CA 2E

Command Reference Guide

Release 8.7

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- YCPYMDLOBJ New Override Target Model Locks Parameter

  YCPYMDLOBJ (Copy Model Objects) Command (see page 138) - Added the description of the new OVRTGTLCK parameter.

- Refresh Action Diagram Statements
  - YCHKFUNACT Enhanced OPTION Parameter

    YCHKFUNACT (Check Function Action Diagram) Command (see page 99) - Updated the description of the OPTION parameter to note that it now refreshes built-in functions.

  - YPRCSFLSEL command

    YPRCSFLSEL (Process Subfile Selection) Command (see page 321) - Documented the command.

- YCVTSPLF Flexibility

  YDUPTKOBJ (Duplicate Toolkit Objects) (see page 269) - Documented the command.

- Allow SQL Record Level Access

  Access Path Generation Values (see page 81) and Model Values in Alphabetical Order (see page 70) - Added YSQLFMT in the tables.

- Allow RLA Access over DDL Database
  - Access Path Generation Values (see page 81)
  - SQLLIB (see page 161)
  - DBFGEN (see page 162)
- Allow SQL/DDL generation without hard-coded schema name
  - YEXCSQL (Execute SQL Statements) Command (see page 289)
- Meaningful Names for SQL/DDL
  - Access Path Generation Values (see page 81)
- Option to Generate RLA against DDL
  - Function Generation Values (see page 82)
  - Model Values in Alphabetical Order (see page 70)
- YDUPTKOBJ Missing Validation
  - TOLIB - YDUPTKOBJ (see page 269)
- Meaningful Names for SQL/DDL - Control Table vs Fields

Contents

Chapter 1: Introduction

This preface introduces the Command Reference Guide. It provides you with information on how the module is organized, identifies the contents, and lists the established conventions. Command Reference is part of a documentation set that provide instructions on how to use the CA 2E product (formerly known as Advantage 2E).

This section contains the following topics:

Purpose

This reference module describes CA 2E commands. The next chapter tells you how to execute a CA 2E command.

Organization

The information in this guide is organized for quick and easy access. All CA 2E commands are listed in alphabetical order. Each command is documented according to the i OS convention.

Contents

The Command Reference module includes an introductory chapter telling you how to execute a CA 2E command, the commands listed in alphabetical order, and an appendix grouping the commands by functional area. For each command there is:

- A summary of the function of the command.
- A command diagram.
- A description of each parameter and allowed values for the command.
- A list of notes regarding any restrictions or extra considerations.
- An example or examples using the command.

Acronyms Used in this Module

Descriptions of the acronyms used in this module are defined once, in this chapter. Thereafter, only the acronym is used.

(TH9)API	Application Program Interface
(TH9)CL	control language
(TH9)DRDA	Distributed Relational Database Architecture
(TH9)HLL	high level language
(TH9)NPT	non-programmable terminal
(TH9)PWS	programmable work station
(TH9)RDB	relational database
(TH9)SQL	Structured Query Language
(TH9)UIM	User Interface Manager
(TH9)VNM	valid name

Contacting Customer Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.

For telephone assistance, call:
U.S. and Canada 1-800-645-3042
International (1) 631-342-4683

Chapter 2: Executing Commands

This chapter describes how to execute a CA 2E command and how to determine which running environment is valid for each command. The following instructions provide information for executing a command interactively or submitting the command to batch.

This section contains the following topics:

Executing a CA 2E Command

You can execute CA 2E commands interactively or in batch.

Interactively

To execute a command interactively:

1. At an i OS command line, enter the command and press F4 to prompt. The command prompt entry displays
2. If there are additional parameters available for the command, the function key F10 appears at the bottom of your panel. Press F10 to use the additional values. You will occasionally be prompted for additional parameters if there is a dependency on values you enter.
3. If you are unsure of what value to enter in a field, enter "F4" or "?" to prompt for the allowed values of the field. Alternatively, you can press help to obtain online Help for the field.
4. Select the appropriate value. You will return to the previous panel with the value entered.
5. Press Enter when you are satisfied with the values you have entered. The command will now execute.

In Batch

To submit a command to batch:

At the command prompt, enter "SBMJOB" and press F4. Enter the command in the Command to run (CMD) parameter. Then follow the above instructions, steps 2 through 4. Press Enter on the SBMJOB panel to submit your job to batch.

Determining a Command's Running Environment

A box in the lower right corner of each command diagram contains an entry code that indicates the environment in which the command can be run. The entry code contains the following symbols:

- **Job**—Indicates that the command can be run independently as a separate function, in other words, outside a compiled CL program.
- **Pgm**—Indicates that the command can be included in a compiled CL program.
- **B**—Indicates that the command can be run in batch.
- **I**—Indicates that the command can be run interactively.

The following table shows the possible combinations and their meanings.

code	Meaning
Job: B	The command can be run in batch as a separate function.
Job: I	The command can be run interactively as a separate function.
Job: B,I	The command can be run either in batch or interactively as a separate function.
Pgm: B	The command can be run as part of a compiled CL program that is called in batch.
Pgm: I	The command can be run as part of a compiled CL program that is called interactively.
Pgm: B,I	The command can be run as part of a compiled CL program that is called either in batch or interactively.

Chapter 3: Commands (YADDMDLLE - YCHKMDLOBJ)

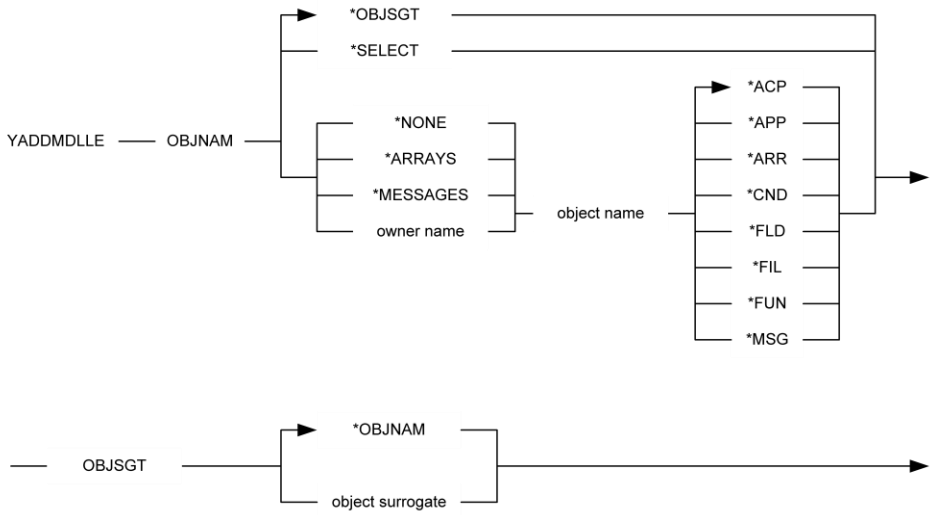
This chapter contains details for CA 2E commands YADDMDLLE through YCHKMDLOBJ. These commands appear in alphabetical order and include descriptions of their functions, parameters and allowed values, notes, and examples. Each command is also accompanied by a command diagram.

This section contains the following topics:

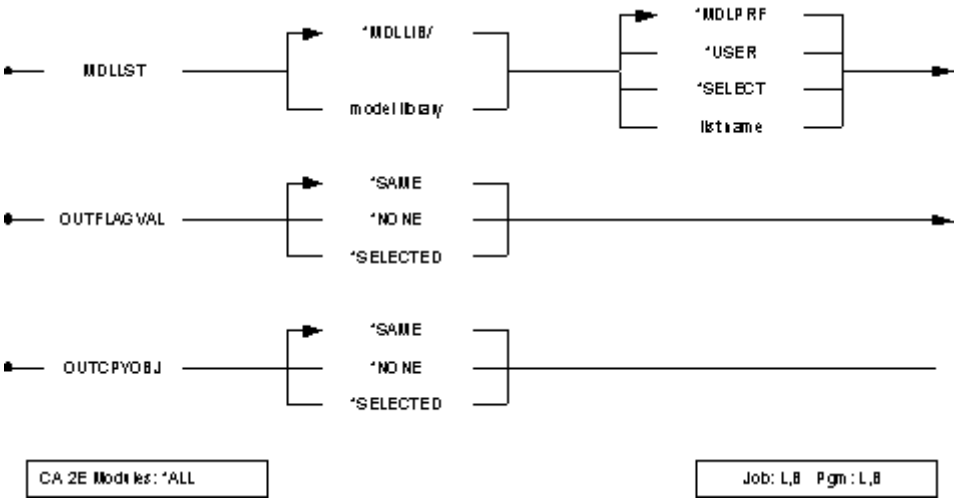
YADDMDLLE (Add a Model List Entry) Command

This command allows a user to add a single model object list entry. The list may or may not exist. If it does not exist, it is created by this command.

Required



Optional



Parameters

The following are parameters for the YADDMDLLE command.

OBJNAM

The object name to be added. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate is used to identify the model object to be added.
- ***SELECT**—Single value indicating that the object to be added is selected using an interactive display function.
- **Object owner name**—The character name of the object that owns the object to be added. Thus, for a function, the owning file would be entered.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **Object name**—The character name of the object to be added.
- **Object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- *MSG—Object is of type message.

OBJSGT

Unique number identifier of the model object that is added. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the model object to be added.
- **Object surrogate**—The surrogate number of the model object.

MDLLST

The qualified name of the model object list to which the entry is added. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The name of the list to be added.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the model library.

OUTFLAGVAL

This parameter specifies the initial value to be placed in the object selected flag associated with each list entry. Values for this parameter are described in the following:

- ***SAME**—(default) No flag value is to be used. New entries are written with the flag indicating that the entry is not selected. There is no change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing entries are flagged as selected. This flag can be used by other list commands when selecting list entries.

OUTCPYOBJ

This parameter specifies the initial value to be placed in the copy object flag associated with each list entry. This flag is used by the Copy Model Object Command (YCPYMDLOBJ) when selecting objects to copy to a target model. Values for this parameter are described in the following:

- ***SAME** —(default) No flag value is to be used. New entries are written with the flag indicating that the entry is not selected. There is no change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing entries are flagged as selected.

Notes

- A value other than *MDLLIB for MDLLST can result in the library list being changed during execution of this command. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- Model objects can either be identified by object name (OBJNAM) or by object surrogate key number (OBJSGT). If the OBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).
- The object to be added to the target list must exist in the model.

Example

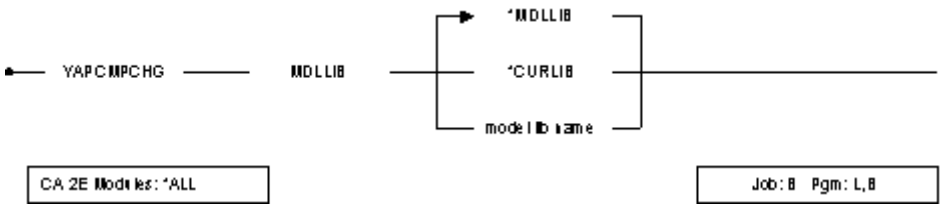
To add the Display Product Details function to model object list WRKLST, and to have the new entry flagged as selected:

```
YADDMDLLE OBJNAM('Product' 'Display Product + Details' *FUN)
 MDLLST(*MDLLIB/WRKLST) + OUTFLAGVAL(*SELECTED)
```

YAPYCMPCHG (Apply Component Changes) Command

Component change processing is the process by which a change to a particular object is reflected in the model by the propagation of that change throughout the users of the changed object. The object that is changed can be a component of many other objects in the model, and this process attempts to make the repercussions visible. This command can be used to invoke this process in batch for all changed objects which were not processed interactively.

Required



Parameters

The following are parameters for the YAPYCMPCHG command.

MDLLIB

The name of the model library in which component changes are applied. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value meaning that the model library is the first model found in the current job's library list.
- ***CURLIB**—Special value meaning that the model library is the current library for the job.
- **model library name**—The name of the model.

Notes

- Processing involves the examination of all objects in the model that have been changed but that have not had component change processing applied. The using objects of each of these objects are expanded and the change type associated with the changed object is propagated throughout them. The changed object is then updated to show that component change processing has been applied.
- Processing includes a call to an exit program for each using object encountered. The details of the call are as follows:

```
Program      :     YCMPCHGR1C

Interface    :     Return code          :   7 characters
                   Original surrogate   :   7 decimals
                   Object surrogate     :   7 decimals
                   Change type          :   3 characters (GEN or EDT)
```

  Developers can use the program to include any additional processing they desire. For example, it can be used to update an external database tracking changes in the model, or to place an entry onto a model list for later resolution of the component change that has happened to a given object.

  Additional information is included in the source for the exit program which is shipped in file QCLSRC in library Y2SYSRC.

Example

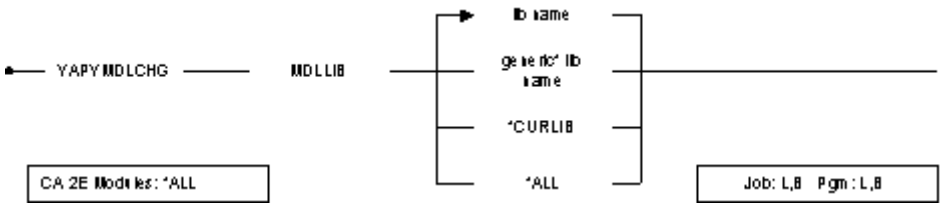
To apply component changes in the model library that is the highest model in the current library list:

YAPYCMPCHG MDLLIB(*MDLLIB)

YAPYMDLCHG (Apply Model Changes) Command

Upgrades a design model or models with any changes required by a new release of CA 2E. New releases of CA 2E can include modifications that must be applied to each design model before the new release of CA 2E can be used with that model. This command applies any such changes to a named model.

Required



Parameters

The following are parameters for the YAPYMDLCHG command.

MDLLIB

Name of library containing a design model to which the changes are applied. A generic name is allowed; for example, AB*. Values for this parameter are described in the following:

- ***ALL**—All models on the machine are upgraded. (Models which have already been upgraded are ignored).
- ***CURLIB**—Use current library for invoking job.

Notes

- You should make a backup of the design model before running this command.
- You must have object existence rights to the model to be able to run this command; that is, to all the objects in the model library.
- The current release level of the design model is indicated by a data area YMDLLVLRFA, a copy of which resides in each model library. You can see the value on the model values display.
- The current release level of the design model expected by CA 2E is indicated by a data area YMDLLVLSYA in the product library. You can see the value on the system parameter display.
- The YAPYMDLCHG command is generally cumulative. See your installation instructions regarding any restrictions. If you have an old model that you have archived to cartridge or tape and then restored, you can use the YAPYMDLCHG command to bring the model up to date.
- The YAPYMDLCHG command invokes the command Apply System Data to Model **(**YAPYSYSMDL**)** if necessary.
- Once a model has been upgraded to a new level using the YAPYMDLCHG command, it cannot be converted downwards again.
- If this command fails, you should resolve the problem with your product support representative, then restore the backup copy of your model from tape and start again.
- If all of the prior level files are not deleted by YAPYMDLCHG, an error message is issued at the end of the conversion. These files will have been renamed to start with Z. The usual cause of this problem is user logical files that are built over design model physical files. The model is properly converted, but you must manually delete these Z files.

Example

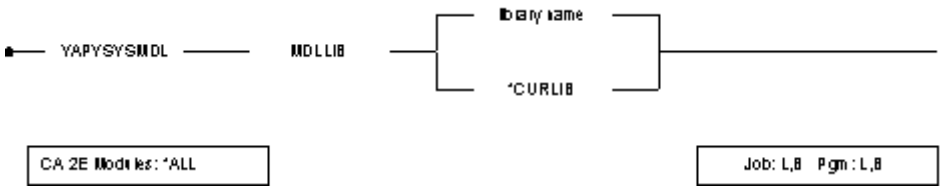
To update the model MYMDL:

YAPYMDLCHG MDLLIB(MYMDL)

YAPYSYSMDL (Apply System Model Data) Command

Updates the system data part of a CA 2E model. Each design model contains, apart from user-defined objects, a number of the shipped system objects, such as standard program functions and default message functions. From time to time updates are issued to the system objects. This command is for applying the updates to your own models.

Required



Parameters

The following are parameters for the YAPYSYSMDL command.

MDLLIB

Name of library containing a design model. The system object part of this model is to be updated. The value for this parameter is described in the following:

- ***CURLIB**—Use current library for invoking job.

Notes

- The current release level of the system objects in a model is indicated by a data area YMDLNBRRFA, a copy of which resides in each model library. You can see the current value on the model values display.
- The current release level of the system objects required by CA 2E is indicated by the data area YMDLNBRSYA in the product library. The current value can be seen on the model values display.
- You automatically invoke this command by using the command Apply Model Changes **(**YAPYMDLCHG**)**

Example

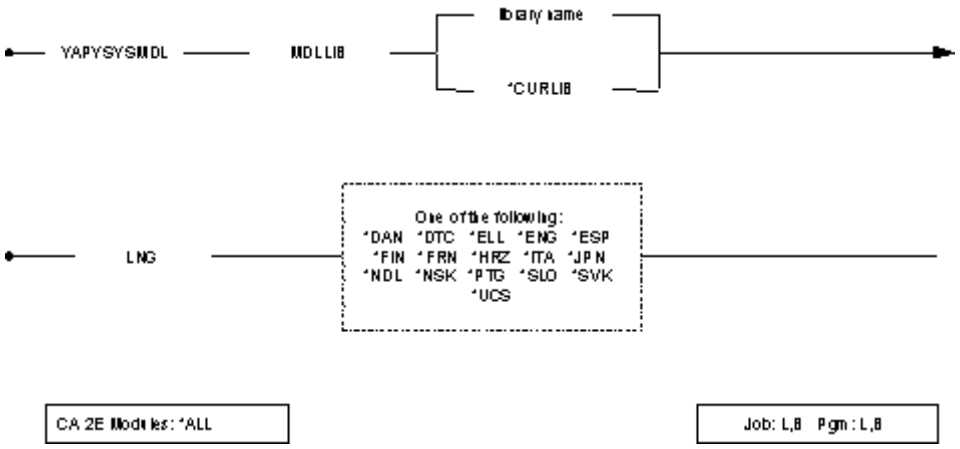
To update the system objects in model MYMDL:

YAPYSYSMDL MDLLIB(MYMDL)

YAPYTRNMDL (Apply Translation to Model) Command

Transposes data from a national language library into the system portion of a design model. This command translates the data in the CA 2E shipped files that are created by the Create Model Library command **(**YCRTMDLLIB**)**, such as *Standard header/footer, *Built-in functions and *Program data.

Required



Parameters

The following are parameters for the YAPYTRNMDL command

MDLLIB

Name of library containing a design model to which the translations are applied. The value for this parameter is described in the following:

- ***CURLIB**—Use current library for invoking job.

LNG

National language used for translation. Values for this parameter are described in the following:

- **DAN**—Danish
- **DTC**—German
- **ELL**—Greek
- **ENG**—English
- **ESP**—Spanish
- **FIN**—Finnish
- **FRN**—French
- **HRZ**—Croatian
- **ITA**—Italian
- **JPN**—Japanese
- **NDL**—Dutch
- **NSK**—Norwegian
- **PTG**—Portuguese
- **SLO**—Slovenian
- **SVK**—Swedish
- **UCS**—English Upper Case

Notes

- You must have the appropriate national language product library installed on your machine in order for the command to function.
- You must have object existence rights to all the objects in the model library to be able to run this command.
- The current national language of the data model is shown on the system parameter display (YDSPMDLVAL).
- The release level of the data model and the national language product level must be compatible.

Example

To translate system data files in the model MYMDL into French:

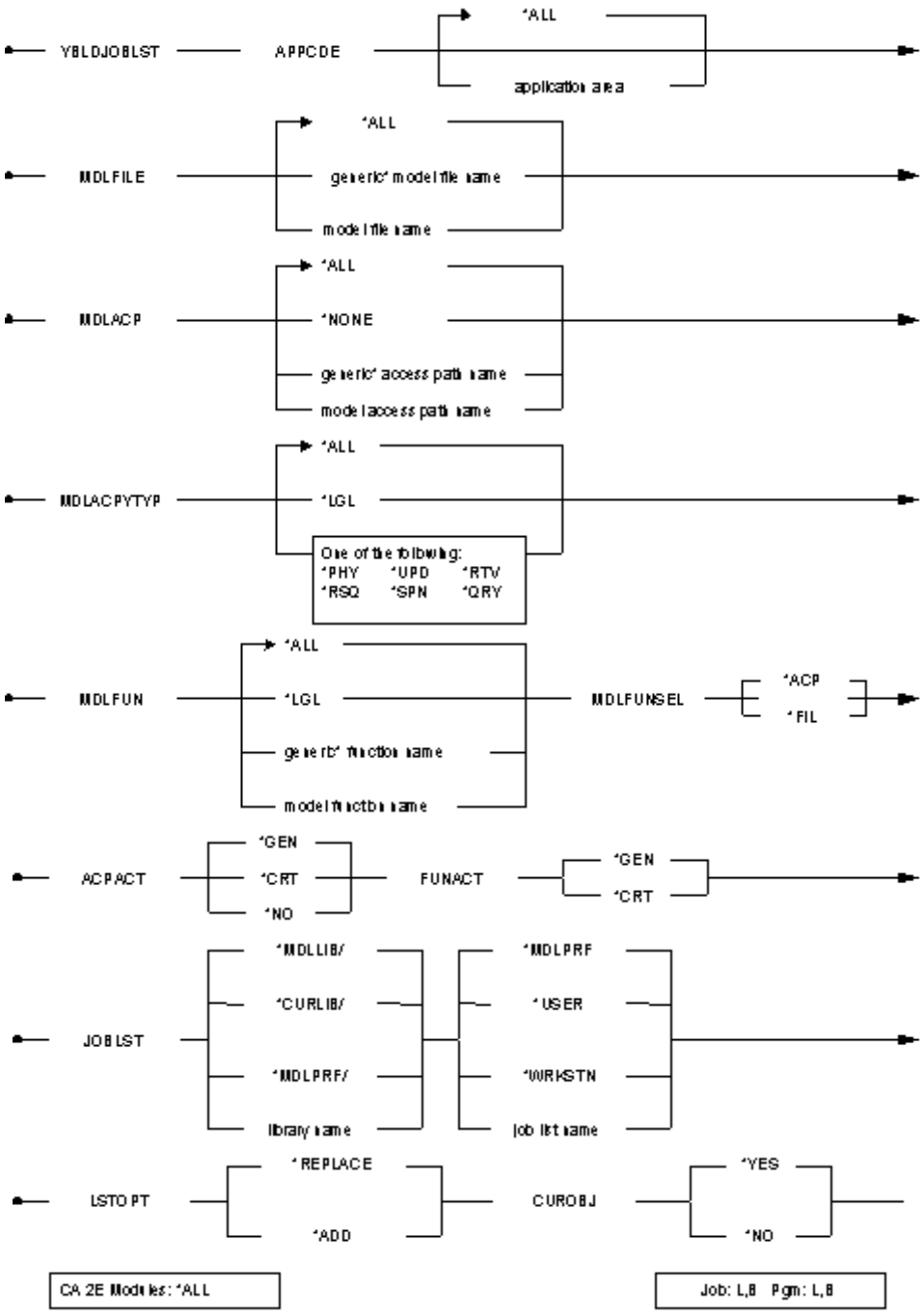
```
YAPYTRNMDL MDLLIB(MYMDL) LNG(*FRN)
```

YBLDJOBLST (Build Job List) Command

Builds a job list of selected objects in a design model. You can use this list in the following commands:

- YSBMMDLCRT submits the job list to be generated and/or compiled.
- YCHKJOBLE checks the existence of corresponding source members and i OS objects for each job list entry.
- YCVTJOBLST converts the job list to a CA 2E Toolkit object list.

Optional



Parameters

The following are parameters for the YBLDJOBLST command.

APPCDE

Name of the application area from which the objects to be included in the job list are selected. The value for this parameter is as follows:

- ***ALL**—(default) Include functions and access paths from all application areas.

MDLFILE

Generic name of files whose dependent access paths and/or functions are included in the job list. Values for this parameter are described in the following:

- ***ALL**—(default) Include the objects dependent on all user-defined files in the model.
- ***GENERIC**—Use generic name.

MDLACP

Generic name of access paths that are included in the job list. Values for this parameter are described in the following:

- ***ALL**—(default) All access paths for the specified files are included.
- ***NONE**—No access paths are included.
- ***GENERIC**—Use generic name.

MDLACPTYP

Specifies the type or types of access paths you want to include in the job list. Values for this parameter are described in the following:

- ***ALL**—(default) Include all access path types.
- ***LGL**—Include all types of access paths other than PHY.
- ***PHY**—Include only physical access paths.
- ***UPD**—Include only update access paths.
- ***RTV**—Include only retrieval access paths.
- ***RSQ**—Include only resequence access paths.
- ***QRY**—Include only query access paths.
- ***SPN**—Include only span access paths.

MDLFUN

Generic name of functions included in the job list. Values for this parameter are described in the following:

- ***ALL**—(default) Include all functions for the specified files or access paths.
- ***NONE**—Do not include any functions.
- ***GENERIC**—Use generic name.

MDLFUNSEL

Specifies whether the functions you want to include in the job list are those based on the selected files or those based on the selected access paths. Values for this parameter are described in the following:

- ***ACP**—(default) Select functions based on the access paths specified by the MDLACP and MDLACPTYP parameters.
- ***FIL**—Select functions based on the files specified by the MDLFIL parameter.

ACPACT

Activity to be requested for job list entries for access paths. Values for this parameter are described in the following:

- ***GEN**—(default) Generate and compile.
- ***CRT**—Compile only.
- ***NO**—Do not add to job list, only use to select functions.

