

# ISPF File Tailoring Skeleton Parser

ISPF File Tailoring Skeleton Parser .....	1
Introduction .....	3
Installation.....	3
The Parser Engine .....	4
Customization .....	4
XML Tags .....	5
<QueryDisplayName> .....	5
<Directory> .....	5
<DirectoryName> .....	5
<DirectoryLabel>.....	5
<DirectoryHostName>.....	5
Output file types.....	5
Variables.....	6
Skeletons .....	6
Tables(DOT) .....	6
Functions.....	6
Libraries .....	6
Keywords .....	6
VarTypes.....	6
FieldOffsets .....	8
<ExcelOutput> .....	9
<ExcelOutputFolder> .....	9
<ExcelOutputHLQ>.....	9
<XMLOutput> .....	10
<XMLOutputFolder>.....	10
<XMLOutputHLQ> .....	10
<FixedOutput> .....	11
<ConfigurationName> <ConfigurationNumber> .....	11
<FixedOutputFolder> .....	12
<FixedOutputHLQ>.....	12
<IgnoreSkel>.....	12
<Unimbedded Skeleton> .....	13
<ImbedRecurseFail>.....	13
<DebugInputLine>.....	14

Creating Different Run Time Versions.....	14
The Graphical Front-End .....	16
Load a New Configuration.....	17
Skeleton to Variable Cross-Reference.....	18
Displaying the Skeleton .....	19
Exporting data .....	20
Skeleton to Skeleton Cross-Reference .....	21
Display the Skeleton .....	22
Exporting data .....	23
Skeleton to Table (DOT) cross-reference .....	24
Display the Skeleton .....	25
Exporting data .....	25
Skeleton to Program Cross-Reference .....	27
Display the Skeleton .....	28
Exporting data .....	28
Skeleton Expansion.....	30
Expansion Context Menu.....	31
Filtering Text Within a skeleton line.....	31
Using a Filter to Hide Skeleton Expansions .....	32
Removing Filters.....	33
Positioning on a Specific Line (Goto).....	33
Skeleton Logic Display For A Selected Line .....	35
Variables Display Within Lines.....	35
Graphic View .....	36

## ***Introduction***

The ISPF File Tailoring Skeleton Parser reads file tailoring input files (skeletons) and generates several cross-reference files:

- Skeleton file-to-variable cross-reference
- Skeleton file-to-embedded skeleton file cross-reference
- Skeleton file-to-ISPF skeleton built-in function cross-reference

For each cross-reference dataset, three optional file formats can be created:

- XML records
- Comma separated value records, suitable for loading into an Excel spreadsheet
- Fixed format records, suitable for loading into relational tables.

This version of the parser is not an update to any previous version. At present there is a limited graphic user interface front-end. The parser itself runs as a command-prompt .exe file and it is invoked internally by the graphic front-end.

## ***Installation***

There are two installation files:

- SETUP.EXE
- INSTALLER.MSI

You must have administrator rights to the system on which you will be installing the parser. Also, you must have the right to install applications.

To install the parser, simply run the setup program. Note that this program will install the Microsoft .NET 3.5 run time. This run time will be downloaded from Microsoft, therefore the installer needs Internet access. It is the installation of the .NET run time that requires administrator rights on your PC.

Following the run time installation the SETUP.EXE program will install the parser and its supporting files. You will be able to override the default install destination directory. The rest of the installation is automated. All of the installation files related to the utility (save for the .NET run time) are installed in a single directory.

The application directory installed files are:

Config1.xml	A sample XML file. You can use this as a base copy. If you accidentally wipe out the base copy there is a backup in the Samples directory.
imbeds2.ico	The icon for the uninstaller
Library.dll	Program support library
LibraryConcatenation.dll	Program support library

Parser.dll	Program support library
README.TXT	Updated program information.
SkeletonParserDS.dll	Program support library
SkeletonParser.exe	The parser program.
SkeletonParserQuery.exe	The Graphical Front End for the parser engine (see below). This is a GUI query program for displaying the Skeleton-Variable cross reference, Skeleton-Skeleton cross-reference and the Skeleton Expansion as table data.
SkelParser.cmd	A sample CMD file to invoke the parser program.
Utility.dll	Program support library
XMLParmsReader.dll	Program support library

Below the application directory are two other directories, Samples and Doc. *This* documentation is in the Doc directory:

SkeletonParser\_Readme.PDF This documentation.

The Samples directory installed files are:

BASE.xml	A read only sample XML parameters file. This should not be modified, but can be copied to create new configurations.
BASE.xsd	A read only schema for the XML file
parser.rex	A sample Regina REXX program showing how to parse the program output.
parserDB2.rex	A sample Regina REXX program showing how to parse the fields offsets fixed format file to create DB2 <i>CREATE TABLE</i> control cards.
SkelParserDB2R.cmd	Invokes parserDB2.rex, passing in a configuration XML file name.
SkelParserR.cmd	Invokes parser.rex, passing in a configuration XML file name.

## ***The Parser Engine***

The parser.dll file implements the parser engine. The engine does the actual analysis of skeleton data passed in the XML parameter file. The parser engine generates internal data stores that are then used to create the various output or query data defined below. The parser engine can be invoked by the batch parser and the Graphical Front End (GFE).

## ***Customization***

You must create a separate directory on the target PC system for each host PDS. There must be one dataset for each member of the PDS. At this time, no utilities are supplied to facilitate transferring files from the z/OS host to the PC.

The parser supports multiple host PDS concatenations. Each unique concatenation of datasets must be specified in a separate XML file. A sample XML file has been provided for customization. This is Config1.xml. Config1.xml is a copy of Base.xml. Note that Config1.xml and Base.xml are identical, but BASE.XML is marked READONLY. You should leave BASE.XML as a model for any configurations you want to create. But if you copy BASE.XML, be sure to change the file properties of the newly created file to remove the READONLY property.

## XML Tags

### <QueryDisplayName>

A character string that is used in the Graphical Front End Query GUI to identify the configuration on all of the dialog windows. Note that this does *not* have to match the <ConfigurationName> value described below in the fixed file output tags definitions.

### <Directory>

Find the <Directory> tag. In BASE.XML it is:

```
<Directory>
  <DirectoryName>C:\mypath\mycustomSkels</DirectoryName>
  <DirectoryLabel>Custom</DirectoryLabel>
  <DirectoryHostName>somnode.CUSTOM.SKELS</DirectoryHostName>
</Directory>
<Directory>
  <DirectoryName>C:\mypath\myvendorSkels</DirectoryName>
  <DirectoryLabel>Vendor</DirectoryLabel>
  <DirectoryHostName>somnode.VENDOR.SKELS</DirectoryHostName>
</Directory>
```

You may specify one or more directories. Each represents one host ISPF skeleton PDS. The order of the <Directory> tags should match the order of the concatenation that is being simulated. The above XML fragment represents this JCL:

```
//ISPSLIB DD DISP=SHR,DSN=somnode.CUSTOM.SKELS
//          DD DISP=SHR,DSN=somnode.VENDOR.SKELS
```

### <DirectoryName>

This is the PC directory containing the datasets copied from one host PDS. At the time the parser is run, this directory must exist, or the parser will terminate with an error. The directory does not have to be on the C: drive, and it can be on a network drive that is read accessible. Note that reading files in network folders from a parser running on a remote machine will be a slower process.

### <DirectoryLabel>

The directory labels are not critical to the program operation. They are written as specified to one of the output files. The label is just a way to give a more meaningful title to the specified Host/PC file combination in the <Directory> element.

### <DirectoryHostName>

Like the <DirectoryLabel>, the data supplied for the <DirectoryHostName> is not critical to the program operation. It is useful for identifying what host file was used to fill the members in the PC directory.

## Output file types

There are three output file types: CSV (comma separated values), XML and fixed format. Within each

of the file types there are several output datasets:

### ***Variables***

The Variables dataset contains fields showing the Skeleton-Variable relationships.

### ***Skeletons***

The Skeletons dataset contains fields showing the Skeleton-)IM'd skeleton relationships.

### ***Tables(DOT)***

The DOT dataset contains fields showing the Skeleton-)DOT (table) relationships.

### ***Functions***

Each function is shown in the Variables reference. But the Functions dataset contains extra information, showing the arguments passed to each function.

### ***Libraries***

The information in the Libraries file is directly extracted from the XML <Directory> elements in your input.

### ***Keywords***

For the fixed format file output, the Keywords file contains information that can be used to build a look-up table. Most keywords are ISPF Skeleton commands, such as IM, SEL, SET or DO. But there are some that further delineate some of the commands. Thus DOWHILE and DOUNTIL are refinements to how a )DO statement can be written, but neither DOWHILE nor DOUNTIL are ISPF commands.

The non-ISPF keywords are: DOWHILE, DOUNTIL and DATALINE.

### ***VarTypes***

The VarTypes dataset is a lookup table for variable type numbers used in the Variables dataset. This dataset is only generated if the fixed format output files are generated.

#### **VARIABLE\_REFERENCE**

Variable found in a non-command line.

#### **CONSTANT\_REFERENCE**

A constant found in a )SET or )SETF statement. Up to the first eight characters are shown.

#### **CONSTANT\_TEST**

A constant found in a )SEL or )IF statement. Up to the first eight characters are shown.

#### **VARIABLE\_TEST**

A variable used in a )SEL or )IF statement

#### **LVAL\_ASSIGNMENT**

The left side (LVAL) of an assignment statement, )SET or )SETF. The variable is specified without a leading &.

#### **LVAL\_INDIRECT\_ASSIGNMENT**

The left side (LVAL) of an assignment statement, )SET or )SETF. The variable name has a

leading &, so it is an indirect assignment. For example:

```
)SET FRED = ETHEL  
)CM The following is the same as )SET ETHEL = 3  
)SET &FRED = 3
```

#### **RVAL\_CONSTANT\_ASSIGNMENT**

The right side (RVAL) of an assignment statement is a constant.

