA.CHAR

When producing an output file in comma-quote format, data values are separated by a literal comma. If you wish to use a different separator, specify it using the COMMA.CHAR option. See also VALUE.SEPARATOR and SUBVALUE.SEPARATOR.

Contrast:

```
DOWNLOAD CUSTOMERS NAME PHONE
"ABC Tooling","937-555-1212"
"Middletown Catering","800-555-9898"
```

with:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    COMMA.CHAR "@"
"ABC Tooling"@"937-555-1212"
"Middletown Catering"@"800-555-9898"
```

A common use of the COMMA.CHAR option is to insert a tab between fields. This can be done as follows:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    COMMA.CHAR ^9
```

The carat ("^") tells DOWNLOAD that the "9" references ASCII character 9 (the tab character) rather than a literal "9". An easier way to obtain this result is to specify FORMAT TAB (no need to specify COMMA.CHAR or QUOTE.CHAR).

The value specified by COMMA.CHAR does not have to be a single character. The following is an acceptable use of COMMA.CHAR:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    COMMA.CHAR “<—>"
```

CONV

You can override the dictionary conversion (or specify a conversion for literals, subroutines, and "@" variables) using the CONV field qualifier.

```
DOWNLOAD STUDENTS NAME GPA CONV "MD34"
```

will produce the field GPA using three decimal positions (of the four that are stored on the file), rather than using whatever conversion was specified in the STUDENTS dictionary.

```
DOWNLOAD CUSTOMERS @DATE CONV "D4/"
```

will produce the system date in MM/DD/YYYY format, rather than the (default) internal format.

“CNV” can be used as a synonym for “CONV”:

```
DOWNLOAD STUDENTS NAME GPA CNV "MD34"
```

DEBUG.LEVEL

This option is designed to be used by system administrators who are resolving problems with running DOWNLOAD. The following debug levels are available:

1	General program flow between external subroutines
2	Detailed program flow of internal subroutines
4	General parsing of the command line
8	Detailed parsing of the command line

16 General calculation of output results
32 Detailed calculation of output results

The following DOWNLOAD statement will show the general parsing of the command line:

```
DOWNLOAD CUSTOMERS NAME ZIP DEBUG.LEVEL 4
```

Multiple debug levels can be invoked by adding the numeric values for the respective levels. For example, to get levels 1 and 4, using a debug level of 5:

```
DOWNLOAD CUSTOMERS NAME ZIP DEBUG.LEVEL 5
```

DEFAULT

The DEFAULT option lets you change DOWNLOAD's defaults. For example, only the first value of a multi-value list appears unless a NUM.VALUES clause modifies the default behavior. If there are many multi-valued fields being used, you may want to set the default behavior to "NUM.VALUES ALL". The following two statements will produce identical results:

```
DOWNLOAD STUDENTS REG.TERMS NUM.VALUES ALL \
    REG.STATUS NUM.VALUES ALL \
    REG.ACTION NUM.VALUES ALL
DOWNLOAD STUDENTS REG.TERMS REG.STATUS REG.ACTION \
    DEFAULT NUM.VALUES ALL
```

DEFAULT.VALUE

If a field being produced by DOWNLOAD is null, it can be replaced (on the output file) by a default value you specify. Contrast the two situations below:

```
DOWNLOAD CUSTOMERS NAME PHONE
"American Plastics","937-555-1212"
"Billings Ink",""
"Cameron Catering","513-666-8888"

DOWNLOAD CUSTOMERS NAME \
    PHONE DEFAULT.VALUE "No Phone"
"American Plastics","937-555-1212"
"Billings Ink","No Phone"
"Cameron Catering","513-666-8888"
```

DETAIL

Unless otherwise specified, each field listed on the command line is assumed to be part of the "detail" output, as opposed to being part of a heading or footing line. In some complicated situations, you may want to explicitly define which fields belong to the detail line.

```
DOWNLOAD CUSTOMERS \
    HEADING @DATE CONV "D2/" @SYSTEM.RETURN.CODE \
    DETAIL NAME PHONE
```

will ensure that NAME and PHONE are part of the detail line, not the heading line. The heading line will contain the system date and system return code.

While not advisable, you may use the keyword DETAIL as many times as you wish. (Your logic will be clearer if you name all of the detail fields at one time.) Consider the clarity of the following statement, which is equivalent to the previous example:

```
DOWNLOAD CUSTOMERS HEADING @DATE CONV "D2/" \
    DETAIL NAME \
    HEADING @SYSTEM.RETURN.CODE \
    DETAIL PHONE
```

DET.SUP

DET.SUP is used in conjunction with a BREAK.ON or BREAK.SUP phrase. Using DET.SUP will cause DOWNLOAD to skip the production of the detail lines. Your output will only contain the heading and footing lines.

DICT

The keyword DICT can be used to specify an alternate dictionary. For example, suppose you want to use data from file CUSTOMERS but you want to use the dictionary from the file CONTACTS. You could do this by creating a VOC entry which points to CUSTOMERS data and CONTACTS dictionary, or you could use the following command:

```
DOWNLOAD CUSTOMERS USING DICT CONTACTS \
    PERSON HOME.PHONE
```

Note that PERSON and HOME.PHONE are dictionary entries from CONTACTS. The data, however, will be obtained from the CUSTOMERS file. The keywords 'USING' and 'DICT' must be adjacent.

DISPLAY.COUNT

DISPLAY.COUNT causes the progress-meter asterisks to display on the screen as DOWNLOAD produces the output file. This is the default behavior. To turn off the display of the asterisks, use NO.DISPLAY.COUNT. To control how many asterisks are printed (i.e., how many records are processed before an asterisk is printed), use PROGRESS.INTERVAL.

END

The keyword END is used to terminate a BEGIN/END block. See the keyword BEGIN for more information.

END.COL

END.COL can be used to specify where data values will end when using FIXED format output. Consider:

```
DOWNLOAD CUSTOMERS NAME BAL.DUE END.COL 75 \
        FORMAT FIXED FILE DOCS CUSTOMER.DAT
```

The NAME field will begin in column 1. The BAL.DUE field will end in column 75 of the output. If the format for BAL.DUE is 10R, then BAL.DUE will start in column 66 (so that the ending column is 75).

EOR.CHAR

When transferring data between operating systems, it is sometimes necessary to adjust the characters which appear between records of the output. For example, if you are creating a file on a Unix system and that file will be processed as ASCII text on a personal computer, you may need to add a carriage return to the end of each line. This can be done as follows:

```
DOWNLOAD CUSTOMERS NAME PHONE \
        FILE DOCS CUSTOMER.DAT \
        EOR.CHAR ^13
```

The carat ("^") tells DOWNLOAD that the "13" references ASCII character 13 (the carriage-return character) rather than a literal "13".

EVAL

The EVAL keyword lets you specify a run-time calculation in situations where you do not want to create a dictionary entry. Note that EVAL imposes a severe performance penalty when used on detail lines compared with using a virtual field to accomplish the same results.

```
DOWNLOAD CUSTOMERS \
        NAME \
        LOCPHONE \
        EVAL "'(937)-':LOCPHONE"
"American Plastics","555-1212","(937)-555-1212"
"Billings Ink","","(937)-"
"Cameron Catering","666-8888","(937)-666-8888"

DOWNLOAD CUSTOMERS \
        NAME \
        LOCPHONE \
        EVAL "IF LOCPHONE = '' THEN '' ELSE '(937)-':LOCPHONE"
```

```
"American Plastics","555-1212",”(937)-555-1212"
"Billings Ink",”“,”“
"Cameron Catering","666-8888",”(937)-666-8888"
```

The prefixes TOTAL, AVERAGE, etc. cannot be used with an EVAL expression. (It is best to create a virtual field for situations where you need break values of an EVAL expression.)

FIELD.GAP

If you wish to "spread out" data produced in a FIXED format, use the FIELD.GAP option. Contrast these two examples:

```
DOWNLOAD STUDENTS ID.NO CLASS GPA \
       FORMAT FIXED
8076513FR3.568
9134892SO2.500
2342382JR4.000
2912833FR1.897

DOWNLOAD STUDENTS ID.NO CLASS GPA \
       FORMAT FIXED \
       FIELD.GAP 3
8076513   FR   3.568
9134892   SO   2.500
2342382   JR   4.000
2912833   FR   1.897
```

For XML output, FIELD.GAP controls the indentation of elements. If FIELD.GAP is not specified, a default indentation of 2 spaces is used.

```
DOWNLOAD STUDENTS ID.NO CLASS GPA \
       FORMAT XML
<?xml version="1.0"?>
<download>
<students>
  <idno>12345</idno>
  <class>FR</class>
  <gpa>4.000</gpa>
</students>
<students>
  <idno>87654</idno>
  <class>SO</class>
  <gpa>3.097</gpa>
</students>
</download>

DOWNLOAD STUDENTS ID.NO CLASS GPA \
       FIELD.GAP 8 \
       FORMAT XML
```

```
<?xml version="1.0"?>
<download>
<students>
        <idno>12345</idno>
        <class>FR</class>
        <gpa>4.000</gpa>
</students>
<students>
        <idno>87654</idno>
        <class>SO</class>
        <gpa>3.097</gpa>
</students>
</download>
```

FIELD.LABELS

The FIELD.LABELS option provides an easy method for generating a heading when using COMMA, FIXED, HTML, or TAB formats. FIELD.LABELS tells DOWNLOAD to insert the field label (item 4 of the dictionary) of each field on the HEADING line. (See also "HEADING" for creating customized headings.)

Consider this example:

```
DOWNLOAD CUSTOMERS \
        DETAIL NAME PHONE ZIP \
        HEADING FIELD.LABELS
```

```
"Customer Name","Telephone","Zip Code"          {heading}
"Harrison Electric","616-444-9283","49418"      {detail}
"General Toy Repair","800-123-4400","88012"     {detail}
"My Pet Store","912-421-1234","20001"           {detail}
```

FIELD.NAMES

The FIELD.NAMES option provides an easy method for generating a heading when using COMMA, FIXED, HTML, or TAB formats. FIELD.NAMES tells DOWNLOAD to insert the name of each field on the HEADING line. (See also "HEADING" for creating customized headings.)

Consider this example:

```
DOWNLOAD CUSTOMERS \
        DETAIL NAME PHONE ZIP \
        HEADING FIELD.NAMES
```

```
"NAME","PHONE","ZIP"                            {heading}
"Harrison Electric","616-444-9283","49418"      {detail}
"General Toy Repair","800-123-4400","88012"     {detail}
"My Pet Store","912-421-1234","20001"           {detail}
```

FILE

By default, DOWNLOAD sends all of its output to the screen. If you want to send your output to a file, use the FILE option. This is especially important if you are creating a WordPerfect or dBASE output file! Following the FILE keyword, you should specify the directory and file name (record name) for storing the output.

```
DOWNLOAD CUSTOMERS NAME PHONE \
    FILE MYDOCS CUSTOMER.DAT
```

will place the output file named CUSTOMER.DAT in directory MYDOCS.

If you are running DOWNLOAD repetitively, you may want to use the OVERWRITING option:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    FILE MYDOCS CUSTOMER.DAT OVERWRITING
```

By default, DOWNLOAD will not create a new output file if a file of the same name already exists. You must use the OVERWRITING option if you want the new output file to replace the old one. The APPEND option will let you add records to an existing output file.

The FILE clause will also accept "@" variables. For example, suppose you wish to have the date and time of the program execution become part of the file name. The following example shows how this can be done:

```
SELECT CUSTOMERS WITH BALANCE GT 0
DOWNLOAD CUSTOMERS NAME PHONE BALANCE \
    FILE MYDOCS OWE_@DATE_@TIME
```

The resulting file name will have this appearance:

```
OWE_11956_36040
```

The underscores are optional delimiters when using "@" variables, so you could use:

```
SELECT CUSTOMERS WITH BALANCE GT 0
DOWNLOAD CUSTOMERS NAME PHONE BALANCE \
    FILE MYDOCS OWE@DATE@TIME
```

The resulting file name will have this appearance:

```
OWE1195636040
```

Be careful about introducing undesired characters such as asterisks ("*"), greater-than signs (">"), and slashes ("/") when using "@" variable file names. In particular, you would ordinarily not want to use @ACCOUNT or @PATH in a variable file name.

There may be occasions when the system administrator wants to grant permission to create DOWNLOADed files in a certain directory or any of its subdirectories. This can be accomplished by creating a VOC entry only for the main directory. For example, suppose that the following Unix directories exist:

/disk3/regdata
/disk3/regdata/FALL
/disk3/regdata/WINTER
/disk3/regdata/SPRING

DOWNLOAD can access all of these directories by creating a single VOC entry:

VOC REGDATA
001: DIR
002: /disk3/regdata
003: D_DIR

The following DOWNLOAD commands would write to the various directories:

DOWNLOAD STUDENTS NAME FILE REGDATA MYDAT
DOWNLOAD STUDENTS NAME FILE REGDATA/FALL MYDAT
DOWNLOAD STUDENTS NAME FILE REGDATA/WINTER MYDAT
DOWNLOAD STUDENTS NAME FILE REGDATA/SPRING MYDAT

The "/" character is the system delimiter for path names on Unix systems. If you are operating on a Windows system or a Prime system, you would use a backslash “\” or greater-than sign ">", respectively.

FINAL

The FINAL keyword is used to specify that the FOOTING being defined is the "grand total" line, the very last footing line to be produced.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
      FOOTING.ON STATE \
      LITERAL "SUBTOT" TOTAL BAL.DUE
      FOOTING.ON FINAL \
      LITERAL "GRANDTOT" TOTAL BAL.DUE
```

```
"FL",552.87
"FL",300.00
"SUBTOT",852.87              {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"SUBTOT",425.00              {this is the detail break line}
"TN",985.12
"SUBTOT",985.12              {this is the detail break line}
"GRANDTOT",2289.99           {this is the final break line}
```

FMT

FMT can be used to override the existing dictionary format or the default format for literals, subroutines, and "@" variables. The syntax is identical to the FMT function within UniBASIC.

```
DOWNLOAD CUSTOMERS \
        NAME FMT "35L" \
        BAL.DUE FMT "12R" \
        NUMBER.OF.ORDERS FMT "8'0'R"
```

In this example, the NAME will be produced using 35 columns and will be left-justified. The BAL.DUE field will be right-justified using 12 columns. The NUMBER.OF.ORDERS field will be right-justified using 8 columns, and any empty columns will be zero-filled.

FOOTING

The FOOTING option will create a single record in the output file after all other output records have been produced. Typical usage would be to add a "trailer" record when creating data for a service bureau.

```
GET.LIST MAILING
295 records retrieved to list 0.
DOWNLOAD CUSTOMERS FORMAT FIXED \
        NAME STREET CITY STATE ZIP \
        FOOTING @SYSTEM.RETURN.CODE FMT "8'0'R" \
        FILE DOCS MAILING.DAT
```

The output file in this case will contain 296 records. There will be 295 detail records (showing customer name, street, city, state, and zip) and one last record containing "00000296".

FOOTING.ON

This option is used in conjunction with the BREAK.ON and BREAK.SUP options. FOOTING.ON is used to specify the contents of control-break output.

```
SELECT CUSTOMERS BY ZIP BY NAME WITH BAL.DUE GT 0.00
DOWNLOAD CUSTOMERS \
        DETAIL ZIP NAME BAL.DUE \
        BREAK.SUP ZIP \
        FOOTING.ON ZIP ZIP LITERAL "TOTALS" TOTAL BAL.DUE \
        FOOTING.ON FINAL LITERAL "GRAND" LITERAL "TOTALS" \
                TOTAL BAL.DUE
```

```
"45314","Adams Excavating",500.00
"45314","Yonker's Donuts",300.00
"45314","TOTALS",800.00
"46517","Elko Camera",250.00
"46517","Furniture by Bill",100.00
"46517","Home Town",200.00
"46517","TOTALS",550.00
"GRAND","TOTALS",1350.00
```

The keyword “NONE” can be used to suppress the footing line. This would typically be done when you are using a break with HEADING.ON (to get some information at the top of each group) but do not want a break line at the end of each group.

```
SELECT CUSTOMERS BY ZIP BY NAME WITH BAL.DUE GT 0.00
DOWNLOAD CUSTOMERS \
      DETAIL ZIP NAME BAL.DUE \
      BREAK.SUP ZIP \
      HEADING.ON ZIP LITERAL “STARTING” ZIP
```

```
“STARTING”,”45314"                          {heading}
"45314","Adams Excavating",500.00
"45314","Yonker's Donuts",300.00
"45314","Yonker's Donuts",300.00            {footing}
“STARTING”,”46517"                          {heading}
"46517","Elko Camera",250.00
"46517","Furniture by Bill",100.00
"46517","Home Town",200.00
"46517","Home Town",200.00                  {footing}
```

```
SELECT CUSTOMERS BY ZIP BY NAME WITH BAL.DUE GT 0.00
DOWNLOAD CUSTOMERS \
      DETAIL ZIP NAME BAL.DUE \
      BREAK.SUP ZIP \
      HEADING.ON ZIP LITERAL “STARTING” ZIP
      FOOTING.ON ZIP NONE
```

```
“STARTING”,”45314"                          {heading}
"45314","Adams Excavating",500.00
"45314","Yonker's Donuts",300.00
“STARTING”,”46517"                          {heading}
"46517","Elko Camera",250.00
"46517","Furniture by Bill",100.00
"46517","Home Town",200.00
```

FORMAT

DOWNLOAD can produce output files in a variety of formats. You can specify which format you want using the FORMAT option. See the chapter "Available File Formats" for details.

FROM

You may instruct DOWNLOAD to process an active select list other than list zero by using the FROM option. Compare:

```
GET.LIST MY.LIST
```

17 records retrieve to list 0.
DOWNLOAD CUSTOMERS NAME PHONE

with:

GET.LIST MY.LIST TO 3
17 records retrieve to list 3.
DOWNLOAD CUSTOMERS NAME PHONE FROM 3

The FROM option can be useful when a virtual field calls a subroutine which manipulates select list zero.

HEADING

The HEADING option generates record(s) in front of all the other output records. It can be used as a heading in the traditional sense (showing field names, for instance), or it can be used to show record counts, dates, etc. that might be required by the application which will use the output file.

Examples:

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
      DETAIL NAME PHONE ZIP \
      HEADING FIELD.NAMES

"NAME","PHONE","ZIP"                              {heading}
"Harrison Electric","616-444-9283","49418"        {detail}
"General Toy Repair","800-123-4400","88012"       {detail}
"My Pet Store","912-421-1234","20001"             {detail}
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
      DETAIL NAME PHONE ZIP \
      HEADING LITERAL “Customer” LITERAL “Phone” \
            LITERAL “Zip”

"Customer","Phone","Zip"                          {heading}
"Harrison Electric","616-444-9283","49418"        {detail}
"General Toy Repair","800-123-4400","88012"       {detail}
"My Pet Store","912-421-1234","20001"             {detail}
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
```

```
        DETAIL NAME PHONE ZIP FORMAT FIXED \
        HEADING @SYTEM.RETURN.CODE FMT "8'0'R"
```

```
00000003                                        {heading}
Harrison Electric   616-444-928349418           {detail}
General Toy Repair  800-123-440088012           {detail}
My Pet Store        912-421-123420001           {detail}
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
        DETAIL NAME PHONE ZIP FORMAT FIXED \
        HEADING EVAL “OCONV(DATE(),’D4/’)”
```

```
12/24/2003                                      {heading}
Harrison Electric   616-444-928349418           {detail}
General Toy Repair  800-123-440088012           {detail}
My Pet Store        912-421-123420001           {detail}
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
        DETAIL NAME PHONE ZIP FORMAT FIXED \
        HEADING EVAL “OCONV(DATE(),’D4/’)” FMT “12L” \
                EVAL “OCONV(TIME(),’MTH’)”
```

```
12/24/2003  05:17PM                             {heading}
Harrison Electric   616-444-928349418           {detail}
General Toy Repair  800-123-440088012           {detail}
My Pet Store        912-421-123420001           {detail}
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS \
        DETAIL NAME PHONE ZIP FORMAT FIXED \
        HEADING LITERAL “Acme Products” \
        LITERAL “Customer Listing” LINE 2 \
        LITERAL “As of ‘D’” LINE 3 \
        LITERAL “ ” LINE 4 \
        LITERAL “Name” FMT “20L” \
        LITERAL “Phone” FMT “12L” \
        LITERAL “Zip” \
        LITERAL “ ” LINE 5
```

```
Acme Products                                    {heading}
Customer Listing                                 {heading}
As of 03/22/02                                   {heading}
                                                 {heading}
Name                Phone       Zip              {heading}
                                                 {heading}
Harrison Electric   616-444-928349418            {detail}
General Toy Repair  800-123-440088012            {detail}
My Pet Store        912-421-123420001            {detail}
```

HEADING.ON

If you need a special record in front of each "control-break" group of output records, use the HEADING.ON option to produce that record.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        DETAIL STATE NAME \
        BREAK.SUP STATE \
        HEADING.ON STATE LITERAL "STARTING" STATE \
        FOOTING.ON STATE NONE
```

```
"STARTING","AR"
"AR","Razorback Industries"
"AR","Alpha Connections"
"STARTING","MI"
"MI","Motor City News"
"MI","Michigan Outdoors"
"MI","Alpena Journal"
```

HTML.BODY

This clause is used specify the body for the output Web page. The default is an empty BODY clause (white background).

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.BODY ‘ BGCOLOR=”#FFFF33#” ’  \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

HTML.BOTTOM

This clause is used to enter HTML tags and text which will appear just after the output data table (but before the '</body>' tag).

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE "Acme Customer List" \
        HTML.BOTTOM \
         "<H2><CENTER>***CONFIDENTIAL***</CENTER></H2>"\
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will center the text "***CONFIDENTIAL***" horizontally on the page just after the data table showing the customers.

The keyword EVAL may also be used with HTML.BOTTOM to generate a custom result at run-time. If this EVAL expression would references data fields, those fields will be obtained from the last record in the active list.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE "Acme Customer List" \
        HTML.BOTTOM EVAL \
         "'<H2>Produced by ':@LOGNAME:'</H2>'"\
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will place text like "Produced by jsmith" on the page just after the data table showing the customers.

HTML.CELL

This clause is used supply HTML code which will be applied to a particular cell in the output table (within the <td> tag).

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME HTML.CELL ' BGCOLOR="BLUE" ' \
```

```
        STATE \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will produce a table with three columns (id number, name, state). Each of the customer names will be in cells with a blue background.

The HTML.CELL clause can include an EVAL expression. The example below will place a red background in the CUST.BALANCE cell whenever that balance is over $50.00.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        CUST.BALANCE \
        HTML.CELL EVAL \
        "IF CUST.BALANCE GT 5000 THEN 'BGCOLOR=RED' ELSE '"" \
        STATE \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

HTML.HEAD

This clause is used specify HTML code (such as style-sheet references) which will appear within the <HEAD> section of the output file.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
HTML.HEAD \
'<style type="text/css"> .ourhue {background-color: #3333FF;} </style>' \
@ID \
CUSTOMER.NAME \
STATE \
HTML.ROW ' class=ourhue' \
HEADING LITERAL "ID" \
        LITERAL "Name" \
        LITERAL "State" \
        HTML.ROW 'class=ourhue' \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will generate an HTML file where each row in the table has color #3333FF. As with standard HTML, lots of other attributes could be set in an embedded style sheet.

You can also reference an external style sheet:

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
HTML.HEAD \
‘<link href=”http://my.company.com/styles/our.css” rel=”stylesheet”
type=”text/css”>’ \
@ID \
CUSTOMER.NAME \
STATE \
HTML.ROW ‘ class=ourhue’ \
HEADING LITERAL “ID” \
        LITERAL “Name” \
        LITERAL “State” \
        HTML.ROW ‘class=ourhue’ \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

Often, the URL for the stylesheet will be abbreviated, since the style sheet and web pages will likely reside on the same server:

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
HTML.HEAD \
‘<link href=”/styles/our.css” rel=”stylesheet” type=”text/css”>’ \
@ID \
CUSTOMER.NAME \
STATE \
HTML.ROW ‘ class=ourhue’ \
HEADING LITERAL “ID” \
        LITERAL “Name” \
        LITERAL “State” \
        HTML.ROW ‘class=ourhue’ \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

HTML.END

This clause is used specify the closing tag of an HTML container-style tag. See HTML.START for an example.

HTML.ROW

This clause is used to specify HTML code which applies to an entire row of the output table (within the TR tag).

```
SELECT CUSTOMERS BY STATE SAMPLE 5
```

```
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.ROW ‘ BGCOLOR=”BLUE” ’ \
        HEADING LITERAL “ID” \
                LITERAL “Name” \
                LITERAL “State” \
                HTML.ROW ‘ BGCOLOR=”RED” ’ \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will produce a table in which the heading row is red and the detail rows are blue.

HTML.ROW can also use an EVAL expression to set row characteristics:

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
@ID \
CUSTOMER.NAME \
STATE \
HTML.ROW EVAL \
“IF STATE = ‘NY’ THEN ‘BGCOLOR=BLUE’ ELSE ‘BGCOLOR=RED’” \
HEADING LITERAL “ID” \
        LITERAL “Name” \
        LITERAL “State” \
        HTML.ROW ‘ BGCOLOR=”RED” ’ \
FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

HTML.START

This clause is used to specify HTML code which will affect the contents of a particular cell in the output table. Note that HTML.START affects the actual contents of the cell, not the <td> tag (see HTML.CELL to do that). A corresponding HTML.END phrase may be used for container tags (tags that have a start-tag and end-tag).

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUST.NAME \
                HTML.START ‘<B>’ HTML.END ‘</B>’ \
        CUST.STATE \
```

```
FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will produce a table in which each customer name is shown in bold.

HTML.START can also use an EVAL expression. This example will use bold text for the ranking of any customer with a rank less than 50.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
    @ID \
    CUST.NAME \
    CUST.RANK \
    HTML.START EVAL \
    "IF CUST.RANK GE 50 THEN '' ELSE '<B>'"
    HTML.END EVAL \
    "IF CUST.RANK GE 50 THEN '' ELSE '</B>'"\
    CUST.STATE \
    FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

HTML.TABLE

This clause is used to specify characteristics of the output data table. The default is a border size of 1 (small, visible lines). To use invisible lines, set the border to zero.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
    @ID \
    CUSTOMER.NAME \
    STATE \
    HTML.TABLE ' BORDER="0" '  \
    FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

The HTML.TABLE clause can use an EVAL expression instead of a literal. The expression will be evaluated using the first record in the active select list.

HTML.TITLE

This clause is used to set an HTML title (the text which will be displayed on the top menu bar of the Web browser). Note that the TITLE might be different from a text line appearing just in front of the data table.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
```

```
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE “Acme Customer List” \
        HTML.TOP “<H2>Customers</H2>” \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will produce an HTML page with the following structure:

```
<HTML>
<HEAD>
<TITLE>
Acme Customer List
</TITLE>
</HEAD>
<BODY>
<H2>Customers</H2>
<TABLE BORDER=”1">
...
</TABLE>
</BODY>
</HTML>
```

The keyword EVAL may also be used with HTML.TITLE to generate a custom result at run-time. If this EVAL expression would references data fields, those fields will be obtained from the first record in the active list.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE EVAL “‘Customer List ‘:OCONV(DATE(),’D4/’)” \
        HTML.TOP “<H2>Customers</H2>” \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will produce an HTML page with the following structure:

```
<HTML>
<HEAD>
<TITLE>
Customer List 05/17/2003
</TITLE>
</HEAD>
<BODY>
```

```
<H2>Customers</H2>
<TABLE BORDER=”1">
...
</TABLE>
</BODY>
</HTML>
```

HTML.TOP

This clause is used to enter HTML tags and text which will appear prior to the output data table.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE “Acme Customer List” \
        HTML.TOP \
         “<H2><CENTER>**** CONFIDENTIAL ***</CENTER></H2>” \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will center the text “*** CONFIDENTIAL ***” horizontally on the page just prior to the data table showing the customers. Note that this text need not be the same as the text specified in the title.

The keyword EVAL may also be used with HTML.TOP to generate a custom result at run-time. If this EVAL expression would references data fields, those fields will be obtained from the first record in the active list.

```
SELECT CUSTOMERS BY STATE SAMPLE 5
5 records selected to list 0.
DOWNLOAD CUSTOMERS \
        @ID \
        CUSTOMER.NAME \
        STATE \
        HTML.TITLE “Acme Customer List” \
        HTML.TOP EVAL \
         “‘<H2>’:OCONV(DATE(),’D4/’):’ REPORT</H2>’” \
        FORMAT HTML FILE _HOLD_ CUSTOMER.HTM
```

will generate a line like:

```
05/17/2003 REPORT
```

on the page just prior to the data table showing the customers.

KEY

This clause is used to identify the record key for a secondary file. Consider this example:

```
DOWNLOAD CUSTOMERS \
        SECONDARY.FILE EMPLOYEES \
        KEY SALES.REP \
        NAME EMPLOYEES->EMP.NAME
```

This DOWNLOAD command uses a primary file named CUSTOMERS and a secondary file named EMPLOYEES. For each record in CUSTOMERS, the output will contain the NAME (from the CUSTOMERS file) and the EMP.NAME (from the EMPLOYEES file). The key to the EMPLOYEES file is the field SALES.REP on the CUSTOMERS file.

The example above is equivalent to creating an I-descriptor called EMP.NAME on CUSTOMERS as shown below and then using the DOWNLOAD command shown below.

```
DICT CUSTOMERS EMP.NAME
001: I
002: TRANS('EMPLOYEES',SALES.REP,'EMP.NAME','X')

DOWNLOAD CUSTOMERS \
        NAME EMP.NAME
```

See the description for ALIAS and SECONDARY.FILE for further explanations.

LENGTH

The LENGTH field qualifier can be used when producing FIXED output to control the size of a field. (This could also be done using the FMT field qualifier.)

```
DOWNLOAD CUSTOMERS FORMAT FIXED \
        NAME LENGTH 35 ZIP LENGTH 12 PHONE LENGTH 15
```

will produce an output file with NAME in columns 1-35, ZIP in columns 36-47, and PHONE in columns 48-62.

LINE

If you need to produce more than one output line for each input record, use the LINE command. The examples below illustrate a couple of variations achieved using the LINE field qualifier. Note that LINE is a field qualifier, appearing after the field that it references.

Producing each field on a line by itself (could also be achieved using RECORD.ORIENTATION):

```
DOWNLOAD CUSTOMERS FORMAT FIXED \
    NAME LINE 1 \
    ZIP LINE 2 \
    PHONE LINE 3 \
    BAL.DUE LINE 4
```

```
Adams Manufacturing
46514
219-555-0876
  568.12
Smith Furniture
60606
312-498-1234
   49.00
```

Multiple fields on one line:

```
DOWNLOAD CUSTOMERS FORMAT FIXED \
    NAME \
    ZIP \
    PHONE \
    BAL.DUE LINE 2
```

```
Adams Manufacturing   46514   219-555-0876
  568.12
Smith Furniture       60606   312-498-1234
   49.00
```

Once specified, the LINE field qualifier remains in effect until it is overridden:

```
DOWNLOAD CUSTOMERS FORMAT FIXED \
    NAME LINE 1\
    ZIP \
    PHONE LINE 2\
    BAL.DUE
```

```
Adams Manufacturing   46514
219-555-0876  568.12
Smith Furniture       60606
312-498-1234   49.00
```

The LINE qualifier cannot be used for structured output (WP50, WP51, HTML, DIF, DBF, and XML.)

LITERAL

The LITERAL command lets you produce the same character string on all output records. This character string can be entered in a paragraph or prompted at execution time.

```
DOWNLOAD CUSTOMERS \
        NAME BAL.DUE \
        LITERAL "05/17/2002"
```

```
"Adams Welding",56.87,"05/17/2002"
"Miller Inc.",123.98,"05/17/2002"
"Excitement",1000.00,"05/17/2002"
```

To have the literal string change each time you execute the command, try an inline prompt:

```
DOWNLOAD CUSTOMERS \
        NAME BAL.DUE \
        LITERAL "<<DUE DATE,2N/2N/4N>>"
```

For commands which have several LITERAL fields, you may want to use LIT as a shorter keyword than LITERAL:

```
DOWNLOAD CUSTOMERS \
        NAME BAL.DUE \
        LIT "05/17/2002" \
        LIT “Reminder” \
        LIT “Urgent” \
        CUSTOMER.CONTACT
```

LPTR

Adding LPTR to the DOWNLOAD command will cause the output-record layout to be sent to the line printer. This clause is ignored if the layout is not being printed (see PRINT.LAYOUT).

MAX

The MAX modifier is used in conjunction with a BREAK.ON or BREAK.SUP option. MAX will generate the maximum value in the group that it follows.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
        FOOTING.ON STATE \
        LITERAL "LARGEST" MAX BAL.DUE
        FOOTING.ON FINAL \
        LITERAL "GRANDTOT" TOTAL BAL.DUE
```

```
"FL",552.87
"FL",300.00
"LARGEST",552.87              {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"LARGEST",250.00              {this is the detail break line}
"TN",985.12
"LARGEST",985.12              {this is the detail break line}
"GRANDTOT",2289.99            {this is the final break line}
```

MIN

MIN finds the minimum value within a control-break group. Here is an example:

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
      FOOTING.ON STATE \
      LITERAL "SMALLEST" MIN BAL.DUE
      FOOTING.ON FINAL \
      LITERAL "GRANDTOT" TOTAL BAL.DUE
```

```
"FL",552.87
"FL",300.00
"SMALLEST",300.00             {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"SMALLEST",50.00              {this is the detail break line}
"TN",985.12
"SMALLEST",985.12             {this is the detail break line}
"GRANDTOT",2289.99            {this is the final break line}
```

MULTI.VALUE

You can have a field which is defined to be single-valued treated as a multi-valued field by DOWNLOAD by using the MULTI.VALUE modifier on the field.

```
DOWNLOAD CUSTOMERS \
      NAME STREET MULTI.VALUE NUM.VALUES ALL
```

will treat the STREET as a multi-valued field and will show all of the values for each record, even if the dictionary for CUSTOMERS shows STREET as a single-valued field.

MV.ORIENTATION

For some applications like spreadsheets, you want multi-valued output to line up in columns. You can accomplish this with the MV.ORIENTATION option.

Here is an example using comma-quote format:

```
DOWNLOAD STUDENTS \
      NAME \
      REG.TERMS NUM.VALUES ALL MV.ORIENTATION VERTICAL
```

```
"Harris, Amy","1998FA"
,"1999G1"
,"1999SP"
,"2000FA"
"Jones, Thomas","1997SP"
,"2002FA"
```

Here is the same example using fixed format:

```
DOWNLOAD STUDENTS FORMAT FIXED \
      NAME \
      REG.TERMS NUM.VALUES ALL MV.ORIENTATION VERTICAL
```

```
Harris, Amy     1998FA
                1999G1
                1999SP
                2000FA
Jones, Thomas   1997SP
                2002FA
```

If you want the single-valued fields to repeat on each output record, you can use an exploded select list:

```
SELECT STUDENTS SAMPLE 2 BY NAME BY.EXP REG.TERMS
6 records retrieved to list 0.
DOWNLOAD STUDENTS FORMAT FIXED \
      NAME \
      BY.EXP REG.TERMS REG.TERMS \
```

```
Harris, Amy     1998FA
Harris, Amy     1999G1
Harris, Amy     1999SP
Harris, Amy     2000FA
Jones, Thomas   1997SP
Jones, Thomas   2002FA
```

Note in this last example that MV.ORIENTATION and NUM.VALUES would be meaningless, because we are referencing each of the values explicitly through the exploded select list.

NONE

The keyword NONE can be used to suppress the generation of a break record. In the example below, break line on STATE is being suppressed.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
      FOOTING.ON STATE NONE \
      FOOTING.ON FINAL \
      LITERAL "GRANDTOT" TOTAL BAL.DUE

"FL",552.87
"FL",300.00
"OH",250.00
"OH",125.00
"OH",50.00
"TN",985.12
"GRANDTOT",2289.99           {this is the final break line}
```

NO.DISPLAY.COUNT

NO.DISPLAY.COUNT turns off the progress meter (printing of asterisks) that usually occurs when DOWNLOAD is processing a large select list.

NO.LINEFEED

NO.LINEFEED tells DOWNLOAD to produce each output record without generating a line feed. This type of file can be used by programs which accept streaming input rather than record-based input. This option is applicable only to the FIXED format layout. Compare these two examples:

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS ID.NO STATE FORMAT FIXED

102324MI
402832IN
239482OH
```

```
GET.LIST MY.LIST
3 records retrieved to list 0.
DOWNLOAD CUSTOMERS ID.NO STATE FORMAT FIXED \
      NO.LINEFEED

102324MI402832IN239482OH
```

NO.NULLS

NO.NULLS excludes null-values from calculation of averages (see AVERAGE).

NO.PAGE

NO.PAGE turns off the screen pausing which is typical in the Unidata/Universe database environment. In particular, as DOWNLOAD generates the progress meter (rows of asterisks), the screen may fill. The system will pause at the full screen, indicating that you should press <new line> to continue. If the job is being run at night, there might not be anyone around to press the <new line> key. By including NO.PAGE in the DOWNLOAD command, you will be assured that the program will not be waiting on keyboard input.

NO.PRINT.ERRORS

When doing repetitive processing with DOWNLOAD, you may have a situation where some records in your select list will generate errors (record not found, illegal field value, etc.), but you want to consider this "normal" and not have DOWNLOAD generate an error report. The NO.PRINT.ERRORS option will suppress printing of the error report.

NUM.SUBVALUES

If a multi-valued field is being printed and that field contains sub-values, DOWNLOAD will only copy the first subvalue of each value to the output. If you want more than one subvalue included, specify the NUM.SUBVALUES field qualifier.

```
DOWNLOAD STUDENTS NAME \
        TERM.CONTACTS NUM.VALUES ALL NUM.SUBVALUES 3
```

would produce output with all values for TERM.CONTACTS and (for any particular TERM.CONTACT), include up to 3 subvalues.

The NUM.SUBVALUES phrase can also be used on an exploded select list. Suppose that student records contain a TERMS field showing the terms for which the student was enrolled. For each particular term value, there is an associated set of subvalues in field COURSES showing the courses the student took that term. Compare the following DOWNLOAD commands:

```
SELECT STUDENTS \
        BY NAME BY.EXP TERMS
DOWNLOAD STUDENTS \
        BY.EXP TERMS \
        NAME \
        TERMS \
        COURSES
```

```
"Adams, Henry","2002FA","MATH-101"
"Adams, Henry","2003SP","HIST-304"
```

```
"Arbuckle, Jane","2001FA","BIO-104"
```

```
SELECT STUDENTS \
        BY NAME BY.EXP TERMS
DOWNLOAD STUDENTS \
        BY.EXP TERMS \
        NAME \
        TERMS \
        COURSES NUM.SUBVALUES ALL
```

```
"Adams, Henry","2002FA","MATH-101","ENG-104","PHYS-109"
"Adams, Henry","2003SP","HIST-304","MATH-102","PHYS-110"
"Arbuckle, Jane","2001FA","BIO-104","HIST-399"
```

NUM.VALUES

By default, DOWNLOAD only uses the first value of a multi-valued field. If you want more values included in the output, use the NUM.VALUES field qualifier. You may specify a particular number of values to be used or you may use the keyword ALL to indicate that you want all values.

Note that specifying a particular number of values will cause each record of the output to **always** have that number of values. If you specify the keyword ALL, then the number of values from one record to the next may vary, depending on the number of values on the input data.

Consider these data records:

```
Record one:  001:   ADAMS
             002:   1996}1997}1998}1999
Record two:  001:   SMITH
             002:   1994}1997
```

The default behavior (no NUM.VALUES clause):

```
DOWNLOAD CUSTOMERS NAME ACTIVE.YR
"ADAMS","1996"
"SMITH","1994"
```

Specifying two values per record:

```
DOWNLOAD CUSTOMERS NAME ACTIVE.YR \
        NUM.VALUES 2
"ADAMS","1996","1997"
"SMITH","1994","1997"
```

Specifying five values per record:

```
DOWNLOAD CUSTOMERS NAME ACTIVE.YR \
        NUM.VALUES 5
```

```
"ADAMS","1996","1997","1998","1999",""
"SMITH","1994","1997","","",""
```

Specifying all values:

```
DOWNLOAD CUSTOMERS NAME ACTIVE.YR \
    NUM.VALUES ALL
"ADAMS","1996","1997","1998","1999"
"SMITH","1994","1997"
```

OVERWRITING

If the DOWNLOAD command specifies an output file which already exists, the default behavior is to terminate with an error message. If you want the output file to be deleted and a new one created in its place, use the OVERWRITING option.

```
DOWNLOAD CUSTOMERS NAME ZIP \
    FILE DOCS CUSTOMER.DAT OVERWRITING
```

See also APPEND.

PRINT.ERRORS

This option merely reinforces the default behavior: DOWNLOAD generates an error report (on screen) if it encounters processing errors such as record not found, illegal field value, etc.

PRINT.LAYOUT

The PRINT.LAYOUT option generates a report describing the layout of the output file. This report is sometimes useful for debugging and for use by external service bureaus to document the record layouts. The report contains header information (file being processed, date, time, format) and a field listing. For FIXED format output, the report also shows beginning and ending column numbers.

PROGRESS.INTERVAL

By default, the progress meter shows an asterisk for every 10 records processed. You may change this interval using the PROGRESS.INTERVAL option.

```
DOWNLOAD CUSTOMERS NAME ZIP \
    PROGRESS.INTERVAL 50
```

will print an asterisk for every 50 records processed.

QUOTE.CHAR

When producing an output file in comma-quote format, non-numeric data values are surrounded by quotation marks. If you wish to use a different character around these values, specify it using the QUOTE.CHAR option.

Contrast:

```
DOWNLOAD CUSTOMERS NAME PHONE
"ABC Tooling","937-555-1212"
"Middletown Catering","800-555-9898"
```

with:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    QUOTE.CHAR "$"
$ABC Tooling$,$937-555-1212$
$Middletown Catering$,$800-555-9898$
```

You may set the quote character to null, obtaining results like the following:

```
DOWNLOAD CUSTOMERS NAME PHONE \
    QUOTE.CHAR ""
ABC Tooling,937-555-1212
Middletown Catering,800-555-9898
```

RECORD.LENGTH

If each record of the output must have the same length (i.e., must be padded with spaces), use the RECORD.LENGTH option.

```
DOWNLOAD CUSTOMERS NAME ZIP \
    FORMAT FIXED RECORD.LENGTH 60
```

will produce output records that are always 60 characters long.

RECORD.ORIENTATION

The default record orientation is horizontal (each line in the output represents a single record from the input). If you want each output field to appear on a line by itself, use the RECORD.ORIENTATION VERTICAL option.

```
DOWNLOAD CUSTOMERS NAME PHONE ZIP
"ABC Tooling","937-555-1212","46514"
"Middletown Catering","800-555-9898","55012"
```

```
DOWNLOAD CUSTOMERS NAME PHONE ZIP \
    RECORD.ORIENTATION VERTICAL
"ABC Tooling"
"937-555-1212"
"46514"
"Middletown Catering"
"800-555-9898"
"55012"
```

REMOVE.PUNCTUATION

Some vendors (like the United States Postal Service) require output without punctuation (commas, apostrophes, etc.). The REMOVE.PUNCTUATION option will delete these characters from your output file. Compare the following examples:

```
DOWNLOAD STUDENTS MAIL.NAME CITY
"Miss Monica J. Billings","St. Louis, MO 63115"
"Mr. Henry Jordan, Jr.","Xenia, OH 45385"

DOWNLOAD STUDENTS MAIL.NAME CITY REMOVE.PUNCTUATION
"Miss Monica J Billings","St Louis MO 63115"
"Mr Henry Jordan Jr","Xenia OH 45385"
```

See also UPCASE.

SAMPLE

The SAMPLE keyword functions just like the SAMPLE keyword in the LIST statement. You can use SAMPLE to select just the first few records from your file or active select list. The following two examples are equivalent:

```
DOWNLOAD CUSTOMERS NAME SAMPLE 8

SELECT CUSTOMERS SAMPLE 8
DOWNLOAD CUSTOMERS NAME
```

If you do not specify the number of records, SAMPLE will use a default of ten records.

SECONDARY.FILE

The SECONDARY.FILE option lets you reference fields from other files without creating a lot of I-descriptors. The SECONDARY.FILE option is also useful when a field in a file references another record in that same file.

```
DOWNLOAD PEOPLE \
    SECONDARY.FILE STUDENTS KEY @ID \
    LAST.NAME FIRST.NAME STUDENTS->CLASS
```

references fields LAST.NAME and FIRST.NAME from the PEOPLE file and the field CLASS from the students file (the same record key is used for both files).

```
DOWNLOAD STUDENTS \
    SECONDARY.FILE STUD.SCHEDS KEY LAST.SS.KEY \
    NAME STUD.SCHEDS->COURSE NUM.VALUES ALL
```

references field NAME from the STUDENTS file and field COURSE from the

STUD.SCHEDS file. The record key for STUD.SCHEDS is computed in field LAST.SS.KEY of the STUDENTS file.