FUNACT

Activity to be requested for job list entries for functions. Values for this parameter are described in the following:

- ***GEN**—(default) Generate and compile.
- ***CRT**—Compile only.

JOBLST

Qualified name of a job list where you place entries. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The job list name is retrieved from the model profile details for the current user.
- ***MDLLIB/*USER**—Default the job list name to user profile name, and store list in model library.
- ***WRKSTN**—Default the job list name to job name of invoking job.
- ***MDLPRF/**—The job list library is retrieved from the model profile details for the current user.

LSTOPT

List replacement option. Values for this parameter are described in the following:

- ***REPLACE**—(default) Create a new list, replacing any previous list's contents.
- ***ADD**—Add to any existing list's contents.

CUROBJ

Controls whether only current objects are included in the output list. Values for this parameter are described in the following:

- ***YES**—(default) Non-current objects are ignored in the build.
- ***NO**—All objects are included in the build.

Notes

- If a value of *NONE is specified for the MDLACP parameter, then a value of *ACP is not allowed for the MDLFUNSEL parameter.
- If a value of *NONE is specified for the MDLACP parameter, then a value of *NONE is not allowed for the MDLFUN parameter.
- For each access path or function selected, the YBLDJOBLST command will add entries to the job list for the necessary implementation objects. For example, for an interactive function, both the program and the display file will be added to the job list.
- The library specified for the job list must be the same as the model library.
- A completion message is returned, giving a count of the number of access paths and functions added to the list.
- Only external functions based on a specified file/access path are selected. If an internal function is based on the specified file/access path, the external functions which use the internal function will not be selected (unless they also are based on the specified file/access path).
- File to file dependencies are not handled by this command. For example, if an order detail file is owned by an order file and the order file is changed, then the access paths/functions based on the order detail file will only be selected by YBLDJOBLST if the order detail file is specified in the MDLFILE parameter.
- When you to try to build a job list in a model and the list of selected access paths contain one or more *DDL-based access paths which meets the DDL limitations; such access paths are not added to the job list.
- The current implementation of the DDL generation mode is not valid for the following cases:
  - Access paths that have virtual fields
  - SPN access path
  - QRY access path
  - Multi-member files

  Workaround for Virtual Fields, SPN, and QRY Access Paths: If the earlier generation mode is *DDS, revert to it and regenerate the access path. You need not regenerate the functions that use this access path. If you want to have an SQL type database, regenerate the access path using *SQL generation mode. The functions using this access path must be regenerated.

  Workaround for Multi-Member Files: If you want to have more than one member for the access paths, revert to *DDS generation mode.

Note: If you want to change an access path, which is previously defined as *DDS with a MAXMBR compiler override, to *DDL, you must revert to *DDS generation mode and must remove the compiler override, and then change back to *DDL generation mode.

Example

To build a job list of all access paths and functions in a design model:

```
YBLDJ0BLST
```

To build a job list of all functions in a design model over all files and access paths, but omitting the access paths from the job list:

```
YBLDJ0BLST
```

To build a job list of all logical access paths and functions in a design model which are based on files whose names start with "Order":

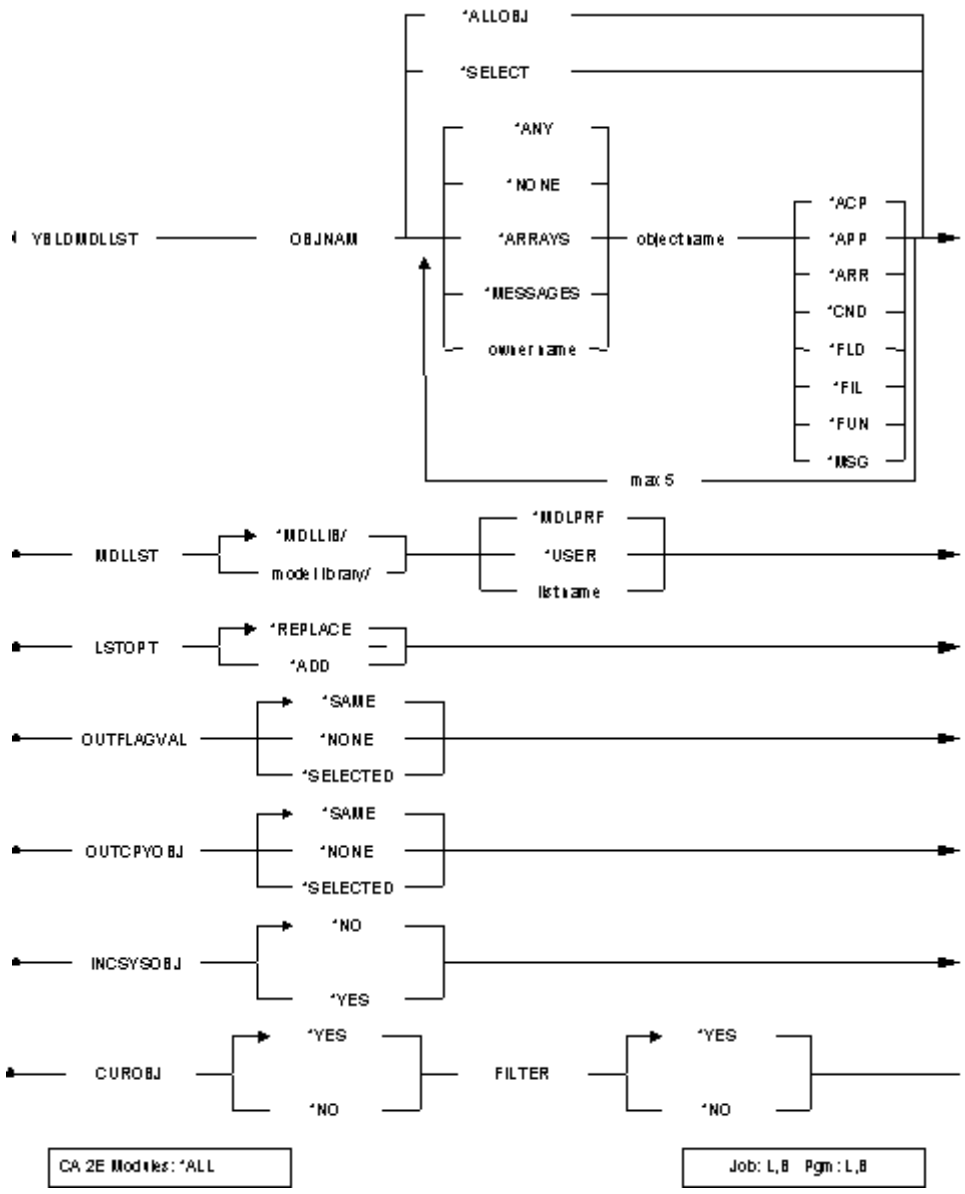
YBLDJOBLST MDLFILE(ORDER*) + MDLACPTYP(*LGL)

YBLDMDLLST (Build a Model Object List) Command

This command allows a user to build a model object list. The list may or may not exist. If it exists, it can be added to or replaced by taking the appropriate option on the LSTOPT parameter.

Only certain model objects can be selected for inclusion in a list. See the command diagram for a list of valid object types.

Once the list has been built, it can be used by the other model list processing commands. Normally a user will edit a list after creation using the Edit Model Object List command (YEDTMDLLST), or process entries in the list using one of the model list processing commands. See the command diagram of these commands for more details.



Parameters

The following are parameters for the YBLDMDLLST command.

OBJNAM

The object name to be added. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- ***ALLOBJ**—(default) Single value indicating that all model objects are included.
- ***SELECT**—Single value indicating that the object to be added is selected using an interactive display function.
- **object owner name**—Generic name of the object that owns the object(s) to be added. Thus, for a function, the owning file would be entered.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **object name**—Generic name of the object(s) to be added.
- **object type**—The object type of the object.
- ***ALL**—All object types are added.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

MDLLST

The qualified name of the model object list that is built. All output is directed to the list specified in this parameter. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- **list name**—The name of the list to be built can be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the target model for the list.
- **library name**—The name of the model library to be the target of the command.

LSTOPT

This parameter specifies the action taken if the list already exists in the model. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list is replaced with the output from this command.
- ***ADD**—The existing model object list is augmented with the output from this command.

OUTFLAGVAL

This parameter specifies the initial value to be placed in the object selected flag associated with each list entry. Values for this parameter are described in the following:

- ***SAME**—(default) No flag value is used. New entries are written with the flag indicating that the entry is not selected. There is no change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing list entries are flagged as selected. This flag may be used by other list commands when selecting list entries.

OUTCPYOBJ

This parameter specifies the initial value to be placed in the copy object flag associated with each list entry. This flag is used by the Copy Model Object command (YCPYMDLOBJ) when selecting objects to copy to a target model. Values for this parameter are described in the following:

- ***SAME**—(default) No flag value is used. New entries are written with the flag indicating that the entry is not selected. There is no change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing list entries are flagged as selected.

INCSYSOBJ

This parameter allows the user to control whether system objects are included in the output list. Values for this parameter are described in the following:

- ***NO**—(default) The system objects are not included.
- ***YES**—System objects are included.

CUROBJ

This parameter allows the user to control whether only current objects are included in the output list. Values for this parameter are described in the following:

- ***YES**—(default) Non-current objects are ignored in the build.
- ***NO**—All objects are included in the build.

FILTER

The initial output of the command can be filtered. If specified, this parameter invokes the Filter Model Object List command (YFLTMDLLST). Values for this parameter are described in the following:

- ***NO**—(default) The filter command is not invoked.
- ***YES**—Filtering is required.

Notes

- The target library for output must be a valid model library. Object lists are be created in file YMDLLSTRFP which resides in the model library. Each list is a separate member in that file. If a value other than *MDLLIB is used for the output list, the specified model library will also be used for object selection.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- *SELECT is invalid for OBJNAM if the job in which the command is running is of type batch.
- The LSTOPT parameter is ignored if the model object list does not already exist.
- The FILTER parameter is ignored if the job running the command is a batch job, since the method for defining the filter criteria is via the command prompter, requiring an interactive display.

Example

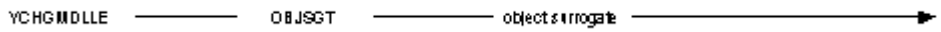
To build model object list MYLIST from all access paths and functions in the model contained in the current library list:

```
YBLDMDLLST OBJNAM( (*ANY *ALL *ACP) +
(*ANY *ALL *FUN) )
```

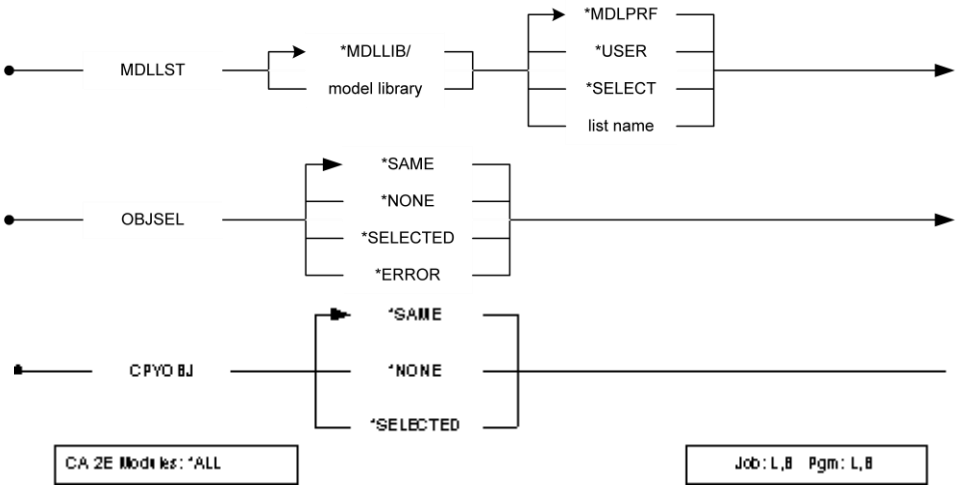
YCHGMDLLE (Change a Model Object List Entry) Command

This command allows a user to change the selection flags of a model object list entry.

Required



Optional



Parameters

The following are parameters for the YCHGMDLLE command.

OBJSGT

Unique number identifier of the model object that is changed. The value for this parameter is as follows:

- object surrogate—The surrogate number of the model object is required.

MDLLST

The qualified name of the model object list in which the entry to be changed exists. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is to be used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The name of the list.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the model library.

OBJSEL

This parameter specifies the value to be placed in the object selected field of the list entry. Values for this parameter are described in the following:

- ***SAME**—(default) The value currently in this field is not changed.
- ***NONE**—The entry is flagged as not selected.
- ***SELECTED**—The entry is flagged as selected.
- ***ERROR**—The entry is flagged in error.

CPYOBJ

This parameter specifies the value placed in the copy object field of the list entry. Values for this parameter are described in the following:

- ***SAME**—(default) The value currently in this field is not changed.
- ***NONE**—The entry is flagged as not selected.
- ***SELECTED**—The entry is flagged as selected.

Notes

- The flagging of list entries is intended to allow users to perform operation on subsets of model objects contained in model object lists. Note that the object selected field (OBJSEL) is recognized by many of the list commands. However, the copy object field (CPYOBJ) is recognized only by the Edit Copy List command (YEDTCPYLST) and the Copy Model Objects command (YCPYMDLOBJ). A value of *SELECTED in the copy object field effectively means explicitly selected or selected for copying for the purposes of the Copy Model Objects command (YCPYMDLOBJ).
- Both the model object list and the list entry must exist prior to running this command.

Example

To change the model object list entry for the object identified by surrogate number 1100911, in model object list DEVLST in the first model library to be found in the current library list, and to flag the entry as *SELECTED for use in another list command:

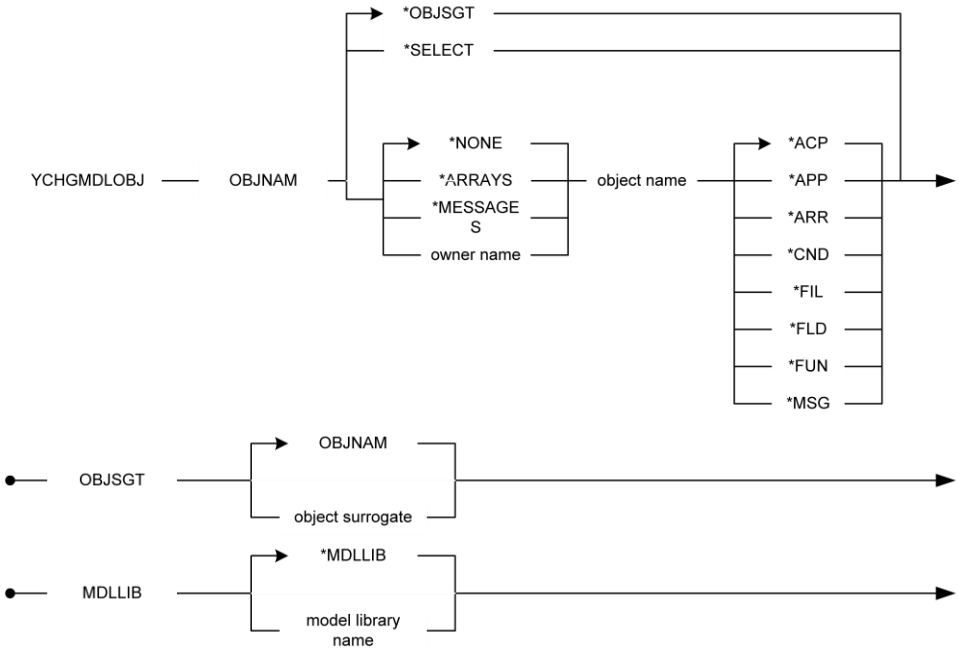
YCHGMDLLE OBJSGT(1100911) MDLLST + (*MDLLIB/DEVLST) OBJSEL(*SELECTED)

YCHGMDLOBJ (Change Model Object) Command

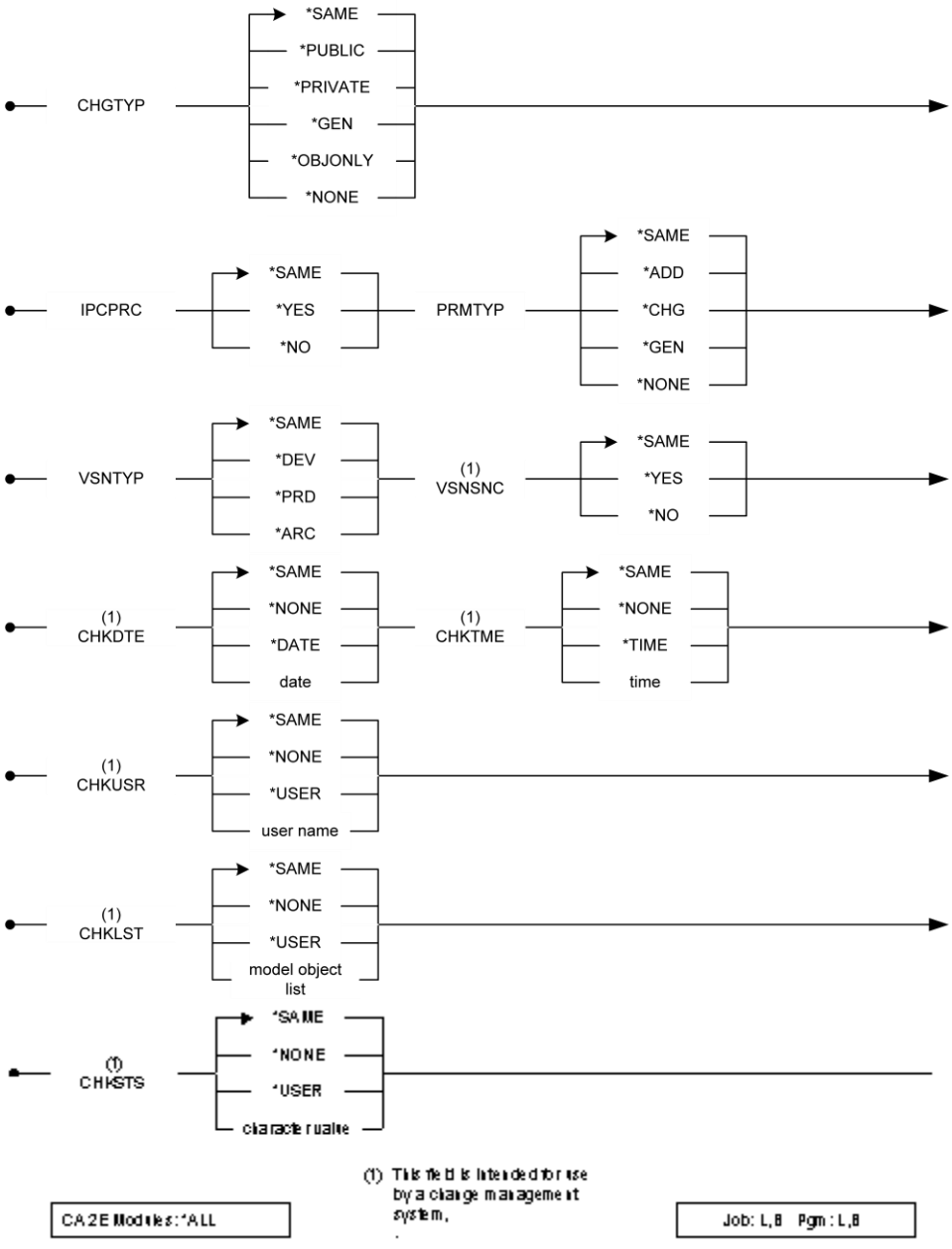
This command provides access to certain user-definable data that is stored for each model object. The command diagram described below shows which data are available to be updated.

We recommend that you use this information to support a user-defined change control facility operating on a model.

Required


YCHGMDLOBJ
OBJNAM
*OBJSGT
*SELECT
*NONE
*ARRAYS
*MESSAGES
owner name
object name
*ACP
*APP
*ARR
*CND
*FIL
*FLD
*FUN
*MSG
OBJSGT
OBJNAM
object surrogate
MDLLIB
*MDLLIB
model library name

Optional



Parameters

The following are parameters for the YCHGMDLOBJ command

OBJNAM

The object name to be changed. This parameter consists of three elements that together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate is to be used to identify the model object that is to be changed.
- ***SELECT**—Single value indicating that the object to be changed is selected using an interactive display function.
- **object owner name**—The character name of the object that owns the object to be changed. Thus, for a function, the owning file would be entered.
- ***NONE**—Special value indicating that the object concerned does not have an owner; for example, objects of type *FIL.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **object name**—The character name of the object to be changed.
- **object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

OBJSGT

Unique number identifier of the model object whose details are to be changed. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the model object to be changed.
- **object surrogate**—The surrogate number of the model object.

MDLLIB

The data model that is to be edited. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model to be edited is the first one found in the current job's library list.
- **model name**—The name of a specific data model.

CHGTYP

Reflects the type of change that has been made to the object. Note that changing this field may also cause the Action required flag (ACTRQD) for objects that use this object to be updated. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***PUBLIC**—The object has been publicly changed. Appropriate component change processing will be performed.
- ***PRIVATE**—The object has been privately changed. Appropriate component change processing will be performed.
- ***GEN**—The object has been changed requiring generation of the implementation object only. This is only valid for generatable objects. No component change processing will be performed.
- ***OBJONLY**—The object has been changed. However, the change does not affect using objects and no source generation is required.
- ***NONE**—The object has been changed but requires no further action. No component change processing will be performed.

IPCPRC

Indicates whether users of the object have been processed to reflect the effect on them resulting from the change made to the object. A Change type (CHGTYP) of *PRIVATE or *PUBLIC recorded for the object will have an effect on its users. When the effect is established during component change processing, the impact processed flag is set to *YES. As a result, changing this flag will either cause component change processing to be performed or avoided, depending on the value to which it is set. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not to be changed.
- ***YES**—The object is to be treated as if component change processing had been performed.
- ***NO**—The object is to be treated as if component change processing had not been performed. As a result, if the Change type (CHGTYP) value is *PRIVATE or *PUBLIC, component change processing will be invoked either the next time the object is changed if component change processing is set to occur interactively when the Apply Component Changes (YAPYCMPCHG) command is run.

PRMTYP

Promotion type to be performed on this object, as part of a change control system. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***ADD**—The object is to be added to the target environment.
- ***CHG**—The object is to replace the same object in the target environment.
- ***GEN**—The object is to be regenerated only in the target environment; the design object is not to be promoted.
- ***NONE**—No promotion type is to be associated with the object.

VSNTYP

This field identifies the status of an object with respect to other members of a group of objects. A group is defined by those objects that originated (were copied) from the same object or another member of the group. This field is intended for use with a promotion procedure and identifies which version of a group is the development version, the production version, and the archive version. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***DEV**—Change the model object value to indicate that it is the development version.
- ***PRD**—Change the model object value to indicate that it is the production version.
- ***ARC**—Change the model object value to indicate that it is the archive version.

VSNSNC

This user-defined field is intended for use by a change management system. You could set it in a checkout procedure to indicate that a version of an object is in conflict with another version; for example, when an object is checked out by two users. The conflict could then be detected when you attempt to promote one of the versions. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***YES**—Change the model object value to indicate that the condition represented by yes is true; for example, *YES could indicate that a conflict exists.
- ***NO**—Change the model object value to indicate that the condition represented by no is true; for example, *NO could indicate that no conflict exists.

CHKDTE

This user-defined field is intended for use by a change management system, to contain the date a checkout procedure was run to check out the model object. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***NONE**—A special value meaning that any value in this field is to be replaced with zero.
- ***DATE**—A special value meaning that the current system date is to be used.
- **date**—A date value may be entered.

CHKTME

This user-defined field is intended for use by a change management system, to contain the time a checkout procedure was run to check out the model object. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***NONE**—A special value meaning that any value in this field is to be replaced with zero.
- ***TIME**—A special value meaning that the current system time is to be used.
- **time**—A time value may be entered.

CHKUSR

This user-defined field is intended for use by a change management system, to contain the user profile of the user who ran a checkout procedure to check out the model object. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***NONE**—A special value meaning that any value in this field is to be replaced with blanks.
- ***USER**—The name of the current user profile is used to update the model object value.
- **user name**—A user profile name can be entered.

CHKLST

This user-defined field is intended for use by a change management system, to contain the name of the model object list from which the model object was checked out as part of a checkout procedure. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***NONE**—A special value meaning that any value in this field is to be replaced with blanks.
- ***USER**—The name of the current user profile is used to update the model object value.
- **list name**—A model object list name can be entered.

CHKSTS

This user-defined field is intended for use by a change management system, to contain the checkout status of the model object when used as part of a checkout procedure. Values for this parameter are described in the following:

- ***SAME**—(default) The current model object value is not to be changed.
- ***NONE**—A special value meaning that any value in this field is to be replaced with blanks.
- **character value**—A user-defined value can be entered.