#### **RVAL\_VARIABLE\_ASSIGNMENT**

The right side (RVAL) of an assignment statement is a variable.

#### **LVAL\_NULL\_ASSIGNMENT**

The )SET or )SETF statement is a null (&Z) assignment. The LVAL variable has no leading &.

#### **LVAL\_INDIRECT\_NULL\_ASSIGNMENT**

The )SET or )SETF statement is a null (&Z) assignment. The LVAL is indirect (variable name has a leading &).

#### **REXX\_CALL\_CONSTANT\_REFERENCE**

A constant used in an in stream )REXX statement.

#### **REXX\_CALL\_VARIABLE\_REFERENCE**

A variable used in an in stream )REXX statement.

#### **REXX\_PROCEDURE**

An external procedure specified in a )REXX statement

#### **FUNCTION**

The listed variable is an ISPF skeleton built-in function.

## FieldOffsets

The FieldOffsets file is only generated for fixed format output. It contains the field names, types and offsets of the data in each file. This file can be used to generate load datasets for relational databases.

```

Table: Variables
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Variable           036 072 072 String      False
Skeleton           108 008 008 String      False
LineNumber         116 006 004 Int32      False
Position           122 005 002 Int16      False
Command            127 009 009 String      False
CommandCode        136 005 002 Int16      False
Type               141 030 030 String      False
TypeCode           171 005 002 Int16      False
Table: Skeletons
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Skeleton           036 008 008 String      False
SkelOffset         044 005 002 Int16      False
Childskeleton      049 025 025 String      False
ChildskelOffset    074 005 002 Int16      False
LineNumber         079 006 004 Int32      False
OPT                085 001 001 Boolean    False
NOFT               086 001 001 Boolean    False
EXT                087 001 001 Boolean    False
Table: Functions
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Function           036 008 008 String      False
Skeleton           044 008 008 String      False
LineNumber         052 006 004 Int32      False
Position           058 005 002 Int16      False
Argument           063 060 060 String      False
Table: Libraries
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Offset             036 005 002 Int16      False
UserTag            041 050 050 String      False
HostFileName       091 044 044 String      False
PCFileName         135 256 256 String      False
Table: Keywords
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
CommandCode        001 005 002 Int16      True
Command            006 020 020 String      False
Table: VarTypes
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
TypeCode           001 005 002 Int16      True
Type               006 030 030 String      False
Table: Configurations
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
ConfigNum          001 005 002 Int16      True
Configuration      006 030 030 String      False
Table: DOT
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Skeleton           036 008 008 String      False
TableName          044 030 030 String      False
LineNumber         074 006 004 Int32      False
Table: Programs
.....0.....0.....0.....0.....0.....0.....0.....0.....0.....0
Configuration      001 030 030 String      False
ConfigNum          031 005 002 Int16      False
Skeleton           036 008 008 String      False
ProgramName        044 030 030 String      False
LineNumber         074 006 004 Int32      False

```

The table names are only suggestions. They match the generated low level qualifier of the above specified output file names. The fields in the file are defined thus:

Column	Length / Type	Description
1	32 / character	Field Name
34	3 / numeric	Offset (first field is always at offset 1)
38	3 / numeric	Display length



Column	Length / Type	Description
42	3 / numeric	Internal length
46	15 / Character	Type Int16 = 2 byte integer Int32 = 4 byte integer String = variable length character string.
62	5	Key field indicator (True or False)

See the sample Rexx file ParserDB2.REX. This Regina Rexx program processes the above offsets file to create z/OS DB2 DDL and utility LOAD statements.

## <ExcelOutput>

```
<ExcelOutput>
  <ExcelOutputFolder>C:\myCSVOutputPath\</ExcelOutputFolder>
  <ExcelOutputHLQ>cProd</ExcelOutputHLQ>
</ExcelOutput>
```

Use the <ExcelOutput> element to define the characteristics of a set of output files consisting of comma separated values (CSV). These files can be loaded into a spreadsheet program, such as MS Excel. If this element is not defined, then no CSV files will be generated or written.

The ExcelOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## <ExcelOutputFolder>

Use this element to define the full path to the directory in which the files will be generated. At the time the parser is run this directory must exist or the parser will terminate with an error.

As noted above, the ExcelOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## <ExcelOutputHLQ>

Use this element to define the High Level Qualifier(HLQ) of the CSV output datasets. To this HLQ will be appended one of the output dataset types: Variables, Skeletons, Tables (DOT), Programs (skeleton to executed program cross reference), Functions or Libraries. The Keywords, VarTypes and FieldOffset files are not generated for CSV output. For the output folder and HLQ shown above, the following files would be generated in the directory [C:\myCSVOutputPath](#)

```
cProd.Variables.csv
cProd.Skeletons.csv
cProd.DOT.csv
cProd.Programs.csv
cProd.Functions.csv
cProd.Libraries.csv
```

As noted above, the ExcelOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## **<XMLOutput>**

```
<XMLOutput>
  <XMLOutputFolder>C:\myXMLOutputPath</XMLOutputFolder>
  <XMLOutputHLQ>xProd</XMLOutputHLQ>
</XMLOutput>
```

Use the <XMLOutput> element to define the characteristics of a set of output files consisting of XML tags for the parsed data. Some databases can be defined with XML file input. Newer languages can directly query data in XML files.

The XMLOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## **<XMLOutputFolder>**

Use this element to define the full path to the directory in which the files will be generated. At the time the parser is run, this directory must exist, or the parser will terminate with an error.

As noted above, the XMLOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## **<XMLOutputHLQ>**

Use this element to define the High Level Qualifier(HLQ) of the XML output datasets. To this HLQ will be appended one of the output dataset types: Variables, Skeletons, Tables (DOT), Programs (skeleton to executed program cross reference), Functions or Libraries. The Keywords, VarTypes and FieldOffset files are not generated for XML output. For the output folder and HLQ shown above, the following files would be generated in the directory [C:\myCSVOutputPath](#)

- xProd.Variables.xml
- xProd.Skeletons.xml
- xProd.DOT.xml
- xProd.Programs.xml
- xProd.Functions.xml
- xProd.Libraries.xml

As noted above, the XMLOutput related tags are ignored when the XML file is used as the control for the Graphical User Interface.

## <FixedOutput>

```
<FixedOutput>
  <Configuration>
    <ConfigurationName>MyConfigName</ConfigurationName>
    <ConfigurationNumber>1</ConfigurationNumber>
  </Configuration>

  <FixedOutputFolder>C:\myFixedFilesOutputPath</FixedOutputFolder>
  <FixedOutputHLQ>fProd</FixedOutputHLQ>
</FixedOutput>
```

Use this compound element to define the characteristics of a set of output files that can be used to load tables for a relational database such as DB2 or MS SQL Server.

Only the ConfigurationName and ConfigurationNumber tags are used when the XML file is used as the control for the Graphical User Interface. The FixedOutput related file and folder tags are ignored when the XML file is used as the control for the Graphical User Interface.

## <ConfigurationName> <ConfigurationNumber>

The reason for generating the fixed file input is create data to load relational database tables. The configuration name and number exist to allow output from different runs of the parser to be stored in the same tables. Use the Configuration data to retrieve the proper data for the concatenations you defined.

This might be clearer with an example.

If you have a test system with a test skeletons library and then a production skeletons concatenated following it, you can define a test configuration thus:

```
<Directory>
  <DirectoryName>C:\mypath\test</DirectoryName>
  <DirectoryLabel>Custom</DirectoryLabel>
  <DirectoryHostName>P.CUSTOM.SKELS</DirectoryHostName>
</Directory>
<Directory>
  <DirectoryName>C:\mypath\vendor</DirectoryName>
  <DirectoryLabel>Vendor</DirectoryLabel>
  <DirectoryHostName>P.VENDOR.SKELS</DirectoryHostName>
</Directory>
:
<FixedOutput>
  <Configuration>
    <ConfigurationName>Test</ConfigurationName>
    <ConfigurationNumber>2</ConfigurationNumber>
  </Configuration>

  <FixedOutputFolder>C:\myFixedFilesOutputPath</FixedOutputFolder>
  <FixedOutputHLQ>fTest</FixedOutputHLQ>
</FixedOutput>
```

With a production system that only uses the production library, the XML for the configuration would

look thus:

```
<Directory>
  <DirectoryName>C:\mypath\vendor</DirectoryName>
  <DirectoryLabel>Vendor</DirectoryLabel>
  <DirectoryHostName>P.VENDOR.SKELS</DirectoryHostName>
</Directory>
.
<FixedOutput>
  <Configuration>
    <ConfigurationName>Production</ConfigurationName>
    <ConfigurationNumber>1</ConfigurationNumber>
  </Configuration>

  <FixedOutputFolder>C:\myFixedFilesOutputPath</FixedOutputFolder>
  <FixedOutputHLQ>fProd</FixedOutputHLQ>
</FixedOutput>
```

Note that the test and production configurations share an output folder, but that the HLQs of *test* and *prod* keep the output files distinct.

### **<FixedOutputFolder>**

Use this element to define the full path to the directory in which the files will be generated. At the time the parser is run, this directory must exist, or the parser will terminate with an error.

As noted above, the FixedOutput related file and folder tags are ignored when the XML file is used as the control for the Graphical User Interface.

### **<FixedOutputHLQ>**

Use this element to define the High Level Qualifier(HLQ) of the fixed output datasets. To this HLQ will be appended one of the output dataset types: Variables, Skeletons, Tables (DOT), Programs (skeleton to executed program cross reference), Functions, Libraries, Keywords, VarTypes and FieldOffsets. For the output folder and HLQ shown above, the following files would be generated in the directory [C:\myFixedFilesOutputPath](#)

```
fProd.Variables.txt
fProd.Skeletons.txt
fProd.DOT.txt
fProd.Programs.txt
fProd.Functions.txt
fProd.Libraries.txt
fProd.Keywords.txt
fProd.VarTypes.txt
fProd.FileOffsets.txt
```

As noted above, the FixedOutput related file and folder tags are ignored when the XML file is used as the control for the Graphical User Interface.

