Type, Point and Click – A Practical Guide to SAS Enterprise Guide® Prompts

Patricia Hettinger, SAS Professional, Oakbrook Terrace, IL

ABSTRACT

Your SAS Enterprise Guide project is set up. Processes have been created, linking code in the desired order. Now if only you didn't have to keep changing your code every time your input file name changed or your date range or ….

This paper gives an overview of the prompts available in SAS Enterprise 5.1 and ways to automate your project. We will also touch on the general nature of macro variables, and the various quoting and unquoting functions. We will also explore the many additional prompt features available in stored processes.

INTRODUCTION

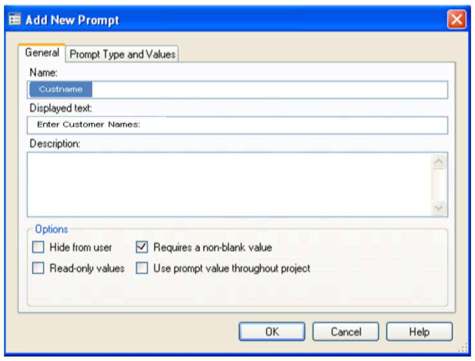
SAS® Enterprise Guide® is a fantastic tool for visualizing your data and processes. If you are already linking your programs together, you have done the first step of true automation. Now the next step is to make your process truly reusable by letting whoever is running it choose various options such as file names, dates, and even passwords. At the end of this paper, we will set up a practical demonstration, interfacing to a ConnectDirect script (formerly know as NDM) that will let us transfer any file visible to your SAS Enterprise Guide UNIX session to a select mainframe. Although developed for SAS Enterprise Guide 5.1, most of this information will still be valid for earlier releases.

THE NATURE OF SAS ENTERPRISE GUIDE PROMPTS

All prompts work by passing parameters to a SAS program in batch mode. These are picked up by SAS as macro variables. Since all SAS macro variables are character in nature, any prompts set by SAS Enterprise Guide are character as well. However SAS Prompt Manager has code manipulation behind the scenes that will let us create variables to name file directories, files, numbers and dates. Versions 4.3 and 5.1 even act like Windows Explorer, allowing you to point and click to file names as in our NDM example later.

Several of the prompt options will allow you to select more than one value at run time. These include text, numeric and date prompts. The old Parameters Manager in SAS Enterprise Guide 4.0 and 4.1 would resolve these multiple values as value1-delimiter'-value2-delimiter etc. Version 4.2, 4.3 and 5.1 handle this differently, setting each value into its own variable, plus a variable keeping track of the count. For example, suppose you wanted a prompt that would let you enter a number of last names as a customer search tool. Let's name the macro variable custname.

We'll create the prompt and enter the text we want to see when the prompt is in use (figure 1). We will require at least one value, set here and on the next screen (Figure 3). We will not use the prompt value throughout the project for reasons discussed later. Figure 2 shows how we can set the minimum number of values allowed to one and the maximum to five. The length allowed for each value will range from two to twenty. Note that we also had to specify multiple values in the 'Number of values' drop box:



Name: Custname

Displayed text: Enter Customer Names:

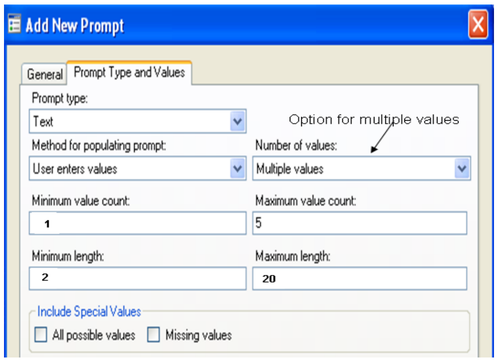
Description:

Options

[ ] Hide from user [✓ Requires a non-blank value]

[ ] Read-only values [ ] Use prompt value throughout project

Figure 1: Naming the prompt



Prompt type: Text

Method for populating prompt: User enters values

Number of values: Multiple values

Minimum value count: 1

Maximum value count: 5

Minimum length: 2

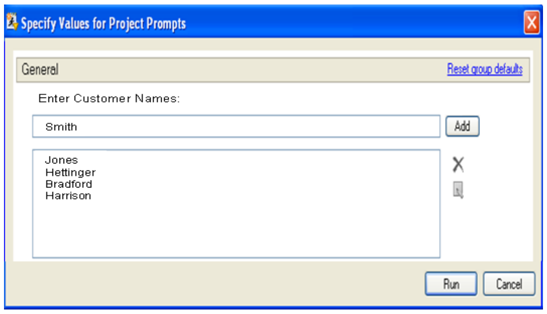
Maximum length: 20

Include Special Values

[ ] All possible values [ ] Missing values

Figure 2: Setting the value and lengths allowed

At run-time:



Smith

Jones
Hettinger
Bradford
Harrison

Figure 3: Run-time view for custname

After the last value is entered, Smith, Prompt Manager creates eight macro variables. These are:

custname – Harrison	custname1 – Harrison	custname2 – Bradford
custname3 – Hettinger	custname4 – Jones	custname5 – Smith
custname0 – 5	custname_count – 5	

If only one value was entered, just custname and custname_count would exist. Referring to custname without a suffix gives us the first value entered, Harrison. Custname0 and custname_count contain the number of values entered from the screen. The other variables, custname1 through custname5 may be thought of as an array, referring to each customer name by reference. Processing these can be tricky. The appendix of this paper has one example using a macro do-loop and the various masking/unmasking functions. If we had defined this variable as a text ***range*** instead, we would have created two variables, custname_MIN and custname_MAX.

Why didn't we want to use these values throughout the project when we set up this prompt (figure1)? If we had, all of the values in the array would have remained except where they were changed. This is despite actually deleting the values we didn't want anymore. So if we decided we simply wanted three values the next time we ran, like 'Patel', 'Whang' and 'Butkovich', the custname_count variable would have still been five and we would have ran with 'Jones' and 'Smith' again. Not using them throughout the project gives us a clean slate each time

Selecting multiple dates is even more complicated. Figure 4 shows what a date prompt named Pickdates would look like at runtime. You have your choice of picking today's date, some increment backward or forward or a specific date.

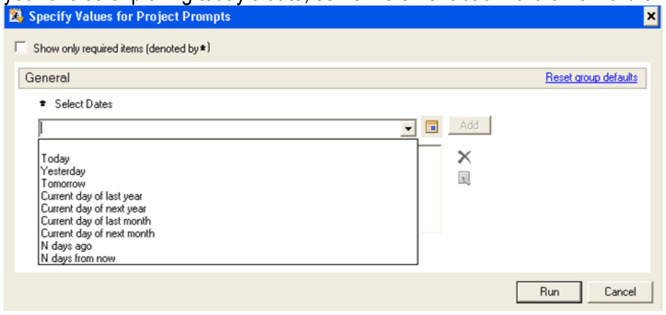


Figure 4: Selecting multiple dates – the choices

If we chose 'Today' and 'Yesterday' when running this on July 7, 2012 and July 20, 2012, even more variables would be created than in the previous example:

PICKDATES_REL3 - D-1D	PICKDATES_REL2 - D0D	PICKDATES_COUNT - 3
PICKDATES0 – 3	PICKDATES - 20Jul2012	
PICKDATES1 - 20Jul2012	PICKDATES2 - 07Jul2012	PICKDATES3 - 06Jul2012
PICKDATES_LABEL - July 20, 2012	PICKDATES_LABEL1 - July 20, 2012	
PICKDATES_LABEL2 Today	PICKDATES_LABEL3 Yesterday	

We've seen the array variables before but what are the REL and LABEL variables? The REL variables refer to the displacement from the run date and the labels may be used in reports and analyses. DOD refers to the current date and D-1D one day prior. You will probably never use the REL variables but the labels would be very useful in a report.

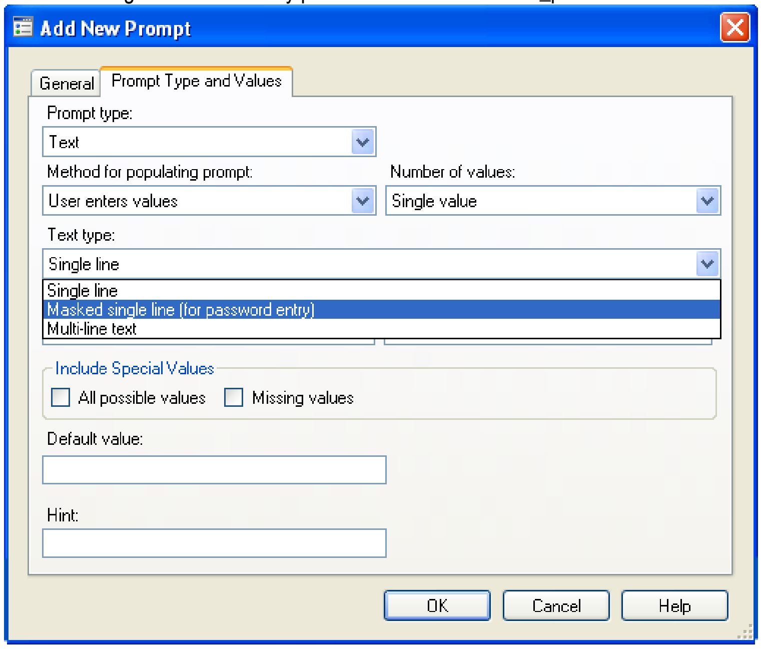
Note that none of the date variables are enclosed with single quotes and a 'd' at the end. 20Jul2012 should be '20Jul2012'd. You will need to put it in quotes and append a 'd' to the end. Sounds simple, doesn't it? PICKDATES1 right off the prompt looks like 20Jul2012. This is not a valid date literal so you decide to enclose in quotes and add the 'd' suffix to get '20July2012'd. However, just setting a variable to '&pickdates1'd results in just that: '&pickedates1'd. You can use the %bquote function to resolve the &pickdates1 and add the other characters as %bquote('&pickdates1'd). The log will show this as '20Jul2012'd but when you try to use it an expression, you will get an error. Unmask the resulting value so that it can be used as a literal in the data step as %unquote(%bquote('&date1'd)).