Example

To change the Order Details file, enter the following:

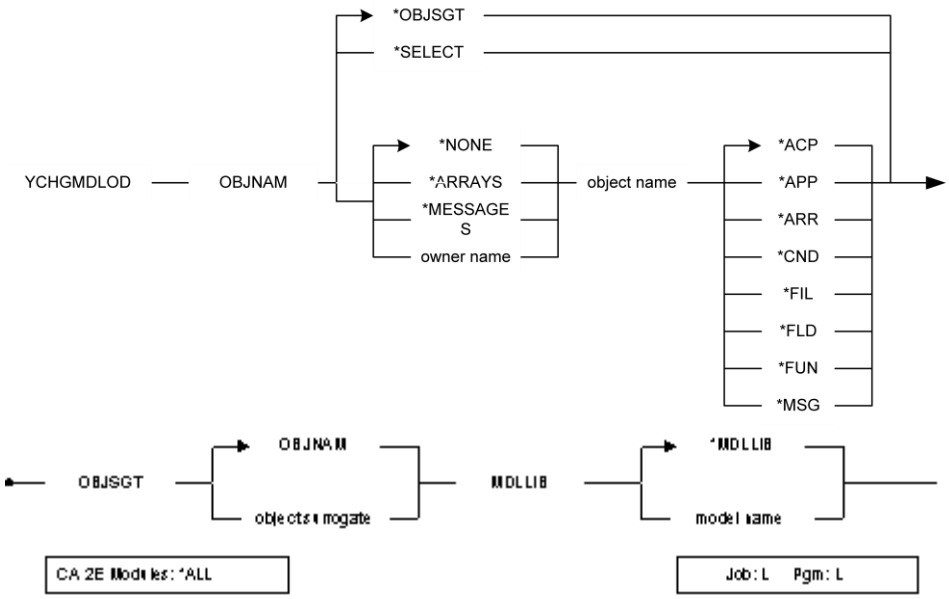
YCHGMDLOBJ OBJNAM(*NONE 'Order Details' + *FIL)

YCHGMDLOD (Change Model Object Description) Command

This command accesses an interactive panel that displays the details for a given model object. All details are display only.

For more information on using the interactive panel, see online Help for the specific panel.

Required



Parameters

The following are parameters for the YCHGMDLOD command.

OBJNAM

The name of the model object to be changed. This parameter consists of three elements that together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate is to be used to identify the model object that is to be changed.
- ***SELECT**—Single value indicating that the object to be changed is selected using an interactive display function.
- **object owner name**—The character name of the object that owns the object to be changed. Thus, for a function, the owning file would be entered.
- ***NONE**—Special value indicating that the object concerned does not have an owner; for example, objects of type *FIL.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **object name**—The character name of the object to be changed.
- **object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

OBJSGT

Unique number identifier of the model object that is to be changed. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the model object to be changed.
- **object surrogate**—The surrogate number of the model object.

MDLLIB

The data model in which the object whose description is to be changed resides. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model is the first one to be found in the current library list.
- **model name**—The name of a specific data model.

Notes

- A value other than *MDLLIB for MDLLIB may result in the library list being changed. If you are currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- Model objects can either be identified by object name (OBJNAM) or by object surrogate key number (OBJSGT). If the OBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).

  Model object names are structured as follows:

Type	Name
ACP	File name/Access path name/'ACP'
APP	---/Application area code/'APP'
ARR	*Arrays/Array name/'ARR'
CND	Field name/Condition name/'CND'
FIL	---/File name/'FIL'
FLD	---/Field name/'FLD'
FUN	File name/Function name/'FUN'
MSG	*Messages/Message name/'MSG'

Example

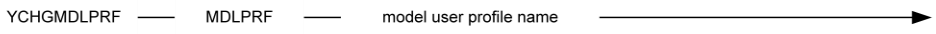
To change the Edit Order Details function, which is owned by the Order Details file, enter the following:

YCHGMDLOD OBJNAM('Order Details' 'Edit Order + Details' *FUN)

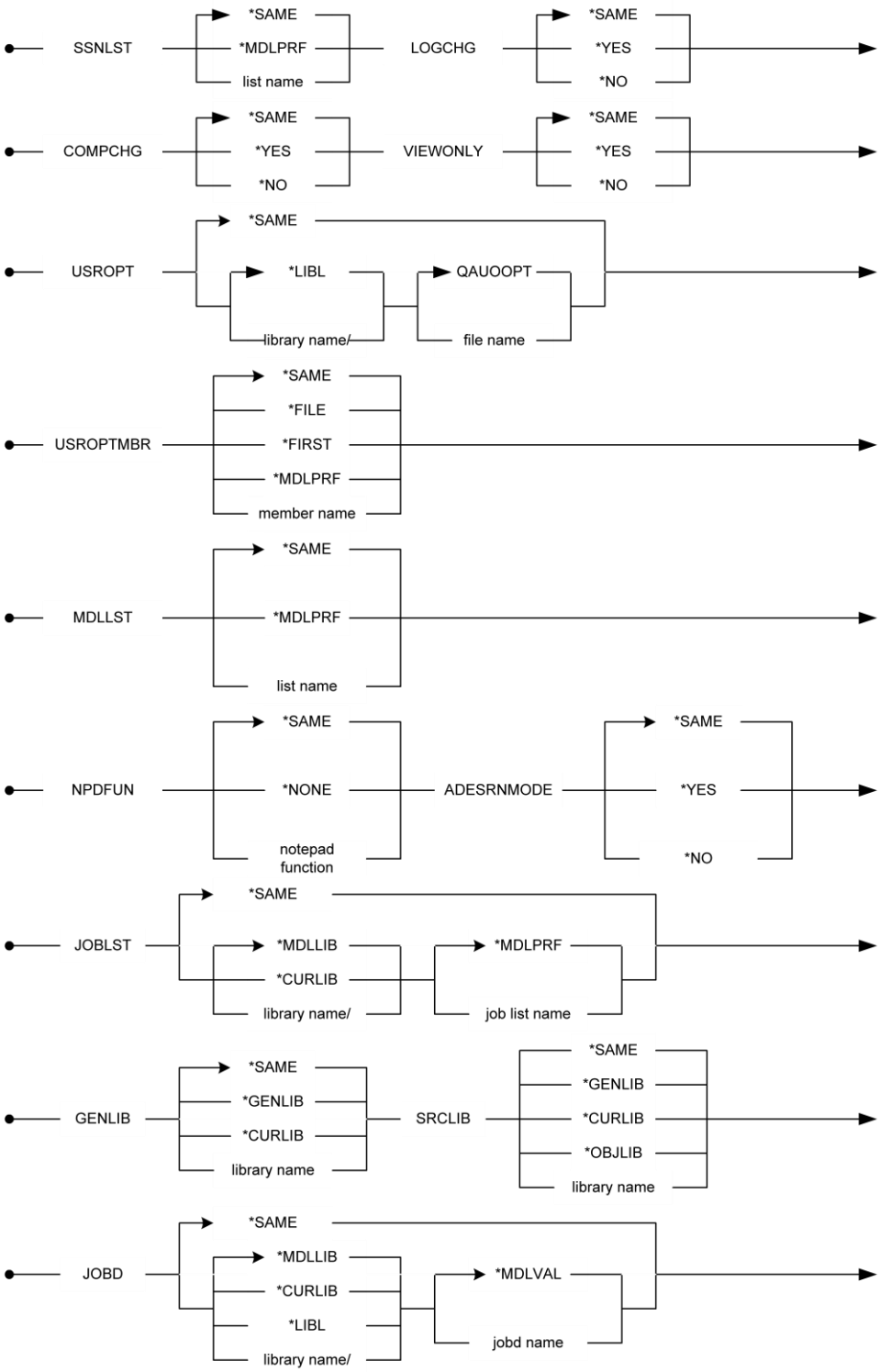
YCHGMDLPRF (Change Model Profile) Command

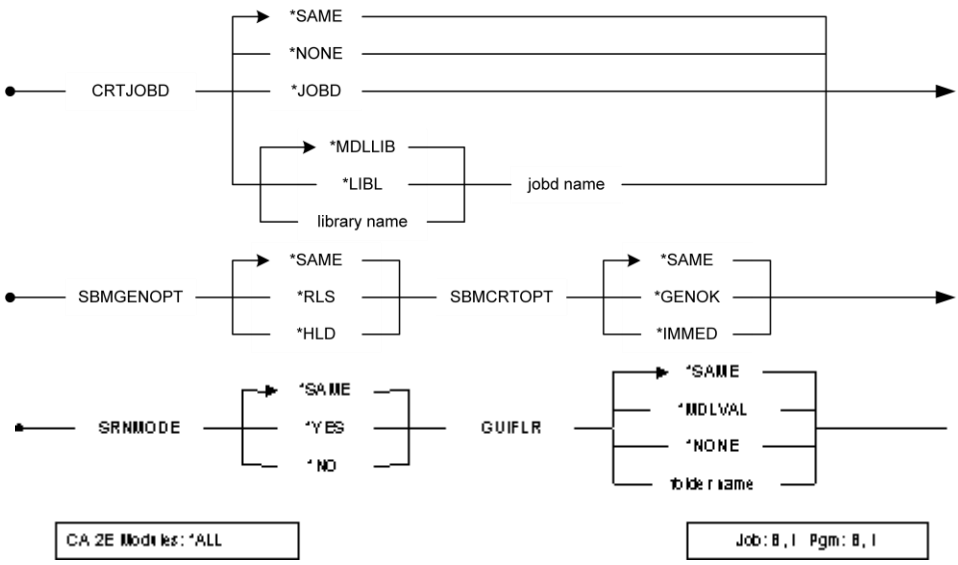
Certain data is stored in each data model associated with user profiles of developers. This command allows these values to be adjusted.

Required



Optional


SSNLST
*SAME
*MDLPRF
list name
LOGCHG
*SAME
*YES
*NO
COMPCHG
*SAME
*YES
*NO
VIEWONLY
*SAME
*YES
*NO
USROPT
*SAME
*LIBL
library name/
QAUOOPT
file name
USROPTMBR
*SAME
*FILE
*FIRST
*MDLPRF
member name
MDLLST
*SAME
*MDLPRF
list name
NPDFUN
*SAME
*NONE
notepad function
ADESRNMODE
*SAME
*YES
*NO
JOBLST
*SAME
*MDLLIB
*CURLIB
library name/
*MDLPRF
job list name
GENLIB
*SAME
*GENLIB
*CURLIB
library name
SRCLIB
*SAME
*GENLIB
*CURLIB
*OBJLIB
library name
JOBD
*SAME
*MDLLIB
*CURLIB
*LIBL
library name/
*MDLVAL
jobd name



Parameters

The following are parameters for the YCHGMDLPRF command.

MDLPRF

The model user profile name that is changed. The value for this parameter is as follows:

- **user profile name**—The name of the user profile must be entered.

SSNLST

The name of the session list used when editing a model. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***MDLPRF**—This special value indicates that the name of the current user profile is loaded to the session list field.
- **list name**—The name of the list can be entered.

LOGCHG

This parameter indicates whether changed objects are logged to the specified session list while editing the model. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***YES**—Changed objects are logged to the session list.
- ***NO**—Changed objects are not logged.

COMPCHG

This parameter indicates whether objects that use a changed object are to be flagged as having had a component changed. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***YES**—Objects which use a changed object are flagged when the object is changed.
- ***NO**—Component change processing is not performed when the object is changed. It may be performed later using the Apply Component Change command (YAPYCMPCHG).

VIEWONLY

This parameter indicates whether the current user is navigating the model in *VIEW mode or *EDIT mode. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***YES**—Navigation is in *VIEW mode.
- ***NO**—Navigation is as normal (*EDIT mode).

USROPT

The Edit Model List panel supports the use of user options to be applied to subfile records. This field stores the user option file name to use. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- **QAUOOPT**—The system-supplied user option file name.
- **list name**—User-defined file name.
- ***LIBL**—This special value is used as the library name.
- **library name**—A library name can be entered.

USROPTMBR

The member in the user options file that is used. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***FILE**—The member name is the same as the user options file name.
- ***FIRST**—The first member in the file is to be used.
- ***MDLPRF**—The member name is the same as the current user profile name.
- **member name**—The user options member name.

MDLLST

The list used if *MDLPRF is specified on a model object list command. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***MDLPRF**—A list with the same name as the user profile is used.

NPDFUN

The user-defined notepad function to use when using the action diagram editor. The function specified must either be an Execute External Function (EXCEXTFUN) or an Execute Internal Function (EXINTFUN). Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***NONE**—No particular notepad function is specified, the action diagram editor will provide an empty notepad for use during editing sessions.
- **notepad function**—The function name must be specified.

ADESRNMODE

This parameter provides the default for full screen mode when using the action diagram editor. Values for this parameter are described in the following:

- ***SAME**—The current value is not changed.
- ***YES**—The default for the action diagram editor full screen mode is yes.
- ***NO**—The default for full screen mode is no.

JOBLST

When a user invokes the Submit Model Create command (YSBMMDLCRT), this qualified value can be used to determine the job list that is defaulted. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***MDLPRF**—A job list with the same name as the user profile is the default.
- **list name**—Job list name.
- ***MDLLIB**—This special value is used as the default library name.
- ***CURLIB**—This special value is used as the default library name.
- **library name**—Default library name.

GENLIB

The default generation library name may be specified. When a user invokes the Submit Model Create command (YSBMMDLCRT), this value will be used by default. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***GENLIB**—This special value is the default.
- ***CURLIB**—This special value is the default.
- **library name**—Default library name.

SRCLIB

The default source library name may be specified. When a user invokes the Submit Model Create command (YSBMMDLCRT), this value will be used by default. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***GENLIB**—This special value is the default.
- ***CURLIB**—This special value is the default.
- ***OBJLIB**—Special value meaning that the source library name is the same as for the GENLIB parameter.
- **library name**—Default library name.

JOBD

The default job description name to use when a user invokes the Submit Model Create command (YSBMMDLCRT). This qualified value will be used as the default. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***MDLVAL**—This special value is the default.
- **job description name**—The job description name.
- ***MDLLIB**—This special value is the default library in which to find the job description.
- ***CURLIB**—This special value is the default library in which to find the job description.
- ***LIBL**—This special value is the default library in which to find the job description.

CRTJOBD

The default create job description name to use when a user invokes the Submit Model Create command (YSBMMDLCRT). This qualified value will be used as the default. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***JOBD**—This special value is the default.
- ***NONE**—This special value is the default.
- **job description name**—The create job description name.
- ***MDLLIB**—This special value is the default library in which to find the job description.
- ***LIBL**—This special value is the default library in which to find the job description.

SBMGENOPT

The default generation option value to use when a user invokes the Submit Model Create command (YSBMMDLCRT). Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***RLS**—This special value is the default.
- ***HLD**—This special value is the default.

SBMCRTOPT

The default compilation option value to use when a user invokes the Submit Model Create command (YSBMMDLCRT). Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***GENOK**—This special value is the default.
- ***IMMED**—This special value is the default.

SRNMODE

This parameter determines whether or not the default panel on the Edit Model List command (YEDTMDLLST) is displayed in full screen mode. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***YES**—The default for full screen mode is yes.
- ***NO**—The default for full screen mode is no.

GUIFLR

This parameter records the default value for the GUI folder value used when prompting the Submit Model Create (YSBMMDLCRT) command for the specified user. Values for this parameter are described in the following:

- ***SAME**—(default) The current value is not changed.
- ***MDLVAL**—The value to be prompted is derived from the model value.
- ***NONE**—The default value is *NONE.
- **folder name**—Enter the name of a folder.

Notes

- User profile values for a particular developer are updated while using the Edit Model Object List command (YEDTMDLLST). The YCHGMDLPRF command may be used to adjust values outside the YEDTMDLLST command.
- The processing of the VIEWONLY parameter occurs in the user access exit program (YUSROBJR1C). The shipped default processing switches from *EDIT to *VIEW mode if the VIEWONLY parameter is set. Users should be careful, therefore, when changing the processing of this exit program, to take account of such processing.

Example

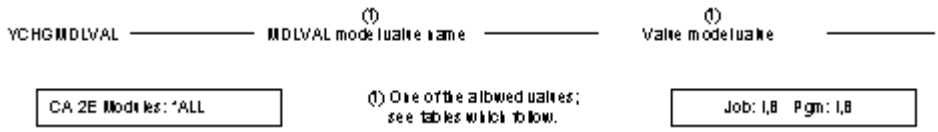
To log changed objects to model list CHGOBJ and to perform component change processing the next time user KATHY edits the model:

YCHGMDLPRF MDLPRF(KATHY) CHGLST + (CHGOBJ) LOGCHG(*YES) COMPCHG(*YES)

YCHGMDLVAL (Change Model Value) Command

Changes a model value.

Required



Parameters

The following are parameters for the YCHGMDLVAL command.

MDLVAL

Name of model value that is changed. One of the model values.

VALUE

Value that the model value is changed to.

Notes

- Initial values for model values are set by the parameters specified when the command Create Model Library (YCRTMDLLIB) was run
- On the following page is an alphabetical listing of model values. You can then find related values grouped according to the tables below:
  - Design Option Values
  - Name Allocation Values
  - Access Path Generation Values
  - Function Generation Values
  - Environment Values
  - Help Generation Values
  - PWS Environment Values
  - System Defaults
  - Protected Model Values
  - Unchangeable System Model Values

Model Values in Alphabetical Order

Model Value	Description	Grouping
YABRNPT	Action/DDS Menu Bars	Design Option
YACTCND	Action diagram compound symbols	Design Option
YACTFUN	Action diagram compute symbols	Design Option
YACTSYM	Action diagram structure symbols	Design Option

YACTUPD	Default update flag value on action diagram exit	Environment
YALCVNM	Automatic name allocation	Name Allocation
YBNDDIR	ILE Binding directory	Environnent
YCHGCTL	Change control library	Environment
YCMPCHG	Component change processing control	Environment
YCMPTXT	Company text	Environment
YCMTCDE	Include Inactive Code	Environment
YCNFVAL	Confirm value	Function Generation
YCPYLIB	Copy library	Environment
YCPYMSG	Copy back messages	Function Generation
YCRTENV	Creation environment	Environment
YCRTJBD	Job description	Environment
YCUAEXT	CUA device extension	Design Option
YCUAPMT	CUA prompt	Design Option
YCUTOFF	Date range for 2-digit year input dates	Function Generation
YDATFMT	Date format	Function Generation
YDATGEN	Date generation	Function Generation
YDBFACC	Database access method	Access Path Generation
YDBFGEN	Database implementation	Access Path Generation
YDDLDBA	Database access method	Function generation
YDFTCTX	Parameter default context	Function Generation
YDSTFIO	Distributed file I/O control	Environment
YERRRTN	Error routine	Function Generation
YEXCENV	Execution environment	Environment
YFILPFX	File name prefix	Name Allocation
YFRFPFX	Field reference prefix	Access Path Generation
YFRFTXT	Field reference file text	Access Path Generation
YFRFVNM	Field reference file name	Access Path Generation
YGENCMT	Generate comments in source	Function Generation
YGENGUI	Generate GUI by default (Synon/TC	Function Generation

YGENHLP	Generate help text	Function Generation/ Help Generation
YGENLIB	Generation library	Environment
YGENLMT	Maximum job list entries for single YGENSRC job	Function Generation
YGENRDB	Generation RDB name	Environment
YGUIAKY	Suppress aid keys (Synon/TC	PWS Environment
YGUICMD	Command key descriptor text (Synon/TC	PWS Environment
YGUIFLR	Folder for GUI objects (Synon/TC	PWS Environment
YGUIHLP	Default GUI help language (Synon/TC	Help Generation
YGUILIB	Library partitioning for GUI (Synon/TC	PWS Environment
YGUILSO	Suppress list options (Synon/TC	PWS Environment
YGUIOPR	Operator between prefix & text (Synon/TC	PWS Environment
YGUISEL	Subfile select descriptor text (Synon/TC	PWS Environment
YGUITKY	Display command keys per tab (Synon/TC	PWS Environment
YHLLCBL	HLL Cobol	Function Generation
YHLLGEN	HLL to generate	Function Generation
YHLLVNM	HLL naming convention	Name Allocation
YHLPCSR	Generate cursor sensitive text	Help Generation
YIGCCNV	IGC support	Environment
YLHSFLL	Leaders for device design	Design Option
YLIBLST	Model library list name	Environment
YMDLLIB	Model library	Environment
YMDLLNG	Model language	Environment
YMDLTXT	Model text	Environment
YMSGNBR	Message number	Name Allocation
YMSGPFX	Message id prefix	Name Allocation
YMSGVNM	Message file name	Name Allocation
YNLLMDL	Null model	Unchangeable System Model Value

YNLLUPD	Null update suppression	Function Generation
YNPTHLP	NPT help default generation type	Help Generation
YOBJPFX	Member name prefix	Name Allocation
YOLDDBF	Old DBF names	Access Path Generation
YOLDLIB	Old library	Environment
YOPNACC	Open access to the model	Environment
YPKYLIB	Print key file library	Function Generation
YPKYVNM	Print key file	Function Generation
YPMTGEN	Prompt implementation	Function Generation
YPMTMSF	Prompt message file	Name Allocation
YPMTNBR	Prompt number	Name Allocation
YPUTOVR	DDS put with override	Function Generation
YRPGHDR	RPG header specification	Function Generation
YRPGHSP	RPG Control (H) Specification	Function Generation
YRP4HSP	RPGIV Control (H) Specification (*PGM)	Function Generation
YRP4HS2	RPGIV Control (H) Specification (*MODULE)	Function Generation
YRP4SGN	RPGIV Generation Options	Function Generation
YSAAFMT	SAA format	Design Option
YSCNLMT	Scan Limit	Function Generation
YSFLEND	Subfile end	Design Option
YSHRDFT	Sharing default	Design Option
YSHRSBR	Share function subroutines	Function Generation
YSNDMSG	Send error message	Function Generation
YSQLFMT	Generate SQL RCDFMT clause	Function Generation
YSQLLCK	SQL locking	Function Generation
YSQLLEN	SQL naming length	Access Path Generation
YSQLLIB	SQL collection library	Environment
YSQLVNM	SQL naming	Access Path Generation
YSQLWHR	SQL Where clause	Function Generation
YSYSCHG	Change control library	System Defaults

YSYSDBF	System database access	System Defaults
YSYSHLL	Default HLL	System Defaults
YSYSLNG	System language	Unchangeable System Model Value
YSYSNPT	System default NPT help generation type	System Defaults
YSYSPMT	System prompt	System Defaults
YSYSSAA	Default SAA formatting option	System Defaults
YTRGLIB	Trigger Runtime library	Environment
YTRNAPI	Convert Case API	Function Generation
YUIMBID	Bi-directional UIM help text	Help Generation
YUIMFMT	Default UIM format	Help Generation
YUIMIDX	UIM search index	Help Generation
YVLSPFX	Value list prefix	Name Allocation
YWBDATR	Window border attribute	Design Option
YWBDCHR	Window border characters	Design Option
YWBDCLR	Window border background color	Design Option
YWSNGEN	Workstation environment	Function Generation
YW2ELIB	Web Option Product Library	Function Generation

Model Values Grouped By Role

Following are the model values grouped according to their role.

Design Option Values

Model Value	Description	Allowed Values
YABRNPT	Allows you to choose between creation of CA 2E Action Bars or DDS Menu Bars for a given function.	*DDSMNU *ACTBAR

YACTCND	Symbols used in editing a compound condition.	5 pairs of up to 3 characters each used to represent the edit and display values for logical operators and parenthesis. Then, up to 3 characters for the conditions.
YACTFUN	Symbols used in editing a compute expression.	8 sets of up to 3 characters each separated by blanks used to represent the arithmetic operations, parenthesis and functions.
YACTSYM	Symbols used to indicate the action diagram symbols :, |, . Used in action diagram editor and YDOCMDLFUN.	*SAA—Default to symbols, :, |, , or you can pick three symbols of your own choice. See the notes that follow.
YCUAEXT	For panel device designs, defines whether right-hand side text, provision for instructions area on entry panels, and padding of field label trailers will be used as a default.	*C89EXT—This value will provide right-hand side text and extra spacing in line with the 1989 CUA extensions. *DEFAULT—This value will not provide right-hand side text and extra spacing.
YCUAPMT	Enables the CUA prompt. The default is F4.	*MDL—Will pick it up from YSAAFMT. If YSAAFMT = CUATEXT, F4 will be enabled within the generated application. *YES—F4 will be enabled within the generated application. *NO—F4 will not be enabled. *CALC—See the notes that follow.
YLHSFLL	Symbols to be used as leaders between screen text and screen fields.	*SAA—Will provide default left-hand side filler characters for input and output fields or you can pick characters of your own choice. See the notes that follow.

YSAAFMT	Screen device convention to be defaulted for use in the device design editor. (System/38, CUA Entry, or CUA Text)	*CUAENTRY—Defaults to a header/footer without windows or action bars. *CUATEXT—Defaults to windows and action bar style header/footers. All function types with the exception of SELRCD will default to action bars. THE SELRCD will default to a window. *S38—Defaults to System/38 style header/footer.
YSFLEND	Controls whether + or More. . . is displayed if additional subfile records are available for display.	*PLUS—Display the + sign to indicate that the subfile contains more records. *TEXT—Display More. . . to indicate that the subfile contains more records. Display Bottom to indicate that the last subfile record is displayed.
YSHRDFT	Defines the sharing default for the model when editing file relations. This affects how key fields are shared by default when the key field is already present in the file.	*ALL—All keys that match existing fields, other than the last key of the relation, are automatically shared. *NONE—No keys are automatically shared. This is a recommended default when using Gateway/SR.
YWBDATR	Window border attribute that is to be the default for device designs.	*CUA—Shadowed window. *SHADOW—Shadowed window. *NOSHADOW—Straight lined window.

YWBDCHR	Window border characters. The field length is 8, with each number referencing a border section: 1 Top left corner character 2 Horizontal top line 3 Top right corner character 4 Vertical left line character 5 Vertical right line character 6 Bottom left corner character 7 Horizontal bottom line 8 Bottom right corner character	*CUA—Dots provided at the top and bottom, colon at the left and right. CHAR (8)—You can choose which characters you want for each border section. See the notes that follow.
YWBDCLR	Window border background color.	*CUA—Default color is blue. *RED, *GRN, *WHT, *PNK, *TRQ, *YLW, *BLU

YACTSYM

YACTSYM is made up of three characters:

- The symbol used to indicate iteration brackets in action diagrams
- The symbol used for sequence brackets
- The symbol used for condition brackets

The three characters are shipped with the following initial values:

- Iteration..: '|'
- Sequence...: ':'
- Condition..: ' '

YCUAPMT

The *CALC value for YCUAPMT enables F4 prompting in the generated application, and processes the CALC: user points in the action diagram of the function where you pressed F4. The main use for this feature is to provide Retrieve Condition functionality (*RTVCND built-in function) when you press F4.