### **<IgnoreSkel>**

You may specify zero, one or more skeleton names that are to be ignored in the parsing. Use one <IgnoreSkel> element per skeleton. Only specify the eight character skeleton name, not the extension.

The utility ignores file extensions.

## <Unimbedded Skeleton>

You may specify zero, one or more skeletons that are to be added to the Skeleton-to-Skeleton cross reference table. These skeletons are normally shown in the table as being )IM'd by “no parent”. You can change the parent to the name of a process or program, which would otherwise be unknown by the parser.

```
<Unimbedded Skeleton="CMN$$AUD" Parent="..AUDIT Program"/>
<Unimbedded Skeleton="CMN$$AUD" Parent="..AUDIT Program"/>
<Unimbedded Skeleton="CMNIMRPM" Parent="..PROMOTION"/>
<Unimbedded Skeleton="CMNIMPRM" Parent="..PROMOTION"/>
<Unimbedded Skeleton="CMN$$D2J" Parent="..DB2Jobs" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN$$OTH" Parent="LIBTYPE_OTHER" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN10" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN11" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN14" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN15" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN17" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN18" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN19" Parent="..Distribution" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN20" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN20I" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN21" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN24" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN25" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN28" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN29" Parent="..Install" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN30" Parent="..Baseline" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN30I" Parent="..Baseline" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN31T" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN31TI" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN32" Parent="..Baseline" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN34T" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN35T" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN37" Parent="..Baseline" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN38T" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN39T" Parent="..Temp" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN49" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN50" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN50I" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN50T" Parent="..TempBkout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN50TI" Parent="..TempBkout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN51" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN54" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN55" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN55I" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN55T" Parent="..TempBkout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN55TI" Parent="..TempBkout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN56" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN57" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN58" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN59" Parent="..Backout" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN64" Parent="..Revert" NOFT="false" OPT="false" EXT="false"/>
```

The Unimbedded element has five attributes: Skeleton, Parent, NOFT, OPT and EXT. The Skeleton is the name of a skeleton that is not )IM'd by any other skeleton in the concatenation. The Parent is the name of a process or dialog that uses the ISPF FTINCL to invoke the skeleton. You may specify up to 40 characters here. If longer strings are specified, they are truncated to 40 characters. NOFT, OPT and EXT are the file tailoring options for “no file tailoring”, “optional” and “extend ON”. These have either a *true* or a *false* value. The default is false.

The Unimbedded element must have at the minimum a Skeleton and a Parent attribute, or the XML configuration file will be rejected. If a duplicate entry occurs for with the same Skeleton and Parent values, only the first will be added to the cross reference table, the second will be ignored.

## <ImbedRecurseFail>

This is a true/false element. It affects the expansion function. Setting the value to *true* will cause an expansion to halt if a skeleton expansion encounters a previously expanded skeleton. The following

code fragments represent a valid run-time use of recursion:

**Skeleton A:**

```
...
)SET VAR = 1
)IM B
...
```

**Skeleton B:**

```
...
)SEL &VAR = 1
)IM C
)ENDSEL
...
```

**Skeleton C:**

```
...
)SET VAR = 2
)IM B
)ENDSEL
...
```

Because the )IM of skeleton C in skeleton B is conditional, this logic causes no problems at run time. The )IM statements work as follows:

$A \rightarrow B \rightarrow C \rightarrow B$  (C is not processed again due to the value of &VAR)

However, the expansion function in the skeleton analysis utility does not know the run-time values of variables. To the expansion utility, the )IM B in skeleton C could cause an infinite recursion:

$A \rightarrow B \rightarrow C \rightarrow \text{B} \rightarrow C \rightarrow B \rightarrow \dots$

<ImbedRecurseFail>true</ImbedRecurseFail> generates an error when the second )IM of skeleton B (marked red) is encountered.

If <ImbedRecurseFail>>false</ImbedRecurseFail> is specified, then the expansion continues, however the expansion will not include the second )IM of skeleton B. Therefore any logic relative to C imbedding B will not be evident.

## <DebugInputLine>

This is a true/false element. Setting the value to *true* will cause the input skeleton line to be written to the CSV and XML files. This element has no effect on the fixed format file output. Note that this is not needed, but may be requested as documentation if you find that one or more skeleton lines are not being parsed correctly.

## Creating Different Run Time Versions

When creating files that will be loaded into relational databases, the ConfigurationNumber or the ConfigurationName can be used to distinguish between the cross-reference data for different PDS concatenations. There must be one configuration XML file for each set PDS concatenation that you wish to define. You must change the ConfigurationNumber and/or ConfigurationName in each XML input file that you process.

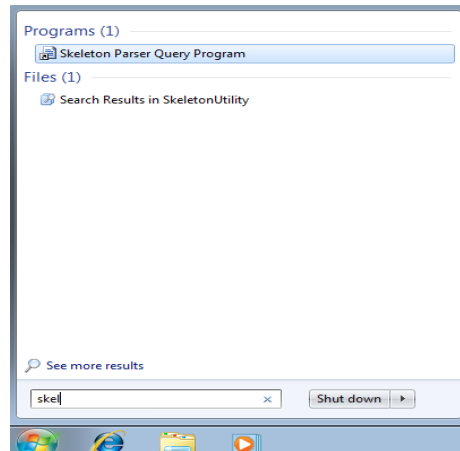
Sample batch .CMD files have been supplied showing how the utility is invoked. The Samples

directory also contains files written in REXX showing how to invoke the parser from REXX. You must have a REXX interpreter installed on your system to use these files. The samples were tested with Regina REXX, which is a free, open source implementation of a REXX interpreter.

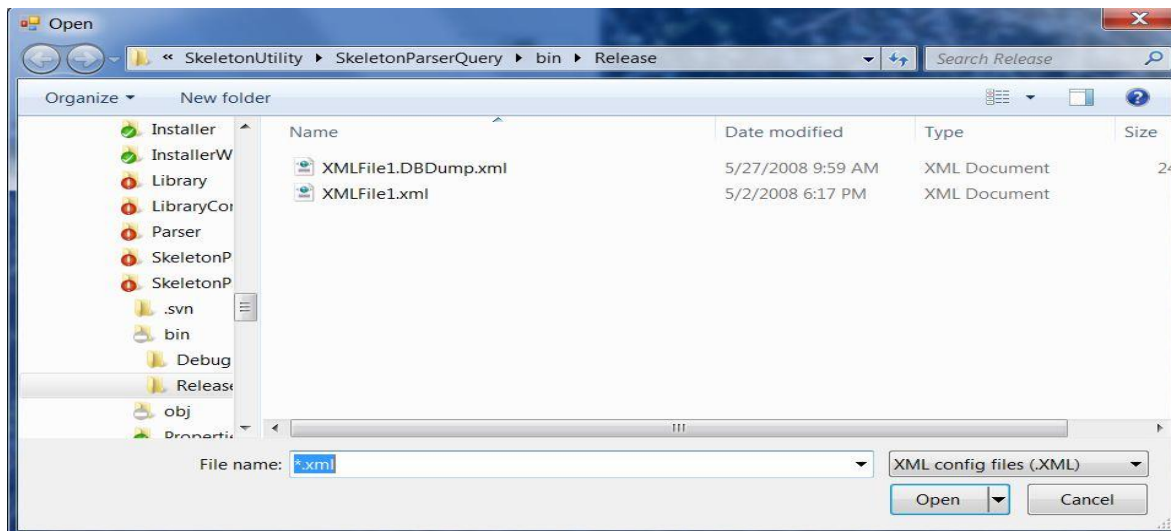
## The Graphical Front-End

The graphical front-end (GFE) can be used to pass an XML configuration file to the parser engine. This configuration file is not required to generate any of the output files (CSV, XML, fixed format) normally created in the batch run of the parser. The GFE invokes the parser engine and then uses the transient intermediate data objects created by the parser as relational data.

When the ISPF skeleton analyzer is installed, the GFE is added to the Start Menu's program items under the name “Skeleton Parser Query Program”:



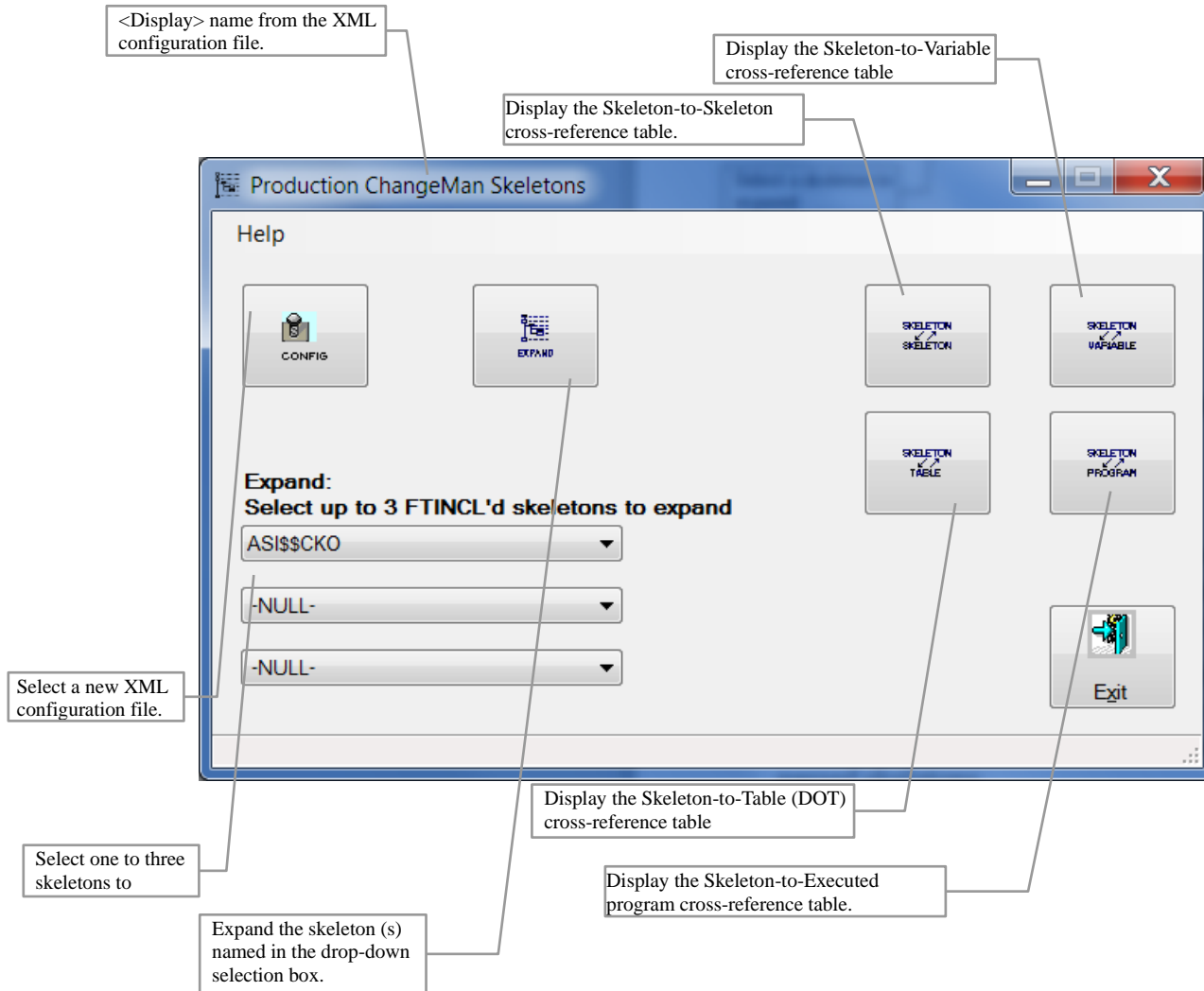
Each time you run the program, it will prompt for a configuration XML file:



The file you select is validated. Then the skeleton files in the folders specified in the <Directory> entries in the XML file are scanned. The cross-reference information is placed in transient tables, internal to the program.



Following the scan the query program's home dialog is displayed:



## Load a New Configuration



The *New Configuration* button allows you to browse for a new configuration file. The skeletons indicated by the new file will be scanned. Any prior data relationships will be deleted. The internal transient tables will be loaded with new data relationships.

If any of the queries (Skeleton-Skeleton, Skeleton-Variable, Skeleton-Table, Skeleton-Program or the Skeleton Expansion) are open, you will be prompted to either close them and load and parse skeletons from a new configuration or to cancel and return to using the current configuration with the already parsed skeletons.

# Skeleton to Variable Cross-Reference



The Skeleton-Variable button displays the Variable cross-reference Panel:

Production ChangeMan Skeletons - Variable Cross Reference

Export

Skeleton	Variable	Line	Cmd	Type
#PRINTX	1	53	SEL	CONSTANT_TEST
#PRINTX	2	79	SEL	CONSTANT_TEST
#PRINTX	B	36	SEL	CONSTANT_TEST
#PRINTX	B	68	SEL	CONSTANT_TEST
#PRINTX	CCLASS	74	DATALINE	VARIABLE_REFERENCE
#PRINTX	CCLASS	80	DATALINE	VARIABLE_REFERENCE
#PRINTX	COPT	53	SEL	VARIABLE_TEST
#PRINTX	COPT	79	SEL	VARIABLE_TEST
#PRINTX	CPGM	51	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPGM	69	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPGM	72	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPKGID	42	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPKGID	72	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPKGNO	42	DATALINE	VARIABLE_REFERENCE
#PRINTX	CPKGNO	72	DATALINE	VARIABLE_REFERENCE
#PRINTX	CSRCE	36	SEL	VARIABLE_TEST
#PRINTX	CSRCE	40	SEL	VARIABLE_TEST

Skeleton: -All- Variable: -All- Type: -All- Order By: [ ]

Filter [Filter Icon] [All Icon] [Return Icon]

Filters

Sorting

Filter Execution




Filter Discontinue

When the table is first displayed, all variables of all reference types in all skeletons are displayed. You can filter the results by selecting a particular skeleton and/or variable and/or reference type; then push the “Filter” button. Here are the results filtered by a specific skeleton:

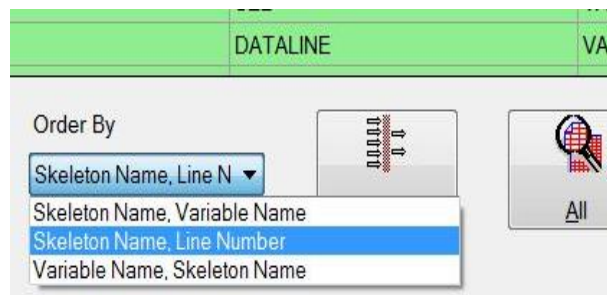
Production ChangeMan Skeletons - Variable Cross Reference

Export

Skeleton	Variable	Line	Cmd	Type
CICPHAIN	SCISCOPE	27	SET	LVAL_ASSIGNMENT
CICPHAIN	NEWC@ALL	27	SET	CONSTANT_REFERENCE
CICPHAIN	SCASS020	31	IF	VARIABLE_TEST
CICPHAIN	Y	31	IF	CONSTANT_TEST
CICPHAIN	NULL(&Z)	39	IF	VARIABLE_TEST
CICPHAIN	NXPRMLV	39	IF	VARIABLE_TEST
CICPHAIN	USER	41	DATALINE	VARIABLE_REFERENCE
CICPHAIN	USER	41	DATALINE	VARIABLE_REFERENCE
CICPHAIN	40	42	IF	CONSTANT_TEST
CICPHAIN	NXPRMLV	42	IF	VARIABLE_TEST
CICPHAIN	RMTSITE	52	IF	VARIABLE_TEST
CICPHAIN	RZ1	52	IF	CONSTANT_TEST
CICPHAIN	\$SPROJC	54	DATALINE	VARIABLE_REFERENCE
CICPHAIN	JNMPNUM	54	DATALINE	VARIABLE_REFERENCE
CICPHAIN	RMTSITE	58	IF	VARIABLE_TEST
CICPHAIN	RZ2	58	IF	CONSTANT_TEST
CICPHAIN	\$SPROJC	60	DATALINE	VARIABLE_REFERENCE

Skeleton: 
 Variable: 
 Type: 
 Order By: 
 Filter
  All
  Return

Note that the sort order was changed by selecting the Order By drop down and choosing “Skeleton Name, Line Number”. There are three ways the data can be sorted:



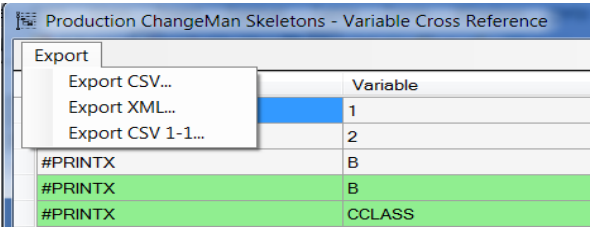
It is not possible to select multiple skeletons or variable names when filtering the data. But you can have multiple instances of the cross-reference dialog open simultaneously. Each can be separately filtered.

To discontinue filtering, simply press the “All” button.

### Displaying the Skeleton

Double click on any line in the Skeleton/Variable cross-reference grid to open the skeleton and position on the line number where the variable reference occurs.

# Exporting data



There are three data export functions. The first Export menu item on this form is **Export CSV...** Select the export to create a comma separated value file of the currently active data. This file can be loaded into a spreadsheet or database:

	A	B	C	D	E	F	G	H	I
1	Skeleton	Variable	Line Number	Position	Command(N)	Command(A)	Type(N)	Type(A)	Line Data
2	#PRINTX		1	53	1	15 SEL	15	CONSTANT_TEST	)SEL &COPT = 1
3	#PRINTX		2	79	1	15 SEL	15	CONSTANT_TEST	)SEL &COPT = 2
4	#PRINTX	B		36	1	15 SEL	15	CONSTANT_TEST	)SEL &CSRCE = B
5	#PRINTX	B		68	1	15 SEL	15	CONSTANT_TEST	)SEL &CSRCE = B
6	#PRINTX	CCLASS		74	1	120 DATALINE	5	VARIABLE_REFERENCE	//DDLST DD SYSOUT=&CCLASS,OUTPUT=* OUT
7	#PRINTX	CCLASS		80	1	120 DATALINE	5	VARIABLE_REFERENCE	//SYSUT2 DD SYSOUT=&CCLASS,OUTPUT=* OUT
8	#PRINTX	COPT		53	1	15 SEL	20	VARIABLE_TEST	)SEL &COPT = 1
9	#PRINTX	COPT		79	1	15 SEL	20	VARIABLE_TEST	)SEL &COPT = 2
10	#PRINTX	CPGM		51	1	120 DATALINE	5	VARIABLE_REFERENCE	&CPGM
11	#PRINTX	CPGM		69	1	120 DATALINE	5	VARIABLE_REFERENCE	PgmName &CPGM Baseline CMN.U0000.P0.RZ
12	#PRINTX	CPGM		72	1	120 DATALINE	5	VARIABLE_REFERENCE	PgmName &CPGM PackageID CMN.DIV.P0.&CI
13	#PRINTX	CPKGID		42	1	120 DATALINE	5	VARIABLE_REFERENCE	// INDSN(CMN.DIV.P0.&CPKGID.#&CPKGID
14	#PRINTX	CPKGID		72	1	120 DATALINE	5	VARIABLE_REFERENCE	PgmName &CPGM PackageID CMN.DIV.P0.&CI

The second Export menu item **Export XML...** is used to export the currently active data as an XML data file.

The last Export menu item **Export CSV 1-1...** is used to export a comma separated value file containing only the skeleton and variable names in a one-to-one relationship. This can allow you to document variable usage within a skeleton. Once you manually update the exported data with your comments, you should not export to that CSV again, or you will wipe out your updates.