Another way to get the proper quotes for your dates and text variables is to set them in a data step. You could use a null data set as output as in the example below. Note the single quote is enclosed in double quotes.

```
        DATA _null_;
        pickdates1 = "'"||"&pickdates1"||"'d";
        CALL SYMPUT('pickdates1',pickdates1);
        run;
```

ABOUT PASSWORDS

SAS Enterprise Guide version 5.1 offers considerable improvement for protecting passwords obtained with a prompt. As figure 5 below demonstrates, you can specify a masked value when selecting how the run-time prompt will look. The log will show NDM_PASSWORD = XXXXXX. You will want to keep symbolgen turned off though, as well as avoiding %put statements as either will still show the unencrypted value in the log. The suffix of any password variable must be _password.



Add New Prompt

Prompt type: Text

Method for populating prompt: User enters values

Number of values: Single value

Text type: Single line

Include Special Values: All possible values, Missing values

Default value:

Hint:

Figure 5: Masking a password

When you use the suffix, it will ask how you want the password encrypted. If you are going from one operating system to another, you may find no encrypted value works. In fact, the file transfer application whose build is discussed later in this paper cannot use an encrypted password.

For those using older versions of SAS Enterprise like 4.2 and 4.3, you will find SAS Enterprise still showing the unencrypted value first thing in the log. To avoid this, you will have do two things. One is to uncheck the 'Use prompt throughout project' box when adding the password prompt. The other has to do with your SAS Enterprise Guide settings. Uncheck the 'show generated wrapper code in SAS log' which is one of the Results General options. This may be found by selecting 'Options' under the Tools menu bar. If you do not uncheck this, everything set by a prompt will be clearly visible in the log, including your password. There is nothing you can do code-wise to suppress this unless you run a separate program first that turns off the log altogether. This would be one line of code: `Options nosource nosymbolgen;`

A PRACTICAL DEMONSTRATION – SETTING UP A FILE TRANSFER PROGRAM

You might have wondered why so many SAS job requirements specify being able to write UNIX shell scripts. This is because, believe it or not, there are some things involving the system environment too complicated for SAS, as in this example. We will interface to a Korn shell calling ConnectDirect aka NDM. The shell sends a file from a UNIX box to a mainframe selected from a list. See the appendix for the full code. SAS will be used to set up a GUI interface and to do some editing on the front end. Keep in mind that mainframe files are considerably more complicated than UNIX ones. We have an lrecl as we do on UNIX but we also have various record formats like variable, fixed, variable block and fixed block. Our file we are creating on the mainframe will need the record format and lrecl specified but the Korn script will do some rudimentary block size calculations where appropriate.

First we'll select Prompt Manager from the SAS project view in figure 6:

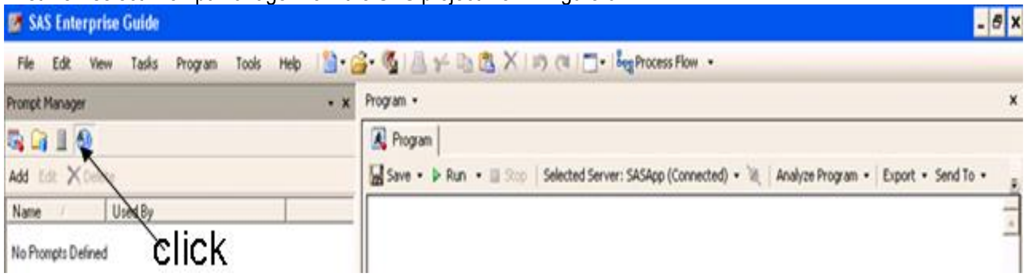
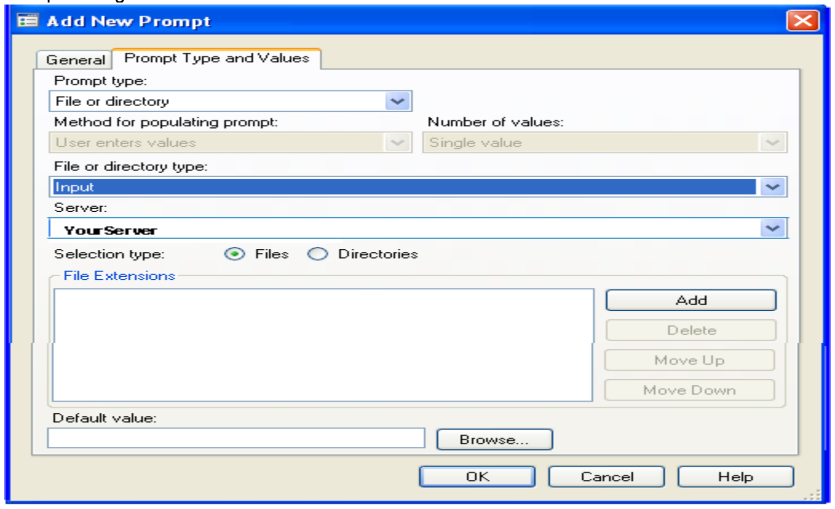


Figure 6: Prompt Manager

The first prompt we will need is one for the UNIX file to be transferred. Enterprise Guide 4.3 lets us set up a prompt that will look much like MS/Windows in selecting a file. We will call our first prompt UNIXFILE. The displayed text will read 'Enter UNIX File to be Sent:' and it will be set up like Figure 7:

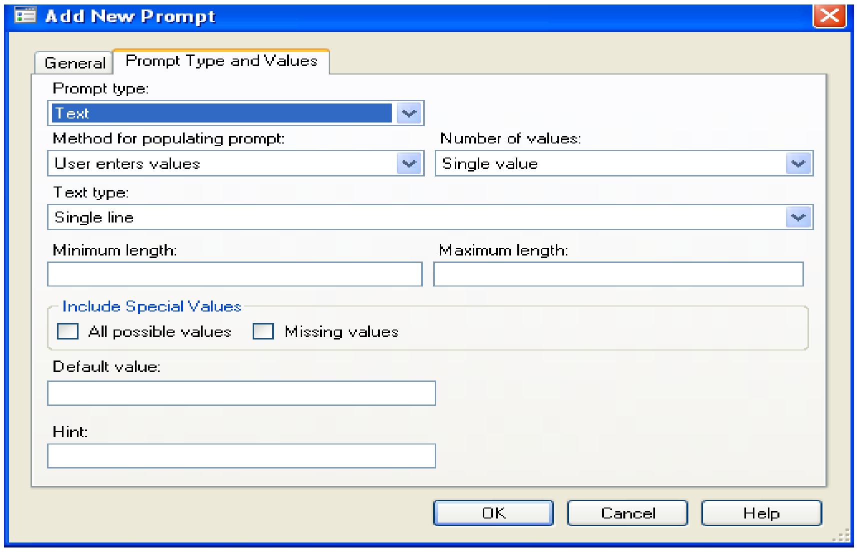


Prompt type: File or directory
Method for populating prompt: User enters values
Number of values: Single value
File or directory type: Input
Server: YourServer
Selection type: (● Files) ○ Directories
Default value:

Figure 7: File/directory prompt setup
Note that we could select either files or directories. Since this application transfers a file, we will select 'Files'.

The next prompt will need is the mainframe file name. This should be a prefix you have –CREATE- authority over. File security works differently on the mainframe than UNIX. With UNIX, it is possible to have write authority without read authority. The mainframe is different, having a hierarchy of READ, WRITE and CREATE where CREATE authority automatically gives you READ and WRITE.