Note: If you have Retrieve Condition logic within a USER: user point that you want processed when you press F4, you will need to move the logic to a CALC: user point.

If a CALC: user point contains logic that you do not want processed when you press F4, you can bypass the logic by checking for the F4 prompt condition. For example:

```
>CALC:
.──
.     DTL.Gender name = Condition name of DTL.Gender                    <<<
.     +─CASE                                                           <<<
.     |─DTL.*CMD key is *Prompt                                        <<<
.     |─*OTHERWISE                                                     <<<
.     |  :                                                             <<<
.     | (Actions to be bypassed when you press F4=Prompt)              <<<
.     |  :                                                             <<<
.     +─ENDCASE                                                        <<<
```

Note: Some function types contain logic that reinitializes function fields as shown in the following table. As a result, if you bypass function field calculation, when you press F4 the function fields will be reinitialized but not recalculated.

Function Type	Function Field Initialization
PMTRCD	No
DSPFIL	No
SELRCD	No
DSPRCD,2,3	Yes
EDTRCD, 2,3	Yes
EDTFIL	Yes
DSPTRN	Yes
EDTTRN	Yes

The action diagram logic that processes the key screen for the following functions does not contain a CALC: user point:

- DSPRCD
- DSPRCD2
- DSPRCD3
- EDTRCD
- EDTRCD2
- EDTRCD3.

As a result, to process Retrieve Condition logic before processing the detail screens for these functions you can:

- Set the Bypass Key Screen function option for the function to Y.
- Create a Prompt Record (PMTRCD) function that includes the Retrieve Condition logic at its CALC: user point and calls the Display Record or Edit Record function to process the detail screen.

YLHSFLL

YLHSFLL takes either the special value *SAA, indicating that the SAA convention is to be used (. . . :), or a 9 character string is specified, typically :.:..:..:. The first character (":" in the example) is used as the leader on screen designs when there is one space between the text and the following field. The next two characters (".:") are used when there are two spaces between the text and the following field. The next three characters ("..:") are used when there are three spaces between the text and the field. The final three characters ("..:") are used when there are more than three spaces, with the middle character being repeated as required.

YWBDCHR

YWBDCHR is made up of eight characters that correspond to the eight sides and corners of a window frame as follows:

Character	Position	Value
5	Top left corner	.
1	Top horizontal	.
6	Top right corner	.
3	Left vertical	:
4	Right vertical	:
7	Bottom left corner	:

2	Bottom horizontal	.
8	Bottom right corner	:

Name Allocation Values

Model Value	Description	Allowed Values
YALCVNM	Specifies whether field and object names are to be allocated automatically by CA 2E	*YES *NO *MNC—Will be allocated mnemonically. See the note that follows this table.
YFILPFX	The Last Used File Prefix (YFILPFX) model value contains the last two-character identifying mnemonic CA 2E used when creating a new file. These two characters occupy positions three and four of the new file name, following the model object prefix.	Two-character prefix.
YHLLVNM	Specifies HLL or HLLs with which new names will comply.	*RPG *CBL *RPGCBL—Allocate both RPG and Cobol. *VNM—Same language as YHLLGEN option is allocated.
YMSGNBR	The last assigned message number in the message file named by YMSGVNM.	AAAA to 9999
YMSGPFX	Message prefix to be given to message identifiers issued automatically by CA 2E.	Message prefix or *NO VNM(3)
YMSGVNM	Name of default message file in GENLIB to place message descriptions in.	*NONE *MSGPFX VNM(10)
YOBJPFX	Prefix to use when generating system object names.	Two-character prefix that will be attached to your generated objects.
YPMTMSF	Device prompt message file.	Valid System Name—Specify your own. *NONE *PMTMSF—Default name provided.
YPMTNBR	The last assigned prompt message number in the message file named by YPMTMSF.	0001 to 9999
YVLSPFX	Prefix to be given to value list profile and the program that calls it. See YCVTCNDVAL.	Two-character prefix

Note: If you set the model value for YALCVNM to *MNC, a column of input fields is created on the define objects panel. Place an M in this column and the newly defined object name will be added to a file of object names or mnemonics of 1, 2, and 3 characters. The three variants can then be defined and made available to the name allocation program

Access Path Generation Values

Model Value	Description	Allowed Values
YDBFACC	Specifies whether data is accessed using the SQL table directly or an SQL view.	*DBFGEN—Access data using a view. *TABLE—Access data directly from the SQL table.
YDBFGEN	Defines the method for database definition, either DDS, SQL or DDL. The default is the value defined by the YSYSDBF model value.	*DDS *SQL *DDL
YFRFPFX	Prefix for fields in field reference file.	Valid System Name—Specify your own.
YFRFTXT	Text for field reference file.	Character string
YFRFVNM	Name of field reference file.	Valid System Name or *NONE
YOLDDBS	Controls display of the old DBF name field on the Edit Field Details display.	*YES—Display YES. *NO—Display NO.
YSQLLEN	Controls the length of the extended SQL name.	A numeric value up to 25. This model value is used only when YSQLVNM is *SQL.
YSQLFMT	Specifies whether the RCDFMT keyword must be generated for SQL tables, views, and indexes.	*NO *YES

YSQLVNM	Specifies whether to assign DDS names or model object names to SQL tables and columns.	*DDS—Use DDS names. *SQL—Use the names of the CA 2E model objects; in other words, use extended SQL naming. *LNG—Use the long names of the CA 2E objects in the model along with the DDS or implementation names. *LNF—Use the long field names of the CA 2E objects in the model along with the DDS or implementation names. *LNT—Use the long table names of the CA 2E objects in the model along with the DDS or implementation names.

Function Generation Values

Modern value	Description	Allowed values
YCNFVAL	Specifies initial value for confirm prompt	*YES
YCPYMSG	Specifies whether outstanding messages are to be copied back to the calling program.	*YES *NO
YCUTOFF	Specifies the first of 100 consecutive years that can be entered as 2 digit years. It is specified as 19YY, which represents the hundred years 19YY to 20YY-1. Values between YY and 99 are assumed to be in the 20th century; values between 00 and YY-1 are assumed to be in the 21st century.	1940 19YY—where YY is a numeric value between 00 and 99; in other words, a 4-digit year between 1900 and 1999
YDATFMT	Specifies the display format for dates at run time (DMY, YMD, MDY) if YDATGEN = *VRY was specified. It is stored as Y2DTFMA data area in GENLIB.	*DMY—Date, Month, Year *YMD—Year, Month, Date *MDY—Month, Date, Year
YDATGEN	Date validation generation option: MDY, YMD, DMY, VRY - if VRY see YDATFMT value	*DMY *YMD *MDY *VRY
YDDLDBA	Specifies a method of accessing the database (RLA or SQL), when a function's Generation Mode option is set to A(ACPVAL) or M(MDLVAL), which resolves to DDL type.	*RLA—Specifies that the external function generates with RLA access. *SQL—Specifies that the external function generates with SQL access.

Modern value	Description	Allowed values
YDFTCTX	Specifies the default context to be used for a given function call in the action diagram editor when no context has been supplied by the user.	*WRK—Use WRK context for parameter defaulting. *LCL—Use LCL context for parameter defaulting.
YERRRTN	Specifies whether an error handling routine is to be generated.	*YES—Generate error handling routine. *NO—Do not generate error handling routine.
YGENCMT	Determines whether or not comments are placed in the resulting generated source code. The time required to generate a function can be significantly improved if comments are not required for the source. Note: Refer to the YGENHLP model value for another method that can be used to improve function generation times.	*ALL—All comments are generated into the source for a function. *HDR—Only header comments are generated. *NO—No comments are generated. *STD—Currently the same as *ALL. To be used in a future release.
YGENGUI	Generate GUI by Default. Indicates whether to generate SDF source for functions that use a screen device design. This applies to all programmable workstation functions; namely, if the YWSNGEN model value is set to *GUI, *VB, or *JVA. YGENGUI is ignored when YWSNGEN or its function option is set to *NPT. For Synon/TC, also indicates whether to generate Windows Help and depends on YGENHLP. See YGENHLP.	*YES—Generate a GUI interface for functions that use a screen device design. *NO—Do not generate a GUI interface. *ONLY—Generate only a GUI interface that use a screen device design. For functions where SDF is not applicable, for example, Print File functions, no source is generated.
YGENHLP	Determines whether help will be generated/created by default for functions in the model. This can be overridden by a function option. For Synon/TC the result depends on YGENGUI. See YGENHLP.	*YES—Generate help text (UIM or TM) with all other HLLs. *NO—Do not generate help text. *ONLY—Generate help text only (without other HLLs).

Modern value	Description	Allowed values
YGENLMT	Specifies the maximum number of job list entries that can be processed by a single YGENSRC job. If a generation or compilation job is submitted which contains more entries (objects) than this value, the list will be automatically 'split' and processed by multiple YGENSRC jobs. **Note**: YGENLMT is shipped with a value of 99999. This value should not be changed unless directed by CA support.	1-99999 **Note:** If an object assigned to a YGENSRC job is associated with another object assigned to another YGENSRC job, PGM and DSPF of an EDTFIL, for example, the YGENSRC job assigned to the PGM will also process the DSPF and update the job list accordingly. In this scenario the YGENSRC that was originally assigned the DSPF will not process that object. Therefore it is possible for one or more of the submitted YGENSRC jobs to not actually perform any source generation at all; this is expected behavior.
YHLLCBL	Specifies Cobol version.	*CBL85 *CBL74
YHLLGEN	Default HLL type to be given to new functions.	*RPG *CBL
YNLLUPD	Sets the default for whether CHGOBJ functions update or release the database record if the record has not changed. This may be overridden with a matching function option.	*NO—CHGOBJ functions always update the database whether the record changed or not. *AFTREAD—CHGOBJ checks whether to suppress database update after the After Data Read user point. *YES—CHGOBJ checks whether to suppress database update both after the After Data Read and after the Data Update user points.
YPKYLIB	Print file library name.	Valid System Name—Specify your own. *LIBL—Will pick up name from your library list. *NONE
YPKYVNM	Print file name for use on DSPF DDS PRINT keyword: specifies name of spool file for PRINT key output.	Valid System Name *NONE

Modern value	Description	Allowed values
YPMTGEN	Device text prompt implementation which specifies whether the text on your device designs will be externalized to message files. The default is the value defined by the YSYSPMT model value.	*OFF—This model is incapable of generating message IDs for screen text literals. (default *MSGID—This model will default to generated external message IDs for every screen text constant. *LITERAL—This model is capable of generating external message IDs but will not default to do so. You can override at the function level.
YPUTOVR	Specifies whether code is to be generated using PUTOVR option in DDS.	*YES—Generate PUTOVR in the DDS. *NO—Do not generate PUTOVR in the DDS.
YRPGHSP	Specifies H specification line to be used for RPG programs.	H specification (80)
YRP4HSP	Specifies H Specification line to be used for RPGIV Programs (*PGM)	H Specification VNM(80)
YRP4HS2	Specifies H Specification line to be used for RPGIV Modules (*MOD)	H Specification VNM(80)
YRP4SGN	The options which are used to generate RPGIV (RPGLE) source in the model. See help text for more information	1st character – U,M,L 2nd character – G,W,R,B,P
YSCNLMT	Allows you to specify the scan limit for subfile functions. The shipped value is 500, but can be changed using YCHGMDLVAL. It can also be changed within a function through the use of the *PGM.*Scan limit value.	1- 999999999
YSHRSBR	indicates whether generated source code for subroutines is to be shared and whether the subroutine's interface is internal or external.	*NO—Generate source code each time the subroutine is called. The subroutine's interface is internal. *YES—Share the first instance of generated source for all subsequent calls to the subroutine. The subroutine's interface is external.
YSNDMSG	Specifies whether all messages are to be sent, or only the first one detected.	*YES *NO

Modern value	Description	Allowed values
YSQLLCK	Specifies whether a row to be updated will be locked at the time it is read or at the time it is updated.	*UPD—Lock row at time of update. *FET—If a SELECT is done, lock row at time of read; else lock row at time of update. *IMG—Lock row at time of read for CHGOBJ's embedded in the DBF record user point of standard Edit functions.
YSQLWHR	Specifies whether to use OR or NOT logic when generating SQL WHERE clauses.	*OR—Use OR logic when generating SQL WHERE clauses. *NOT—Use NOT logic when generating SQL WHERE clauses.
YWSNGEN	Defines whether interactive functions are to operate on non-programmable terminals (NPT) or on programmable work stations (PWS) communicating with an IBM i host. For programmable workstations, you also specify the PC runtime environment.	*NPT—Will generate for non-programmable terminals. *GUI—Generates for non-programmable terminals together with a Windows executable. *JVA—Generates for non-programmable terminals together with a Windows executable and a Java executable. *VB—Generates for non-programmable terminals together with a Visual Basic executable.
YW2ELIB	The name of the Web Option product library that is associated with this model if HTML templates are being generated	VNM (10) - Valid library name.

YRP4SGN

RPGIV generation options: The options which are used to generate RPGIV (RPGLE) source in the model. This model value consists of 2 characters as follows:

1st character: Case of generated source. This specifies the case of the actual source statements (excluding comments). It applies to variable names and RPGIV op-codes. The default is 'U' (upper-case). If any value other than those given below is specified, the default value will be used when generating RPGIV source.

U

Source code is generated in upper-case (e.g. 'NNNNNN')

M

Source code is generated in mixed-case (e.g. 'Nnnnnn')

L

Source code is generated in lower-case (e.g. 'nnnnnn')

2nd character: Comment color in generated source. This specifies the color of any comments generated in the source. The default is 'G' (green). If any value other than those given below is specified, the default value is used when generating RPGIV source.

G

Comments are generated in green

W

Comments are generated in white

R

Comments are generated in red

B

Comments are generated in blue

P

Comments are generated in pink

YRP4HSP

This model value controls characters 1-80 of data area YRP4HSPRFA.

YRP4HS2

This model value controls characters 81-160 of data area YRP4HSPRFA 2 model values are used because in RPGIV, the H specification can be 160 characters.

Environment Values

Model Value	Description	Allowed Values
YACTUPD	Defines the default value for the change/create function option on the exit function definition panel.	*YES—Option will always default to Y. *NO—Option will always default to N. *CALC—Option is set to Y only when a change to the function's action diagram or panel design has been detected.
YCHGCTL	Name of the library in which the change control exit programs reside. Restricted exit programs will be called qualified by this library name. For more information on the change management facilities, please see the Start Change Control command (YSTRCHGCTL).	*NONE—Restricted exit programs will not be called. Change control additional features are effectively off. Valid library name.

YCMPCHG	Determines whether component change processing happens interactively and the access which developers have to the component change processing flag on their model.	*NONE—No interactive component change processing occurs while editing model objects unless running the YAPYCMPCHG command. *LIMITED—Component change processing will occur if developer's model profile is set to Y. Only *DSNR authorized developers can change the model profile value. *UNLIMITED—Component change processing will occur if developer's model profile is set to Y. Developers may change the model profile value as they wish. *GEN—The YAPYCMPCHG command will be invoked for each object being generated by a YGENSRC job. This will keep the effect of changes to model objects as up to date as possible. Note: *NONE and *GEN also imply *LIMITED authority to developer access to the component change processing flag on their model profile.
YCMPTXT	Text description of company using CA 2E. Used in banner of generated code. It is displayed on the right-hand side and header/footer.	TEXT (30)

YCMTCDE	Determines whether commented-out (Inactive) code in the Action Diagram will be included when determining Object Usages and References	*YES – Inactive code is included. *NO – Inactive code is not included *IGN - there is no differentiation between active code and inactive code.
YCPYLIB	Name of library from which old physical file data is to be copied.	Valid System Name *NONE
YCRTENV	Object creation environment: IBM i or System/38	QCL—Create environment for S/38. QCMD—Create environment for IBM i.
YCRTJBD	Name of job description to use when submitting compilations. Default to QBATCH.	VNM (10)—Valid job description name.
YDSTFIO	For SQL DRDA support, the distributed file I/O control option.	*SYNON—Synon provides for control rolling between databases. *USER—User provides for control rolling between databases. *NONE—No distributed functionality is generated.
YEXCENV	Runtime environment: Specifies default initial value for execution environment for EXCMSG functions.	QCL—Prepare execution for S/38. QCMD—Prepare execution for IBM i.
YGENLIB	Name of the generation library.	VNM (10) - Valid library name.
YGENRDB	For SQL DRDA support, the name of the default database. Used in the creation of the SQL package. This should typically correspond to the local database as defined in the IBM i RDB directory.	Valid database name.

YIGCCNV	Generate IGC (Kanji ideographic characters) support keywords in DDS.	0—Do not generate IGC support keywords 1—Generate IGC support keyword.
YLIBLST	Name of the model library list.	VNM (10)—Valid library list name. Users are advised to ensure that the CA 2E Toolkit data object YLIBLST is available in the library list of jobs running CA 2E commands.
YMDLLIB	Name of model library: must be the same as the library containing the model.	VNM (10)—Valid library name.
YMDLLNG	National language for system portion of model.	Valid model language code; for example *ENG.
YMDLTXT	Text description of the model. Use in banner of generated code.	TEXT (30)
YOLDLIB	Name of library into which old physical files are archived.	Valid System Name
YOPNACC	Value defines whether concurrent access by *DSNR(s) and *PGMR(s) is allowed.	*YES—Allow *DSNRs and *PGMRs in the model concurrently. *NO—Disallow concurrent *DSNR and *PGMR access.
YSQLLIB	Name of library in which to place the collection to implement the SQL database.	SQL Valid System name. *NONE—There is no SQL library associated with the model.
YTRGLIB	Specifies the name of the library into which model trigger references should be converted using the YCVTTRGDTA command.	*GENLIB, VNM(10)
YTRNAPI	Specifies the name of the API to be invoked at runtime by DSPFIL and SELRCD function types to convert non-key field filters to upper case values for comparison with subfile record field values.	VNM(8)

Help Generation Values

Model Value	Description	Allowed Values
YGENHLP	Determines whether help will be generated/created by default for functions in the model. This can be overridden by a function option. For Synon/TC the result depends on YGENGUI. See YGENHLP.	*YES—Generate help text (UIM or TM) with all other HLLs. *NO—Do not generate help text. *ONLY—Generate help text only (without other HLLs).
YGUIHLP	Default GUI Help Language. Indicates the language generated for online help on programmable workstation functions; namely, if the YWSNGEN model value is set to *GUI, *VB, or *JVA. This can be overridden by a function option.	*NONE—Do not generate GUI help. *WIN—Generate GUI help using Windows help. *HTML—Generate GUI help using HyperText Markup Language (HTML) for the Internet.
YHLPCSR	Specifies whether to generate cursor sensitive text.	*YES—Generate help text that is cursor sensitive. *NO—Do not generate cursor sensitive help text.
YNPTHLP	Determines what type of help text will be generated by default for NPT functions in the model.	*UIM—Generates help text in UIM (User Interface Manager) by default. *TM—Generates help text in TM (Text Management) by default. Both are displayed using CA 2E utility.
YUIMBID	Tells the UIM generator whether to specify bi-directional support for UIM.	*NONE—No bi-directional support for generated UIM panel groups. *LTR—Help is to be displayed with left-to-right orientation. *RTL—Help is to be displayed with right-to-left orientation.

YUIMFMT	Specifies the default formatting of narrative text, when included in generated UIM help.	*AUTO—Format the narrative by default with a UIM Paragraph tag (:P). *FIXED—Format the narrative by default with a UIM Lines tag (:L).
YUIMIDX	Defines the valid name used for Help Search Index when generating UIM keywords in all DDS source for the model and when generating calls to CA 2E Display Help API displaying UIM help.	*NONE—Do not generate any references to a search index. *VLSPFX—Use a name for the search index derived from the values list prefix.

YGENHLP

For Synon/TC applications, which source components get generated is determined by derivation of the YWSNGEN, YGENHLP, and YGENGUI model value as shown in this table:

Model Values			**Source Components Generated**			
YWSNGEN	**YGENHLP**	**YGENGUI**	**NPT HLL**	**NPT HELP**	**SDF**	**Windows Help**
*NPT	*YES	n/a	yes	yes	---	---
	*NO		yes	---	---	---
	*ONLY		---	yes	---	---
*GUI	*YES	*YES	yes	yes	yes	yes
		*NO	yes	yes	---	---
		*ONLY	---	---	yes	yes
	*NO	*YES	yes	---	yes	---
		*NO	yes	---	---	---
		*ONLY	---	---	yes	---
	*ONLY	*YES	---	yes	---	yes
		*NO	---	yes	---	---
		*ONLY	---	yes	---	yes

PWS Environment Values

Model Value	Description	Allowed Values
YGUIAKY	Suppress Aid Keys. Specifies whether description of the keys should be displayed or not on the pull-down menu. Aid-keys are the command keys defined functionally rather than graphically on the screen.	■ *NO—Do not suppress generation of aid-keys description on the pull-down menu; in other words, display aid keys. ■ *YES—Suppress generation of aid-keys description.
YGUICMD	Command Key Descriptor Text. Indicates how command key text appears on your green-screen device designs. You can specify up to two ways. The default is F,CF as in F3=Exit and CF3=Exit.	■ *NONE—No descriptor text is used. ■ text1,text2—Up to two ways in which command key text appears on your green-screen device designs, separated by a ‘,’; for example, ‘F,PF’, ‘,PF’. Each value may be up to five characters.
YGUIFLR	Application Folder for GUI. Name of the shared folder into which the Synon/TC source (SDF, SIF, ADF, MSF) is placed for processing on the PC. This must be a character string of up to six characters that follows the IBM i document path naming convention.	■ valid system name—Name of shared folder. This name defaults to the name of the model. ■ *NONE—There is no folder.

YGUILIB	Library Partitioning for GUI. Specifies the way in which SDF panels are partitioned into libraries within a Synon/TC application. There is a maximum of 1000 functions in a single library. Library names must be unique across all Synon/TC applications residing on the same PC. Names exceeding seven characters will be truncated.	■ valid library name—A valid PC directory name. This will place all GUI functions into a single library. ■ *AUTO—Library partitioning will be done automatically for each SDF panel using the shipped routine YALCLIBR1C. By default this builds the library name from the YGUIFLR value and the third character of the SDF name, allocating panels across libraries.
YGUILSO	Suppress List Options. Indicates whether subfile-select options should be displayed on the pull-down menu.	■ *NO—Do not suppress display of subfile-select descriptions on the pull-down menu; in other words, display subfile-select descriptions. ■ *YES—Suppress display of subfile-select descriptions.
YGUIOPR	Operator Between Prefix and Text. Indicates how a command key or subfile selector value is separated from the text that describes it on your green-screen device designs. The default is =,– as in F3=Exit and 5–Display.	■ *NONE—No operator is used. ■ text1,text2—Up to two ways in which a command key or subfile selector value is separated from the text that describes it on your green-screen device designs, separated by a comma; for example, ‘=,-’, ‘,--’. Each value may be up to two characters.

YGUISEL	Subfile Select Descriptor Text. Indicates how subfile selector text appears on your green-screen device designs. You can specify up to two ways. The default is Opt as in Opt1-Select.	■ *NONE—No descriptor text is used. ■ text1,text2—Up to two ways in which subfile selector text appears on your green-screen device designs, separated by a comma; for example, 'Opt,Sel', ',Sel'. Each value may be up to five characters.
YGUITKY	Display command keys per tab. Indicates how command keys should be displayed on the non-NPT implementation of functions using tab dialogs; namely, Edit Record 2 and 3 and Display Record 2 and 3.	■ *NO—Command keys are displayed once for the function, covering all tab dialogs. The command key text is taken from the first last detail screen. ■ *YES—Command keys are displayed on each tab dialog. The command key text is taken from each detail screen.

System Defaults

Model Value	Description	Allowed Values
YSYSCHG	Gives the default YCHGCTL model value for new models.	■ *NONE—Restricted exit programs are not called. Change control additional features are effectively off. ■ Valid library name.
YSYSDBF	Defines system database access method, either DDS and HLL or SQL Data Definition Language (DDL) and Data Manipulation Language (DML).	■ *DDS ■ *SQL
YSYSHLL	Default HLL given to new models. It is the value for YHLLGEN model value.	■ *RPG ■ *CBL

YSYSNPT	Gives the default YNPTHLP on YCRTMDLLIB.	■ *UIM—Generates help text in UIM by default. ■ *TM—Generates help text in TM by default.
YSYSPMT	Defines system prompts.	■ *OFF—This model is incapable of generating message IDs for screen text literals. ■ *MSGID—This model will default to generate external message IDs for every screen text constant. ■ *LITERAL—This model is capable of generating external message IDs, but will not default to do so. You can override at a function level.
YSYSSAA	Defines system default for SAA formatting option YSAAFMT.	■ *CUAENTRY—Defaults to a header/footer without windows or action bars. ■ *CUATEXT—Defaults to windows and action bar style header/footers. All function types with the exception of SELRCD will default to action bars. The SELRCD will default to a window. ■ *S38—Defaults to System/38 style header/footer.

Unchangeable Model Values

The following two tables show model values which cannot be changed with YCHGMDLVAL.

Protected Model Values

Modern Values	To change and use this command
YMDLLIB	YRNMMDL
YMDLLVL	YAPYMDLCHG
YMDLNBR	YAPYSYSMDL
YMDLLNG	YAPYMDLTRN
YPMTNBR	YCRTMDLLIB

You can change the following system values using the i OS CHGDTAARA command. The name of the data area is the model value name with the suffix RFA; for example, the name of the null model for the system is contained in the YNLLMDLRFA data area in the CA 2E base product library.