# Skeleton to Skeleton Cross-Reference



The Skeleton-Skeleton button displays the Skeleton cross-reference panel:

Production ChangeMan Skeletons - Skeleton Cross Reference

Export CSV...

Skeleton	Offset	Line	Imbedded	ImbedOffset	EXT	NOFT	OPT
BNDCLRPM	1	9	CMN\$\$\$NM	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
BNDCLRPM	1	10	CMN\$\$\$PJ	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
BNDCLRPM	1	19	SKIRMLIB	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
BNDCLRPM	1	22	BNDCLPTA	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$017	1	7	SKM\$\$VAR	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$017	1	43	SKMAPPLV	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	7	SKM\$\$VAR	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	62	CMN00INS	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	131	SKM\$\$SLB	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	424	CMN00INS	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	442	CMN99INS	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	479	CICPHAIN	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$020	1	481	CICPHAI8	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$050	1	28	SKM\$\$SLB	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CASS\$050	1	50	SKM\$\$SLB	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CHU0751	1	2	CHU0750	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CHU0752	1	2	CHU0750	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CHU0852	1	15	SKM\$\$JLB	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Skeleton  
-All-

Filter

All

Return

Filter

Filter Execution

Filter Discontinue

When the table is first displayed, all skeletons that imbed other skeletons are shown along with the skeletons they imbed. You can filter the results by selecting a particular skeleton; then push the “Filter” button. Here are the results filtered by a specific skeleton, CMN\$\$MFS:

Production ChangeMan Skeletons - Skeleton Cross Reference

Export CSV...

	Skeleton	Offset	Line	Imbedded	ImbedOffset	EXT	NOFT	OPT
▶	CMN\$\$MFS	1	37	CMN\$\$SYC	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	CMN\$\$MFS	1	108	CMN\$\$SYC	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	CMNMFSGN	1	28	CMN\$\$MFS	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Skeleton  
CMN\$\$MFS

Filter

All

Return

Note that the filtered skeleton cross-reference shows all the skeletons that issue )IM for CMN\$\$MFS as well as those skeletons for which CMN\$\$MFS issues )IM's.

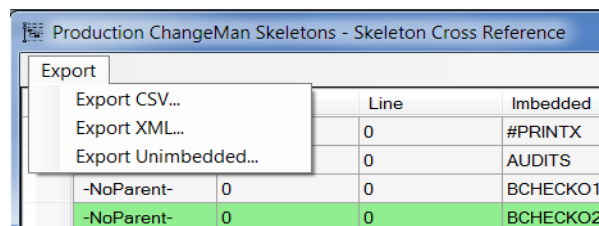
Columns:

- Skeleton** This is the imbedding (parent) skeleton.
- Offset** The PDS in the concatenation where the member is located.
- Line** The line in the parent skeleton where the )IM statement was found
- Imbedded** The imbedded (child) skeleton
- ImbedOffset** The PDS in the concatenation where the imbedded member is located. If the value is -1, then the imbedded skeleton does not exist in the concatenation.
- EXT** If the )IM statement contains the EXT (extensions) option, this box will be checked
- NOFT** If the )IM statement disables file tailoring (the NOFT option) for the imbedded skeleton, this box will be checked.
- OPT** If the )IM statement indicates that the imbedded skeleton is optional (that is no error will occur if the )IM'ed skeleton does not exist), then this box will be checked.

### ***Display the Skeleton***

Double click on any line in the Skeleton/Skeleton cross-reference grid to open the skeleton and position on the line number where the )IM reference occurs. For imbedded skeletons, the parent or imbedding skeleton will be displayed. For skeletons that are top level skeletons, the top level skeleton will be displayed. In the latter case, since there is no )IM statement involved, the display will be positioned at the first line of the skeleton

## Exporting data



There are three data export functions. The first Export menu item on this form is **Export CSV...** Select the export to create a comma separated value folder of the currently active data. This file can be loaded into a spreadsheet or database:

	A	B	C	D	E	F	G	H
1	Parent Skel	Concatenation Offset	Imbedded Skel	Concatenated Offset	Line Number	Optional	No File Tailoring	Extended
2	BNDICMRPM		1 CMN\$\$\$SNM	1	9			NOEXT
3	BNDICMRPM		1 CMN\$\$\$RPJ	1	10			NOEXT
4	BNDICMRPM		1 SKIRMLIB	1	19			NOEXT
5	BNDICMRPM		1 BNDICMPTA	1	22			NOEXT
6	CAS\$\$017		1 SKM\$\$\$VAR	1	7			NOEXT
7	CAS\$\$017		1 SKMAPPLV	1	43			NOEXT
8	CAS\$\$020		1 SKM\$\$\$VAR	1	7			NOEXT
9	CAS\$\$020		1 CMNNOINS	1	62			NOEXT
10	CAS\$\$020		1 SKM\$\$\$VAR	1	124			NOEXT

The second Export menu item **Export XML...** is used to export the currently active data as an XML data file.

The last Export menu item **Export Unimbedded** exports an set of XML data lines that can be copied into the configuration file, where you can define REXX, z/OS assembly language or a higher level language program or function as the parent for skeletons that are not imbedded by other skeletons, but rather by FTINCL from a program or dialog. The exported data looks like this:

```
<Unimbedded Skeleton="#PRINTX" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="AUDITS" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="BCHECKO1" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="BCHECKO2" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0016A" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0016B" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0018" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0019" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0300" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0751" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0752" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0852" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CHU0862" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
<Unimbedded Skeleton="CMN$$$RPJ" Parent="-NoParent-" NOFT="false" OPT="false" EXT="false"/>
```

# Skeleton to Table (DOT) cross-reference



The Skeleton-Table button displays the Table cross-reference panel:

Production ChangeMan Skeletons - )DOT Cross Reference

Export

Skeleton	Table	Line
BCHECKO1	BCHECKO2	7
BCHECKO1	BCHECKO1	10
BCHECKO2	BCHECKO1	10
BCHECKO2	BCHECKO1	22
BCHECKO2	BCHECKO1	32
BNDCMRPM	CMNDB2SS	6
CASS\$017	GND\$CTBL	22
CASS\$017	IMPINTBL	28
CASS\$020	GND\$CTBL	23
CASS\$020	IMPINTBL	29
CASS\$020	&RMLBTBL	167
CASS\$020	&STGLTYP.CPYTB	212
CASS\$020	&STGLTYP.DELTB	260
CASS\$020	&STGLTYP.CPYTB	315
CASS\$020	&STGLTYP.DELTB	361
CASS\$020	&STGLTYP.RENTB	670
CASS\$020	&RMLBTBL	
CHU0751	CHTYPTS	

Skeleton

-All-

Table

-All-

Filter

All

Return

Filter displayed data using the skeletonon drop down on the left.

Filters

Filter Execution

Filter Discontinue

When the cross-reference is first displayed, all skeletons and table loops (DOT) are shown. You can filter the results by selecting a particular skeleton and/or table; then push the “Filter” button. Here are the results filtered by a specific skeleton, CMN\$LNK:



Skeleton	Table	Line
CMNSSLNK	XPOLLTBL	50
CMNSSLNK	XPOLLTBL	60
CMNSSLNK	XPOLLTBL	83
CMNSSLNK	XPOLLTBL	93

Columns:

**Skeleton** This is the imbedding (parent) skeleton.

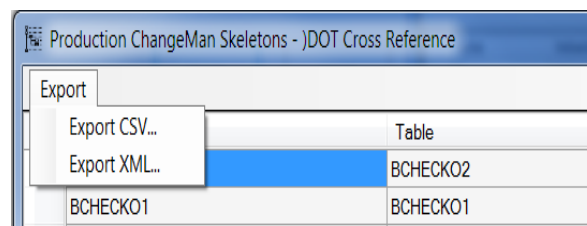
**Table** The table name. Note: Tables with names containing variables are also shown, if they exist.

**Line** The line in the skeleton where the )DOT statement was found

### ***Display the Skeleton***

Double click on any line in the Skeleton/ )DOT cross-reference grid to open the skeleton and position on the line number where the )DOT reference occurs.

### ***Exporting data***



There are two data export functions. The first Export menu item on this form is **Export CSV...** Select the export to create a comma separated value folder of the currently active data. This file can be loaded into a spreadsheet or database:

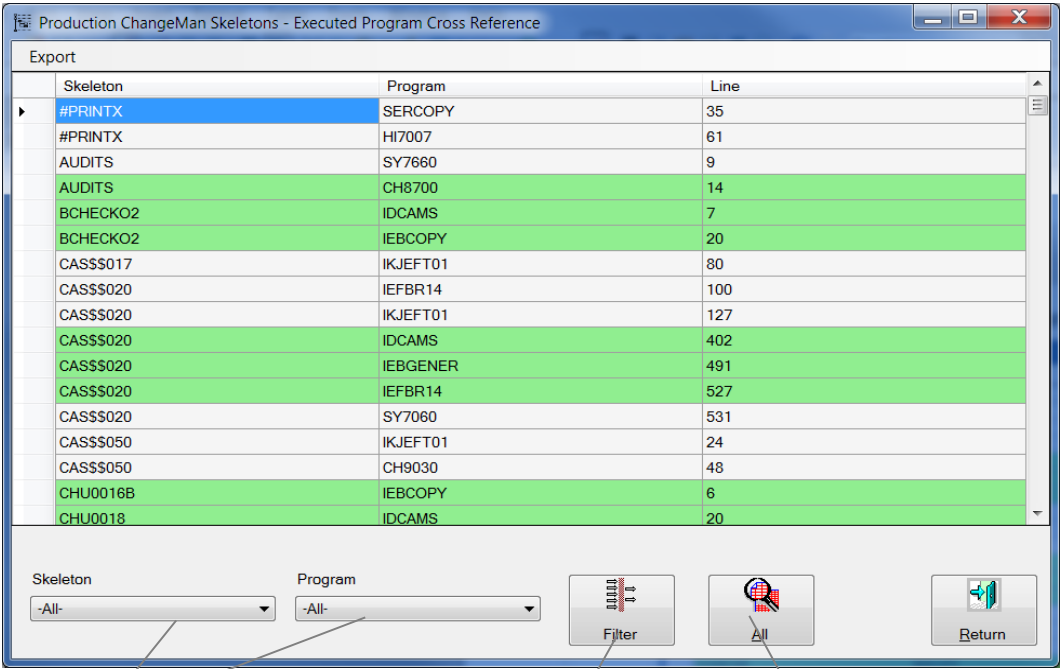
	A	B	C	D
1	Skeleton	Table	Line Number	
2	BCHECKO1	BCHECKO1	10	
3	BCHECKO1	BCHECKO2	7	
4	BCHECKO2	BCHECKO1	10	
5	BCHECKO2	BCHECKO1	22	
6	BCHECKO2	BCHECKO1	32	
7	BNDCMRPM	CMNDB2SS	6	
8	CAS\$\$017	GNDSTBL	22	
9	CAS\$\$017	IMPINTBL	28	
10	CAS\$\$020	&RMLBTBL	167	

The second Export menu item **Export XML...** is used to export the currently active data as an XML data file.