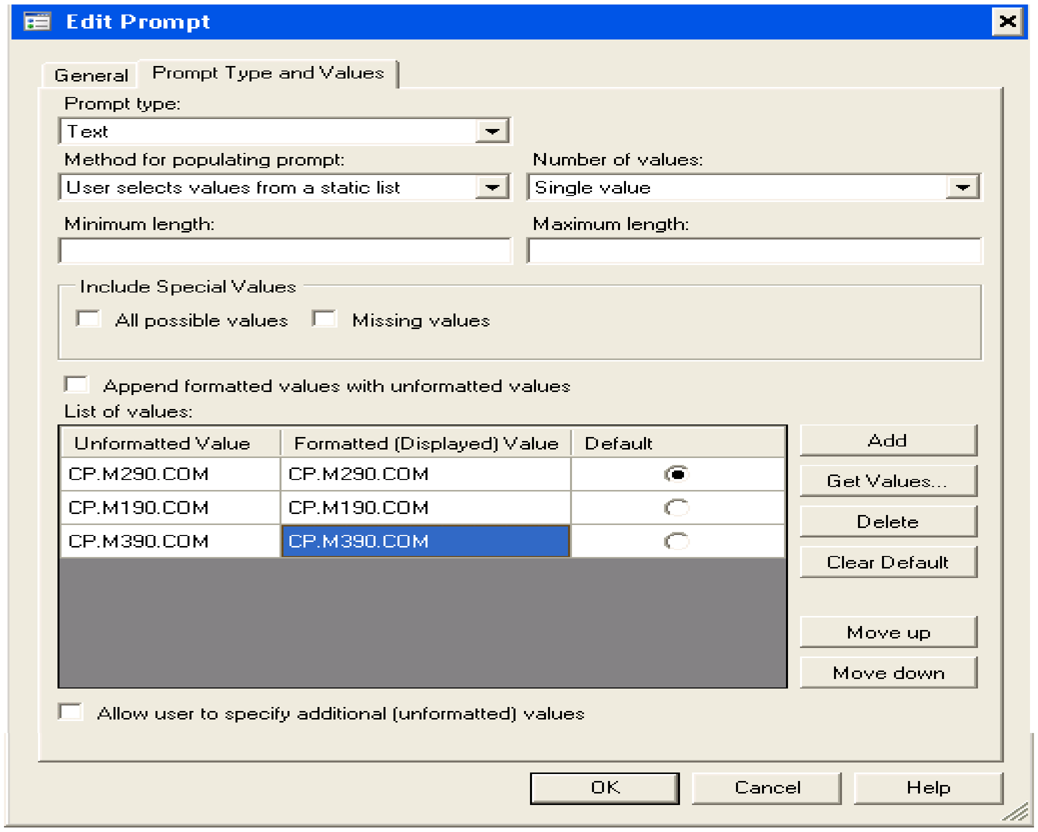
Our prompt will be called MVSFILE and will be set up as straight text. There will be no default value or guidelines on how to enter the name as in figure 8. The display text will be 'Enter Target MVS File Name:'



Prompt type: Text
Method for populating prompt: User enters values
Number of values: Single value
Text type: Single line
Minimum length:
Maximum length:
☐ All possible values ☐ Missing values
Default value:
Hint:

Figure 8: Unformatted text set up

Our next prompt will be the mainframe id. We will call this MVSID and let the user select from a list at run-time. 'Select MVS ID: ' will be the display text. We will set the default value to be Cp.m290.com which will look like 'My Mainframe' at run-time. The other possible values will be Cp.m190.com and Cp.m390.com. This will be a static list. To use a dynamic list, there must be a data source available to you in a SAS folder. How to put this source in such a folder is beyond the scope of this paper but for values that don't change that much, a static list should be fine. We could also select 'Allow user to specify additional (unformatted) values' but we'll go with the addresses in Figure 9 for right now.


Edit Prompt

General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User selects values from a static list

Number of values: Single value

Minimum length:

Maximum length:

Include Special Values: All possible values, Missing values

Append formatted values with unformatted values

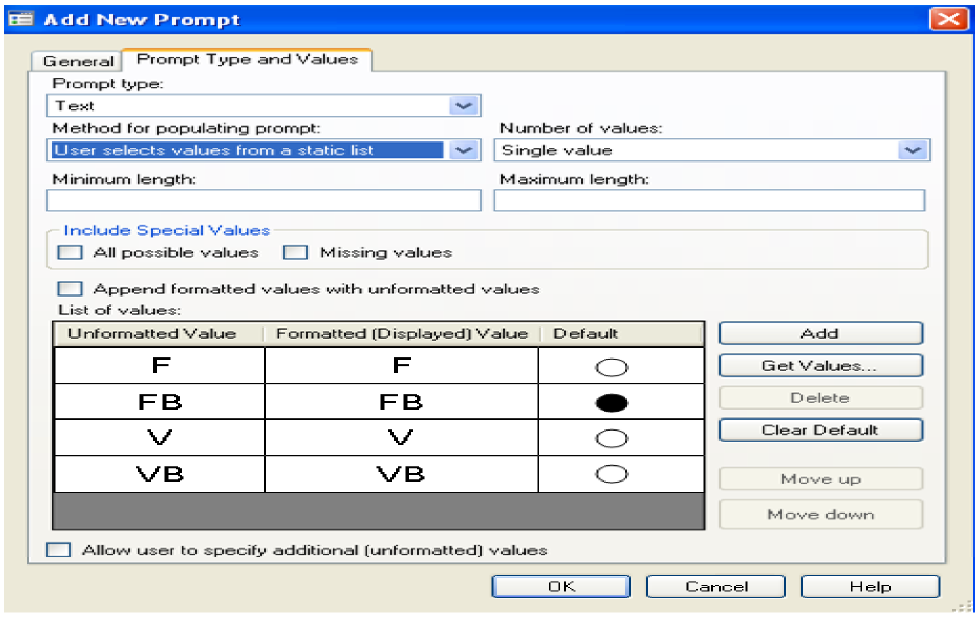
List of values:

Unformatted Value	Formatted (Displayed) Value	Default
CP.M290.COM	CP.M290.COM	●
CP.M190.COM	CP.M190.COM	○
CP.M390.COM	CP.M390.COM	○

Allow user to specify additional (unformatted) values

Figure 9: Entering allowable values

We will do the record format much the same way, since the four basic types, fixed, variable, fixed block, and variable block should suffice for the vast majority of our file transfers. The prompt will be named RECFM, displayed as 'Select Record Format: ' at run-time. We will make fixed block (FB) the default (Figure 10)


Add New Prompt

General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User selects values from a static list

Number of values: Single value

Minimum length:

Maximum length:

Include Special Values: All possible values, Missing values

Append formatted values with unformatted values

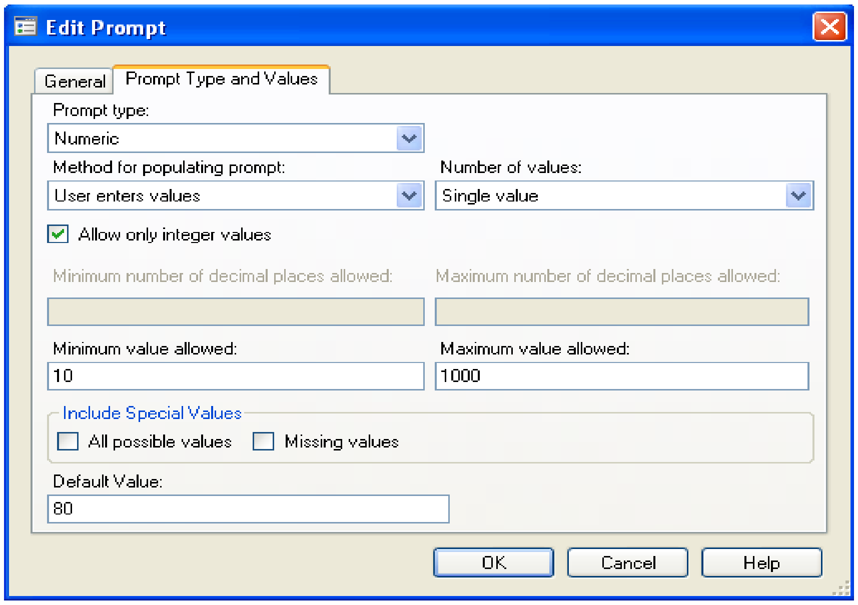
List of values:

Unformatted Value	Formatted (Displayed) Value	Default
F	F	○
FB	FB	●
V	V	○
VB	VB	○