Unchangeable System Model Values

Modern value	Set by	Role
YNLLMDL	Install procedure	Defines the name of the null model.
YSYSLNG	Install procedure	Defines the language of the installed product.

Example

To change the design standard to CUATEXT:

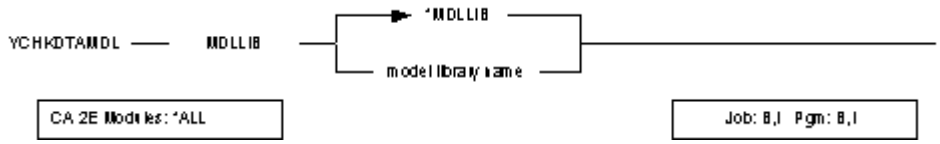
```
YCHGMDLVAL MDLVAL(YSAAFMT) + VALUE(*CUATEXT)
```

YCHKDTAMDL (Check Data Model) Command

Check data model is a process which reports on any integrity issues within a model.

Any integrity issues found are listed on the spooled output. The messages indicate the surrogate numbers that are in error, as well as the file in which the errors were found. Contact Product Support if any errors are reported by the Check Data Model command.

Optional



Parameters

The following are parameters for the YCHKDTAMDL command.

MDLLIB

Specifies the name of a CA 2E model to check for integrity. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value meaning that the first model library in the library list is to be used.
- **model library name**—The name of the model library over which an integrity check should be run.

Example

To examine objects in model library UURMDL for corruption:

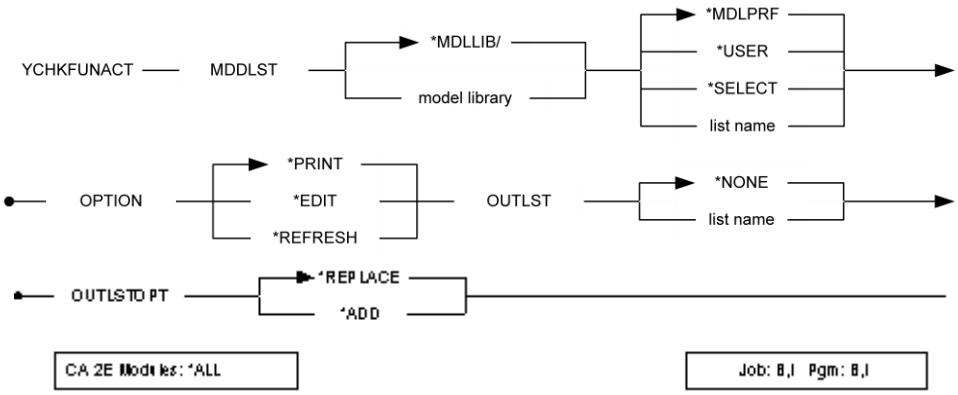
YCHKDTAMDL MDLLIB(UURMDL)

YCHKFUNACT (Check Function Action Diagram) Command

This command allows a user to analyze the action diagram of a list of functions. Only functions that appear on the input model list are processed.

A report provides details of any errors found. Functions that contain errors may optionally be added to an output model list.

Required



Parameters

The following are parameters for the YCHKFUNACT command.

MDLLST

The qualified name of the model object list that contains the functions to be checked. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The model object list name must be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—The model library name for the list.

OPTION

Specifies the action to be taken if an error is found in an action diagram. Values for this parameter are described in the following:

- ***PRINT**—(default) The error is written to a report.
- ***EDIT**—The action diagram of the function that has the error is to be edited. The program will position to the first offending action diagram block to allow the user to correct the problem.
- ***REFRESH**—The title of each action element of the action diagram is examined. If the title is out of date, for example because the referenced function or message (or field, in the case of built-in functions) has been renamed, it is refreshed.

OUTLST

The name of the model object list that receives output from the command. Values for this parameter are described in the following:

- ***NONE**—(default) Special value meaning that no outlist processing is to be performed.
- **list name**—The model object list name is to be used as output.

OUTLSTOPT

This parameter specifies the action taken if the output list already exists. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list should be replaced with the output from this command.
- ***ADD**—The existing model object list should be augmented with the output from this command.

Example

To check the action diagrams of all functions contained in the model object list APLIST in the current model library and print a report of all errors found:

YCHKFUNACT MDLLST(*MDLLIB/APLIST)

YCHKFUNPAR (Check Parameter Interfaces) Command

Use the YCHKFUNPAR (Check Parameter Interfaces) command to identify any functions that have duplicate parameter fields when the *Duplicate parameters* function option is set to *N*. Earlier, it was possible to enter duplicate parameters even when the *Duplicate parameters* function option was set to *N*. This command allows you to analyze an entire model and identify functions that violate the *Duplicate Parameters* restriction and exception. Executing the command produces a joblog that contains error messages identifying problematic functions.

Important! After gathering the results from the YCHKFUNPAR command, it is your responsibility to rectify any problems by modifying, as necessary, the parameter interfaces.

The command searches for two specific parameter interface problems:

- Invalid Duplicate Parameter field:

  For a given function (where function option *Duplicate parameters* is set to N) each parameter field must be unique, regardless of usage. The only exception is that a field can appear once for Input and once for Output.

  Sends error message (Y2V0719) to joblog.

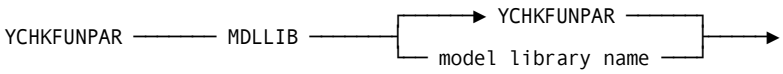
  Note: If a function has *Duplicate parameters* function option changed from Y to N, previously valid duplicate fields may now become invalid (however no warning or error will be sent). It is your responsibility to revisit the parameter interface and ensure that any invalid duplicate parameters are modified or removed accordingly. The YCHKFUNPAR can be used to identify such invalid duplicate parameters.

- Non-unique Parameter SEQ:

  When defining parameters on the EDIT FUNCTION PARAMETERS panel, SEQ does not need to be unique. This has no adverse impact when function option Duplicate Parameters = *N*. However, when function option Duplicate Parameters = *Y*, and two parameters are specified with the same sequence number, the same duplicate parameter context is used for both parameters. Therefore in the AD, a field can appear in a given duplicate parameters context more than once.

  Sends error message (Y2V0720) to joblog.

Optional



Parameter

The following parameter is for the YCHKFUNPAR command:

MDLLIB

The name of the model to be analyzed. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value indicating the model library is the first one in the current library list.
- **model library name**—The name of the model to be analyzed may be entered explicitly.

Examples

YCHKFUNPAR processing includes processing to provide feedback on the progress. If the job is running interactively, a message is sent that indicates the current object being processed and the percentage complete. For example:

```
Processing Object 837 of 2476. 33% Completed.
```

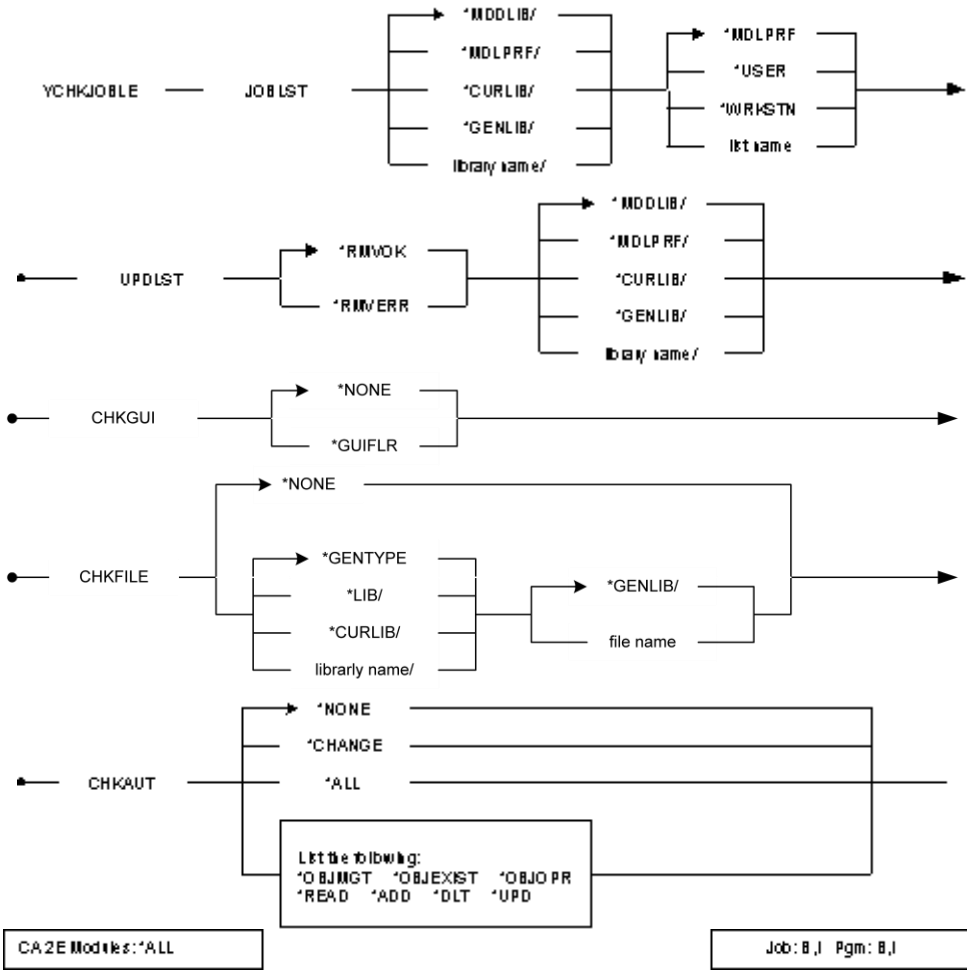
If the job is running interactively or in batch, a message is sent to the joblog to indicate the total number of objects to be processed, and a message is sent when every 10% block is reached. For example:

```
YCHKFUNPAR (MYMDL)
Total number of objects to process: 169
Processing Object 1 of 169. 0% Completed.
Processing Object 17 of 169. 10% Completed.
Processing Object 34 of 169. 20% Completed.
Processing Object 51 of 169. 30% Completed.
Processing Object 68 of 169. 40% Completed.
Invalid duplicate parameter field on Function 'CHANGE BOOKING' (@@1100454) on File
'BOOKING'. Processing Object 85 of 169. 50% Completed.
Invalid duplicate parameter field on Function 'CANCEL BOOKING' (@@1100541) on File
'BOOKING'.
Non-unique parameter SEQ on Function 'PREPARE LOGIN'  (@@1100562) on File 'SECURITY'.
Processing Object 102 of 169. 60% Completed.
Invalid duplicate parameter field on Function 'PROCESS BILL' (@@1100624) on File
'BILL'.
Invalid duplicate parameter field on Function 'PROCESS TICKET' (@@1100645) on File
'TICKET'.
Processing Object 119 of 169. 70% Completed.
Non-unique parameter SEQ on Function 'VALIDATE ORDER' (@@1100742) on File 'ORDER'.
Processing Object 136 of 169. 80% Completed.
Processing Object 153 of 169. 90% Completed.
Processing Object 169 of 169. 100% Completed.
Non-unique parameter SEQ on Function 'CHECK CUSTOMER'  (@@1100872) on File 'CUSTOMER'.
4 functions identified that contain an invalid duplicate parameter field.
3 functions identified that contain non-unique parameter SEQ & DUP PARMS=Y.
Model MYMDL examined to identify functions with invalid parameter interfaces. See job
log for details.
```

YCHKJOBLE (Check Job List Entries) Command

Checks whether a source member or i OS object exists for each entry on a job list. Checks your authority to each i OS object found. You can also use the command to check your authority level to each i OS object found. If the object already exists, it is dropped from your job list.

Optional



Parameters

The following are parameters for the YCHKJOBLE command.

JOBLST

Qualified name of a job list where you place entries. Name of the job list whose entries are checked. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The job list name is retrieved from the model profile details for the current user.
- ***WRKSTN**—Defaults job name to job list name of invoking job.
- ***MDLLIB/**—Defaults job list name to user profile name in model library.

UPDLST

List update option. Values for this parameter are described in the following:

- ***RMVOK**—(default) Remove verified items from list; that is, if the corresponding source member (CHKFILE parameter), i OS object (CHKLIB parameter), or specified authorities (CHKAUT parameter) are found.
- ***RMVERR**—Remove items from list that are not verified.

CHKLIB

Name of the library to use when checking for the existence of i OS objects corresponding to each job list entry. Values for this parameter are described in the following:

- ***NONE**—(default) Do not check for any object's existence.
- ***GENLIB**—Check for objects in the generation library of the model (YGENLIB model value).
- ***CURLIB**—Check for objects in the current library.
- ***LIBL**—Use the current job's library list.

CHKGUI

Name of shared folder to check for the existence of Synon/TC source objects corresponding to each job list entry. Values for this parameter are described in the following:

- ***NONE**—(default) Do not check for any object’s existence.
- ***GUIFLR**—Check for objects in the Synon/TC shared folder of the model (YGUIFLR).

CHKFILE

Qualified source file name to check for the existence of a source member corresponding to each job list entry. Values for this parameter are described in the following:

- ***NONE**—(default) Do not check for a corresponding source member.
- ***GENTYPE**—Use the default source file name as specified by the edit generation types display; for example, QRPGSRC for RPG III source, QCLSRC for CL source.

CHKAUT

i OS authorities to check after checking for object or source member existence, as specified by the CHKLIB or CHKFILE. The value for this parameter is as follows:

- ***NONE**—Do not check for authorities when checking existence of corresponding source members or i OS objects.

For more information on i OS authority values, refer to the IBM *CL Programmers* guide.

Notes

The job list must already exist. Job lists are created using either the command Build Job List **(**YBLDJOBLST**)** or Convert Model List **(**YCVTMDLLST**)** from a command line, or by prompting the command from within a design model.

Example

To check for the existence of i OS objects in the generation library corresponding to the entries on job list MYJOBLST, removing all the entries for which a corresponding i OS object is found:

```
YCHKJOBLE UPDOPT(*RMVOK) +
CHKLIB(*GENLIB) JOBLST(MYJOBLST)
```

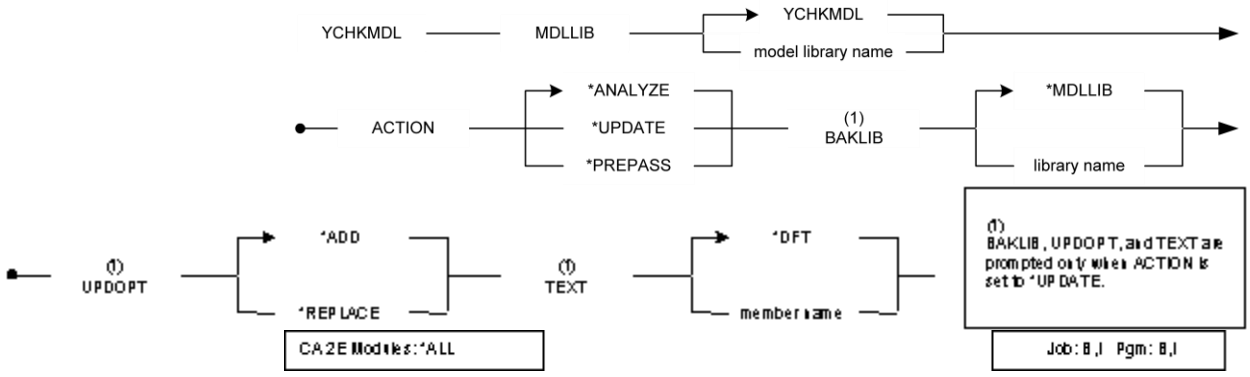
YCHKMDL (Check Model) Command

This command checks your model for inactive internal records and unresolvable model object references. These can occur, for example, if a power outage results in an incomplete update or if a developer makes an error while using YWKRF to change model file data.

The ACTION parameter lets you specify whether the internal files or the objects in your model are to be checked. It also lets you specify whether the command will run in prepass or update mode. All errors found are reported, and in update mode, inactive internal records are backed up to files in a library you specify before being deleted from the model.

Note: If you need to restore data from a backup file, you should take great care to specify the correct member name. It will be the same as the name of your model library. When copying data back, be sure to specify *ADD for the MBROPT parameter on the Copy File (CPYF) command.

Optional



Parameters

The following are parameters for the YCHKMDL command.

MDLLIB

The name of the model to be analyzed. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value indicating the model library is the first one in the current library list.
- **model library name**—The name of the model to be analyzed may be entered explicitly.

ACTION

Specifies the type of check to perform and whether to run in prepass or update mode. A report is produced listing any data in error. Values for this parameter are described in the following:

- ***ANALYZE**—(default) The command checks each object in the model and all objects that use the object (usages) and all objects that it refers to (references). This can be a lengthy process for large models. No update is performed.

This option produces a separate report that shows any unresolvable references that are encountered during the expansion of a given object; for example, if a model object refers to an object that has been deleted from the model.

If the report shows any errors, first attempt to remove them using *UPDATE and run the *ANALYZE analysis again. If there are any entries remaining after running the second analysis, please contact your local product support representative who will assist you with their removal.

- ***PREPASS**—The command analyzes those internal files that can automatically be updated in *UPDATE mode and reports any inactive records. No actual update is performed.
- ***UPDATE**—The command performs the same analysis as in the *PREPASS mode, and also deletes inactive records from the internal files.

BAKLIB

The name of the library that is to receive backup data.

The data is written to a file that has a name similar to that of the originating file; the member name is the same as the name of your model library. For example, an inactive record in the YSCRENTRFP file (screen or report device entries) in model library MYMODEL, would be written to file YSCRENTXXP and member MYMODEL in the backup library. You can therefore specify a single backup library to store all deleted data from all the models on the system.

This parameter is prompted only in *UPDATE mode. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value indicating the model library is to be the backup library.
- **library name**—The name of the backup library may be entered explicitly.

UPDOPT

This parameter specifies whether new data written to the backup members should replace or be added to existing data. This parameter is prompted only in *UPDATE mode. Values for this parameter are described in the following:

- ***ADD**—(default) Special value indicating that the new data should be added to any existing data in the backup members.
- ***REPLACE**—The existing data should be cleared during processing.

TEXT

The text to be attached to any new members added to the backup files. This parameter is prompted only in *UPDATE mode. Values for this parameter are described in the following:

- ***DFT**—(default) Special value indicating that a standard message will be used as new member text.
- **member text**—The user may specify the text to be used when adding new members to the backup files.

Example

To examine internal files in model library GR5MDL and to produce a report without updating data:

```
YCHKMDL  MDLLIB(GR5MDL)  ACTION(*PREPASS)
```

Additional Processing

YCHKMDL processing now includes some additional processing to provide feedback on the progress of YCHKMDL.

- If the job is running interactively, a message is sent to the screen to indicate the current object being processed and the percentage complete. For example:

Processing Object 837 of 2476. 33% Completed.

- If the job is running interactively *or in batch*, a message is sent to the joblog to indicate the total number of objects to be processed, and a message is sent when every 10% block is reached. For example:

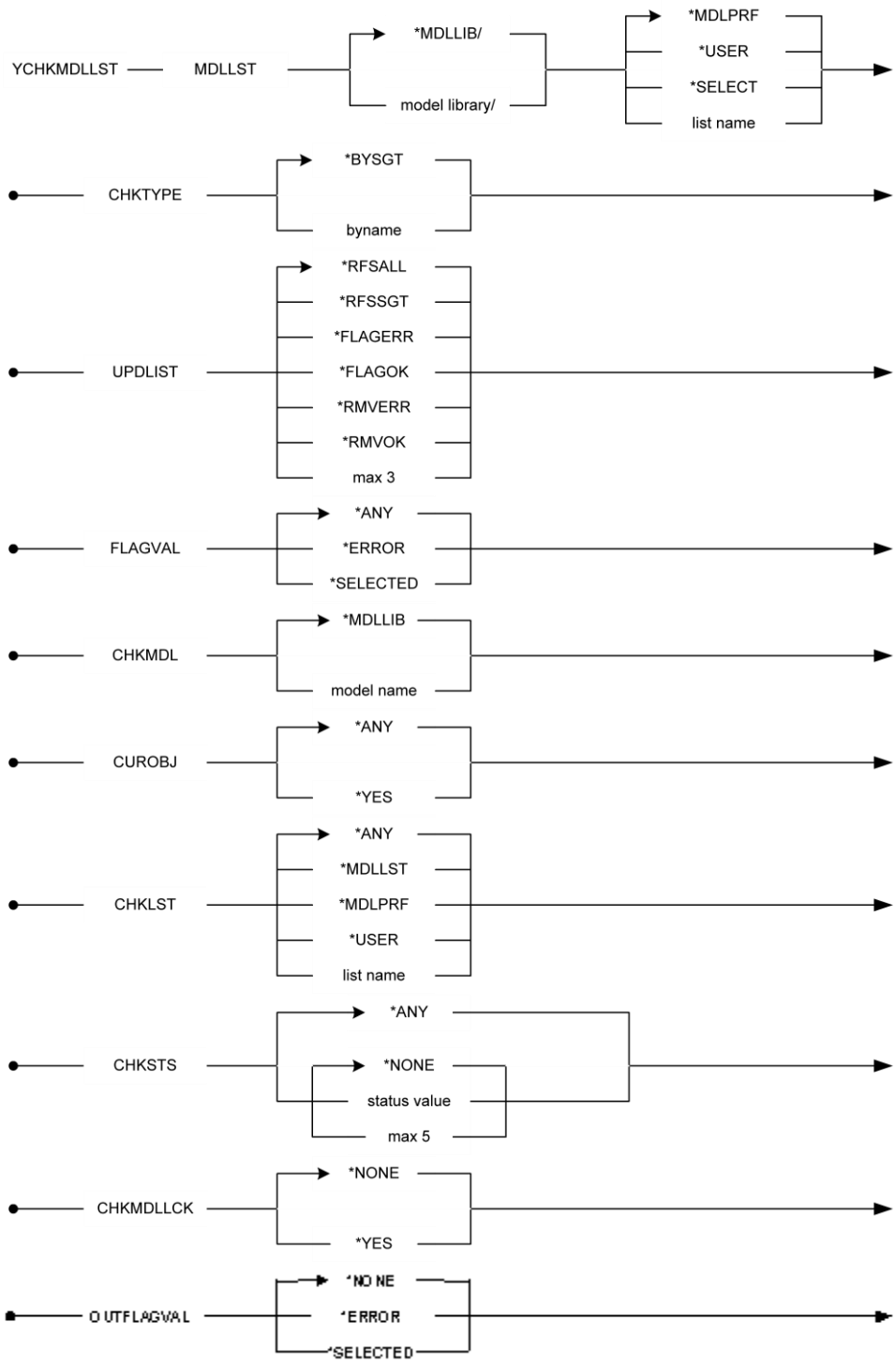
Total number of objects to process: 2476
Processing Object 1 of 2476. 0% Completed.
Processing Object 248 of 2476. 10% Completed.
Processing Object 496 of 2476. 20% Completed.
Processing Object 743 of 2476. 30% Completed.
Processing Object 991 of 2476. 40% Completed.
Processing Object 1238 of 2476. 50% Completed.
Processing Object 1486 of 2476. 60% Completed.
Processing Object 1734 of 2476. 70% Completed.
Processing Object 1981 of 2476. 80% Completed.
Processing Object 2229 of 2476. 90% Completed.
Processing Object 2476 of 2476. 100% Completed.

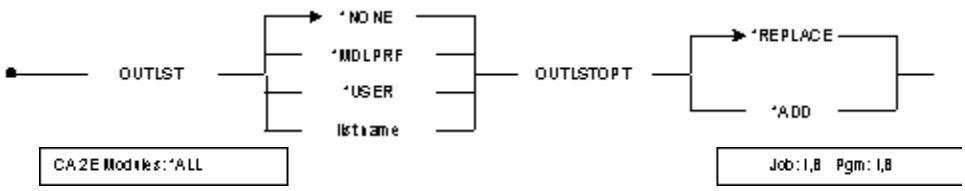
YCHKMDLLST (Check a Model Object List) Command

This command allows a user to simultaneously check model object list entries and to remove certain, unwanted entries. The command also allows entry details to be refreshed from the originating object.

An outlist capability allows users to filter entries to an alternative list leaving the input list unchanged by the filtering process. In other words, if you specify an output list, the input list will not be updated.

Optional


YCHKMDLLST
MDLLST
*MDLLIB/
model library/
*MDLPRF
*USER
*SELECT
list name
CHKTYPE
*BYSGT
byname
UPDLIST
*RFSALL
*RFSSGT
*FLAGERR
*FLAGOK
*RMVERR
*RMVOK
max 3
FLAGVAL
*ANY
*ERROR
*SELECTED
CHKMDL
*MDLLIB
model name
CUROBJ
*ANY
*YES
CHKLST
*ANY
*MDLLST
*MDLPRF
*USER
list name
CHKSTS
*ANY
*NONE
status value
max 5
CHKMDLLCK
*NONE
*YES
OUTFLAGVAL
*NONE
*ERROR
*SELECTED



Parameters

The following are parameters for the YCHKMDLLST command.

MDLLST

The qualified name of the model object list that is checked. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the model profile details for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The name of the list.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the model library.

CHKTYPE

Check list entry method. Values for this parameter are described in the following:

- ***BYSGT**—(default) The list entries are checked by object surrogate number.
- ***BYNAME**—The list entries are checked by object name.

UPDLST

List update option. Values for this parameter are described in the following:

- ***RFSALL**—(default) The data contained in each list entry are refreshed from the corresponding model object details.
- ***RFSSGT**—Only the surrogate number of each list entry is refreshed from the corresponding model object details. The CHKTYPE parameter must be *BYNAME for this option.
- ***FLAGERR**—If an error occurs during processing of a list entry, the entry is flagged. The value to use is specified by the OUTFLAGVAL parameter. This value may be the subject of a later filtering operation. Note that if entries are flagged, then the OUTFLAGVAL parameter must be set to the desired value.
- ***FLAGOK**—If an error does not occur during processing of a list entry, the entry is flagged. The value to use is specified by the OUTFLAGVAL parameter. This value may be the subject of a later filtering operation.
- ***RMVERR**—If an error occurs during processing of a list entry, the entry is removed from the list.
- ***RMVOK**—If the processing of a list entry completes without error, the entry is removed from the list.