# Skeleton to Program Cross-Reference



The Skeleton-Program button displays the executed program cross-reference panel:



Filters

Filter Execution

Filter Discontinue

When the cross-reference is first displayed, all skeletons and executed program names are shown. You can filter the results by selecting a particular skeleton and/or program name; then push the “Filter” button. Here are the results filtered by a specific skeleton, CMN\$\$LNK:

Production ChangeMan Skeletons - Executed Program Cross Reference

Export		
Skeleton	Program	Line
CMNSSLNK	IEFBR14	9
CMNSSLNK	IEBCOPY	41
CMNSSLNK	HEWL	109
CMNSSLNK	CMNBAT90	228
CMNSSLNK	SERCOPY	265

Skeleton: CMNSSLNK Program: -All-

Filter All Return

Columns:

**Skeleton** This is the imbedding (parent) skeleton.

**Program** The executed program name. Note: Programs with names containing variables are also shown, if they exist.

**Line** The line in the skeleton where the )DOT statement was found

### Display the Skeleton

Double click on any line in the Skeleton/Executed Program cross-reference grid to open the skeleton and position on the line number where the EXEC PGM reference occurs.

### Exporting data

Production ChangeMan Skeletons - Executed Program Cross Reference

Export	
Skeleton	Program
CMNSSLNK	IEFBR14
CMNSSLNK	IEBCOPY
CMNSSLNK	HEWL

There are two data export functions. The first Export menu item on this form is **Export CSV...** Select the export to create a comma separated value folder of the currently active data. This file can be loaded into a spreadsheet or database:

A1				Skeleton
	A	B	C	D
1	Skeleton	Program	Line Number	Line Data
2	#PRINTX	HI7007	61//P02	EXEC PGM=HI7007,PARM='ISASIZE(200K),R'
3	#PRINTX	SERCOPY	35//DECOMP	EXEC PGM=SERCOPY, *** decompress component
4	AUDITS	CH8700	14//P02	EXEC PGM=CH8700,
5	AUDITS	SY7660	9//P00	EXEC PGM=SY7660,PARM='U'
6	BCHECK02	IDCAMS	7//P01	EXEC PGM=IDCAMS
7	BCHECK02	IEBCOPY	20//P02	EXEC PGM=IEBCOPY,REGION=1M
8	CAS\$\$017	IKJEFT01	80//P10	EXEC PGM=IKJEFT01,DYNAMNBR=20
9	CAS\$\$020	IDCAMS	402//P20	EXEC PGM=IDCAMS,DYNAMNBR=20,

The second Export menu item **Export XML...** is used to export the currently active data as an XML data file.

# Skeleton Expansion



The skeleton expansion function operates on the skeleton indicated in the three drop down selectors on the home form. These drop downs are processed as three separate, sequential FTINCL statements. The expansion processes each skeleton in order and then process each )IM statement in the currently processed FTINCL skeleton, adding the imbedded skeleton to the expansion. When an imbedded skeleton also has an )IM statement, the skeleton associated with that statement is processed.

The result is placed in a viewable, scrollable table:

Production ChangeMan Skeletons - Skeleton Expansion CMNIMRPM

Search

Skeleton	Offset	End Offset	Line Number	.....1.....2.....3.....4.....5.....6.....7..
SKM\$\$VAR	304	304	237	)SET \$SSTGINT = &Z
SKM\$\$VAR	305	308	238	)SEL &\$LN EQ COBOL2 OR &\$LN EQ COBOLE
SKM\$\$VAR	306	306	239	)SET \$SSTGINT = Y
SKM\$\$VAR	307	307	240	)SET \$SINTCMP = INC
SKM\$\$VAR	308	308	241	)ENDSEL &\$LN EQ COBOL2 OR &\$LN EQ COBOLE
SKM\$\$VAR	309	312	242	)SEL &LNGNAME EQ PLI
SKM\$\$VAR	310	310	243	)SET \$SSTGINT = Y
SKM\$\$VAR	311	311	244	)SET \$SINTCMP = INT
SKM\$\$VAR	312	312	245	)ENDSEL &LNGNAME EQ PLI
SKM\$\$VAR	313	313	246	
SKM\$\$VAR	314	314	247	)CM *****
SKM\$\$VAR	315	315	248	)CM *
SKM\$\$VAR	316	316	249	)CM * U U SSSS RRRRRR 000000 PPPP *
SKM\$\$VAR	317	317	250	)CM * U U SS R R O O P P *
SKM\$\$VAR	318	318	251	)CM * U U SS RRRRR O O PPP *
SKM\$\$VAR	319	319	252	)CM * U U SS R R O O P *
SKM\$\$VAR	320	320	253	)CM * UUUUUU SSSS R R 000000 P *
SKM\$\$VAR	321	321	254	)CM *
SKM\$\$VAR	322	322	255	)CM * TRANSLATE USER OPTIONS (USROPN) TO MEANINGFUL NAMES *
SKM\$\$VAR	323	323	256	)CM * USER OPTIONS 1-5 ARE SET IN CMN\$\$VAR, 6-20 HERE *
SKM\$\$VAR	324	324	257	)CM *
SKM\$\$VAR	325	325	258	)CM *
SKM\$\$VAR	326	326	259	)CM * USER OPTIONS:
SKM\$\$VAR	327	327	260	)CM * PL/I -----
SKM\$\$VAR	328	328	261	)CM * 01<=>ISA SIZE *
SKM\$\$VAR	329	329	262	)CM * 09<=>IT/ET-COMPILE *
SKM\$\$VAR	330	330	263	)CM * 02<=>IMS 03<=>CICS 10<=>DEC31.8 08<=>FREI 11<=>FREI *
SKM\$\$VAR	331	331	264	)CM * 12<=>STROBE 13<=>JOSKA 14<=>CICS/IMS-GEN *
SKM\$\$VAR	332	332	265	)CM * 04<=>ETB 05<=>CV1/2 06<=>MQM 15<=>PRT7 16<=>FREI *
SKM\$\$VAR	333	333	266	)CM * 17<=>\$AMODE 18<=>\$RMODE 19<=>RENT 20<=>REUSE 07<=>REPORTWR *
SKM\$\$VAR	334	334	267	)CM *

Skeleton logic leading to selected line

Offset	EndOffset	Skeleton	Line
306	306	SKM\$\$VAR	)SET \$SSTGINT = Y
39	2448	CMNIMRPM	)IM CMN\$\$JCD
60	2257	CMN\$\$JCD	)SEL &TYPERUN NE PROMOTE AND &TYPERUN NE DEMOTE
61	2256	CMN\$\$JCD	)IM SKMAPPLV
67	1086	SKMAPPLV	)IM SKM\$\$VAR
290	756	SKM\$\$VAR	)SEL &\$SCLMODE EQ INITIALIZE OR &\$SCLMODE EQ &Z
305	308	SKM\$\$VAR	)SEL &\$LN EQ COBOL2 OR &\$LN EQ COBOLE

Variables on the selected line

Variable	type
\$SSTGINT	LVAL_ASSIGNMENT
Y	CONSTANT_REFERENCE

Graphic View

Return

The selected (current) line.

The skeleton commands that were in effect to reach the selected line.

Variables and/or constants referenced on the highlighted line

## Columns:

**Skeleton** The skeleton currently being processed.

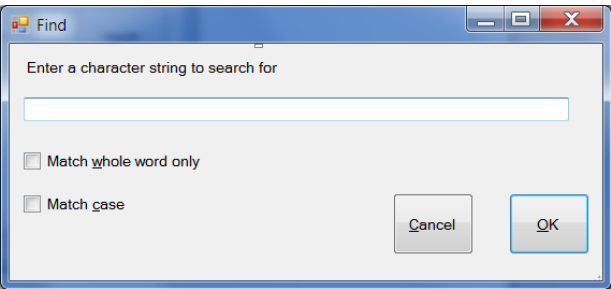
**Offset** The relative offset of this line from the start of the expansion

**LineNo** The offset of the line in the skeleton

This data is followed by the actual line from the skeleton.

### Expansion Main Menu

The main menu for this function allows you to search for character strings starting at the top line of the expansion. Select the Search menu item and submenu item Find or type Ctrl-F.



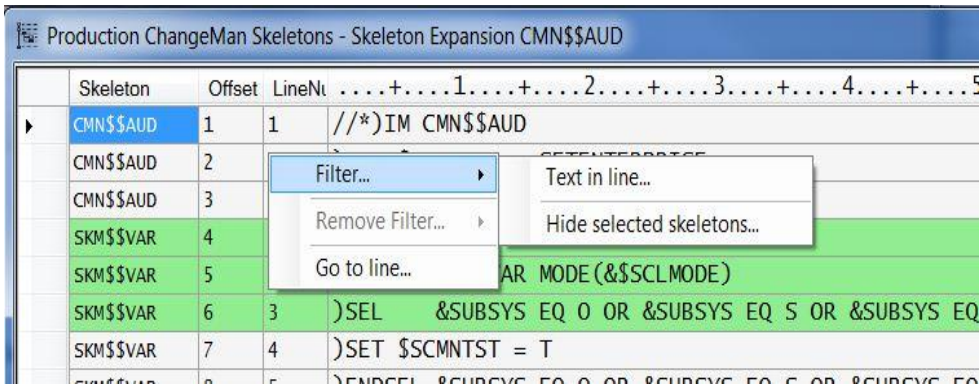
Type the character string you want to find. Choose the “Match whole word only” and/or “Match Case” options. Then press OK to do the search or Cancel to skip the search.