Allow user to specify additional (unformatted) values

Figure 10: Default for record format

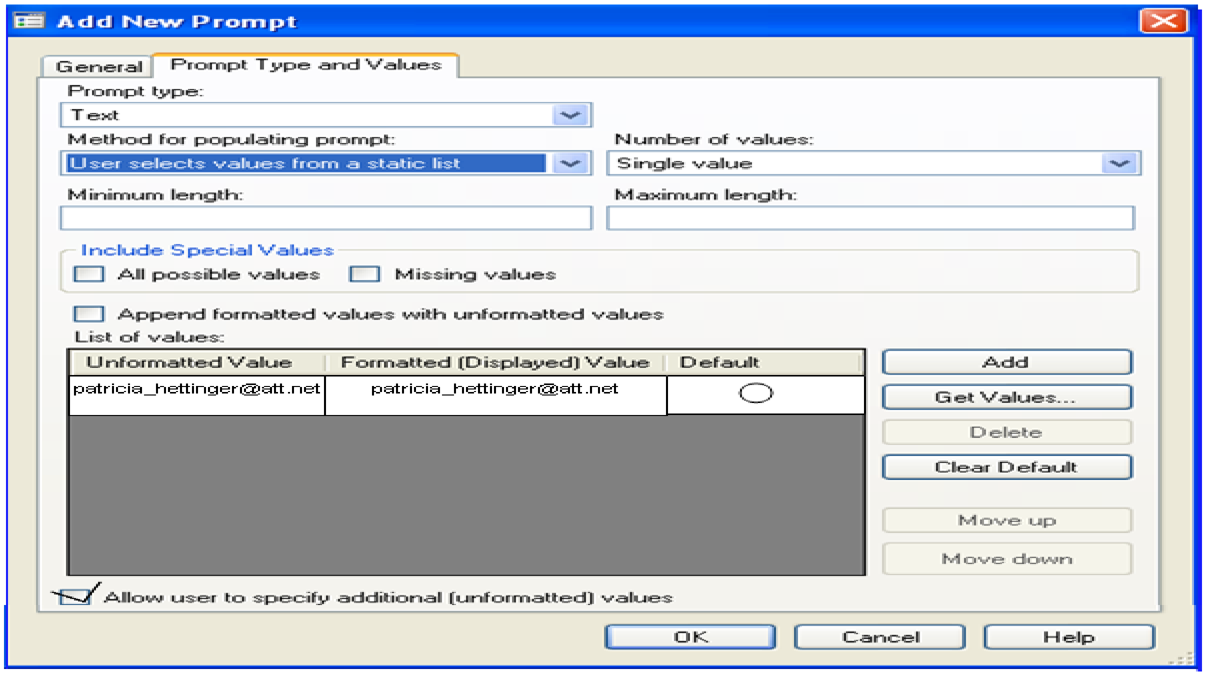
Next we'll set up a prompt for the lrecl which will be named LRECL. The prompt will display as 'Record Length (LRECL)?". Values of ten to a thousand should handle the vast majority of our files (Figure 11)



Prompt type: Numeric
Method for populating prompt: User enters values
Number of values: Single value
[✓] Allow only integer values
Minimum number of decimal places allowed:
Maximum number of decimal places allowed:
Minimum value allowed: 10
Maximum value allowed: 1000
Include Special Values: [ ] All possible values [ ] Missing values
Default Value: 80

Figure 11: Allowable LRECL values

We will want NDM to send an email when it's done so we'll set up a variable named ENOTIFY with the email address. At run-time, this will be displayed as 'Enter Notification Address:' . We can put in some known addresses as in figure 12 but we will also allow the user to specify another address.



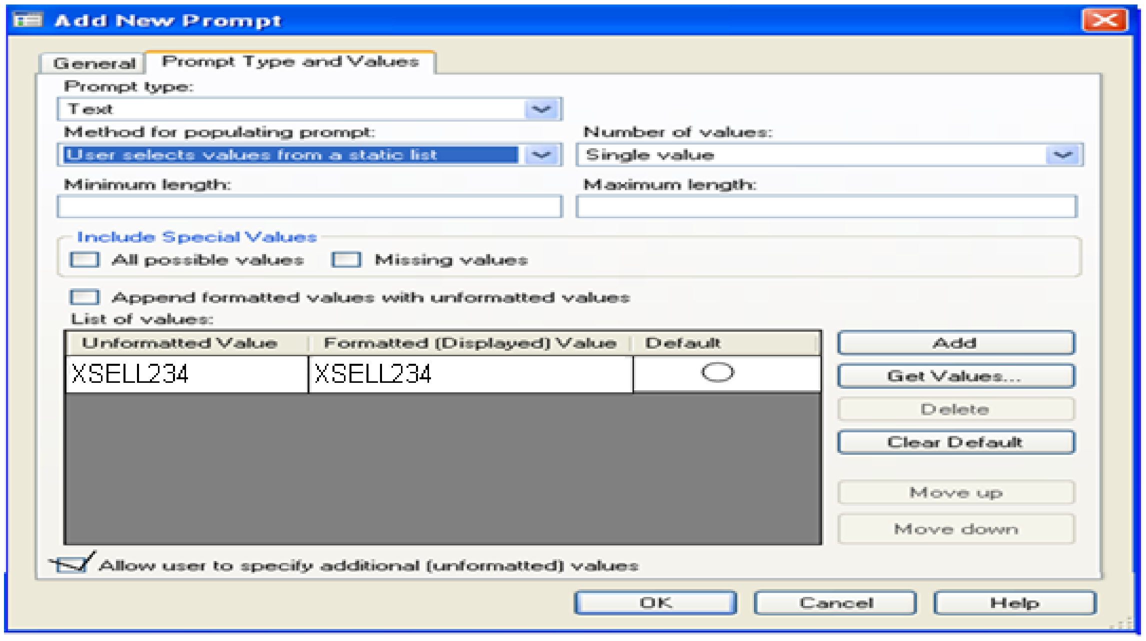
Prompt type: Text
Method for populating prompt: User selects values from a static list
Number of values: Single value
Minimum length:
Maximum length:
Include Special Values: [ ] All possible values [ ] Missing values
[ ] Append formatted values with unformatted values
List of values:

Unformatted Value	Formatted (Displayed) Value	Default
patricia_hettinger@att.net	patricia_hettinger@att.net	○

[✓] Allow user to specify additional (unformatted) values

Figure 12: Letting the user enter other values

We will do the same thing with USERID (mainframe user id), allowing the user to either use some known id like XSELL234 (figure 13) or to specify another one. The display text will be 'Enter MVS User Id:'



Add New Prompt

General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User selects values from a static list

Number of values: Single value

Minimum length:

Maximum length:

Include Special Values: ☐ All possible values ☐ Missing values

☐ Append formatted values with unformatted values

List of values:

Unformatted Value	Formatted (Displayed) Value	Default
XSELL234	XSELL234	○

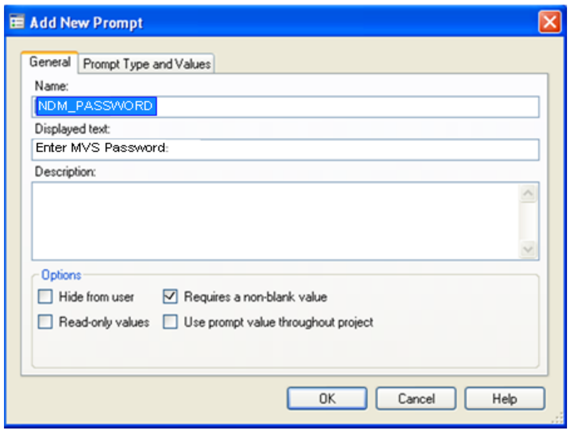
Add | Get Values... | Delete | Clear Default | Move up | Move down

☑ Allow user to specify additional (unformatted) values

OK | Cancel | Help

Figure 13: Mainframe user id

Now for our password. We must pass this to the mainframe as plain text. We will call it NDM_PASSWORD and set it up as illustrated in Figures 14 and 15. Figure 14 shows the displayed text 'Enter MVS Password' and Figure 15 more of the prompt type.



Add New Prompt

General | Prompt Type and Values

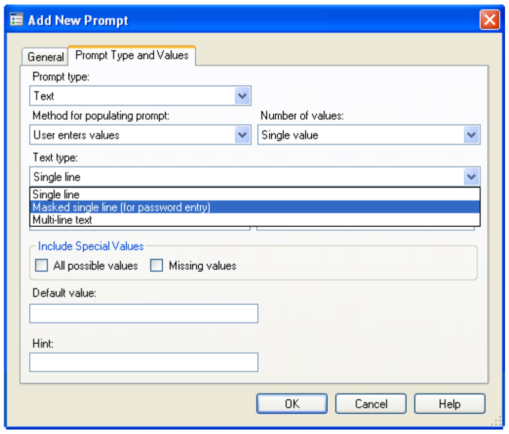
Name: NDM_PASSWORD

Displayed text: Enter MVS Password:

Description:

Options: ☐ Hide from user ☑ Requires a non-blank value ☐ Read-only values ☐ Use prompt value throughout project

OK | Cancel | Help



Add New Prompt

General | Prompt Type and Values

Prompt type: Text

Method for populating prompt: User enters values

Number of values: Single value

Text type: Single line

Single line / Masked single line (for password entry) / Multi-line text

Include Special Values: ☐ All possible values ☐ Missing values

Default value:

Hint:

OK | Cancel | Help

Figure 14 - the _PASSWORD suffix

Figure 15 - masking at run-time

Now that we have our prompts set up, we need a SAS program to use them. Our SAS program that calls the Korn script is very simple - just turning off symbolgen and turning the mainframe variables MVSFILE, USERID and NDM_PASSWORD to upper case on the strong likelihood your receiving MVS system would consider these in error otherwise. We then use the systask command to execute the Korn script.

```
options NOsymbolgen;
%let MVSFILE=%trim(%upcase(&MVSFILE));
%let USERID = %trim(%upcase(&USERID));
%let UNIXFILE=%trim(&UNIXFILE);
%let MVSID=%trim(&MVSID);
%let RECFM=%trim(&RECFM);
%let LRECL=%trim(&LRECL);
%let NDM_PASSWORD=%TRIM(%UPCASE(&NDM_PASSWORD));
SYSTASK COMMAND "/UNIX_path/NDM2MVS.ksh &UNIXFILE &MVSFILE &MVSID &RECFM
&LRECL &USERID &NDM_PASSWORD &ENOTIFY";
RUN;
```