Note: The updating of model lists is controlled by the UPDLST parameter. For example, it must specify *RMVERR if invalid entries are deleted from the input list or omitted from copy to the output list. Users should also note that the flagging of entries occurs if the UPDLST parameter instructs the program to do so. However, the appropriate value to use should be specified in the OUTFLAGVAL parameter to avoid the default value of blank.

When an output list is specified the input list is not updated.

If *RMVERR is specified together with an output list, it has the effect of not copying input list entries to the output list. Such entries are not removed if they already exist on the output list.

FLAGVAL

This parameter allows model object list entries to be filtered by the model object list entry flag value. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on flag value is to be performed.
- ***ERROR**—Only entries flagged as being in error satisfy the filter. Other entries are ignored.
- ***SELECTED**—Only explicitly selected model objects satisfy the filter. Other entries are ignored.

CHKMDL

Check model library name. This parameter enables objects from one model to be checked against objects of the same name in another model. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The list entries are checked against objects in the same model.
- **model name**—The model that entries are checked against.

CUROBJ

This parameter identifies whether the objects on the list are current. Values for this parameter are described in the following:

- ***ANY**—(default) Objects are not checked.
- ***YES**—Only current objects satisfy the check.

CHKLST

The checked out to list may be examined with this parameter. Values for this parameter are described in the following:

- ***ANY**—(default) Any value for checked out to list is acceptable.
- ***MDLLST**—List entries must be checked out to the current list.
- ***MDLPRF**—List entries must be checked out to the model object list specified on the current user's model profile.
- ***USER**—Entries must be checked out to the model list that has the same name as the current user.
- **list name**—A list name may be specified.

CHKSTS

The check out status of list entries may be checked. Values for this parameter are described in the following:

- ***ANY**—(default) Check out status is not examined.
- ***NONE**—Special value meaning that entries should have no value for check out status.
- **status value**—User-defined value. Up to five values may be specified for check out status.

CHKMDLLCK

This parameter allows the lock status of each object on the list to be examined. Values for this parameter are described in the following:

- ***NONE**—(default) No examination of object locks is made.
- ***YES**—Objects on the list must not be locked.

OUTFLAGVAL

This parameter specifies the value used in combination with the UPDLST parameter. Selected list entries are updated with this value if UPDLST is specified as *FLAGERR or *FLAGOK. Values for this parameter are described in the following:

- ***NONE**—(default) No updating of the flag value is performed.
- ***ERROR**—Entries which satisfy the filtering criteria are flagged as *ERROR if UPDLST is specified as *FLAGERR or *FLAGOK.
- ***SELECTED**—Entries that satisfy the filtering criteria are flagged as selected if UPDLST is specified as *FLAGERR or *FLAGOK.

OUTLST

The name of the model object list which is the target for entries that satisfy any filter. Values for this parameter are described in the following:

- ***NONE**—(default) Special value meaning that no outlist processing is performed.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the model profile record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name.
- **list name**—The name of the target list.

OUTLSTOPT

This parameter specifies the action taken if the outlist specified in the OUTLST parameter already exists in the model. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list is replaced with the output from this command.
- ***ADD**—The existing model object list is augmented with the output from this command.

Notes

- The input model object list must already exist prior to running this command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- If an error is detected during processing of the list entries, a diagnostic message is sent for each error encountered, and an escape message is sent at the end of processing. When using this command in CL programs, users may monitor for the escape message which is sent. The message id for this message is Y2E0328.
- The OUTLSTOPT parameter is ignored if *NONE is specified for OUTLST. If the specified OUTLST does not exist prior to running this command, it is created.
- If the CHKMDL parameter is specified (i.e., other than *NONE) then the CHKTYP must specify *BYSGT, since it is not possible to compare objects between models by surrogate key number.

Example

To refresh model object list OLDLST so that all the list entries are completely up to date:

```
YCHKMDLLST MDLLST( *MDLLIB/OLDLST ) + UPDLST( *RFSALL )
```

To check model object list FUNLST to determine that all objects on the list still exist in the model:

```
YCHKMDLLST MDLLST( *MDLLIB/FUNLST ) + UPDLST( *RMVERR )
```

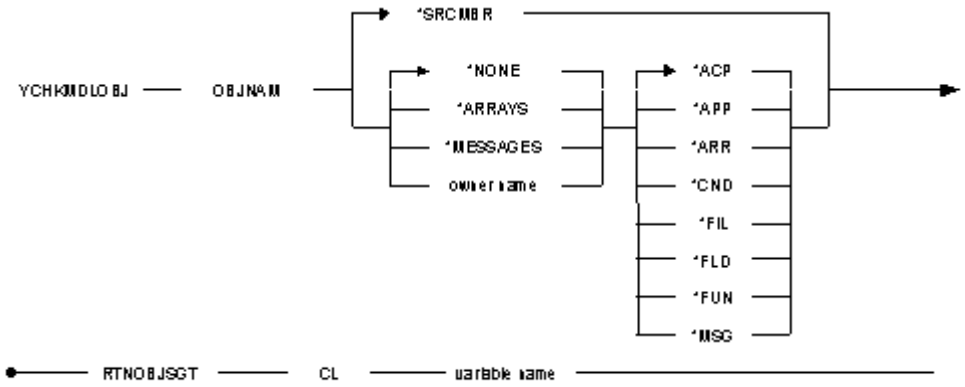
Using the *RMVERR option for UPDLST will ensure that only valid objects remain on the list.

YCHKMDLOBJ (Check Existence of Model Object) Command

This command provides the means by which the existence of an object may be established.

Objects may be identified by name or by implementation name.

Required



Parameters

The following are parameters for the YCHKMDLOBJ command.

OBJNAM

The object name of the object whose existence is checked. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- *SRCMBR—(default) Single value indicating that the object implementation name (source member) is used to identify the model object to be checked.
- object owner name—The character name of the object which owns the object to be checked. Thus, for a function, the owning file would be entered.
- *NONE—Special value indicating that the object concerned does not have an owner (such as objects of type *FIL).
- *ARRAYS—Special value for the product internal file *ARRAYS.
- *MESSAGES—Special value for the product internal file *MESSAGES.
- object name—The character name of the object to be checked.
- object type—The object type of the object.
- *ACP—Object is of type access path.
- *APP—Object is of type application area.
- *ARR—Object is of type array.
- *CND—Object is of type condition.
- *FIL—Object is of type file.
- *FLD—Object is of type field.
- *FUN—Object is of type function.
- *MSG—Object is of type message.

SRCMBR

The source member implementation name of the model object whose existence is checked. Values for this parameter are described in the following:

- *OBJNAM—(default) Use object name to identify the model object to be checked.
- source member name—The source member name of the model object.
- implementation object type—The type of object created by the source member. This value may be required to distinguish between objects that have the same implementation name, such as a display file in a model that has the same name as a program.
- *ANY—(default) The object type is unspecified.
- *PGM—The object is a program.
- *FILE—The object is an access path, display file or print file.
- *HLPTXT—The object is a help text file.

MDLLIB

This parameter allows the user to specify the model library in which to find the object. Values for this parameter are described in the following:

- *MDLLIB—(default) Use the first model library in the job's current library list.
- model library—The name of the model library to use.

RTNOBJSGT

CL variable that is to receive the surrogate key of the object. Decimal variable, seven digits packed.

Notes

- A search is made for the model object in the first model library found in the library list. If the object is found, the command completes normally. If the object is not found, an exception message (Y2EO311) is sent.
- If the object being checked is not a primary object (such as the display file or help file for a panel based function), the surrogate number of the owning object is returned. Thus, for a display file auxiliary object, the function surrogate number would be returned.

Example

To check whether the function Edit Order version 2 exists in the model, enter the following command:

```
YCHKMDLOBJ OBJNAM( 'Order' 'Edit Order + version  2' *FUN )
```

To check whether the function UUG2EFR exists in data model DEVMDL, and, if it does, to receive the corresponding object surrogate key number in parameter &OBJSGT:

```
YCHKMDLOBJ SRCMBR( UUG2EFR ) + MDLLIB(DEVMDL) RTNOBJSGT( &OBJSGT )
```

Chapter 4: Commands (YCLRMDL - YDLTOBJTBL)

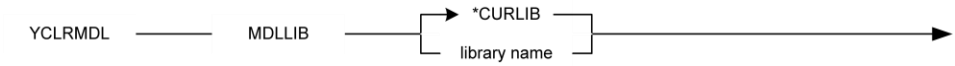
This chapter contains details for CA 2E commands YCLRMDL through YDLTMDLVSN. These commands appear in alphabetical order and include descriptions of their functions, parameters and allowed values, notes, and examples. Each command is also accompanied by a command diagram.

This section contains the following topics:

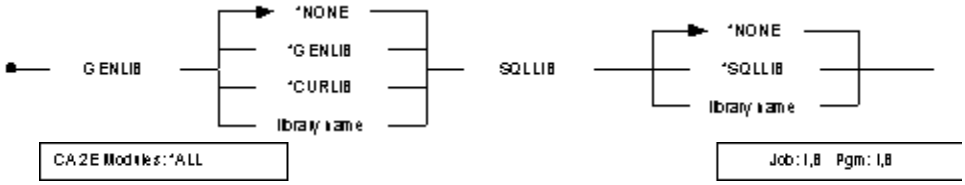
YCLRMDL (Clear Model) Command

Clears a model. All user-defined data is dropped leaving a copy of the null model data but retaining your existing model values.

Required



Optional



Notes

- When you clear a model, the CA 2E null model (Y2SYMDL) must be present. If you specify a library name for the GENLIB parameter, all programs and files in the library beginning with the system prefix are deleted (except those beginning with "Q").
- Model profiles are not cleared by the YCLRMDL command. To clear model profiles, you should clear file YMDLPRFRFP in the model.

Parameters

The following are parameters for the YCLRMDL command.

MDLLIB

Library containing a design model you want to clear. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

GENLIB

Library containing generated source to be cleared. Values for this parameter are described in the following:

- ***NONE**—(default) Do not clear the generation library.
- ***GENLIB**—Clear the generation library whose name is given by the YGENLIB model value.
- ***CURLIB**—Use current library for invoking job.

SQLLIB

Library containing the SQL collection to be cleared. Values for this parameter are described in the following:

- ***NONE**—(default) Do not clear the collection library.
- ***SQLLIB**—Clear the library containing the SQL collection given by the YSQLLIB model value.

Example

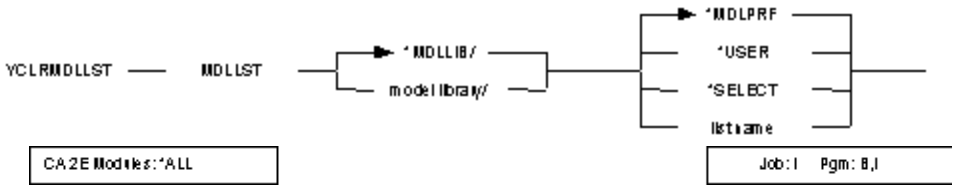
To clear model MYMDL, but not the generation library, enter:

```
YCLRMDL MDLLIB(MYMDL)
```

YCLRMDLLST (Clear a Model Object List) Command

This command clears the entries on a model object list.

Optional



Parameters

The following are parameters for the YCLRMDLLST command.

MDLLST

The qualified name of the model object list that is to be cleared. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The model object list name can be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—Model library name for the list.

Example

To clear entries from model object list LST001 from the model contained in the current library list:

```
YCLRMDLLST MDLLST( *MDLLIB/LST001 )
```

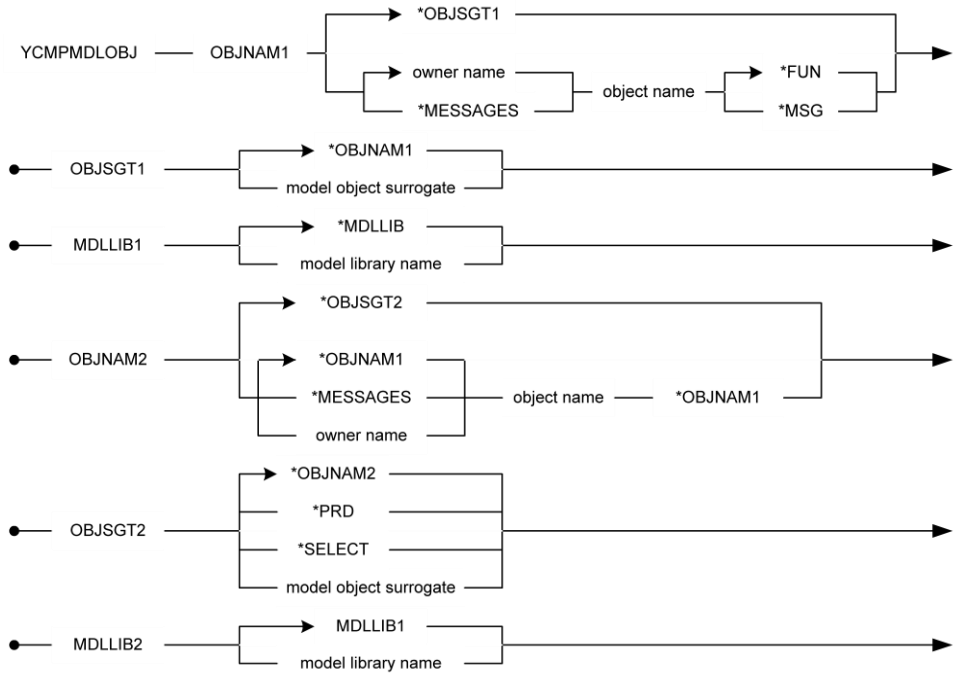
YCMPMDLOBJ (Compare Model Objects) Command

This command provides the means by which two model objects may be compared. This facility would be required, for instance, to identify changes made to one version of an object for retrofitting to another version. For example, you could use this command to identify the differences between versions of a model object.

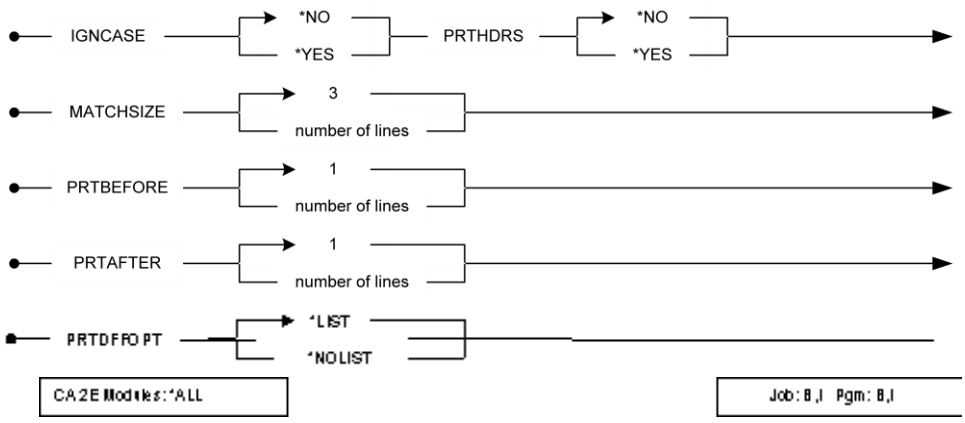
Objects may be identified by name or by object surrogate number.

At present, you can compare functions, messages or files (FIL) only.

Required



Optional



Parameters

The following are parameters for the YCMPMDLOBJ command.

OBJNAM1

The first object name of the two objects which are to be compared. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT1**—(default) Single value indicating that the first object surrogate number parameter is to be used to identify the model object to be compared.
- **object owner name**—The character name of the object which owns the object to be compared. Thus, for a function, the owning file would be entered.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- **object name**—The character name of the object to be compared.
- **object type**—The object type of the object.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

OBJSGT1

Unique number identifier of the first model object that is compared. Values for this parameter are described in the following:

- ***OBJNAM1**—(default) Use the first object name parameter details to identify the first model object.
- **object surrogate**—The surrogate number of the first model object.

MDLLIB1

This parameter identifies the model in which the first object resides. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value indicating that the first model in the library list is used.
- **model name**—The name of the model.

OBJNAM2

The name of the second object that is compared. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT2**—(default) Single value indicating that the second object surrogate number parameter is used to identify the model object to be compared.
- **object owner name**—The character name of the object that owns the second object to be compared.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- ***OBJNAM1**—(default) Special value indicating that the second object name is the same as the first object name.
- **object name**—The character name of the object to be compared.
- ***OBJNAM1**—(default) Special value indicating that the second object type is the same as the first object type. This is a required value. That is, the comparison of different object types is not supported.

OBJSGT2

Unique number identifier of the second model object to be compared. Values for this parameter are described in the following:

- ***OBJNAM2**—(default) Use the second object name parameter details to identify the second model object.
- ***PRD**—Use the surrogate number of the production version of the group to which OBJSGT1 belongs.
- ***SELECT**—Invoke a selection program so you can select a version from the group to which OBJSGT1 belongs.
- **object surrogate**—The surrogate number of the second model object.

MDLLIB2

This parameter identifies the model in which the second object resides. Values for this parameter are described in the following:

- ***MDLLIB1**—(default) Special value meaning that the model is the same as that specified on the MDLLIB1 parameter.
- **model name**—The name of the model.

IGNCASE

Case sensitivity option. Values for this parameter are described in the following:

- ***NO**—(default) Treat upper and lower case text as different when comparing object details.
- ***YES**—Ignore any differences between text which is the result of a case difference only. For example, the character x will match with X.

PRTHDRS

This parameter controls the printing of headings and action diagram user points during processing. Values for this parameter are described in the following:

- ***NO**—(default) Headings will not be printed. If there are no differences between the two objects compared, no spool file will be produced.
- ***YES**—Headings will be printed. A spool file will be produced even if there are no significant differences between the two objects. The forced printing of headings is useful when there is a difference between two large action diagrams as it identifies the user point in which the difference occurs.

MATCHSIZE

The number of lines required for a match. The value for this parameter is as follows:

- **3**—(default) Three lines must match.

PRTBEFORE

The number of preceding lines to be printed when a mismatch is encountered. The value for this parameter is as follows:

- **1**—(default) One line is to be printed before the mismatch.

PRTAFTER

The number of following lines to be printed when a mismatch is encountered. The value for this parameter is as follows:

- **1**—(default) One line is to be printed after the mismatch.

PRTDFFOPT

Indicates whether differences are to be reported. Values for this parameter are described in the following:

- ***LIST**—(default) Print the mismatched lines.
- ***NOLIST**—No mismatch details are to be reported. Simply indicate whether or not the objects match.

Notes

- Each object must exist in the specified model(s) prior to running the command.
- If the two objects are found to differ, an exception message is returned to notify the user.
- Model objects can either be identified by object name (OBJNAM) or by object surrogate key number (OBJSGT). If the OBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).
- Users of this command should note that when applied to a function which has an action diagram, commented code is not ignored.

Example

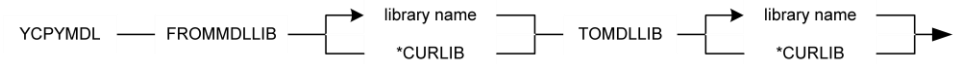
To check whether the function Display Vehicle Dtls #1 differs from Display Vehicle Dtls #2, ignoring any case differences in the object definitions, enter the following command:

```
YCMPMDLOBJ OBJNAM1( 'Vehicle' 'Display + Vehicle Dtls #1' *FUN )
OBJNAM2( *OBJNAM1 + 'Display Vehicle Dtls #2' *OBJNAM1 ) + IGNCASE( *YES )
```

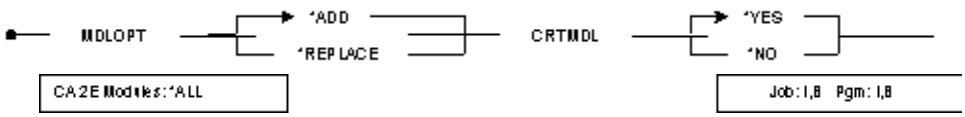
YCPYMDL (Copy Model) Command

Copies an entire model library to a new model or to an existing model.

Required



Optional



Parameters

The following are parameters for the YCPYMDL command.

FROMMDLLIB

Name of design model you want to copy. Must be a model library. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

TOMDLLIB

Name of model library to which the model is to be copied. If the name of an existing library is specified, then it must be a valid design model; that is, it must have been created with the YCRTMDLLIB command. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

MDLOPT

Model copy option. Values for this parameter are described in the following:

- ***ADD**—(default) Add to the contents of a new model. This option is only allowed if *YES is specified for the CRTMDL parameter.
- ***REPLACE**—Replace any existing contents of the model named by the TOMDLLIB parameter.

CRTMDL

Model create option. Values for this parameter are described in the following:

- ***YES**—(default) If the model specified by the TOMDLLIB parameter does not exist, create it.
- ***NO**—If the model specified by the TOMDLLIB parameter does not exist, do not copy the model.

Notes

- You cannot copy a model while it is in use.
- The YCPYMDL command does not create a CA 2E generation library. If you have created a model using the CRTMDL(*YES) parameter of YCPYMDL, you will need to create the generation library yourself.
- To create a generation library:

YCRTMDLLIB MDLLIB(*NONE) GENLIB(NEWGEN)

- If you wish to copy objects from the generation library of the copied model:

YCRTDUPOBJ OBJ (*ALL) +
FROMLIB (old generation library) +
OBJTYPE (*ALL) +
TOLIB (new generation library) +
DATA (*YES) CRTOPT (*ALL)

- The YCPYMDL and YCRTDUPOBJ commands leave your new model library with settings from your old model for some of the Environment model values. For example:
  - **YGENLI**—Generation Library
  - **YSQLLIB**—SQL Collection Library
  - **YCPYLIB**—Copy Library
  - **YOLDLIB**—Old Library
  - **YCMPTXT**—Company Text

  Use the YCHGMDLVAL command to change the settings of these model values in your new model as needed.

  For more information on the Environment model values, refer to this chapter, the YCHGMDLVAL command.

Example

To copy library MYOLDMDL to MYNEWMDL, creating a new model:

```
YCPYMDL FROMMDLLIB(MYOLDMDL) + TOMDLLIB(MYNEWMDL) CRTMDL(*YES)
 YCRTMDLLIB MDLLIB(*NONE) GENLIB(MYNEWGEN)+ SYSTEXT(My new model)
YWRKLIBLST LIBLST(MYNEWMDL)
```

Add the generation library name to the list. Save the library list and the job description on exit.

```
YCHGMDLVAL MDLVAL(model value name) + VALUE(new value)
```

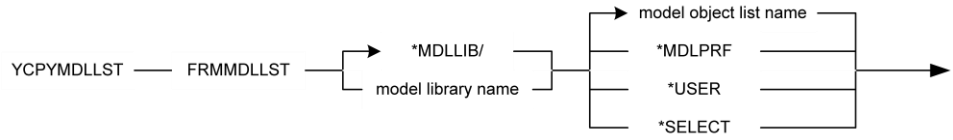
If needed, update Environment model values for the new model.

YCPYMDLLST (Copy Model Object List) Command

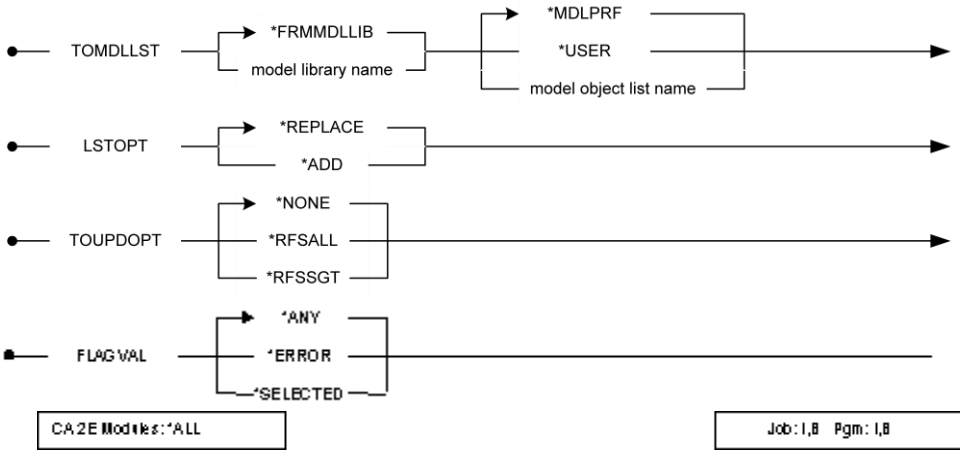
This command allows the user to copy from one model object list to another. It is also possible to copy from one data model to another.

If the target list exists prior to the copy it can be replaced or appended to. At the time of the copy, list entries can be refreshed in the target list and optionally flagged for further processing.

Required



Optional



Parameters

The following are parameters for the YCPYMDLLST command.

FRMMDLLST

The qualified name of the list that is copied. Values for this parameter are described in the following:

- **list name**—The name of the list to copy.
- ***MDLPRF**—Special value meaning that the name of the list is obtained from the model profile details of the current user.
- ***USER**—Special value indicating that the list to copy has the same name as the current user.
- ***SELECT**—Special value meaning that the list to copy is selected from an interactive display.
- ***MDLLIB**—(default) Special value meaning that the first model library in the library list is used.
- **model library name**—The name of the model library may be entered.