After the first line of text matching your search string is displayed, pressing the F3 function key will find the next occurrence of the string. Continue to press F3 until no more occurrences are found. After the “no more data” box is indicated, pressing F3 once more will reposition on the first line matching your search string.

### Expansion Context Menu

#### Filtering Text Within a skeleton line

The expansion grid has a context menu for filtering the displayed lines or for moving to a specific line in the expansion:



The context menu is activated by positioning the mouse pointer in the main expansion grid and clicking on the right mouse button.

The **Filter...Text in Line...** selection allows you to specify a character string to find:



Filtering for )IF finds all lines with )IF somewhere in the text. Then only those lines are displayed:

	Skeleton	Offset	End Offset	Line Number	.....+....1.....+....2.....+....3.....+....4.....+....5.....+....6.....
▶	SKM\$\$VAR	232	232	165	)IF &ZSYSNODE EQ RZ8 THEN
	SKM\$\$VAR	347	347	280	)IF &USROP09 EQ N THEN
	SKM\$\$VAR	353	353	286	)IF &USROP09 EQ N THEN
	SKM\$\$VAR	416	416	349	)IF &CMPTYPE EQ PLB OR &CMPTYPE EQ PLC THEN
	SKM\$\$VAR	418	418	351	)IF &CMPTYPE EQ PLD OR &CMPTYPE EQ PLI OR &CMPTYPE EQ PLO ...
	SKM\$\$VAR	1001	1001	934	)IF &\$FUNC EQ REMOTEPROMOTE AND &NXPRMM EQ STRICH THEN
	SKM\$\$VAR	1003	1003	936	)ELSE )IF &\$FUNC EQ REMOTEPROMOTE AND &NXPRMM EQ RZ8RP THEN
	SKMPAPPL	1102	1102	10	)IF &PROJECT EQ A02E OR &PROJECT EQ CIMM THEN
	SKMPAPPL	1104	1104	12	)IF &PROJECT EQ CRMM OR &PROJECT EQ EDUC OR &PROJECT EQ ZK...
	SKMPAPPL	1106	1106	14	)IF &PROJECT EQ CMAN OR &PROJECT EQ CMNR THEN
	SKMPAPPL	1120	1120	28	)IF &PROJECT EQ ELAB OR &PROJECT EQ ELAW THEN
	SKMPAPPL	1122	1122	30	)IF &PROJECT EQ RETI THEN
	SKM\$\$VAR	1326	1326	165	)IF &ZSYSNODE EQ RZ8 THEN
	SKM\$\$VAR	1441	1441	280	)IF &USROP09 EQ N THEN

Skeleton logic leading to selected line (repositioning disabled while filtering)					Variables on the selected line	
	Offset	EndOffset	Skeleton	Line	Variable	type
▶	232	232	SKM\$\$VAR	)IF &ZSYSNODE EQ RZ8 THEN	ZSYSNODE	VARIABLE_TEST
	39	2448	CMNIMRPM	)IM CMN\$\$JCD	RZ8	CONSTANT_TEST
	60	2257	CMN\$\$JCD	)SEL &TYPERUN NE PROMOTE AND &TYPERUN ...		
	61	2256	CMN\$\$JCD	)IM SKMAPPLV		
	67	1086	SKMAPPLV	)IM SKM\$\$VAR		
	85	269	SKM\$\$VAR	)SEL &\$SCLMODE EQ SETENTERPRISE		

Graphic View	Return
--------------	--------

## Using a Filter to Hide Skeleton Expansions

There is a second filter type, used to hide a skeleton in the expansion. One or more skeletons can be hidden to allow easier analysis of lines:

	Skeleton	Offset	End Offset	Line Number	.....+....1.....+....2.....+....3.....+....4.....+....
▶	CMN\$\$AUD	1	1	1	/(*)IM CMN\$\$AUD
	CMN\$\$AUD	2	2	2	)SET \$SCLMODE = SETENTERPRISE
	CMN\$\$AUD	3	1022	2	)IM SKM\$\$VAR
	SKM\$\$VAR	4	12	7	)SET \$SCMNTST = &Z
	SKM\$\$VAR	5	5	8	)ENDSEL &SUBSYS NE 0 AND &SUBSYS NE S AND &SUBSYS
	SKM\$\$VAR	6	8	9	)ENDSEL &\$SYSIN NE YES
	SKM\$\$VAR	7	7	10	)SET \$LN = &LNNAME
	SKM\$\$VAR	8	8		
	SKM\$\$VAR	9	11		
	SKM\$\$VAR	10	10		
	SKM\$\$VAR	11	11		
	SKM\$\$VAR	12	12		
	SKM\$\$VAR	13	13		

Skeleton name(s) to hide...

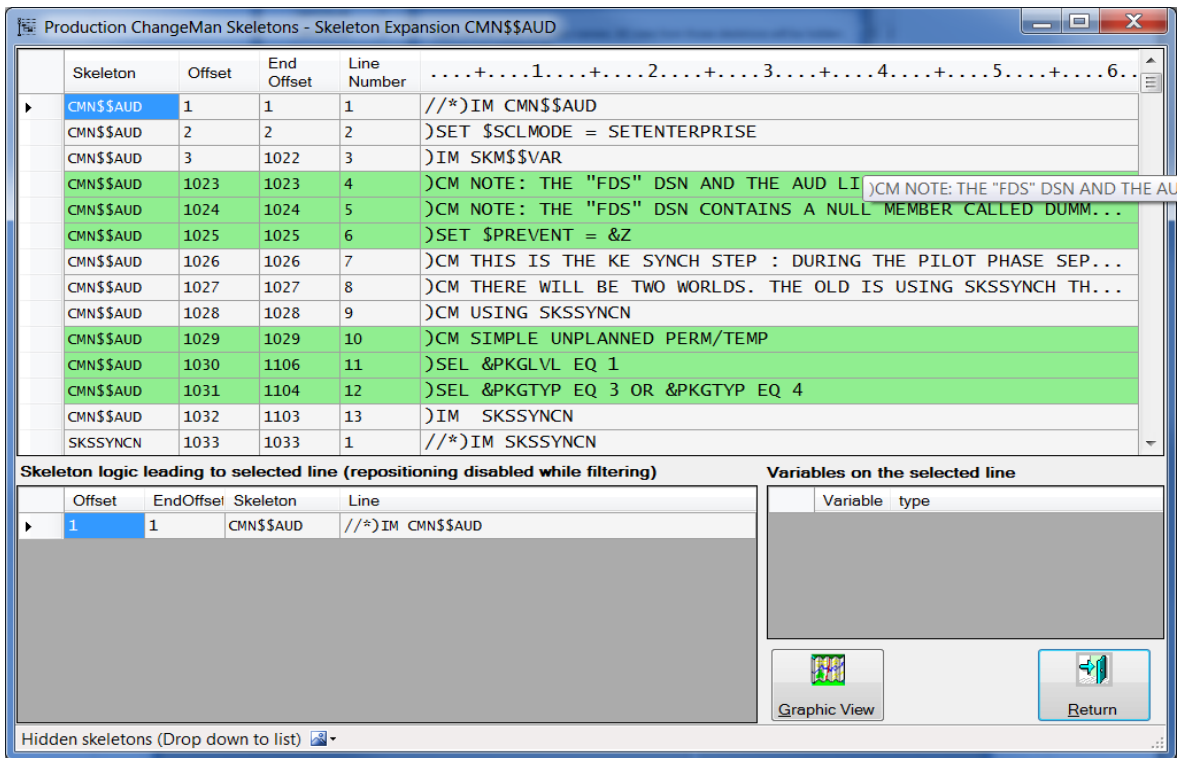
Type one or more skeleton names. All rows from those skeletons will be hidden.  
SKM\$\$VAR SKM\$\$XLB

OK Cancel

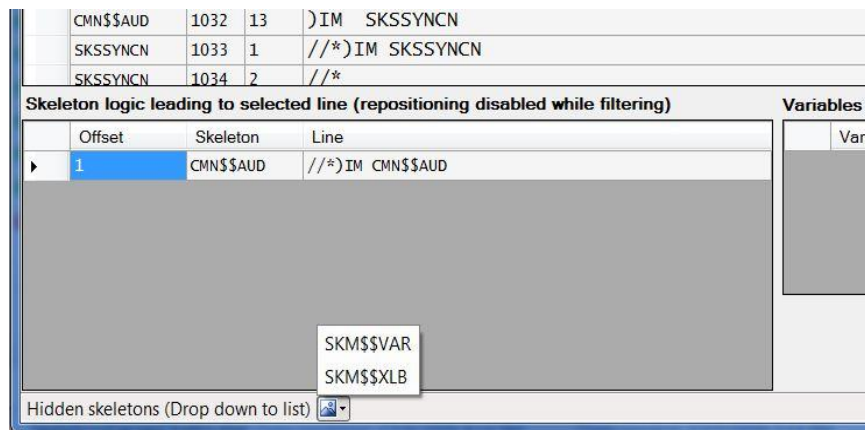
As shown, you may select multiple skeletons in a single request. You can also make multiple requests, hiding more skeletons.

The above filter has the result shown below. Note that after the )IM of SKM\$\$VAR on line 3, that 1019 lines of SKM\$\$VAR are suppressed. Also note that in the Status area at the bottom of the form, that there is now a drop down box titled “Hidden skeletons (Drop down to list)”.