We will call this program SAS2NDM.SAS. Now we have to associate the program with our prompts. We could do this from the editor or by right-clicking on the program icon in Enterprise Guide's Project View and clicking on 'Properties' as in Figure 16:

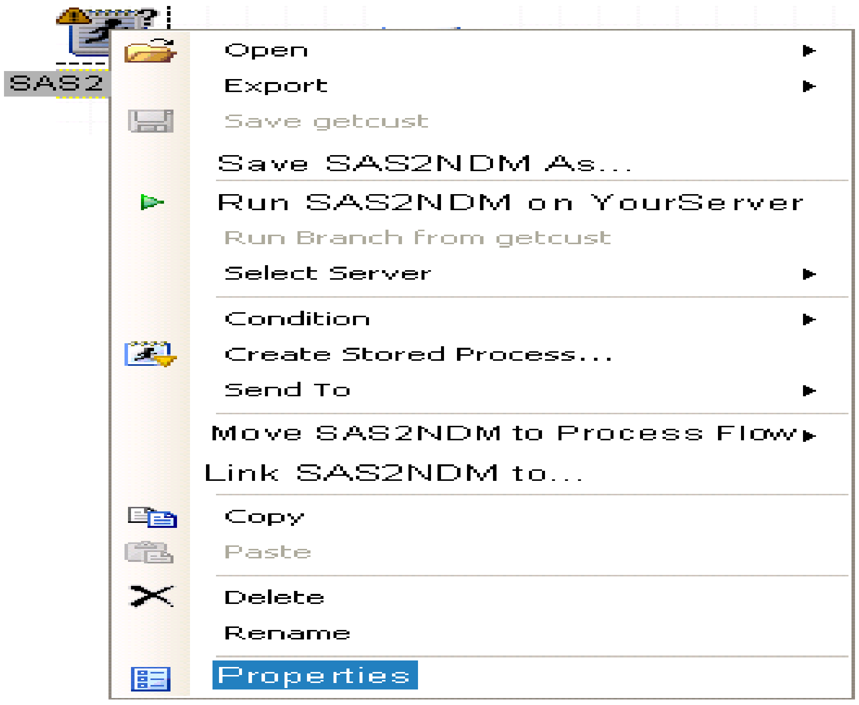
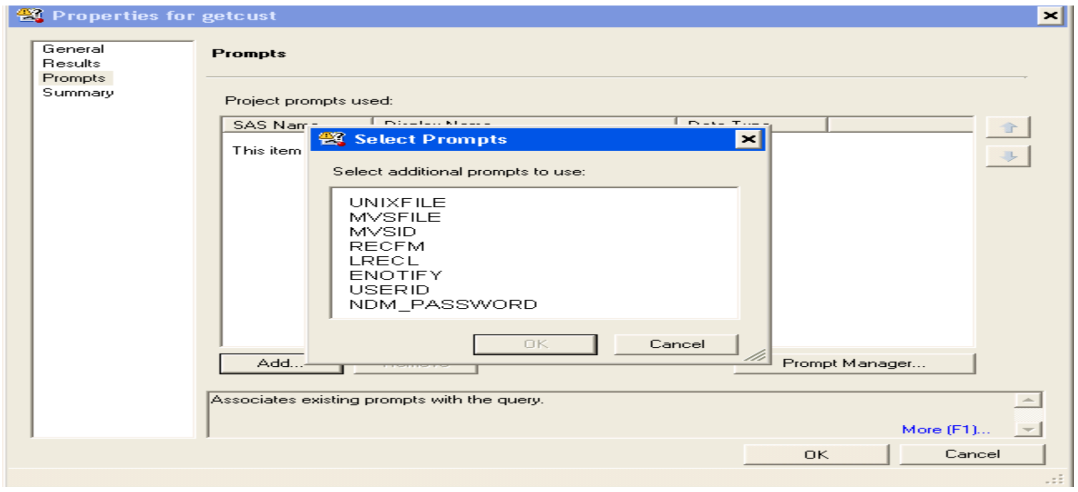


Figure 16 – Selecting properties

A properties box will open up. If you select prompts and then 'add', you will see a list of all the available prompts (Figure 17). We will add all of them in the order given.



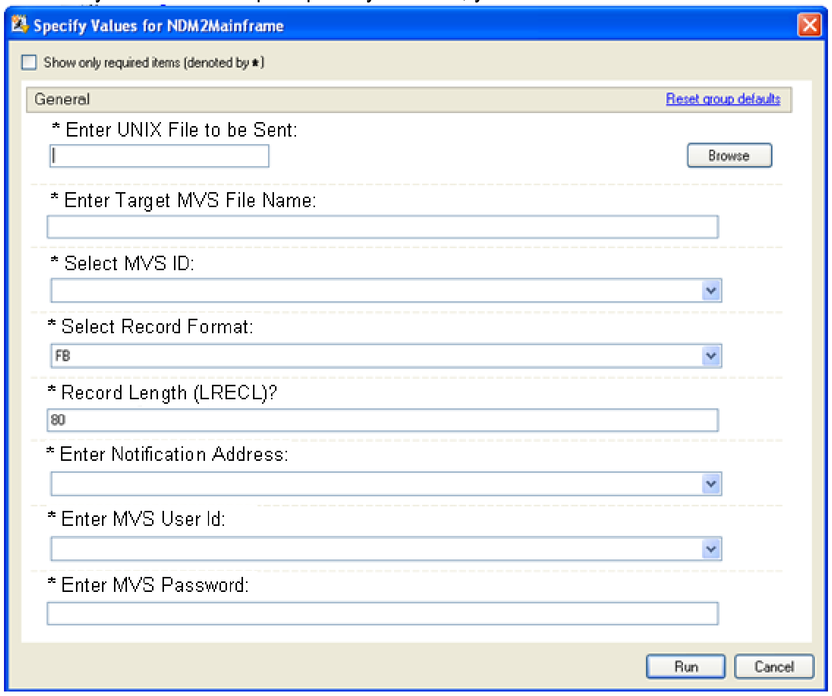
UNIXFILE
MVSFILE
MVSID
RECFM
LRECL
ENOTIFY
USERID
NDM_PASSWORD

Associates existing prompts with the query.

Figure 17: Adding prompts to our code

RUNNING CODE WITH PROMPTS

Now that you'd added the prompts to your code, your screen should look like this at run-time (Figure 18)



Specify Values for NDM2Mainframe

Show only required items (denoted by *)

General — Reset group defaults

* Enter UNIX File to be Sent: [Browse]

* Enter Target MVS File Name:

* Select MVS ID:

* Select Record Format: FB

* Record Length (LRECL)? 80

* Enter Notification Address:

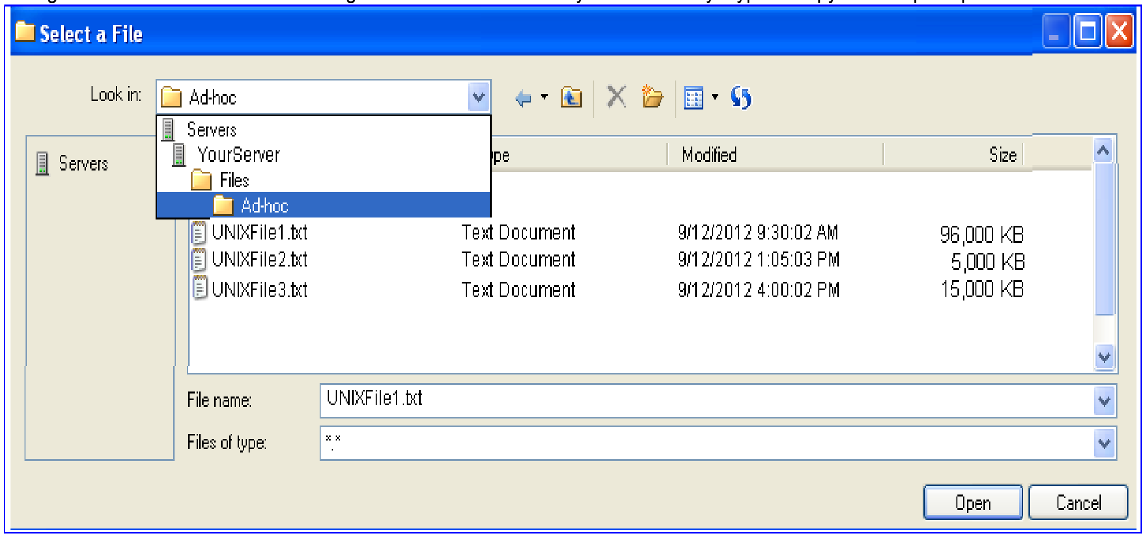
* Enter MVS User Id:

* Enter MVS Password:

[Run] [Cancel]

Figure 18: Run-time

Let's take a better look at the first prompt, selecting the file on UNIX to be sent. In order to browse for a file, you must be able to reach it through a SAS metadata server as in Figure 19 below. Of course you could always type or copy the complete path as well.



Select a File

Look in: Ad-hoc (Servers / YourServer / Files / Ad-hoc)

Name	Type	Modified	Size
UNIXFile1.txt	Text Document	9/12/2012 9:30:02 AM	96,000 KB
UNIXFile2.txt	Text Document	9/12/2012 1:05:03 PM	5,000 KB
UNIXFile3.txt	Text Document	9/12/2012 4:00:02 PM	15,000 KB

File name: UNIXFile1.txt

Files of type: *.*

[Open] [Cancel]

Figure 19 – Browsing files from the metadata server

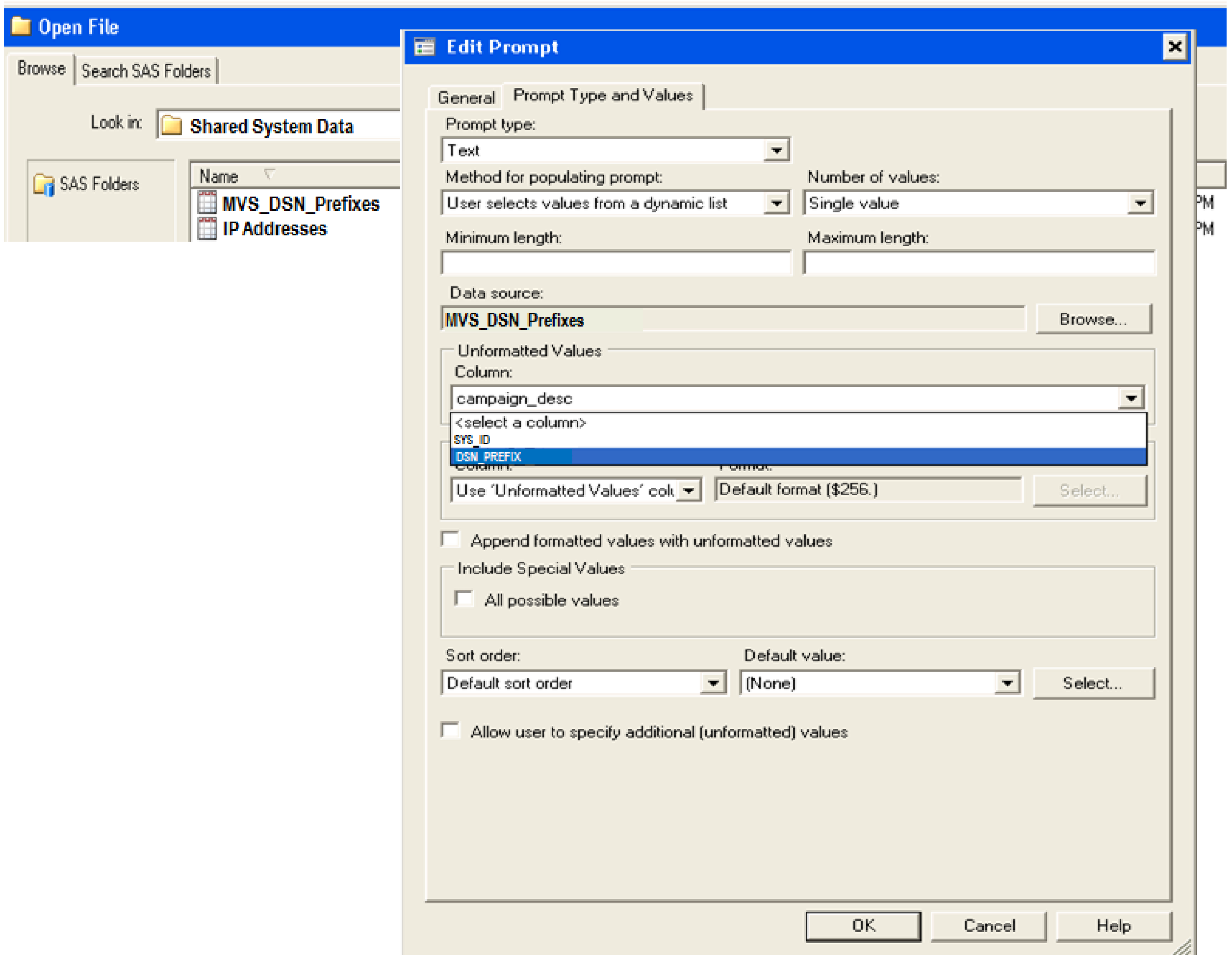
PROMPTS AND STORED PROCESSES

The ConnectDirect program above became so popular that it turned into a stored process, easily accessible from any project that had access. The first version was the one you just saw in this paper. It worked well but didn't address one common problem. The systems all had different allowable data set prefixes that people kept forgetting. They also kept forgetting the MVS data set naming conventions. The maximum allowable name length is 44. The nodes are separated with periods, with each node having the maximum length as eight. No node can begin with a number. Some special characters are allowed but for the sake of simplicity, their use would not be allowed in the revised process.

One thing you can do in a stored process that isn't possible otherwise is set up prompt dependencies. We created a dependency between our MVSID (mainframe id) prompt and a new prompt called MVSPREFIX. For this to work, MVSPREFIX had to be populated from a dynamic list as figure 20 demonstrates. This was a small SAS data set called MVS_DSN_Prefixes with two variables, SYS_ID and DSN_PREFIX. There were only six observations. Note that CP.M190.COM and CP.M390.COM had more than one allowable prefix

SYS_ID	DSN_PREFIX
CP.M290.COM	NANS.OO
CP.M190.COM	SAPM.CIA
CP.M190.COM	HTS.00
CP.M390.COM	FARV.OUT
CP.M390.COM	FARV.IN
CP.M390.COM	INDRA.SOUC

We found the MVS_DSN_Prefixes data set in a SAS folder, set the method for populating the prompt as 'User selects values from a dynamic list', and selected DSN_PREFIX as source variable. The user may not input any other values.

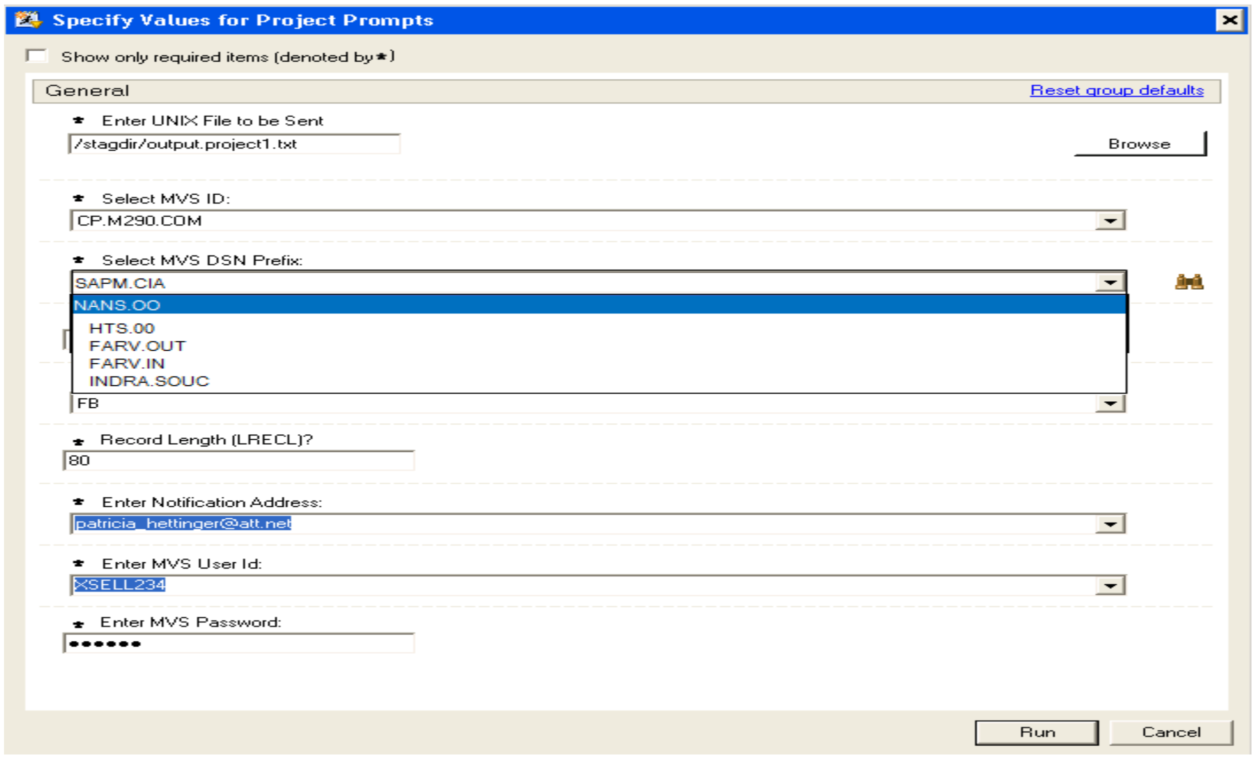


Look in: Shared System Data
MVS_DSN_Prefixes

Prompt type: Text
Method for populating prompt: User selects values from a dynamic list
Number of values: Single value
Minimum length:
Maximum length:
Data source: MVS_DSN_Prefixes
Column: campaign_desc
<select a column>
SYS_ID
DSN_PREFIX
Column: Use 'Unformatted Values' colu
Format: Default format ($256.)
Append formatted values with unformatted values
All possible values
Sort order: Default sort order
Default value: (None)
Allow user to specify additional (unformatted) values

Figure 20 – Populating MVSPREFIX from a dynamic list

Adding the prompt to the program and running it from the project gives us all of the possible data set prefixes regardless of the MVS ID. Another new prompt, mf_file_name was created for the rest of the MVS data set name. The rules for this part of the name had to be put in a SAS program, as they were too complicated for Prompt Manager. See appendix C for the version with the MVS data set validation added.