TOMDLLST

The qualified name of the list that is the target of the copy operation. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the name of the list is obtained from the model profile details of the current user.
- ***USER**—Special value indicating that the list to copy has the same name as the current user.
- **list name**—The name of the target list.
- ***FRMMDLLIB**—(default) Special value meaning that the target model object list exists or should be created in the same data model library as the list to copy.
- **model library name**—The name of the target model object list model library.

LSTOPT

This parameter specifies the action to be taken if the list already exists in the model. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list is replaced with the output from this command.
- ***ADD**—The existing model object list is augmented with the output from this command.

TOUPDOPT

To list update option. Values for this parameter are described in the following:

- ***NONE**—(default) No updates are made to the output list entries. This value is only valid if the target model library is the same as the source model library.
- ***RFSALL**—The data contained in each list entry are refreshed from the corresponding model object details in the target model.
- ***RFSSGT**—The surrogate number of each list entry is refreshed from the corresponding model object details.

FLAGVAL

Specifies the flag value of entries processed from the input list during the copy. Values for this parameter are described in the following:

- ***ANY**—(default) No record selection is performed.
- ***SELECTED**—Only entries flagged as being selected are processed.
- ***ERROR**—Only entries flagged in error are processed.

Notes

- The library for the input list must be a valid model library.
- A value other than *MDLLIB for FRMMDLLST may result in the library list being changed during execution of the command. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- The FRMMDLLST must exist prior to running the command. The target list will be created if it does not already exist.
- The LSTOPT parameter will be ignored if the target list does not already exist.
- The refreshing of list entries is always by name. That is, the details for each entry are ascertained by attempting to find the object by name in the model object file of the target model. If the object cannot be found, the entry in the input list is flagged in error and the object is not copied to the target list.
- If the object already exists in the target list, no action is taken to refresh the entry.
- The refresh surrogate number option (*RFSSGT) refreshes the surrogate number, object type, object attribute, owner surrogate and owner name of the new object entry in the target list.
- The completion message for this command includes a count of the number of records that already existed in the target list, the number of records copied and the number of errors, where objects from the input list could not be found in the target model.
- If any errors were encountered, a diagnostic is sent for each one indicating the reason for the error and at the end of processing an escape message is sent, for which developers may monitor.

Example

To copy model object list WRKLST from the model contained in the current library list to a model object list with the same name as the current user:

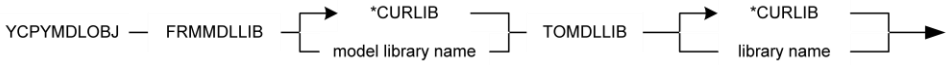
```
YCPYMDLLST  FRMMDLLST(*MDLLIB/WRKLST) +    TOMDLLST(*MDLLIB/*USER) LSTOPT(*REPLACE)
```

YCPYMDLOBJ (Copy Model Objects) Command

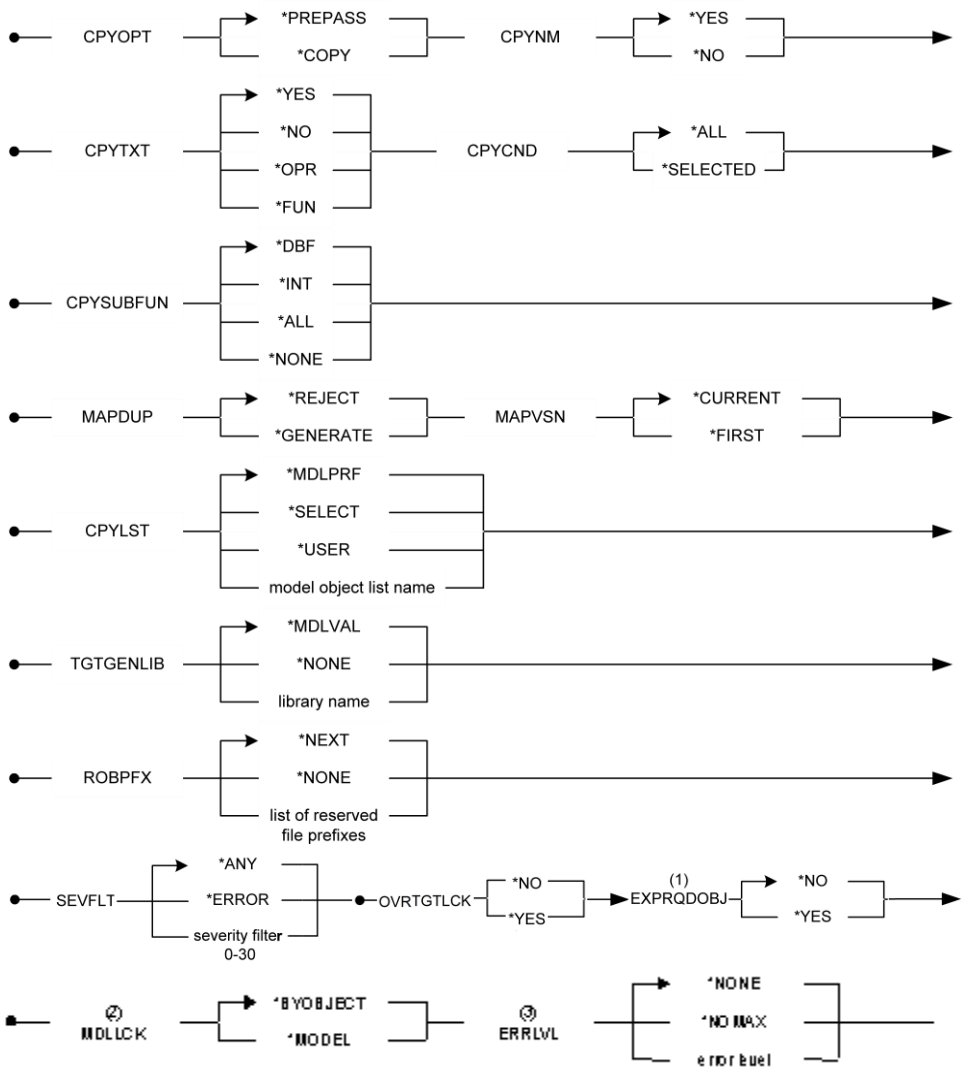
Copies selected objects from one model to another model. The objects to copy are specified on a model list. These objects must be flagged explicitly on the list. To do this, you may use any of the following commands:

- Edit Model Object List for Copy (YEDTCPYLST)
- Build a Model Object List (YBLDMDLLST)
- Change a Model Object List Entry (YCHGMDLLE)
- Add a Model Object List Entry (YADDMDLLE)

Required



Optional



1. EXPRQDOBJ is prompted when CPYSUBFUN is not set to *ALL.
2. MDLLCK is prompted when CPYOPT is set to *COPY.
3. ERRLVL is prompted when MDLLCK is set to *BYOBJECT.

CA2E Modules: *ALL

Job: B,I Pgm: B,I

Parameters

The following are parameters for the YCPYMDLOBJ command.

FROMMDLLIB

Name of design model from which objects are copied. You must have previously built and edited a model list in this library. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

TOMDLLIB

Name of model library to which objects are copied. It must be a valid design model that has been created with the YCRTMDLLIB or YCPYMDL commands. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

CPYOPT

Model copy option. Values for this parameter are described in the following:

- ***PREPASS**—(default) Check the objects to be copied against the objects in the target model named by the TOMDLLIB parameter. Produce a report of any discrepancies found. Do not actually copy objects.
- ***COPY**—Copy the objects in the list to the model named by the TOMDLLIB parameter. Produce a report of any alterations that have to be made to the target model.

CPYVNM

Specifies whether the implementation name of an object should be copied. Only applies when the object being copied already exists in the target model; implementation details will always be copied for objects new to the target model. Values for this parameter are described in the following:

- ***YES**—(default) When copying object details, also copy any implementation names associated with the object (member names, DDS format names, DDS field names, or SQL column names).
- ***NO**—When copying object details, do not copy any implementation names associated with the object. Use the names from the existing objects in the target model.

CPYTXT

Specifies whether narrative text associated with a CA 2E object should be copied. Values for this parameter are described in the following:

- ***YES**—(default) When copying object details, also copy any narrative text associated with the object.
- ***OPR**—Copy only any operational text.
- ***FUN**—Copy only any functional text.
- ***NO**—When copying object details, do not copy any narrative text associated with the object.

CPYCND

This parameter determines which conditions are included in the analysis. Explicitly selected conditions will be copied and will overwrite the same condition in the target model. All new condition which are included in the analysis will be copied to the target model. Values for this parameter are described in the following:

- ***ALL**—(default) The copy process will examine all conditions even if they are not going to be copied.
- ***SELECTED**—Only conditions that are explicitly or implicitly selected will be included in the analysis. Implicit conditions are those, for example, that appear on an explicitly selected list condition.

CPYSUBFUN

Specifies whether and which subordinate functions called by a CA 2E function should be copied when the function is copied. Implicit selections are copied only if they do not exist in the target model. To replace existing subordinate functions in the target model, you must explicitly select them on the model list.

Note that this parameter affects the expansion of explicitly selected objects. If an object that does not qualify for expansion is encountered, and it does not exist in the target model, processing cannot continue past the prepass stage. See the EXPRQDOBJ parameter and the Notes section for more details. Values for this parameter are described in the following:

- ***DBF**—(default) When copying a function, also copy any database functions (for example, CRTOBJ, CHGOBJ) called by the function.
- ***INT**—When copying a function, also copy all internal functions (for example, EXCINTFUN) called by the function.
- ***ALL**—Copy all subordinate functions called by the function.
- ***NONE**—Do not copy any subordinate functions called by the function.

MAPDUP

While attempting to allocate object names in the target model it is possible duplicate names will be encountered. This parameter determines what action is taken in that situation. Values for this parameter are described in the following:

- ***REJECT**—(default) The duplicate name will be reported in the prepass report and the copy process will not be attempted.
- ***GENERATE**—A new name is automatically generated for any object found with a duplicate name in the target. This name will be a combination of the source model object name and the assigned target surrogate number. This action will be reported in the prepass report before any renaming occurs in the target model.

MAPVSN

This parameter is used while identifying which version for a versionable object should be replaced in the target model. Values for this parameter are described in the following:

- ***CURRENT**—(default) The current version of all objects found in the target model will be replaced during the copy process. If there is no current version for an object then the copy will replace the *FIRST version for the group. See the description of the *FIRST parameter value.
- ***FIRST**—The first version encountered chronologically in the target model will be replaced by the copy process. This will generally be the oldest version in a group.

CPYLST

This parameter specifies the model object list that lists the objects to copy to the target model. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The list name is retrieved from the model profile for the current user.
- ***SELECT**—An interactive program is called from which you can select a model object list.
- ***USER**—The list name is the same as the current user name.
- **list name**—The name of a specific model object list.

TGTGENLIB

This parameter specifies the generation library associated with the target model that receives the source for function types Execute User Program or Execute User Source which are included in the copy process. Values for this parameter are described in the following:

- ***MDLVAL**—(default) Use the default generation library name for the target model.
- ***NONE**—No source is copied to the target model.
- **library name**—The name of the library to receive source.

ROBPFX

Specifies allocation of reserved object prefixes for files when the basic limit (AA-T9) has been exceeded. Values for this parameter are described in the following:

- ***NEXT**—(default) Use available system prefixes (UU and up).
- ***NONE**—No allocation.
- **list of reserved file prefixes**—String of up to 5 two-character values can be specified to be used as reserved prefixes for files.

SEVFLT

Specifies the lowest message severity to be printed on the prepass report.

Note: You can change the default value for the YCPYMDLOBJ messages by first duplicating them out into a separate message file and then changing their message severity. If this is do, remember that messages shipped with a severity of 30 are deemed to be *Error messages. Values for this parameter are described in the following:

- ***ANY**—(default) Print all messages. This is equivalent to entering a severity filter of 0.
- ***ERROR**—Only print error messages on the prepass report. This is equivalent to entering a severity filter of 30.
- **severity filter**—A value from 0 to 30. Only messages with a severity greater than or equal to this value will be printed on the prepass report.

OVRTGTLCK

Specifies whether objects in the target model which have a permanent lock can be replaced by the copy operation.

- ***NO**—(default) Objects in the target model which have a permanent lock will not be replaced, and the YCPYMDLOBJ command will fail.
- ***YES**— Objects in the target model which have a permanent lock will be replaced. The lock details for the target object will be replaced with lock details from the source object, if the source object is also locked otherwise, the lock for the target object will be removed.

EXPRQDOBJ

This parameter is prompted when the Copy subfunctions parameter (CPYSUBFUN) is other than *ALL. It lets you specify the action to be taken when a function excluded by CPYSUBFUN does not exist in the target model. This is needed to prevent corruption of the target model or abnormal termination of the YCPYMDLOBJ command. Refer to the Notes section for details. Values for this parameter are described in the following:

- ***NO**—(default) Stop processing and send a message identifying the function that does not exist in the target model. You can then choose to take action to correct the problem; for example, you can explicitly select the excluded function for copy. You should specify *NO if you do not want unexpected objects copied to the target model.
- ***YES**—Expand references for the function and then continue normal processing. The function and any implicitly required objects that do not exist in the target model are copied. A report is produced to notify you of this action. This value effectively overrides the CPYSUBFUN parameter on an object by object basis.

Specifying *YES improves the performance of YCPYMDLOBJ since subfunctions are only expanded when necessary. However, this can cause unexpected objects to be included in the copy. One solution is to run *YES in *PREPASS mode so you can examine any objects that were included unexpectedly before they are actually copied to the target model.

MDLLCK

This parameter specifies how the source model is to be locked during the copy operation. Only prompted if the CPYOPT parameter is set to *COPY. Values for this parameter are described in the following:

- ***BYOBJECT**—(default) The source model is placed in a shared lock state and all objects explicitly or implicitly required are also placed in a shared lock state.

Using this option avoids locking the model during the copy process, however, locking objects individually involves extra processing. All objects must be available for processing to succeed, thus, the copy process will stop if any object cannot be locked.

Since only a read lock is placed on required objects, it will be possible to share objects while copying them. For example, an access path can be copied and, at the same time, shared by a programmer editing a function based on that access path. It would not be possible, however, to attempt to edit the access path.

Use the ERRLVL parameter to specify how many object locks may fail before stopping. This is useful when a small number of objects is involved so that they can be released and the copy process restarted. When a large number of objects is involved it may be easier to stop after the first object lock fails (ERRLVL(*NONE)), so that processing can be deferred until the whole model becomes available.

- ***MODEL**—The source model is exclusively locked for the duration of processing. No concurrent access is allowed in this state.

ERRLVL

Specifies how many object lock errors can occur before processing stops. This applies only where you lock individual objects in the source model, namely when MDLLCK is *BYOBJECT.

Note that processing will stop even if only one object cannot be locked. This parameter simply allows processing to continue so that a given number of errors may be accumulated before stopping. Values for this parameter are described in the following:

- ***NONE**—(default) Processing will stop when the first object that cannot be locked is encountered.
- ***NOMAX**—Processing will stop at the end of the expansion phase if any objects could not be locked.
- **error level**—Processing will stop when the number of objects which cannot be locked equals or exceeds the entered value.

Notes

- You cannot copy to a target model if it is in use by another job. However, you can run the YCPYMDLOBJ command with the CPYOPT(*PREPASS) option. For concurrent use of the source model, refer to the MDLLCK parameter.
- The model library named in the TOMDLLIB parameter (target model) must be the first model library in your library list.
- For SQL you must create an SQL collection before generating source.
- When YCPYMDLOBJ copies the selected entries to the target, it matches objects between the two models by name and type. By default, an object in the source model is copied to the target model using the same object name. You can override this default by specifying a different name for an object in the target model on the YEDTCPYLST rename panel or by changing the object's Copy name before adding the object to the model list for copy. For existing entries you can refresh the copy name from the object's details using option or for all entries on a list using the Check Model List command (YCHKMDLLST).
- When you create a model object, by default its Copy name will be the same as its object name. However, if the default Copy name is not unique within the model for the owner and object type, the Copy name is set to the object name suffixed by the objects 7-digit object surrogate. For object types other than conditions and physical access paths, if you rename an object, its Copy name is not changed; if you rename a condition or physical access path, the Copy name is automatically set to the new object name. You can manually reset an objects Copy name using the Change Model Object Description (YCHGMDLOD) command.
- The Copy Model Objects command does the following:
- **Expands the model list**—Each of the explicitly selected objects in the list are examined in turn to see if it has any associated objects which are required in order for it to be used. Any referenced objects which are found are added to the model list as implicit selections. This means that they will be copied only if they do not already exist in the target model.

  Possible referenced objects for the **file** object type are as follows:

  - Any other files referenced by the relations which define the file.
  - The following messages belonging to the file:

file not found

file already exists

  - The fields arising from the resolution of the relations or entries.

  Possible referenced objects for the **field** object type are as follows:

  - If the field is a REF field, the based-on field.
  - Any field conditions, associated constants, and external message IDs.

  Possible referenced objects for the **access path** object type are as follows:

- The based-on file.
- Any associated access paths and referenced access paths.

Possible referenced objects for the **functions** object type are as follows:

- The based-on access path.
- Any function parameters, any files used to define function parameters.
- Any objects referenced by the function's action diagram: constants, fields and conditions.
- Function fields.
- Action bars.

Note: Functions called by a selected function (other than built-in functions and message functions) are not copied unless they too are selected or unless CPYSUBFUN is specified to be *DBF, *INT or *ALL as appropriate. See also the Note later in this section on CPYSUBFUN and EXPRQDOBJ for details.

The possible referenced object for the **messages** object type is as follows:

- Any parameters (and files and fields named in parameter definition).

There are no referenced objects for the **Application areas** object type.

Note: The expand list does not automatically include functions of type SELRCD which are implicitly, rather than explicitly, referenced by other functions. If there are fields with a user-defined data type of the same name in both models, fields from the source model will take the attributes of the data type in the target model, unless the internal/external conversion functions do not match; that is, the target model data type attributes will not be overwritten.

- **Preliminary Check**—Makes a preliminary (pre-pass) check against the model named in the TOMDLLIB parameter. This checks whether an object with the same CA 2E name is already present in the target model and if so, whether it is the same in all its attributes. If there is a discrepancy, one of two types of messages will be generated on a report:
  - **Warnings**—A warning message indicates an insignificant difference between the two models, (i.e., one that can be handled automatically by the subsequent copy process). Warning messages can be subdivided into two classes by their severity levels:
  - **Severity 10**—(Additional objects) These occur when there are fewer associated objects in the target model than in the source model. In this case, the additional objects can be copied to the target model without changing any existing values in the target model.
  - **Severity 20**—(Property differences) These occur when the objects differ in one or more of their properties (for example, fields have different lengths or data types) but it will be possible to produce a viable new version simply by replacing the existing version in the target model with the version from the source model.

- **Errors**—An error message will be issued if the two versions of an object differ in a way that cannot be handled automatically. For instance:

- The object in the target model has been locked to prevent change.
- The condition values for a given field conflict.
- A required relation does not exist in the target model.

You may find further information about particular messages by using the i OS Display Message Description **(**DSPMSGD**)** command to display the second level text for the message. All YCPYMDLOBJ messages begin with the prefix Y2V5 and reside in the Y2MSG message file.

If any error messages are generated during the pre-pass, no objects can be copied. You must first take action to remove the errors, then rerun the copy. Actions that you may take include:

- Changing, adding or removing objects in either of the two models as appropriate.
- Using the command Edit Copy List (YEDTCPYLST) to specify alternative names under which to copy the new objects into the target model.

- If CPYOPT(*PREPASS) is specified, the YCPYMDLOBJ utility will stop at this point.
- If CPYOPT(*COPY) is specified, and no errors are detected (i.e. warnings only), the YCPYMDLOBJ utility will copy the objects indicated by the model list into the target model. An audit report is generated which lists all the objects copied**.**

In order to produce a viable copy of an object, one or more referenced objects may also need to be present in the target model; the referenced object may in turn require further objects. You should be aware of the implications that the inter-dependencies between the objects in a model have for the copying process, and in particular make sure you understand when an associated object will or will not be copied. The rules for copying an object are:

- **Any object that you explicitly select will always be copied.** If a version of the object already exists in the target model, it is replaced. Otherwise, it is added.
- **Objects that are not explicitly selected**, but are implicitly required by an object that has been selected, **will be added** to the target model **only if they are not already present** in it. Otherwise the existing versions of the referenced objects will be used.

In summary, an object will never be replaced by the YCPYMDLOBJ command unless you explicitly select it.

	Effect on Target Model	
Selection in Model list	`	Not already present
Explicit	REPLACE	ADD
Implicit	DO NOT COPY	ADD

The following table shows for each object type the possible referenced object types that may be copied or replaced.:

Object Selected	Associated objects copied by YCPYMDLOBJ (Unless object already exists in the target model)
FIELD	Any conditions based on the field Any referenced field if it does reference another field External message identifiers
FILE	Relationships that define the file Fields referenced by the relationships Other files referenced by the relationships "Not found" and "already exists" messages associated with the file
ACCESS PATH	File on which the access path is based Any other access paths the access path references
FUNCTION	Access path on which the function is based Any messages the function uses Any function fields the function uses Any fields used as parameter Any access paths used to define the parameters
MESSAGE	Fields used as message parameters Any access paths used to define message parameters Any files upon which the access paths are based
APPLICATION AREA	Nothing extra (i.e. files present in the source model that are absent in the target model will not be copied).

Note that:

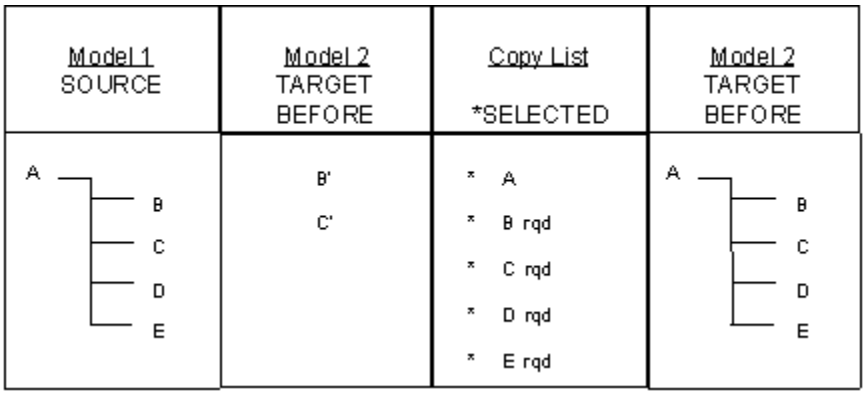
- When a field is replaced, any existing conditions it may have in the target model are retained. In other words, the copy is a merge. New conditions are added but existing conditions are not updated. If there is a conflict over the condition values that would prevent the merge, an error message will be sent.
- When an application area is replaced, any existing files assigned to it in the target model are retained.

Example

To copy objects in a model list in THISMDL to a model THATMDL:

```
YCPYMDLOBJ FROMMDLLIB(THISMDL) + TOMDLLIB(THATMDL) CPYOPT(*COPY) + CPYSUBFUN(*ALL)
```

Contents of Models Before and After Copy



- The following considerations apply to the issuing of generated field names and source member names if CPYVNM(*YES) is specified:
- – If either the name for a field or RPG prefix for a file or source member name from the source model have already been used in the target model, a warning will be generated. If the target model is allocating names automatically (i.e., a value of *YES specified for the YALCVNM model value), a newly generated name or RPG III prefix will be allocated to the object when the copy occurs. If you are not auto-allocating names, they will be left blank for you to supply. The new names will be recorded on the audit listing.

Note: If different model prefixes are used for the two models (specified with the YOBJPFX model value), these will be retained in source member names after copying. In this instance it is very unlikely that there will be any conflict of source member names.

- The following considerations apply to the issuing of message identifiers.
- – If a message identifier already exists in the target model, and you have specified that the target model is to allocate message identifiers automatically (i.e., the YMSGPFX model value is not *NO), a new identifier will be assigned. If you have not, the identifier will be left blank.
- – If the messages in the source and target models are stored in message files of different names (the message file is determined by the message prefix), then a new message file with the same name as that used in the source model will be created in the target model. This fact is recorded on the audit listing.
- The following considerations discuss how the Copy Subfunctions (CPYSUBFUN) and Expand Required Objects (EXPRQDOBJ) parameters affect the copying of functions.
  - – The EXPRQDOBJ parameter applies only when a function is not included by CPYSUBFUN (does not qualify for expansion) and does not exist in the target model.A function qualifies for expansion of references if it is explicitly selected, it meets the criteria set by the CPYSUBFUN parameter, or if it does not exist in the target model and EXPRQDOBJ is set to *YES.
  - – The following table shows how the CPYSUBFUN and EXPRQDOBJ parameters work together to control whether references are expanded for a function and whether the function is copied to the target model. Note that this table applies only if the function is NOT explicitly selected and the CPYSUBFUN parameter is not set to *ALL.

Relationship of CPYSUBFUN and EXPRQDOBJ Parameters

	Expand Required Objects (EXPRQDOBJ = Y)	**Do NOT Expand Required Objects (EXPRQDOBJ = N)**

	Function exists in target model	Function does NOT exist in target model	Function exists in target model	Function does NOT exist in target model
CPYSUBFUN qualifies function for expansion of references	Function expanded but not copied. (1)	Function expanded and copied. (2)	Function expanded but not copied. (1)	Function expanded and copied. (2)
CPYSUBFUN does NOT qualify function for expansion of references	Function neither expanded nor copied. Processing continues normally. (3)	Function expanded and copied. (4)	Function neither expanded nor copied. Processing continues normally. (3)	Function neither expanded nor copied. Processing stops. (5)

Notes:

1.The function is not copied since it already exists in the target model; however, implicitly required objects that do not exist in the target model are copied. Use these settings in *PREPASS mode to identify new objects that have not been explicitly selected and may need to be copied.