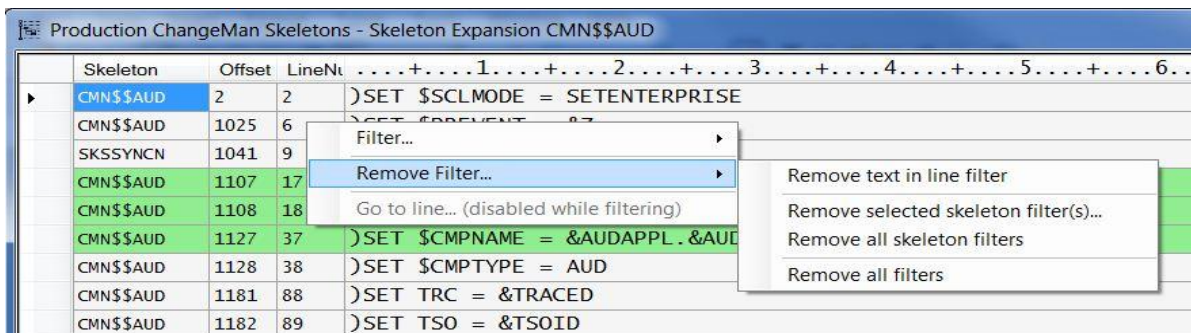


After the drop down is clicked, the list of hidden skeletons is shown.



## Removing Filters

Once any filter has been applied, the **Remove Filters...** context menu item is enabled:



You can remove the active text filter (in this case the filter is on “)SET”). You can also remove any or

all skeleton filters. The first skeleton hiding filter removal menu item is for picking one or more hidden skeletons to remove. The next skeleton hiding filter removal menu item redisplay all skeletons.

The “Remove all filters” selection redisplay all skeletons in the expansion, as well as removing the text filter.

There is one additional way to redisplay a hidden skeleton. Using the “Hidden skeletons” drop down list, you can select any of the skeletons in the list. This will redisplay the selected skeleton.

Positioning on a Specific Line (Goto)

You can view lines in the expansion grid by clicking on the Offset and End-Offset cells in the Skeleton Logic grid. The expansion grid will display the line referenced in the cell you clicked on. If possible, a few lines above the selected line will also be displayed, so that the line may be a few lines down. You may notice in some cases that the grid will be displayed with the indicated line at the top. You may also use the **Go to line...** context menu item to reposition the grid to a specified. Neither the Go to line... context menu item nor the Skeleton Logic Offset/End-Offset click will change the current row pointer.

Production ChangeMan Skeletons - Skeleton Expansion CMN\$\$RPM

Skeleton	Offset	End Offset	Line Number	.....1.....2.....3.....4.....
CMN\$\$ENQ	189	189	17	//&\$SENQDDN DD DISP=(MOD,DELETE),
CMN\$\$ENQ				SEL &\$SENQDDN NE &Z
CMN\$\$ENQ				DSN=CQ.&ENQLIB. ,
CMN\$\$ENQ				UNIT=SYSDA,SPACE=(CYL,(5,5))
CMN\$\$ENQ				
CMN\$\$ENQ				SORRY, I JUST DON'T SEE ANY USE FOR SYSUT4
CMN\$\$ENQ				HOWEVER, LEAVE WELL ENOUGH ALONE. IF USE
CMN\$\$ENQ	196	196	24	)CM THAN ONCE IN A SINGLE STEP IS DESIRED, SET
CMN\$\$ENQ	197	197	25	)CM NON NULL VALUE. THIS WILL EXCLUDE THE DDNA
CMN\$\$ENQ	198	198	26	)CM
CMN\$\$ENQ	199	201	27	)SEL &\$SENQDD4 EQ &Z
CMN\$\$ENQ	200	200	28	//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
CMN\$\$ENQ	201	201	29	)ENDSEL &\$SENQDD4 EQ &Z
CMN\$\$ENQ	202	202	30	)ENDSEL &STGDSNT NE L

Line number  
Type the line number to go to:  
172  
OK Cancel

Skeleton logic leading to selected line

Offset	EndOffset	Skeleton	Line
189	189	CMN\$\$ENQ	//&\$SENQDDN DD DISP=(MOD,DELETE),
164	261	CMN\$\$RPM	)SEL &RSTTTYP EQ IEBCOPY
168	251	CMN\$\$RPM	)SEL &PKGLIB NE &RMTLIB
172	205	CMN\$\$RPM	)IM CMN\$\$ENQ
184	202	CMN\$\$ENO	)SEL &STGDSNT NE L

Variables on the selected line

Variable	type
\$SENQDDN	VARIABLE_

Select a line in the list of commands active for the current line (189).

The current line in the expansion will remain 189 and the Skeleton Logic grid will remain unchanged.

Production ChangeMan Skeletons - Skeleton Expansion CMN\$\$SRPM

	Skeleton	Offset	End Offset	Line Number	.....1.....2.....3.....4.....5.....6.....
	CMN\$\$SRPM	172	205	144	)IM CMN\$\$ENQ
	CMN\$\$ENQ	173	173	1	/*)IM CMN\$\$ENQ
	CMN\$\$ENQ	174	174	2	)CM
	CMN\$\$ENQ	175	175	3	)CM ROUTINE TO SINGLE THREAD CHANGE MAN JOBS
	CMN\$\$ENQ	176	176	4	)CM VARIABLE ENQLIB MUST BE SET BEFORE IMBEDDING THIS SKE...
	CMN\$\$ENQ	177	177	5	
	CMN\$\$ENQ	178	178	6	)CM
	CMN\$\$ENQ	179	179	7	)CM TO ALLOW CMN\$\$ENQ TO BE USED MORE THAN ONCE IN A JO...
	CMN\$\$ENQ	180	180	8	)CM THE DDNAME MUST VARY. IF \$SENQDDN IS SET TO A VALUE,
	CMN\$\$ENQ	181	181	9	)CM USE IT AS A DDNAME, OTHERWISE USE THE DEFAULT
	CMN\$\$ENQ	182	182	10	)CM
	CMN\$\$ENQ	183	183	11	
	CMN\$\$ENQ	184	202	12	)SEL &STGDSNT NE L
	CMN\$\$ENQ	185	187	13	)SEL &\$SENQDDN EQ &Z

Skeleton logic leading to selected line

	Offset	EndOffset	Skeleton	Line
▶	189	189	CMN\$\$ENQ	/*&\$SENQDDN DD DISP=(MOD,DELETE),
	164	261	CMN\$\$SRPM	)SEL &RSTTYP EQ IEBCOPY
	168	251	CMN\$\$SRPM	)SEL &PKGLIB NE &RMTLIB
	172	205	CMN\$\$SRPM	)IM CMN\$\$ENQ
	184	202	CMN\$\$ENQ	)SEL &STGDSNT NE L
	188	190	CMN\$\$ENQ	)SEL &\$SENQDDN NE &Z

Variables on the selected line

	Variable	type
▶	\$SENQDDN	VARIABLE_REFERENCE

Graphic View

Return

Notice that the selected line is still line 189 and that the “Skeleton logic” area has not changed. But the **Go to line...** has positioned the grid so that the selected line is now displayed. In this example, you can return to line 189 either by using the **Go to line...** context menu item and either specifying the line number 189, or by leaving the line number blank, or by simply clicking in the Offset cell in the first line of the Skeleton Logic grid.. The **Go to line...** is not limited to the lines in the logic grid.

To make the **Go to line...** target line the current line, simply click on the line when it is displayed.

### Skeleton Logic Display For A Selected Line

When a line is selected by clicking on it the table on the lower left will show the ISPF Skeleton commands: )SEL/IF/ELSE/DO/DOT/IM that were in effect when this line was encountered. The table has the label *Skeleton logic leading to the selected line*. By clicking with the left mouse button in any line within this table the Expansion table view changes to show that line and two lines preceding and the lines that follow the selected line. Changing the view does not change the selected line, so the skeleton logic table remains unchanged. You can quickly return to the selected line by clicking on the top line in the skeleton logic table.

The left mouse button click functionality of the *Skeleton logic...* table is disabled when the line filter is activated in the expansion grid.

### Variables Display Within Lines

The table on the lower right will show the variables and constants referenced on that line. This table

has the label *Variables on the selected line*. If you double click a line in this table, a Skeleton-Variable cross-reference form will open showing all the skeletons that use the variable or constant:

8 )IM CMN\$\$VAR

1 )CM

2 /\*)IM CMN\$\$VAR

3 )CM

4 )CM ROUTINE TO PROCESS USER 0

5 )CM

6 )CM

7 )SET LISTNO = 0

8 )CM

9 )SET SAVFLKE = &STGLIKE

10 )SET SAVTLLT = &STGTLLT

11 )SET BAT90SEC = &Z

12 )SET BAT90THR = &Z

13 )SET CPYLIBA = NO

14 )CM

15 )CM

16 )CM TEST FOR USER OPTION 2 (D

17 )CM

18 )SET DLITCBL = &Z

19 )SET SSIOPT2 = &Z

20 )SEL &USROP02 EQ Y

21 )CM SET SSIOPT2 = DLITCBL

22 )ENDSEL &USROP02 EQ Y

23 )CM

24 )CM TEST FOR USER OPTION 3 (C

25 )CM

26 )SET CICSPC = &Z

27 )SEL &USROP03 EQ Y

- Variable Cross Reference for DLITCBL

Skeleton	Variable	Line	cmd	type
CMN\$\$VAR	DLITCBL	18	SET	LVAL_ASSIGNMENT
SKSSPSYL	DLITCBL	64	SEL	VARIABLE_TEST

Skeleton

CMN\$\$VAR

Type

All

Order By

Filter

All

Return

Variable	type
DLITCBL	LVAL_ASSIGNMENT
NULL (&Z)	RVAL_VARIABLE_ASSIGNMENT

**Graphic View**

On the expansion form there is a button labeled “Graphic View”. Press this button to show the relationships among all of the )IM'd skeletons in an expansion:

