62 Special exports and imports

Written by Dave Terry Published Date March 3, 1988

Importing and Exporting Fixed Length Records and Comma Delimited Records

4th Dimension has a variety of built-in export/import capabilities. It can export/import in DIF, SYLK and TEXT. In the TEXT format 4th Dimension can delimit fields and records with any *single* character you choose.
4th Dimension even has the ability to remap ASCII characters during exporting. Even with all of these capabilities, users find themselves in situations that require writing special procedures to do importing. Two of these situations are described here along with the procedures to solve the problems.

Exporting Fixed Length Records

In many mainframe databases fields are kept in fixed length format. That is, every field is padded with spaces to the maximum size of the field. Exporting in this format is fairly easy. Using **SEND PACKET** we can build a text file of fixed length records. The method used to pad the fields is the same as the one used for arrays (See Tech Note #36). The argument to the **SEND PACKET** command can be as long as needed. In this example the export was trimmed to only 3 fields.

```
ALL RECORDS([File1])     `In this case all records are sent out.  This could be replaced by a search.
SET CHANNEL(12;"")`Lets the user specify the document name to be created.

APPLY TO SELECTION([File1];SEND PACKET([File1]Field1+(" "*(10-Length([File1]Field1)))
      +[File1]Field2+(" "*(20-Length([File1]Field2)))+[File1]Field3+ (" "*(20-Length([File1]Field3)))))

      `The APPLY TO SELECTION command uses the SEND PACKET command to
      `send out a packet for every record composed of field information with padded
      `spaces.

SET CHANNEL(11)    `Closes then document.
```

Importing Fixed Length Records

This procedure imports fixed length records. The global function *strip* removes any trailing spaces. Because fields might have embedded spaces the *strip* procedure starts from the end and works forward, a slow process. This procedure is specific to [File1] and would need to be changed to work elsewhere. This procedure imports the text file created in the example above. The number and size of each field would need to be known before the import could take place.

```
Flush:=0                                `Setting the Flush variable to 0 speeds up the writing of
                                        `records.
SET CHANNEL(13;"")                      `Lets the user specify the document to use in the import.

While (OK=1)                            `While records still exist to be read continue the loop.

    RECEIVE PACKET(x;50)                `This command will set OK:=0 when it tries to read beyond the
                                        `last record.  Record size was determined by the export
                                        `routine above.

    If ((Length(x)=50)&(OK=1))          `Determines if a record should be created.
       CREATE RECORD
       [File1]Field1:=strip (Substring(x;1;10))     `The "strip" function is listed below.
       [File1]Field2:=strip (Substring(x;11;30))    `It takes a string and
       [File1]Field3:=strip (Substring(x;31;50))
       SAVE RECORD                      `returns a string without any trailing spaces.  The
                                        `size of the fields would be determined by the
                                        `export routine.
    End if
End while

SET CHANNEL(11)                         `Closes the document.
Flush:=1 `Sets Flush back to its default status.
```

where the global procedure *strip* is

```
$i:=Length($1)                          `Puts the length of the passed string into $i
While ((Substring($1;$i;1)=" ")&($i#0))  `Loops until either a non-space is found or $i =0
    $i:=$i-1
End while
$0:=Substring($1;1;$i)                  `Passes back the passed string minus the spaces.
```

Exporting Comma Delimited Records

This procedure exports records as a comma delimited text file. All strings have quotes around them while numbers do not. Each field is delimited by commas but strings may have commas embedded within them. Records are delimited by two characters, a carriage return and then line feed (**Char**(13)+**Char**(10)). In this example no numerical fields are exported. This procedure is specific to [File1] and would need to be altered to be used elsewhere. Once again, only three fields are exported.

```
$Q:=Char(34)      `Quote
$QCQ:=$Q+","+$Q                          `Quote+Comma+Quote
$EOL:=Char(13)+Char(10)                  `Carriage Return-Line Feed
ALL RECORDS([File1])                     `Could be replaced by a search.
SET CHANNEL(12;"")                       `Lets the user specify the document name to be created.

APPLY TO SELECTION([File1];SEND PACKET($Q+[File1]Field1+$QCQ+[File1]Field2+
                                       $QCQ+[File1]Field3+$EOL))

                                         `The APPLY TO SELECTION command uses the SEND
                                         `PACKET command to send out a packet for every record
                                         `composed of field information separated by quotes and
                                         `commas.

SET CHANNEL(11)
```

Importing Comma Delimited Records

This procedure imports comma delimited records. It reads a whole record, parses it into different variables, and then creates a record from the variables. With this procedure the record size is limited to 32,000 characters. A portion of this procedure is specific to [File1] and would need to be changed to work elsewhere. In this example the the number of fields and their order would already have to be known.

```
$Q:=Char(34)                          `Quote
$Qat:=Char(34)+"@"                    `Quote-@ sign
Flush:=0                              `Speed up writing records.
SET CHANNEL(13;"")                    `Open the document.

While (OK=1)                          `While their are still records to recieve.
   $i:=0
   RECEIVE PACKET(x;Char(10))         `Read characters into "x" up to a Form Feed (Char(10)).
                                      `Will set OK:=0 when it tries to read past the last record.

   x:=Substring(x;1;Length(x)-1)+","  `Strip off the carriage return and add a comma.  This is
                                      `needed because the last field will have no trailing
                                      `comma, which the algorithm uses to determine the end
                                      `of a field.

   While ((OK=1)&(Length(x)>1))       `Parse record into variables.
      $i:=$i+1                        `To keep track of which variable to create.

      If(x=$Qat)                      `Starts with a quote.  The @ sign may be used in
                                      `string comparisons.  See Tech Note # 22.

         $PQ:=Position($Q;Substring(x;2;Length(x)-1))     `Look for the next quote
         var{$i}:=Substring(x;2;$PQ-1)                    `Put  this field into a variable.
         x:=Substring(x;$PQ+3;Length(x)-$PQ-2)            `Strip field from "x".

      Else                            `Starts without a quote.
         $PC:=Position(",";x)         `Find next comma.
         var{$i}:=Substring(x;1;$PC-1)    `Put  this field into a variable.
         x:=Substring(x;$PC+1;Length(x)-$PC)              `Strip field from "x".
      End if
End while

   If ($i>0)
                                      `This section of code is specific to [File1] and would
      CREATE RECORD([File1])          `need to be changed to work in a different file.  The
      [File1]Field1:=var1             `variables var1..var3 were assigned in the while
      [File1]Field2:=var2             `loop above.
      [File1]Field3:=var3
      SAVE RECORD([File1])
   End if
```

```
End while
SET CHANNEL(11)
Flush:=1
```