Specify Values for Project Prompts

Show only required items (denoted by *)

General — Reset group defaults

* Enter UNIX File to be Sent: /stagdir/output.project1.txt

* Select MVS ID: CP.M290.COM

* Select MVS DSN Prefix: SAPM.CIA

NANS.OO
HTS.00
FARV.OUT
FARV.IN
INDRA.SOUC

FB

* Record Length (LRECL)? 80

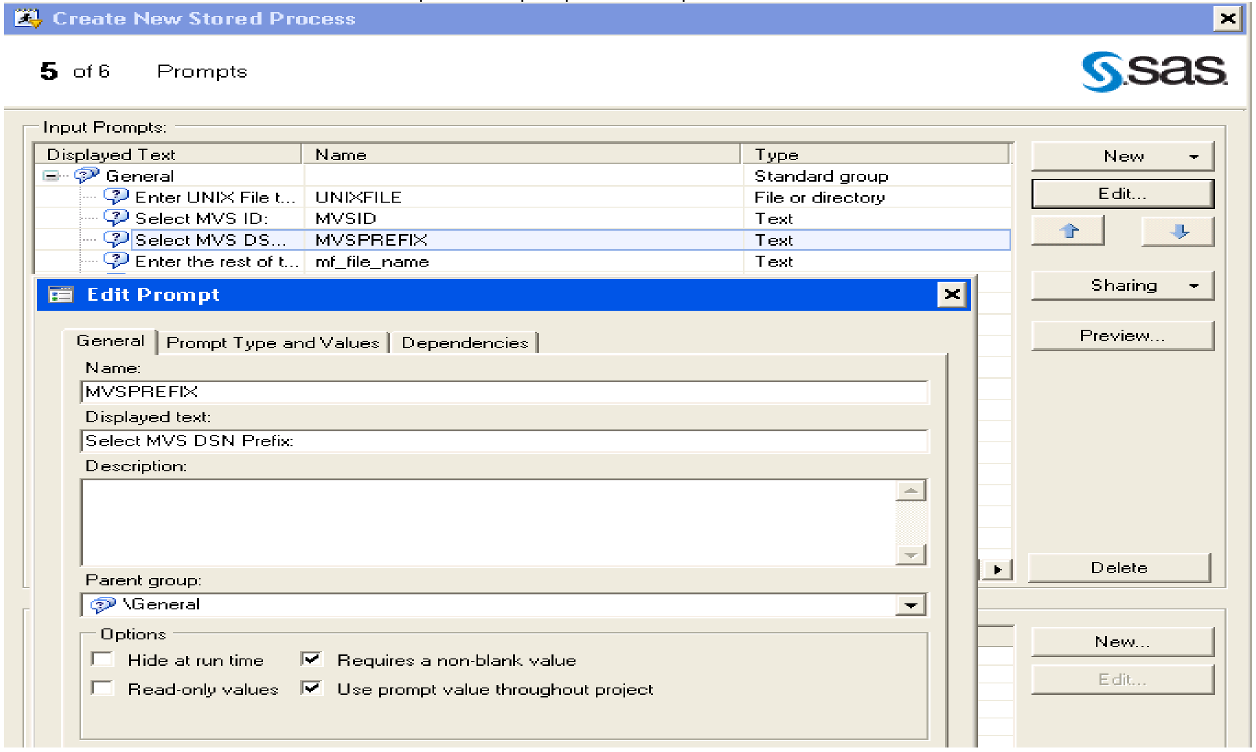
* Enter Notification Address: patricia_hettinger@att.net

* Enter MVS User Id: XSELL234

* Enter MVS Password: ●●●●●●

Figure 21 Run-time from Project MVS DSN Prefix

If we create this as a stored process on either a logical or workspace server, we can set up MVSPREFIX to only show those values associated with the selected MVS id. You'll see an extra option in the prompt section: dependencies


Create New Stored Process

5 of 6 Prompts

Input Prompts:

Displayed Text	Name	Type
General		Standard group
Enter UNIX File t...	UNIXFILE	File or directory
Select MVS ID:	MVSID	Text
Select MVS DS...	MVSPREFIX	Text
Enter the rest of t...	mf_file_name	Text

Edit Prompt

General | Prompt Type and Values | Dependencies

Name: MVSPREFIX

Displayed text: Select MVS DSN Prefix:

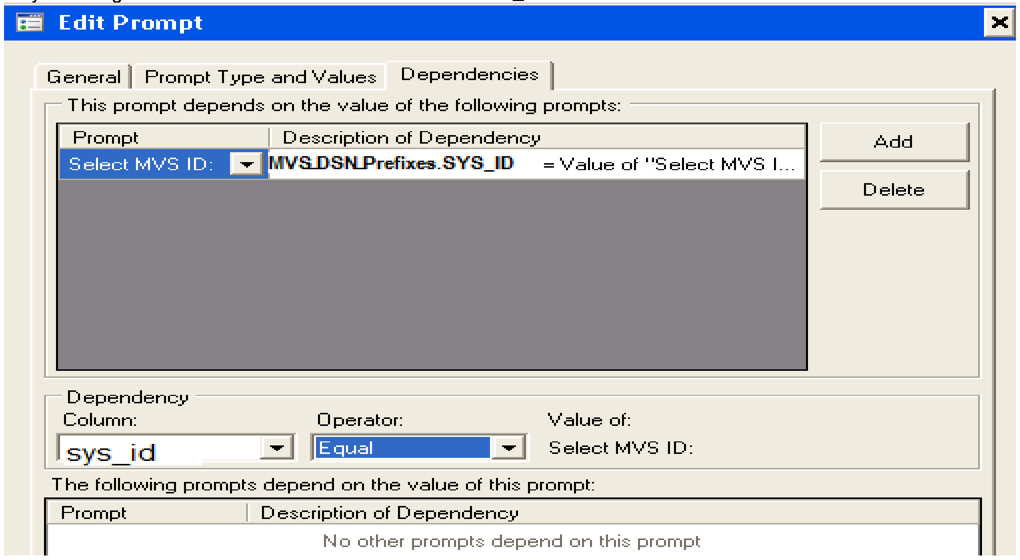
Description:

Parent group: \General

Options
- [ ] Hide at run time
- [x] Requires a non-blank value
- [ ] Read-only values
- [x] Use prompt value throughout project

Figure 22: Editing a prompt in a stored process

In creating a dependency, the system id will determine which DSN prefixes we can select. Note that while MVSPREFIX must be sourced from a dynamic list, the prompt on which it depends can be source from a static list or even straight text entry. For those of us familiar with SQL, this may be thought of as a WHERE statement – WHERE SYS_ID = &MVSID.


Edit Prompt

General | Prompt Type and Values | Dependencies

This prompt depends on the value of the following prompts:

Prompt	Description of Dependency
Select MVS ID:	MVS.DSN.Prefixes.SYS_ID = Value of "Select MVS I...

Add
Delete

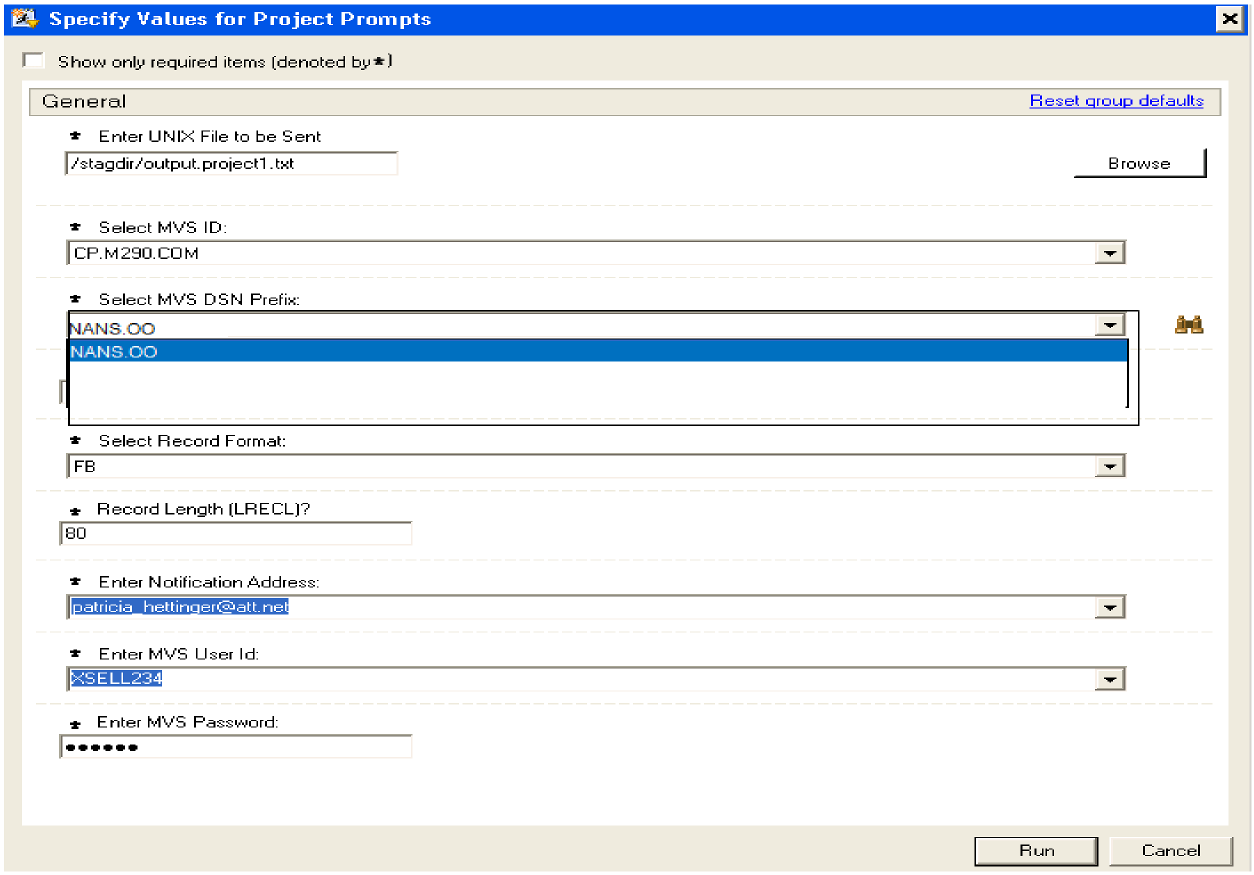
Dependency
Column: sys_id
Operator: Equal
Value of: Select MVS ID:

The following prompts depend on the value of this prompt:

Prompt	Description of Dependency
No other prompts depend on this prompt	

Figure 23: Prompt dependency details

Now when we run this as a stored process, we'll see only the prefixes allowed for that MVS system ID:


Specify Values for Project Prompts

Show only required items (denoted by *)

General — Reset group defaults

* Enter UNIX File to be Sent: /stagdir/output.project1.txt Browse

* Select MVS ID: CP.M290.COM

* Select MVS DSN Prefix: NANS.OO
NANS.OO

* Select Record Format: FB

* Record Length (LRECL)? 80

* Enter Notification Address: patricia_hettinger@att.net

* Enter MVS User Id: XSELL234

* Enter MVS Password: ••••••

Run Cancel

Figure 24: Stored Process MVS DSN Prefix appearance as a function of MVS ID

If your connection has more than one server, make sure you create your stored process to run on the same server where your dynamic list is defined. If you do not, nothing will show up for your prompt. If a value is required for the prompt, you won't be able to run the stored process at all.

CONCLUSION

The NDM demonstration is just one of several applications you can build with prompts to automate common processes, reduce error and just generally make your work day more productive. Just another reason to get started with SAS Enterprise Guide!

ACKNOWLEDGEMENTS

Thank you, Joe and Paul Butkovich for reviewing this paper and presentation

CONTACT INFORMATION

Your comments, questions and experiences are valued and encouraged. Contact the author at:

Patricia Hettinger
Oakbrook Terrace, IL 60523
Phone: 331-462-2142, cell 630-309-3431
Email: patricia_hettinger@att.net

APPENDIX

A. Macro to use multiple character prompts in a 'WHERE' statement

```
%MACRO proctext;
%LET listval = ;    *&listval will be our resulting variable;
    %DO i=1 %TO &custname_count;   *&custname_count is the number of values selected;
            %IF &i = &custname_count %THEN %DO;
                    %LET listval =   *build the string;
                    &listval "%unquote(%NRSTR(&custname)&i. )";  *quoting and unquoting;
            %END;
    %ELSE %DO;
            %LET listval = &listval "%unquote(%NRSTR(&custname)&i. )",;
    %END;
    %END;
            PROC SQL;
            SELECT * FROM
            dictionary.tables
            WHERE memname IN (&listval);

*&listval evaluates to list as in Figure 4 like 'Jones',' Bradford','Hettinger','Harrison';
            QUIT;
%MEND proctext;
%proctext;
```

B. NDM UNIX to Mainframe Korn shell script

```
!/usr/bin/ksh
NDMAPICFGxxxx//ndmapi.cfg
export NDMAPICFG
#above will vary by your system
#######################################################################################
## This script is to transfer one file from a UNIX to one of several OS/390 servers
# There are eight parameters that are needed:  Parameter 1 = The from file - fully qualified,
#      Parameter 2 = The to file - fully qualified,  Parameter 3 = snodep - receiving system
#      Parameter 4 = recfmp - OS/390 record format,  Parameter 5 - lreclp - OS/390 record length
#      Parameter 6 - sid - OS/390 user id,  Parameter 7 - pw - MVS password
#     Parameter 8 - notify - email address for ndm notification
#######################################################################################
fromfile=$1
tofile=$2
snodep=$3
recfmp=$4
lreclp=$5
sid=$6
pw=$7
notify=$8
if [[ $# != 8 ]];then
   print "Need eight parms"
   exit
fi
if [[ $recfmp = "FB" ]];then
             if [[ $lreclp -lt 278 ]];then let blksize=$lreclp*100
             elif [[ $lreclp -gt 277 ]];then let blksize=$lreclp*10
             fi # calculate blocksize if record format is fixed block
elif [[ $recfmp = "VB" ]];then let blksize=32760 #literal for variable blocked
elif [[ $recfmp = "V" ]];then let blksize=$lreclp+4 #variable unblocked
elif [[ $recfmp = "F" ]];then let blksize=$lreclp #fixed unblocked
fi
set -v
/xxx/ndmcli  -x << EOJ # Above path will vary according to your system
submit maxdelay=0 proc1    process
snode=$snodep
snodeid=($sid,$pw) #mainframe id and password
notify=$notify
copy01 copy from (file=$fromfile
                  sysopts=":datatype=text:"
                  PNODE
```

```
            )
 ckpt=5m
          compress extended=(CMP=5
                            WIN=15
                           MEM=9)
to      (file=$tofile
             disp=(RPL,CATLG,CATLG)  #RPL overwrites current file if it exists
           DCB=(DSORG=PS,RECFM=$recfmp,LRECL=$lreclp,BLKSIZE=$blksize)
           unit=XXX   # varies according to your system and whether you want tape or disk
  snode
  )
  if  (copy01 != 0) then
failed run task  pnode
              sysopts="echo '$JOBJSTEP  NDM FAILED' "
     eif
pend;
EOJ
```

C. Data Set Name Validation Code

```
%LET RC=0;
Data _null_;  data step handles this sort of thing better
attrib file_name format=$50.;

file_name="&MVSPREFIX..&mf_file_name";
flength=length(file_name);
rc=0;
call symput('MVSFILE',file_name);  this will be our MVSFILE value
if length(file_name) gt 44 then do;   check for total length
file print;
put @1 'ERROR! File Name Too Long' ;
rc=99;
end;
Array name_part(6) $15.;
do i=1 to 6;
name_part(i)=scan(file_name,i,'.');
if length(name_part(i)) gt 8 then do;  check for length of each node
file print;
put @1 "ERROR! Node #" i "Greater than 8 positions long for " name_part(i);
rc=99;
end;
IF (UPCASE(SUBSTR(NAME_PART(I),1,1)) LT 'A' OR
UPCASE(SUBSTR(NAME_PART(I),1,1)) GT 'Z') AND NAME_PART(I) NE ' '
THEN DO;  node cannot begin with number or special character
file print;
put @1 "ERROR! Node #" i "Must begin with a character " name_part(i);
rc=99;
end;
if notalnum(trim(name_part(i))) gt 0 no special characters at all
and name_part(i) ne ' ' then do;
file print;
```

```
put @1 "ERROR! Node #" i " Special Characters Not Allowed " name_part(i);
rc=99;
end;
end;
call symput('rc',rc);
run;
%put &rc;

run;


%macro handle_name(rc);
%if &rc = 0 %then %do; run ConnectDirect if name OK
options NOsymbolgen;
%let MVSFILE=%trim(%upcase(&MVSFILE));
%let USERID = %trim(%upcase(&USERID));
%let UNIXFILE=%trim(&UNIXFILE);
%let MVSID=%trim(&MVSID);
%let RECFM=%trim(&RECFM);
%let LRECL=%trim(&LRECL);
%let NDM_PASSWORD=%TRIM(%UPCASE(&NDM_PASSWORD));
systask command "/UNIX_path/NDM2MVS.ksh &UNIXFILE &MVSFILE &MVSID &RECFM
&LRECL &USERID &NDM_PASSWORD &ENOTIFY";
RUN;
DATA _NULL_;
FILE PRINT;
PUT @1 'ConnectDirect Transfer Submitted';
run;
%end;
%else %do;

DATA _NULL_;    print message if anything wrong in the name
FILE PRINT;
PUT @1 'TRANSFER NOT RUN - INVALID MVS NAME';
RUN;
%end;
%mend;
%handle_name(&rc);
run;
```