```
DOWNLOAD PEOPLE \
        SECONDARY.FILE PEOPLE KEY PARENT.ID ALIAS PGS \
        SECONDARY.FILE PEOPLE KEY SPOUSE ALIAS SP \
        SECONDARY.FILE STUDENTS KEY @ID \
        NAME \
        STUDENTS->CLASS \
        PARENT.ID PGS->NAME SPOUSE SP->NAME \
        FORMAT FIXED FIELD.GAP 2
```

retrieves data from the PEOPLE and STUDENTS files:

NAME is the person's name from the PEOPLE file
CLASS is the person's class from the STUDENTS file
PARENT.ID is the id number of the person's parent
PGS->NAME is the name of the parent (accessed via PARENT.ID)
SPOUSE is the id number of the person's spouse
SP->NAME is the name of the spouse (accessed via SPOUSE)

In a real situation, we may want all of the student's parents who exist on our files. The following example illustrates getting both parents (or only one if only one is on file) and formatting the results in XML:

```
SELECT PERSON WITH PARENTS SAMPLE 3
DOWNLOAD PERSON \
@ID XML.ATTRIBUTE \
C26PER.NAME \
SECONDARY.FILE PERSON ALIAS PAR KEY PARENTS \
      NUM.VALUES ALL \
PAR->@ID XML.ATTRIBUTE \
PAR->C26PER.NAME \
FILE _HOLD_ PARENT.XML OVERWRITING \
FORMAT XML
```

Output from the above would look something like this:

```
<?xml version="1.0"?>
<download>
<person id="0044316">
  <c26pername>Johnson, Lynn B.</c26pername>
  <personparents parid="0012313">
    <parc26pername>Johnson, Charles V.</parc26pername>
  </personparents>
  <personparents parid="0049813">
    <parc26pername>Johnson, Mary K.</parc26pername>
  </personparents>
</person>
<person id="1423314">
  <c26pername>Gillis, Warren Lee</c26pername>
```

```
  <personparents parid="0088315">
    <parc26pername>Gillis, Warren Lee</parc26pername>
  </personparents>
</person>
<person id="4030747">
  <c26pername>Thompson, Kristen Sue</c26pername>
  <personparents parid="5512447">
    <parc26pername>Thompson, Kevin Robert</parc26pername>
  </personparents>
  <personparents parid="5512408">
    <parc26pername>Thompson-Jones, Millicent</parc26pername>
  </personparents>
</person>
</download>
```

SINGLE.VALUE

If you wish to treat a multi-valued field as single-valued, use the SINGLE.VALUE field qualifier:

```
DOWNLOAD CUSTOMERS NAME ORDER.DATES SINGLE.VALUE
```

The ORDER.DATES field will be treated as single-valued. This is actually the default behavior, and would likely be useful only if you had changed the default:

```
DOWNLOAD CUSTOMERS DEFAULT NUM.VALUES ALL \
    CONTACT.NAMES SITE.CITIES ORDER.DATES SINGLE.VALUE
```

would include all of the CONTACT.NAMES and SITE.CITIES but only the first ORDER.DATE for each record.

SUBR

The SUBR command allows you to call a subroutine to obtain data, as opposed to using a data field from the file. The syntax is identical to the use of SUBR in defining an I-descriptor. The DOWNLOAD command would look like this:

```
DOWNLOAD PROSPECTS \
    NAME HOME.PHONE \
    SUBR("RATE.PROSPECTS",INCOME,ZIP,EDUCATION)
```

where INCOME, ZIP, and EDUCATION are fields in the PROSPECTS file. The subroutine RATE.PROSPECTS uses these arguments to calculate a rating.

For more information on using SUBR, see the chapter on "Defining Output Data".

SUBVALUE.SEPARATOR

The SUBVALUE.SEPARATOR qualifier changes the character(s) used between subvalues of the same field. You may specify a single character for the separator, or a longer string. Specifying a longer string would be useful in generating HTML output as you would be able to specify strings like “<BR>”. SUBVALUE.SEPARATOR can be used with XML output to change the handling

of subvalues.

This example illustrates the default behavior for comma-quote output for a single record:

```
DOWNLOAD CUSTOMER @ID \
      SALESPERSON NUM.VALUES 2 MV.ORIENTATION VERTICAL \
      SALESPERSON.STATES NUM.SUBVALUES 3
```

```
123,"George","IN","KY","OH"
,"Bill","MI","NJ",""
```

Now, using the SUBVALUE.SEPARATOR qualifier:

```
DOWNLOAD CUSTOMER @ID \
      SALESPERSON NUM.VALUES 2 MV.ORIENTATION VERTICAL \
      SALESPERSON.STATES NUM.SUBVALUES 3 \
            SUBVALUE.SEPARATOR '~'
```

```
123,"George","IN"~"KY"~"OH"
,"Bill","MI"~"NJ"~""
```

This example illustrates the default behavior for XML output:

```
DOWNLOAD CUSTOMER @ID \
      SALESPERSON NUM.VALUES 2 \
      SALESPERSON.STATES NUM.SUBVALUES 3 \
      FORMAT XML
```

```
<?xml version="1.0"?>
<download>
<customer>
  <id>123</id>
  <salesassoc>
    <salesperson>George</salesperson>
    <salespersonstates>IN</salespersonstates>
    <salespersonstates>KY</salespersonstates>
    <salespersonstates>OH</salespersonstates>
  </salesassoc>
  <salesassoc>
    <salesperson>Bill</salesperson>
    <salespersonstates>MI</salespersonstates>
    <salespersonstates>NJ</salespersonstates>
    <salespersonstates></salespersonstates>
  </salesassoc>
</customer>
</download>
```

Now, using the SUBVALUE.SEPARATOR qualifier:

```
DOWNLOAD CUSTOMER \
        SALESPERSON NUM.VALUES 2 \
        SALESPERSON.STATES NUM.SUBVALUES 3 \
                SUBVALUE.SEPARATOR '~' \
        FORMAT XML
```

```
<?xml version="1.0"?>
<download>
<customer>
  <id>123</id>
  <salesassoc>
    <salesperson>George</salesperson>
    <salespersonstates>IN~KY~OH</salespersonstates>
  </salesassoc>
  <salesassoc>
    <salesperson>Bill</salesperson>
    <salespersonstates>MI~NJ~</salespersonstates>
  </salesassoc>
</customer>
</download>
```

Default Subvalue Separators by Output Format	
COMMA	comma character (,)
HTML	non-breaking space ( )
QUOTE	comma character (,)
TAB	(tab)
WP50	hard-return (Hrt)
WP51	hard-return (Hrt)
XML	<fieldname> and </fieldname>

TOTAL

Use the TOTAL keyword to define the contents of a control-break (footing) line.

```
SELECT CUSTOMERS BY STATE
DOWNLOAD CUSTOMERS BREAK.ON STATE BAL.DUE \
        FOOTING.ON STATE \
        LITERAL "SUBTOT" TOTAL BAL.DUE
        FOOTING.ON FINAL \
        LITERAL "GRANDTOT" TOTAL BAL.DUE
```

```
"FL",552.87
"FL",300.00
"SUBTOT",852.87              {this is the detail break line}
"OH",250.00
"OH",125.00
"OH",50.00
"SUBTOT",425.00              {this is the detail break line}
"TN",985.12
"SUBTOT",985.12              {this is the detail break line}
"GRANDTOT",2289.99           {this is the final break line}
```

UPCASE

The UPCASE keyword instructs DOWNLOAD to convert all output to upper case. This can be helpful when a vendor (e.g., the United States Postal Service) prefers information in upper case. Compare the following examples:

```
DOWNLOAD STUDENTS NAME CITY
"Billings, Monica","San Jose"
"Jordan, Henry","Wilmington"

DOWNLOAD STUDENTS NAME CITY UPCASE
"BILLINGS, MONICA","SAN JOSE"
"JORDAN, HENRY","WILMINGTON"
```

See also REMOVE.PUNCTUATION.

USING

The USING option allows you to DOWNLOAD one file but use a dictionary from a different file. See the reference for 'DICT' in this chapter or the chapter "Defining Output Data" for more information.

VALUE.SEPARATOR

The VALUE.SEPARATOR qualifier changes the character(s) used between values of the same field. By default, DOWNLOAD uses the COMMA.CHAR between values. (VALUE.SEPARATOR cannot be used with XML output.)

This example illustrates the default behavior:

```
DOWNLOAD CUSTOMER \
      SALESPERSON NUM.VALUES 3

123,"George","IN","KY","OH"
408,"Karen","MD","NJ","DC"
```

Now, using the VALUE.SEPARATOR qualifier:

```
DOWNLOAD CUSTOMER \
      SALESPERSON NUM.VALUES 3 VALUE.SEPARATOR " AND "

123,"George","IN" AND "KY" AND "OH"
408,"Karen","MD" AND "NJ" AND "DC"
```

The VALUE.SEPARATOR option might be useful in customizing HTML output, as the separator could include text like "<BR>" or other HTML code.

Default Value Separators by Output Format	
COMMA	comma character (,)
HTML	non-breaking space ( )
QUOTE	comma character (,)
TAB	(tab)
WP50	hard-return (Hrt)
WP51	hard-return (Hrt)

WHEN

The WHEN option for DOWNLOAD can be used to control when an output value appears. WHEN does **not** control which records get produced; only which values show. The following examples illustrate:

```
SELECT STUDENTS BY NAME BY.EXP REG.TERMS SAMPLE 3
7 records selected to list 0.
DOWNLOAD STUDENTS NAME BY.EXP REG.TERMS \
      REG.TERMS
"Johnson, Susan","94/SP"
"Johnson, Susan","95/FA"
"Johnson, Susan","96/FA"
"Kennedy, Abraham","93/FA"
"Kennedy, Abraham","94/WI"
"Larson, Jenny","94/FA"
"Larson, Jenny","95/FA"
```

```
SELECT STUDENTS BY NAME BY.EXP REG.TERMS SAMPLE 3
7 records selected to list 0.
DOWNLOAD STUDENTS NAME BY.EXP REG.TERMS \
```

```
        REG.TERMS WHEN REG.TERMS LIKE '...FA...'
"Johnson, Susan",""                 {note the null value}
"Johnson, Susan","95/FA"
"Johnson, Susan","96/FA"
"Kennedy, Abraham","93/FA"
"Kennedy, Abraham",""               {note the null value}
"Larson, Jenny","94/FA"
"Larson, Jenny","95/FA"
```

To get rid of the null values in the previous example, modify the SELECT:

```
SELECT STUDENTS BY NAME BY.EXP REG.TERMS SAMPLE 3 \
      WHEN REG.TERMS LIKE '...FA...'
5 records selected to list 0.
DOWNLOAD STUDENTS NAME BY.EXP REG.TERMS \
      REG.TERMS WHEN REG.TERMS LIKE '...FA...'
"Johnson, Susan","95/FA"
"Johnson, Susan","96/FA"
"Kennedy, Abraham","93/FA"
"Larson, Jenny","94/FA"
"Larson, Jenny","95/FA"
```

WRITE.INTERVAL

The WRITE.INTERVAL option is useful if your host system tends to "overrun" the destination system. For example, if you are executing DOWNLOAD on a Unix host but the output is being written to a network file server via an NFS-mounted volume, the host may write data faster than what the file server can accept it. To slow down the output, use the WRITE.INTERVAL command.

```
SELECT CUSTOMERS WITH CUST.SALESPERSON = "SMITH"
DOWNLOAD CUSTOMERS CUST.NAME CUST.ZIP \
      FILE HERO CUSTOMERS \
      WRITE.INTERVAL 20 3 \
      FORMAT FIXED LPTR
```

will cause DOWNLOAD to pause for 3 seconds after processing each set of 20 records. The default WRITE.INTERVAL is 10 records with a sleep time of zero (i.e., no pausing between groups of records).

XML.ALLOW.CHARACTERS

By default, DOWNLOAD removes the following characters from XML tags:

`<` `>` `&` `'` `"`

In values, these characters are replaced by an appropriate expression:

`<` `>` `&` `&apos;` `&quot;`

If you want output to include these special characters in values, you should

specify XML.ALLOW.CHARACTERS. This can be useful when output is specified via a literal or an EVAL expression. In the example below, the heading is used to provide some additional XML information and must include the less-than and greater-than signs in order to be valid XML.

```
DOWNLOAD CUSTOMERS \
        @ID \
        CUST.NAME \
        CUST.CITY \
        HEADING \
                LITERAL '<mycompany>Acme Data Products</mycompany>' \
                XML.ALLOW.CHARACTERS \
        FORMAT XML
```

If you have a situation where you want the "illegal" characters allowed in every output field, you may specify XML.ALLOW.CHARACTERS as the default, as in this example:

```
DOWNLOAD CUSTOMERS \
        DEFAULT XML.ALLOW.CHARACTERS \
        @ID \
        CUST.NAME \
        CUST.CITY \
        HEADING \
                LITERAL '<mycompany>Acme Data Products</mycompany>' \
        FORMAT XML
```

Exercise care when setting this default. In the example above, company names like

Jones & Smith
Sue's Ceramics

will generate invalid XML output if the ampersand and apostrophe are left in place.

XML.ALLOW.PERIODS

By default, DOWNLOAD removes periods from XML element names. Specifying XML.ALLOW.PERIODS on the command line will cause DOWNLOAD to leave periods in XML element names.

```
DOWNLOAD CUSTOMERS @ID CUST.NAME CUST.CITY \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <custcity>West Union</custcity>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <custcity>Montgomery</custcity>
</customers>
</download>
```

DOWNLOAD CUSTOMERS @ID CUST.NAME CUST.CITY \
XML.ALLOW.PERIODS \
FORMAT XML

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <cust.name>Harris Hardware</cust.name>
  <cust.city>West Union</cust.city>
</customers>
<customers>
  <id>3834</id>
  <cust.name>ACE Supply</cust.name>
  <cust.city>Montgomery</cust.city>
</customers>
</download>
```

XML.ASSOC.NAME

The XML.ASSOC.NAME option is used when you want the XML file to use a different name for an association element than one derived from the actual dictionary name for the association.

DOWNLOAD CUSTOMERS @ID CUST.NAME \
CUST.PHONE \
CUST.PHONE.TYPE \
FORMAT XML

will produce a file like this (assuming that the dictionary entry for CUST.PHONE lists ‘PHONES’ as the association name):

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <phones>
    <custphone>800-555-1212</custphone>
    <custphone.type>voice</custphonetype>
```

```
  </phones>
  <phones>
    <custphone>555-400-1212</custphone>
    <custphonetype>fax</custphonetype>
  </phones>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <phones>
    <custphone>404-555-8888</custphone>
    <custphonetype>voice</custphonetype>
  </phones>
</customers>
</download>
```

```
DOWNLOAD CUSTOMERS @ID CUST.NAME \
        CUST.PHONE XML.ASSOC.NAME 'cphone' \
        CUST.PHONE.TYPE XML.ASSOC.NAME 'cphone' \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <cphone>
    <custphone>800-555-1212</custphone>
    <custphonetype>voice</custphonetype>
  </cphone>
  <cphone>
    <custphone>555-400-1212</custphone>
    <custphonetype>fax</custphonetype>
  </cphone>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <cphone>
    <custphone>404-555-8888</custphone>
    <custphonetype>voice</custphonetype>
  </cphone>
</customers>
</download>
```

Specifying different association names will have the effect of breaking the association defined in the dictionary:

```
DOWNLOAD CUSTOMERS @ID CUST.NAME \
        CUST.PHONE XML.ASSOC.NAME 'cphone' \
        CUST.PHONE.TYPE XML.ASSOC.NAME 'newtype' \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <cphone>
    <custphone>800-555-1212</custphone>
  </cphone>
  <newtype>
    <custphonetype>voice</custphonetype>
  </newtype>
  <cphone>
    <custphone>555-400-1212</custphone>
  </cphone>
  <newtype>
    <custphonetype>fax</custphonetype>
  </newtype>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <cphone>
    <custphone>404-555-8888</custphone>
  </cphone>
  <newtype>
    <custphonetype>voice</custphonetype>
  </newtype>
</customers>
</download>
```

If XML.ASSOC.NAME is not specified and the dictionary does not contain an association name, DOWNLOAD will omit the association tags from the output. For example (assuming that INVOICE is multivalued but does not have an association specified in the dictionary), this statement:

```
DOWNLOAD CUSTOMERS \
      @ID \
      INVOICE NUM.VALUES ALL
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <invoice>134AR</invoice>
  <invoice>429AK</invoice>
  <invoice>306RM</invoice>
</customers>
<customers>
  <id>3834</id>
  <invoice>134AR</invoice>
  <invoice>042BY</invoice>
</customers>
</download>
```

In some situations, the use of XML.GROUP.NAME will present data in a more-

desirable form, as shown in the following modifications of the previous examples.

```
DOWNLOAD CUSTOMERS @ID CUST.NAME \
        CUST.PHONE XML.GROUP.NAME “phonenumbers” \
        CUST.PHONE.TYPE \
        FORMAT XML
```

will produce a file like this (assuming that the dictionary entry for CUST.PHONE lists ‘PHONES’ as the association name):

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <phonenumbers>
    <phones>
      <custphone>800-555-1212</custphone>
      <custphone.type>voice</custphonetype>
    </phones>
    <phones>
      <custphone>555-400-1212</custphone>
      <custphonetype>fax</custphonetype>
    </phones>
  </phonenumbers>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <phonenumbers>
    <phones>
      <custphone>404-555-8888</custphone>
      <custphonetype>voice</custphonetype>
    </phones>
  </phonenumbers>
</customers>
</download>
```

```
DOWNLOAD CUSTOMERS \
        @ID \
        INVOICE XML.GROUP.NAME “invoices” \
                XML.ASSOC.NAME “” \
                NUM.VALUES ALL
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <invoices>
    <invoice>134AR</invoice>
    <invoice>429AK</invoice>
    <invoice>306RM</invoice>
  </invoices>
</customers>
<customers>
  <id>3834</id>
```

```
  <invoices>
    <invoice>134AR</invoice>
    <invoice>042BY</invoice>
  </invoices>
</customers>
</download>
```

XML.ATTRIBUTE

By default, each field included in an XML DOWNLOAD is treated as an XML element (i.e., enclosed within its own start and end tags). If you want a particular field to be included as an attribute rather than as an element, include the XML.ATTRIBUTE qualifier after the field name. Note that quotation marks, less-than signs, and greater-than signs are removed from attributes as these characters will cause problems with XML parsers.

```
DOWNLOAD CUSTOMERS @ID \
      CUST.NAME CUST.PHONE CUST.PHONE.TYPE \
      FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers>
  <id>4082</id>
  <custname>Harris Hardware</custname>
  <phones>
    <custphone>800-555-1212</custphone>
    <custphonetype>voice</custphonetype>
  </phones>
  <phones>
    <custphone>555-400-1212</custphone>
    <custphonetype>fax</custphonetype>
  </phones>
</customers>
<customers>
  <id>3834</id>
  <custname>ACE Supply</custname>
  <phones>
    <custphone>404-555-8888</custphone>
    <custphonetype>voice</custphonetype>
  </phones>
</customers>
</download>
```

```
DOWNLOAD CUSTOMERS @ID XML.ATTRIBUTE \
      CUST.NAME CUST.PHONE CUST.PHONE.TYPE \
      FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers id=”4082">
```

```
  <custname>Harris Hardware</custname>
  <phones>
    <custphone>800-555-1212</custphone>
    <custphonetype>voice</custphonetype>
  </phones>
  <phones>
    <custphone>555-400-1212</custphone>
    <custphonetype>fax</custphonetype>
  </phones>
</customers>
<customers id=”3834">
  <custname>ACE Supply</custname>
  <phones>
    <custphone>404-555-8888</custphone>
  <custphonetype>voice</custphonetype>
  </phones>
</customers>
</download>
```

You can also use the prefix DEFAULT with XML.ATTRIBUTE.

```
DOWNLOAD CUSTOMERS @ID \
        CUST.NAME \
        CUST.BAL \
        DEFAULT XML.ATTRIBUTE \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers id=”4082" custname=”Harris Pie” custbal=”500.00" />
<customers id=”3834" custname=”ACE Supply” custbal=”48.56" />
</download>
```

XML.ELEMENT

By default, each field included in an XML DOWNLOAD is treated as an XML element (i.e., enclosed within its own start and end tags). This behavior can be overridden for an individual field by specifying XML.ATTRIBUTE or overridden for all fields by specifying DEFAULT XML.ATTRIBUTE. If you have specified DEFAULT XML.ATTRIBUTE but wish for a few fields to be treated as elements, you can use the qualifier XML.ELEMENT. (You can also specify DEFAULT XML.ELEMENT, but this is the normal behavior if you do not specify otherwise).

```
DOWNLOAD CUSTOMERS @ID \
        DEFAULT XML.ATTRIBUTE \
        CUST.NAME \
        CUST.BAL XML.ELEMENT \
        CUST.CITY \
        CUST.STATE \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<customers id=”4082" custname=”Harris Pies” custcity=”Tipton”
custstate=”IN”>
   <custbal>500.00</custbal>
</customers>
<customers id=”3834" custname=”ACE Supply” custcity=”Newburg”
custstate=”OR”>
   <custbal>48.56</custbal>
</customers>
</download>
```

XML.FILE.ATTRIBUTE

The XML.FILE.ATTRIBUTE option provides a mechanism to pass custom XML information (such as namespaces) to the output file via an attribute in the file element.

DOWNLOAD STUDENTS @ID STU.NAME \
    FORMAT XML \
    XML.FILE.ATTRIBUTE ‘xmlns:”http://mycompany.com/myxml”’
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students xmlns:”http://mycompany.com/myxml”>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students xmlns:”http://mycompany.com/myxml”>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

XML.FILE.NAME

The XML.FILE.NAME sets the name of the element used to identify each record in the primary file. If XML.FILE.NAME is not specified, the default is the name of the primary file.

DOWNLOAD STUDENTS @ID STU.NAME \
    FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
```

```
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME \
XML.FILE.NAME 'people' \
FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<people>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</people>
<people>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</people>
</download>
```

Specifying the keyword "NONE" or using a null string will cause the filename tags to be omitted. This produces output that is not well-formed XML, but is used on occasion by service bureaus.

DOWNLOAD STUDENTS @ID STU.NAME \
XML.FILE.NAME NONE \
FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</download>
```

XML.GROUP.ATTRIBUTE

The XML.GROUP.ATTRIBUTE provides a means of supplying literal text to appear within a tag created for an XML group.

DOWNLOAD STUDENTS @ID STU.NAME \
STU.TERM \
XML.GROUP.NAME "terms" \
XML.GROUP.ATTRIBUTE ' acadprogram="UG"' \
STU.TERM.GPA \
DEFAULT NUM.VALUES ALL \
FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
```

```
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
    <terms acadprogram=”UG”>
      <term>
        <stuterm>2001FA</stuterm>
        <stutermgpa>3.105</stutermgpa>
      </term>
      <term>
        <stuterm>2002SP</stuterm>
        <stutermgpa>2.943</stutermgpa>
      </term>
    </terms>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
    <terms acadprogram=”UG”>
      <term>
        <stuterm>2001SP</stuterm>
        <stutermgpa>4.000</stutermgpa>
      </term>
      <term>
        <stuterm>2001FA</stuterm>
        <stutermgpa>3.948</stutermgpa>
      </term>
      <term>
        <stuterm>2002FA</stuterm>
        <stutermgpa>3.875</stutermgpa>
      </term>
    </terms>
</students>
</download>
```

XML.GROUP.NAME

The XML.GROUP.NAME creates a level of tags surrounding a set of associated fields, as shown in the second example below. The XML.GROUP.NAME must be specified for at least one of the fields in the association, but may be specified for all of the fields.

```
DOWNLOAD STUDENTS @ID STU.NAME \
        STU.TERM \
        STU.TERM.GPA \
        DEFAULT NUM.VALUES ALL \
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
    <term>
      <stuterm>2001FA</stuterm>
```

```
      <stutermgpa>3.105</stutermgpa>
    </term>
    <term>
      <stuterm>2002SP</stuterm>
      <stutermgpa>2.943</stutermgpa>
    </term>
  </students>
  <students>
    <id>3081234</id>
    <stuname>Peters, Kristi Leigh</stuname>
      <term>
        <stuterm>2001SP</stuterm>
        <stutermgpa>4.000</stutermgpa>
      </term>
      <term>
        <stuterm>2001FA</stuterm>
        <stutermgpa>3.948</stutermgpa>
      </term>
      <term>
        <stuterm>2002FA</stuterm>
        <stutermgpa>3.875</stutermgpa>
      </term>
  </students>
  </download>
```

```
DOWNLOAD STUDENTS @ID STU.NAME \
      STU.TERM XML.GROUP.NAME “terms” \
      STU.TERM.GPA \
      DEFAULT NUM.VALUES ALL \
      FORMAT XML
```

will produce a file like this:

```
      <?xml version="1.0"?>
      <download>
      <students>
        <id>1040400</id>
        <stuname>Acker, Jacob P.</stuname>
          <terms>
            <term>
              <stuterm>2001FA</stuterm>
              <stutermgpa>3.105</stutermgpa>
            </term>
            <term>
              <stuterm>2002SP</stuterm>
              <stutermgpa>2.943</stutermgpa>
            </term>
          </terms>
      </students>
      <students>
        <id>3081234</id>
        <stuname>Peters, Kristi Leigh</stuname>
          <terms>
            <term>
              <stuterm>2001SP</stuterm>
              <stutermgpa>4.000</stutermgpa>
            </term>
            <term>
```

```
        <stuterm>2001FA</stuterm>
        <stutermgpa>3.948</stutermgpa>
      </term>
      <term>
        <stuterm>2002FA</stuterm>
        <stutermgpa>3.875</stutermgpa>
      </term>
    </terms>
</students>
</download>
```

XML.NAME

The XML.NAME option is used when you want the XML output to use a different name for an element than one derived from the actual dictionary name for the field. Note that DOWNLOAD strips disallowed characters from all XML element names. Output names will always start with a letter or underscore. Subsequent characters may be letters, numbers, underscores, or periods. Derived names are converted to lower case (unless XML.UPCASE was specified). Names entered via the XML.NAME qualifier retain the case entered on the command line. DOWNLOAD removes all periods from XML element names unless XML.ALLOW.PERIODS is specified on the command line.

DOWNLOAD STUDENTS @ID STU.NAME \
    FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME XML.NAME ‘Person’ \
    FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <Person>Acker, Jacob P.</Person>
</students>
<students>
  <id>3081234</id>
  <Person>Peters, Kristi Leigh</Person>
</students>
```

```
</download>
```

XML.ROOT.ATTRIBUTE

The XML.ROOT.ATTRIBUTE option provides a mechanism to pass custom XML information (such as namespaces) to the output file via an attribute in the root element.

```
DOWNLOAD STUDENTS @ID STU.NAME \
        FORMAT XML \
        XML.ROOT.ATTRIBUTE ‘xmlns:”http://mycompany.com/myxml”’
```

will produce a file like this:

```
<?xml version="1.0"?>
<download xmlns:”http://mycompany.com/myxml”>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

XML.ROOT.NAME

The XML.ROOT.NAME sets the name of the root element. If XML.ROOT.NAME is not specified, the default ‘download’ is used.

```
DOWNLOAD STUDENTS @ID STU.NAME XML.NAME ‘person \
        XML.ROOT.NAME ‘freshmen’
        FORMAT XML
```

will produce a file like this:

```
<?xml version="1.0"?>
<freshmen>
<students>
  <id>1040400</id>
  <person>Acker, Jacob P.</person>
</students>
<students>
  <id>3081234</id>
  <person>Peters, Kristi Leigh</person>
</students>
</freshmen>
```

Specifying the keyword “NONE” or using a null string will cause the root tags to be omitted.

```
DOWNLOAD STUDENTS @ID STU.NAME XML.NAME ‘person \
        XML.ROOT.NAME NONE \
        XML.VERSION NONE \
```

FORMAT XML

will produce a file like this:

```
<students>
  <id>1040400</id>
  <person>Acker, Jacob P.</person>
</students>
<students>
  <id>3081234</id>
  <person>Peters, Kristi Leigh</person>
</students>
```

XML.SUBASSOC.NAME

The XML.SUBASSOC.NAME option is used when you want subvalues within a multi-valued association to be treated as their own association.

For example, suppose that a file contains one record for each clothing item offered by a manufacturer. For each clothing item, multivalued field COLOR lists the available colors. For each particular color, field SIZE lists the sizes available. Here are some sample records from the file (where "}" represents a value mark and "|" represents a subvalue mark):

```
134-BC
001: Sweatshirt
002: Red}Blue
003: S|M|L}M|L|XL|XXL

406-RM
001: Super-hi t-shirt
002: Black}Tan}White
003: L|XL}M}XS|S|M|L|XL|XXL|XXXL
```

```
DOWNLOAD CLOTHING @ID XML.ATTRIBUTE \
    DESCRIPTION \
    COLOR \
    SIZE \
    DEFAULT NUM.VALUES ALL \
    DEFAULT NUM.SUBVALUES ALL \
    FORMAT XML
```

will produce a file like this (assuming that the dictionary entry for COLOR lists 'INVENTORY' as the association name):

```
<?xml version="1.0"?>
<download>
<clothing id=”134-BC”>
  <description>Sweatshirt</description>
  <inventory>
    <color>Red</color>
    <size>S</size>
    <size>M</size>
```

```
      <size>L</size>
    </inventory>
    <inventory>
      <color>Blue</color>
      <size>M</size>
      <size>L</size>
      <size>XL</size>
      <size>XXL</size>
    </inventory>
  </clothing>
  <clothing id=”406-RM”>
    <description>Super-hi t-shirt</description>
    <inventory>
      <color>Black</color>
      <size>L</size>
      <size>XL</size>
    </inventory>
    <inventory>
      <color>Tan</color>
      <size>M</size>
    </inventory>
    <inventory>
      <color>White</color>
      <size>S</size>
      <size>M</size>
      <size>L</size>
      <size>XL</size>
      <size>XXL</size>
      <size>XXXL</size>
    </inventory>
  </clothing>
  </download>
```

```
DOWNLOAD CLOTHING @ID XML.ATTRIBUTE \
        DESCRIPTION \
        COLOR \
        SIZE XML.SUBASSOC.NAME 'sizes' \
        DEFAULT NUM.VALUES ALL \
        DEFAULT NUM.SUBVALUES ALL \
        FORMAT XML
```

will change how the SIZE field is handled, producing a file like this:

```
        <?xml version="1.0"?>
        <download>
        <clothing id=”134-BC”
          <description>Sweatshirt</description>
          <inventory>
            <color>Red</color>
            <sizes>
              <size>S</size>
              <size>M</size>
              <size>L</size>
            </sizes>
          </inventory>
```

```
  <inventory>
    <color>Blue</color>
    <sizes>
      <size>M</size>
      <size>L</size>
      <size>XL</size>
      <size>XXL</size>
    </sizes>
  </inventory>
</clothing>
<clothing id=”406-RM”>
  <description>Super-hi t-shirt</description>
  <inventory>
    <color>Black</color>
    <sizes>
      <size>L</size>
      <size>XL</size>
    </sizes>
  </inventory>
  <inventory>
    <color>Tan</color>
    <sizes>
      <size>M</size>
    </sizes>
  </inventory>
  <inventory>
    <color>White</color>
    <sizes>
      <size>S</size>
      <size>M</size>
      <size>L</size>
      <size>XL</size>
      <size>XXL</size>
      <size>XXXL</size>
    </sizes>
  </inventory>
</clothing>
</download>
```

XML.UPCASE

By default, all XML element tags are generated in lower case. Use this option to generate the tags in uppercase. If you wish to have element tags with mixed case, use the XML.NAME qualifier described above.

DOWNLOAD STUDENTS @ID STU.NAME \
    FORMAT XML

will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
```

```
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME \
XML.UPCASE \
FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<DOWNLOAD>
<STUDENTS>
  <ID>1040400</ID>
  <STUNAME>Acker, Jacob P.</STUNAME>
</STUDENTS>
<STUDENTS>
  <ID>3081234</ID>
  <STUNAME>Peters, Kristi Leigh</STUNAME>
</STUDENTS>
</DOWNLOAD>
```

XML.VERSION

By default, DOWNLOAD produces a standard version tag at the beginning of the XML document being produced. The XML.VERSION option can be used to change this tag or (by specifying a null tag or the keyword NONE) can be used to omit the version tag.

DOWNLOAD STUDENTS @ID STU.NAME \
FORMAT XML
will produce a file like this:

```
<?xml version="1.0"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME \
FORMAT XML XML.VERSION '<?xml localver="3.2"?>'
will produce a file like this:

```
<?xml localver="3.2"?>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
```

```
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME \
FORMAT XML XML.VERSION ‘’
will produce a file like this:

```
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

DOWNLOAD STUDENTS @ID STU.NAME \
FORMAT XML XML.VERSION NONE
will produce a file like this:

```
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

The XML.VERSION option can be used to place multiple XML tags at the beginning of the output file.

DOWNLOAD STUDENTS @ID STU.NAME \
FORMAT XML \
XML.VERSION ‘<?xml VERSION=”1.0"?><mytaghere abc=”demo”>’
will produce a file like this:

```
<?xml version="1.0"?><mytaghere abc=”demo”>
<download>
<students>
  <id>1040400</id>
  <stuname>Acker, Jacob P.</stuname>
</students>
<students>
  <id>3081234</id>
  <stuname>Peters, Kristi Leigh</stuname>
</students>
</download>
```

Version History

```
* Stamped: p41 rotmand, /usr/local/download, user #1784, 21 Jun 07, 11:04AM.
* Version 7.21
*   Bug fix (forced compilation of uncompiled virtual field within a
*       secondary file did not use the correct field name)
*   Bug fix (online documentation did not reference new XML
*       capabilities added in version 7.20)
*
*
*
* Stamped: p3 rotmand, /usr/local/download, user #3461, 25 Apr 07, 07:48AM.
* Version 7.20
*   Add XML.GROUP.ATTRIBUTE qualifier.
*   Add XML.ELEMENT qualifier.
*   Add DEFAULT XML.ELEMENT option.
*   Add DEFAULT XML.ATTRIBUTE option.
*   Bug fix (APPEND in format XML should remove closing root tag)
*   Bug fix (documentation indicated that XML.FILE.NAME NONE will
*      function the same as XML.FILE.NAME "", but this was not happening)
*   Corrected documentation on illegal characters in XML values (the
*      documentation indicated that these characters would be removed,
*      but they are actually replaced by acceptable expressions)
*
*
*
* Stamped: p8 rotmand, /usr/local/download, user #1502, 08 Sep 05, 10:59AM.
* Version 7.13
*   Add @COUNTER (running counter) as a pre-defined AT variable
*   Bug fix (DEFAULT SUBVALUE.SEPARATOR did not work)
*   Bug fix (Using @RECORD required TYPE to be in DICT.DICT).
*   Bug fix (for XML output, a null value in an association caused
*      the rest of the data values to "slide up")
*   Bug fix (output file specified in VOC via @HOME did not work)
*   Bug fix (EVAL expression with embedded quotation marks did not work)
*
*
*
* Stamped: p1 rotmand, /usr/local/download, user #2041, 31 Mar 05, 08:13AM.
* Version 7.12
*   Allow "NONE" keyword for XML.ROOT.NAME and XML.VERSION.
*   Bug fix (FIXED format with DEFAULT MV.ORIENTATION VERTICAL produced
*      error message during consistency check)
*   Bug fix (HEADING FIELD.NAMES did not honor VALUE.SEPARATOR)
*
*
*
* Stamped: p1 rotmand, /usr/local/download, user #1043, 17 Aug 04, 07:41AM.
* Version 7.11
*   Enhance error message when record keys are not found.
*   Bug fix (COMMA.CHAR did not accept multiple characters)
*   Bug fix (QUOTE.CHAR did not accept multiple characters)
*   Bug fix (DBF format header incorrect)
*   Bug fix (NUM.VALUES clause on a field with null association became
*       the default for all fields with null association)
*
*
```

```
*
* Stamped: p2 rotmand, /usr/local/download, user #2463, 17 Feb 04, 10:49AM.
* Version 7.10
*   Added qualifier XML.ALLOW.CHARACTERS.
*   Added DEFAULT XML.ALLOW.CHARACTERS.
*   Allow null association names in XML.
*   Added HTML.HEAD.
*   Return HTML tags in lower case for compatibility with XHTML.
*   Bug fix (allow NUM.SUBVALUES ALL for FORMAT HTML)
*   Bug fix (abort execution if @RECORD returns no entries)
*   Bug fix (honor EOR.CHAR in XML heading lines)
*   Bug fix (use "head" instead of "heading" in HTML format)
*   Bug fix (use "th" for table headings instead of "td")
*   Renamed several external subroutines:
*      EXPAND.ITEMS          to DL.EXPAND.ITEMS
*      FLIP.8TH.BIT          to DL.FLIP.8TH.BIT
*      GET.KEYWORD.VALUE     to DL.GET.KEYWORD.VALUE
*      OPEN.FILE             to DL .OPEN.FILE
*      PARSE.COMMAND.LINE    to DL.PARSE.COMMAND.LINE
*      PROMPT.ANS            to DL.PROMPT.ANS
*      PROMPT.STACK          to DL.PROMPT.STACK
*      VIEW.FILE             to DL.VIEW.FILE
*      VIEW.SEQ.FILE         to DL.VIEW.SEQ.FILE
*   Added options for cataloging globally or locally during instalation.
*
*
*
* Stamped: p5 rotmand, /usr/local/download, user #1208, 26 Sep 03, 08:18AM.
* Version 7.02
*   Bug fix (compilation-time syntax error on DLPROCESS in Universe)
*
*
*
* Stamped: p3 rotmand, /usr/local/download, user #4151, 15 Sep 03, 11:30AM.
* Version 7.01
*    Escape XML output data for characters:  < > & ' "
*
*
*
* Stamped: p4 rotmand, /usr/local/download, user #4051, 05 Sep 03, 03:15PM.
* Version 7.00
*    Added EVAL functionality for output of expressions.
*    Incorporated EVAL and data-field functionality into various
*       HTML and XML options.
*    Added COL.HDG qualifier.
*    Added XML.GROUP.NAME qualifier.
*    Enabled XML.ATTRIBUTE within associated fields.
*    Default XML.ASSOC.NAME TO 'fieldnameASSOC' if no association
*       is listed in the dictionary and XML.ASSOC.NAME is not
*       specified on the command line.
*    Modified the APPEND logic to allow this function to work in
*       Universe.
*    Enabled APPEND for HTML and XML output.
*    Added support for NUM.VALUES when specifying the key
*       to a SECONDARY.FILE (especially useful with XML).
*    Added '@RECORD' as a method for specifying output fields.
*    Consolidated redundant code in DOWNLOAD.PARSE.
*    Added 'DOWNLOAD HELP MAX' to show the various maximum
*       values for literals, fields, etc.
```

```
*    Systematized DEBUG.LEVEL arguments (see documentation file for
*       details).
*    Conformed all variables in I_DOWNLOAD_OUT_REC_COMMON to a
*       consistent naming scheme.
*    Bug fix (recognize VOC entries with type LF and LD)
*    Bug fix (recognize DICT records of type V)
*    Bug fix (comma-quote fields with a format specification shorter
*       than the actual data length would produce unpredictable
*       results)
*    Bug fix (exploded list was not being honored by secondary files)
*
*
*
* Stamped: p6 rotmand, /usr/local/download, user #13226, 19 May 03, 02:17PM
* Version 6.02
*    Added XML.VERSION option (to let users specify the layout of
*       the xml version tag, or omit it altogether).
*    Unidata/Universe now set a default zero character in their
*       configuration files.  The Unidata version of DOWNLOAD now
*       fetches this value by a call to GETENV.  Universe users may
*       need to set this value by hand in routine DBMS.UNIVERSE.SET.UP
*       in the main DOWNLOAD program.  (Typically, the value will
*       be 0, 128, or 131.)  This setting only affects WP50 and WP51
*       formatted output.
*    Bug fix (WP50/WP51 headings were not complete if NUM.VALUES
*       was used for the detail and HEADING FIELD.NAMES was used
*       to specify the headings.)
*
*
*
* Stamped: p7 rotmand, /usr/local/download, user #13273, 29 Jan 03, 04:46PM
* Version 6.01
*    Bug fix (consecutive XML.ASSOC.NAME qualifiers not honored)
*
*
*
* Stamped: p1 rotmand, /usr/local/download, user #184, 04 Nov 02, 06:56AM.
* Version 6.00
*    Added FORMAT XML
*    Added FORMAT TAB (equivalent to FORMAT COMMA COMMAR.CHAR ^9)
*    Added VALUE.SEPARATOR and SUBVALUE.SEPARATOR
*    Increased DLMAX.LITERAL.VALUES from 150 to 200 (literal values)
*    Increased DLMAX.SUBR.NAMES from 100 to 150 (subroutine names)
*    Increased DLMAX.DICT.RECS from 250 to 300 (fields to DOWNLOAD)
*    Added FIELD.LABELS clause to HEADING option
*    Added LIT as a synonym for LITERAL
*    Added CNV as a synonym for CONV
*    Allowed use of environment variables in VOC entry for output file
*    Allowed use of NUM.SUBVALUES with exploded select lists
*    Updated Unidata installation (force $BASICTYPE "U")
*    Created Universe-specific installation (Unix)
*    Created Windows-specific installation (Universe and Unidata)
*    Created test routines to verify a successful installation
*    Bug fix (specifying HEADING twice caused infinite loop)
*    Bug fix (MIN minimum value on footing line was always null)
*
*
*
* Stamped: p3 rotmand, /usr/local/download, user #1542, 01 Feb 01, 08:04AM.
```

```
* Version 5.10
*    Enable "@" variables in file names
*    Add WRITE.INTERVAL option to pause between writing groups of
*       records (necessary on some systems communicating via NFS
*       or Samba)
*    Fix bug in WP51 format where null fields were suppressed if they
*       were at the end of the output record.
*    Fix bug in field qualifier MAX (used in break lines).
*
*
*
* Stamped: pe rotmand, /usr/local/download, user #12980, 06 Jul 00, 08:04AM.
* Version 5.00
*    Added support for HTML files
*    Added APPEND option
*    Added REMOVE.PUNCTUATION option
*    Corrected typographical errors in the documentation
*    Produce PDF version of the documentation
*
*
*
* Stamped: pt rotmand, /usr/local/download, user #16160, 14 Feb 97, 03:35PM.
* Version 4.0
*    Added support for DBF files
*    Changed headings used in FIXED and DBF formats to follow same
*       lengths as detail records unless over-ridden on the command line
*    Added FIELD.NAMES clause to HEADING option
*    Added support for "@" variables such as @DATE, @SYSTEM.RETURN.CODE
*    Added NO.PAGE option to turn off screen pauses on progress meter
*       and screen-based output
*    Added ability to write to a subdirectory without creating a VOC
*       pointer (see documentation for FILE option).
*    Various bug fixes, including:
*       Default @ID when using secondary file
*       More-complete DIF output (required by Excel)
*    Moved version history to separate file
*    Modified on-line help
*    Created WordPerfect documentation
*
*
*
* Version 3.1
* Stamped: pt rotmand, /usr/local/download, user #2968, 05 Jul 95, 01:16PM.
*    Added BREAK.SUP option.
*
*
*
* Version 3.0
* Stamped: pt sjoquist, /usr/local/download, user #3835, 01 Nov 94, 01:38PM.
*    Added file relations (avoid building multiple, complicated i-descriptors)
*
*
*
* Version 2.2
* Last updated by TEST (SJOQUISTD) at 16:41:41 on 02/11/1994.
*    Modified COMMA format (numeric values do not have quotes)
*    Added QUOTE format (functions like COMMA used to)
*    Created DIF format
```

```
*
*
*
* Version 2.1, miscellaneous changes
* Last updated by LIVE (SJOQUISTD) at 13:42:17 on 10/27/1993.
*    Set up new distributable copy (version 2.1)
* Last updated by LIVE (ROTMAND) at 12:26:26 on 09/01/1993.
*    Add 'T' and 'D' option to LITERAL fields.
* Last updated by LIVE (SJOQUIST) at 09:19:31 on 09/09/1992.
*    Add COMMA.CHAR option.
* Last updated by LIVE (ROTMAN) at 17:19:31 on 08/12/1992.
*    Add QUOTE.CHAR option.
* Last updated by LIVE (SJOQUIST) at 08:18:42 on 08/06/1991.
*    Rename DOWNLOAD.LOAD to DOWNLOAD.PARSE
*    Split DOWNLOAD.PROCESS into DOWNLOAD.LOAD & DOWNLOAD.PROCESS
*
*
*
* Version 2.0, HEADING/FOOTING/BREAK.ON
* Last updated by LIVE (SJOQUIST) at 09:34:44 on 07/26/1991.
*    Split into INIT/LOAD/PROCESS subroutines
*
*
*
* Version 1.1, BEGIN ... END keywords with prompting using PROMPT.STACK
* Last updated by LIVE (SJOQUIST) at 08:28:29 on 07/26/1991.
* Last updated by LIVE (SJOQUIST) at 16:20:31 on 04/10/1991.
*
*
```

Universe Considerations

Installation

1. The DLSOURCE directory (referenced as DOWNLOAD.SOURCE in the installation instructions) may be created as either a Type 1 or a Type 19 file.

2. While not necessarily visible to the person running DOWNLOAD, internal names for subroutines and other files have been shorted to 14 characters or less to fit a somewhat standard Universe environment.

Functionality

The following DOWNLOAD functions have some limitations in the Universe version:

1. Specifying a list number other than zero for the active select list does not work if the DOWNLOAD includes one or more exploded fields.

2. Use of environment variables in file names (i.e., “@” in the VOC entry for the file name) is not supported.