2. The function is copied because it does not exist in the target model. Implicitly required objects that do not exist in the target model are always copied.

3. These settings improve performance since functions that exist in the target model are not expanded.

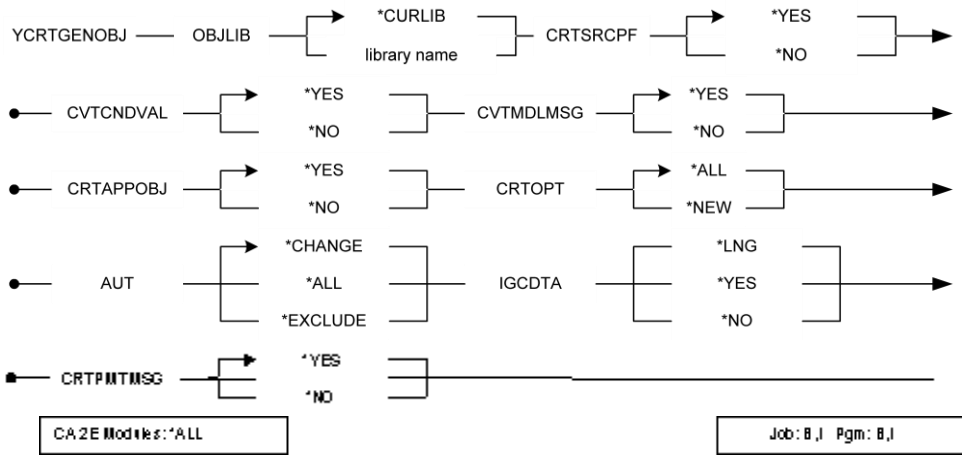
4. Since EXPRQDOBJ caused references for the function to be expanded, it effectively overrides the CPYSUBFUN setting. These settings improve performance since functions are expanded and copied only when they do not exist in the target model. Implicitly required objects that do not exist in the target model are always copied. You should run this in *PREPASS mode and examine the report to ensure that unexpected objects are not copied to the target model.

5. Processing stops since the function's required objects cannot be identified in the target model. A message identifies the function so you can take corrective action; for example, you can explicitly select the function for copy.

YCRTGENOBJ (Create Generation Objects) Command

Duplicates the generation and runtime objects, which are required to run CA 2E generated applications, into a specified target library. This command is useful when you want to execute an application independently of CA 2E. It also allows you to create, as an option, source physical files and to duplicate CA 2E application objects via the YDUPAPPOBJ command.

Optional



Parameters

The following are parameters for the YCRTGENOBJ command.

OBJLIB

Name of the target library into which the required generation objects are duplicated. Values for this parameter are described in the following:

- **Library name**—(default) The name of the target library.
- ***CURLIB**—Use current library for invoking job.

CRTSRCPF

Duplicate source physical files into target library. This command duplicates source physical files for help text or application source. Values for this parameter are described in the following:

- ***YES**—(default) Duplicate all source physical files.
- ***NO**—Source physical files are not duplicated.

CVTCNDVAL

Specifies whether the Convert Condition Values command (YCVTCNDVAL) is to be run as part of processing. Values for this parameter are described in the following:

- ***YES**—(default) Run the command.
- ***NO**—Do not run the command.

CVTMDLMSG

Specifies whether the Convert Model Messages command (YCVTMDLMSG) is to be run as part of processing. Values for this parameter are described in the following:

- ***YES**—(default) Run the command.
- ***NO**—Do not run the command.

CRTAPPOBJ

Create CA 2E application objects. If *YES is specified, the YDUPAPPOBJ command will be invoked, specifying DUPOPT(*ALL). Values for this parameter are described in the following:

- ***YES**—(default) Create CA 2E application objects.
- ***NO**—Do not duplicate CA 2E application objects.

CRTOPT

Duplicate existing objects option. Values for this parameter are described in the following:

- ***ALL**—(default) Duplicate all objects. Replace any existing objects with updated versions.
- ***NEW**—Only duplicate objects that do not already exist in the destination library.

AUT

Authorization rights granted to created libraries and source files. Values for this parameter are described in the following:

- ***CHANGE**—(default) Grant operational rights and all data rights.
- ***EXCLUDE**—Do not grant access rights.
- ***ALL**—Grant all rights.

IGCDTA

Specifies whether CA 2E is to create source files in the generation library that can contain double-byte character set (DBCS) data. Values for this parameter are described in the following:

- ***LNG**—(default) The source files created in the generation library may contain DBCS data depending on the LNG parameter value. The i OS national language versions taking IGCDTA (*YES) by default are Japanese (*JPN) and Uppercase set (*UCS).
- ***YES**—The created files may contain DBCS data.
- ***NO**—The created files may not contain DBCS data.

CRTPMTMSG

Create user prompt message file (NL3). Values for this parameter are described in the following:

- ***NO**—(default) Do not create user prompt message file.
- ***YES**—Create user prompt message file.

Notes

- If *YES is specified on the CRTSRCPF parameter, empty source physical files will be created in the target library. Otherwise, the QTXTSRC file from the first generation library encountered in the library list will be duplicated.
- The CRTAPPOBJ parameter will use the CRTOPT parameter to determine which application objects will be duplicated.
- You must have a GENLIB attached to the model. (Data area MDLLIB/YGENLIBRFA cannot contain *NONE).

Example

To create the required generation and application objects into the library APPLIB creating source physical files:

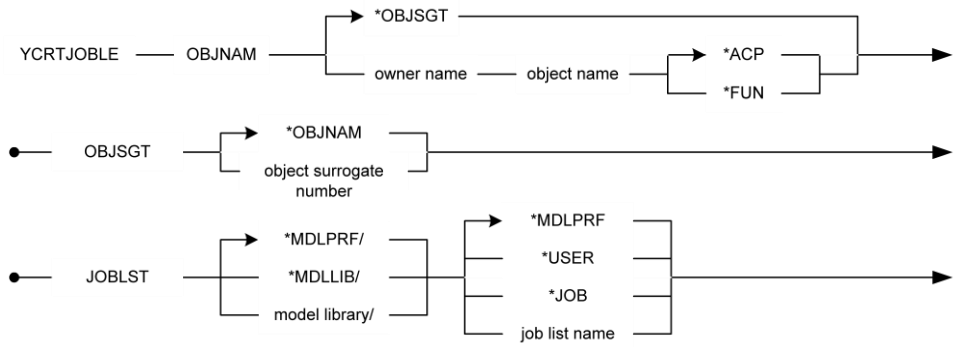
```
YCRTGENOBJ OBJLIB(APPLIB)
```

YCRTJOBLE (Create Job List Entry) Command

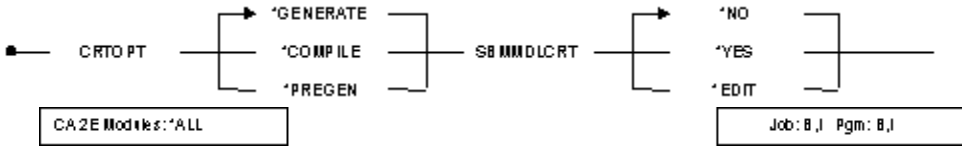
This command provides the means by which an object, together with its auxiliary objects, may be placed onto a job list for regeneration and/or compilation. You can optionally generate the object before placing it on the job list. It accesses the first model found in the library list.

Objects may be identified by name or by object surrogate key number.

Required



Optional



Parameters

The following are parameters for the YCRTJOBLE command.

OBJNAM

The object name of the primary object which is placed on the job list. This parameter consists of three elements which together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate key number is used to identify the model object placed on the job list.
- **object owner name**—The character name of the object which owns the object to be placed. Thus, for a function, the owning file would be entered.
- **object name**—The character name of the object to be placed.
- **object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***FUN**—Object is of type function.

OBJSGT

The object surrogate key number of the model object that is placed on the job list. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the model object.
- **object surrogate number**—The surrogate key number of the model object.

JOBLST

The name of the CA 2E job list that the object is placed on. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the job list name should be taken from the current user's model profile in the model.
- ***USER**—Special value meaning that the user profile name of the current user is used as the job list name.
- ***JOB**—Special value meaning that a job list with the same name as the current job name is used.
- **job list name**—The name of the list to be used.

CRTOPT

This parameter specifies the default action for entries on the target job list. Values for this parameter are described in the following:

- ***GENERATE**—(default) The job list entries should be ready for object regeneration and recompilation.
- ***COMPILE**—The job list entries should be ready for object recompilation only.
- ***PREGEN**—This option specifies that the source for the object is generated prior to the entry being placed on the job list for compile. Note that this will involve an interactive generation if the command is executed in an interactive job.

SBMMDLCRT

This parameter specifies whether the resulting job list is submitted for model generation/creation. Values for this parameter are described in the following:

- ***NO**—(default) The job list is not submitted.
- ***YES**—The job list is submitted after being updated.
- ***EDIT**—The job list is edited prior to submission.

Notes

- If the object and its auxiliaries already exist on the target job list, the status is set to that which is defined on the command. However, note that in certain cases the command will not be able to convert. For example, if the object is currently being generated, then it would not be appropriate to adjust its status. If unable to convert an existing job list entry, an exception message is sent.
- Model objects can either be identified by object name (OBJNAM) or by object surrogate key number (OBJSGT). If the OBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).
- When you to try to place a *DDL-based access path on to a job list for regeneration and/or compilation, and the access path has either of the four DDL limitations, the entry is not added to the job list. The access path source is not generated.
- The current implementation of the DDL generation mode is not valid for the following cases:
  - Access paths that have virtual fields
  - SPN access path
  - QRY access path
  - Multi-member files

  Workaround for Virtual Fields, SPN, and QRY Access Paths: If the earlier generation mode is *DDS, revert to it and regenerate the access path. You need not regenerate the functions that use this access path. If you want to have an SQL type database, regenerate the access path using *SQL generation mode. The functions using this access path must be regenerated.

  Workaround for Multi-Member Files: If you want to have more than one member for the access paths, revert to *DDS generation mode.

Note: If you want to change an access path, which is previously defined as *DDS with a MAXMBR compiler override, to *DDL, you must revert to *DDS generation mode and must remove the compiler override, and then change back to *DDL generation mode.

Example

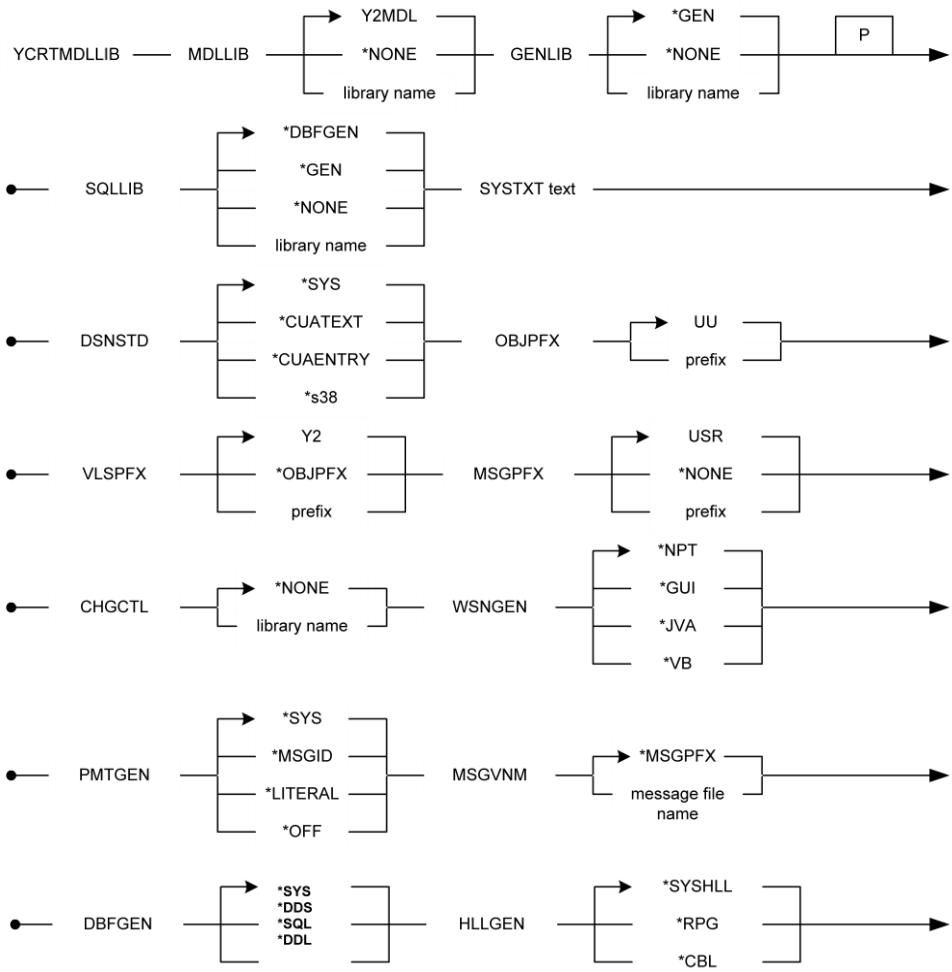
To place the function Process Orders onto the job list of the current user for regeneration and recompilation enter the following command:

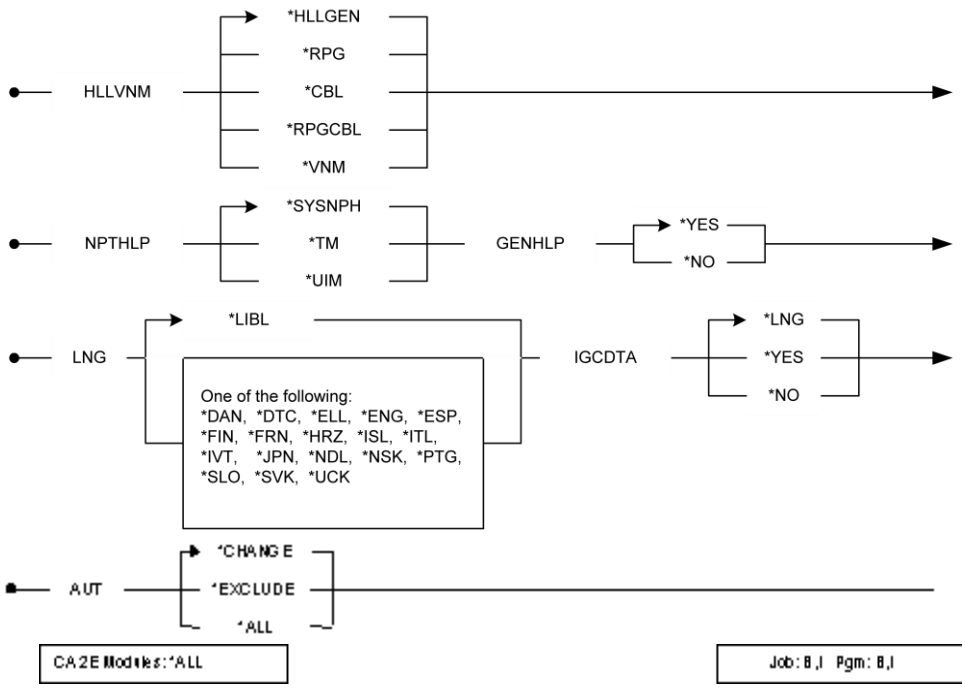
```
YCRTJOBLE OBJNAM( 'Order' 'Process Orders' + *FUN ) JOBLST( *USER ) CRTOPT
 ( *GENERATE )
```

YCRTMDLLIB (Create Model Library) Command

Creates the objects necessary to use CA 2E: (1) library to hold the design model, (2) library in which to place the generated source and compiled objects, and (3) library to hold the SQL collection.

Optional





Parameters

The following are parameters for the YCRTMDLLIB command.

MDLLIB

Name of library to contain the design model and application design. Sets the YMDLLIB model value. This name should have the MDL suffix. The value for this parameter is as follows:

- ***NONE**—Do not create a model library.

GENLIB

Name of library into which source is generated and into which programs and files are compiled. Sets the YGENLIB model value. Values for this parameter are described in the following:

- ***GEN**—(default) Derive name of generated source library according to the rules given in the notes section.
- ***NONE**—Do not create a source library.

SQLLIB

Name of library into which SQL collection for database implementation is placed. Sets the YSQLLIB model value. Values for this parameter are described in the following:

- ***DBFGEN**—(default) If *SQL or *DDL is specified for the DBFGEN parameter, create a library for an SQL collection and generate a name according to the rules given in the notes section.
- ***NONE**—Do not create a collection library.
- ***GEN**—Derive the name of the SQL collection library.

SYSTEXT

Text for the model. Sets the YMDLTXT and YCMPTXT model values.

DSNSTD

Default design standards. Sets the default values for the model values and other parameters controlling aspects of the user interface design standards. Values for this parameter are described in the following:

- ***SYS**—(default) Set default according to YSYSSAA model value.
- ***CUAENTRY**—Set default to SAA CUA Entry model standard.
- ***CUATEXT**—Set default to SAA CUA Text model standard.
- ***S38**—Set defaults to System/38 default standard.

OBJPFX

Application object prefix added to the beginning of all member names that contain source generated by CA 2E.The prefix is only used if automatic generation of member names is specified (system value YALCVNMRFA=Y). For example, UU+CUDAP=UUCUDAP, UU+CHCUR=UUCHCUR. Sets the YOBJPFX model value. The value for this parameter is as follows:

- **UU**—(default) Application prefix.

VLSPFX

Values list object prefix added to the beginning of the names of all objects used to implement the values list facility (xxVLLSP, xxVLLSL0, xxVLLSR, xxVLLLSR#). For example, UU+VLLSP=UUVLLSP. Sets the YVLSPFX model value. Values for this parameter are described in the following:

- **Y2**—(default) Application prefix.
- ***OBJPFX**—Use the same value as that specified for the OBJPFX parameter.

MSGPFX

Message identifier prefix used on all i OS message descriptions generated by CA 2E. For example, USR + 0015 = USR0015. Sets the YMSGPFX model value. Values for this parameter are described in the following:

- **USR**—(default) Message prefix.
- ***NONE**—Allocate message identifiers manually.

WSNGEN

Specifies whether functions are implemented for NPT terminals or for Windows communicating with an IBM i host. This parameter sets the YWSNGEN model value. Values for this parameter are described in the following:

- ***NPT**—(default) Generate an NPT implementation.
- ***GUI**—Generate a Windows implementation.
- ***JVA**—Generate a Windows implementation for use with a Web browser.
- ***VB**—Generate a Windows implementation for use with Visual Basic Forms.

PMTGEN

Device prompt generation option. Sets the YPMTGEN model value. Values for this parameter are described in the following:

- ***LITERAL**—(default) Generate device prompts in display and printer files DDS as fixed literals.
- ***MSGID**—Generate device prompts in display file DDS source as MSGID references and message descriptions.
- ***SYS**—Default to value specified by YSYSPMT system model value.
- ***OFF**—A message ID is allocated but is not generated in the message file.

DBFGEN

Default method for implementing CA 2E access paths as database objects. Sets the YDBFGEN model value. Values for this parameter are described in the following:

- ***SYS**—(default) Default to value specified by YSYSDBF system model value.
- ***DDS**—Implement using DDS and HLL access.
- ***SQL**—Implement using SQL/400 DDL and DML.
- ***DDL**—Implement using SQL/400 DDL.

HLLGEN

Default HLL language in which to generate program source. Sets the YHLLGEN model value. Values for this parameter are described in the following:

- ***SYSHLL**—(default) Default to value specified by YSYSHLL model value.
- ***RPG**—Generate programs in RPG III by default.
- ***CBL**—Generate in COBOL by default.

HLLVNM

HLL language naming conventions for DDS names generated by CA 2E. Sets the YHLLVNM model value. Values for this parameter are described in the following:

- ***RPGCBL**—(default) Naming to be compatible with RPG and COBOL requirements.
- ***RPG**—Naming to be compatible with RPG requirements.
- ***CBL**—Naming to be compatible with COBOL requirements.
- ***HLLGEN**—Default to the same value as specified by the HLLGEN parameter.
- ***VNM**—Apply only name restrictions.

CHGCTL

This parameter allows the new model to run with change control. The library name specified should contain change management exit programs. Values for this parameter are described in the following:

- ***NONE**—(default) Change control is not enabled for the model.
- **library name**—The library containing change control exit programs.

For more information on change control, see the Start Change Control (YSTRCHGCTL) command section in this guide.

MSGVNM

Name of message file used to store message descriptions generated by CA 2E.Sets the YMSGVNM model value. The value for this parameter is as follows:

- ***MSGPFX**—(default) Name is based on the value specified for MSGPFX parameter. If MSGPFX(*NONE) or MSGPFX(*USR) is specified, QUSRMSG is used as the message file name. Otherwise, the message file name is created by concatenating the object prefix (OBJPFX) and the message prefix with the letters MSG, for example, AB +ORD +MSG = ABORDMSG.

GENHLP

Generate help text. Values for this parameter are described in the following:

- ***YES**—(default) Generate help text when generating functions.
- ***NO**—Do not generate help text by default.

NPTHLP

Default NPT help generation language. Values for this parameter are described in the following:

- ***SYSNPH**—(default) Use system-wide model value YSYSNPH.
- ***TM**—Generate help text in TM/38.
- ***UIM**—Generate help text in UIM.

LNG

National language for system data in model. You must have the appropriate national language product libraries loaded onto your machine and referenced in a data area in the base product library. The value for this parameter is as follows:

- ***LIBL**—(default) Use the language of the first product library found in the library list.

Must be one of the national language options.

Note: When specifying a national language it is advisable to make sure the library list of the job running the YCRTMDLLIB command is correct. It should include the following libraries in order:

- CA 2E Toolkit national language library
- CA 2E Toolkit English product library
- CA 2E national language library
- CA 2E English product library

If a model that requires national language support is created without this prerequisite, edit the model library list, using the Edit Library List command (YEDTLIBLST), to correct the error.

IGCDTA

Specifies whether CA 2E is to create source files in the generation library that can contain double-byte character set (DBCS) data. Values for this parameter are described in the following:

- ***LNG**—(default) The source files created in the generation library may contain DBCS data depending on the LNG parameter value. The i OS national language versions taking IGCDTA (*YES) by default are Japanese (*JPN) and Uppercase set (*UCS).
- ***YES**—The created files may contain DBCS data.
- ***NO**—The created files may not contain DBCS data.

LIBLST

Qualified name of a library list that is to be built or updated: the library list will contain the libraries necessary to use the model that is being created. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Library list has same name as the model library.
- ***NONE**—Do not create a library list.

LIBL

Libraries that are to be included in library list specified by the LIBLST parameter, and in the initial library list of the job description specified by the JOBD parameter. Up to 25 libraries may be specified. The value for this parameter is as follows:

- ***MDLLIB**—(default) Creates the following library list:

  QTEMP

  GENLIB parameter value

  SQLLIB parameter value

  MDLLIB parameter value

  QGPL

  National language library depending on LNG

  CA Toolkit product library

  CA 2E product library, depending on HLLGEN and WSNGEN

JOBD

Name of job description that is to be used to compile application objects. This will reside in the library nominated by the MDLLIB parameter. Sets the YCRTJBD model value. The value for this parameter is as follows:

- **QBATCH**—(default) Job description name.

JOBQ

Qualified name of job queue where batch requests are submitted. The value for this parameter is as follows:

- ***JOBD**—(default) Use default job queue name as specified on job description QBATCH in the null model library.

AUT

Authorization rights granted to created libraries and source files. Values for this parameter are described in the following:

- ***CHANGE**—(default) Grant operational rights and all data rights.
- ***EXCLUDE**—Do not grant access rights.
- ***ALL**—Grant all rights.

Notes

- When you run this command you must have all of the appropriate product libraries in your library list.
- To use the command, you must be authorized to the following i OS commands:
  - Create Library **(**CRTLIB**)**
  - Create Physical File **(**CRTPF**)**
  - Create Data Area **(**CRTDTAARA**)**
  - Create Source File **(**CRTSRCPF**)**
  - Create Job Description **(**CRTJOBD**)**
  - Create Duplicate Object **(**CRTDUPOBJ**)**
  - Create Journal **(**CRTJRN**)**
  - Create Journal Receiver **(**CRTJRNRCV**)**
  - Create Data Dictionary **(**CRTDTADCT**)**
- This command creates the following objects:

  MDLLIB (Model library):

  - Creates all files necessary for a CA 2E model.
  - Creates a message file (OBJPFX + USRMSG).
  - Sets model values as indicated on specific parameters.
  - Changes INLLIBL of job description specified by JOBD parameter to LIBL value, unless LIBLST(*NONE) is specified.

  GENLIB (Generation library):

  - A journal (OBJPFX + JRN), and journal receiver (OBJPFX + 00001).
  - A menu file YDSNMNU.
  - An action bar file (OBJPFX + ABDAP).
  - YDSPHLP command help display data areas (YMHPFLA, HMHPLBA).
- You can change the model values that you specified by using the command Change Model Value **(**YCHGMDLVAL**)**.
- If you specify a value of *MDLLIB for the LIBLST parameter, then the appropriate national language product library is included in the library list automatically. If you specify the library list explicitly, then you should ensure that the appropriate national language library is in the library list before either of the CA 2E Toolkit or CA 2E product libraries.
- The DSNSTD parameter controls the initial values for all the parameters affecting design standards, CUA or non-CUA. DSNSTD sets:
  - Which standard header functions are to be the default.

- The default command keys set to use.
- The initial value for the YSAAFMT model value; this value in turn controls design standard for detailed aspects of panel layout, such as how command and subfile explanation instruction text is built.
- The YLHSFLL model value, which controls the appearance of field text leaders on screen designs.
- The YCUAPMT model value, which controls whether the CUA prompt command key to display a list of allowed values, is enabled.
- The YCUAEXT model value provides additional CUA device design compliance if the value is *C89EXT.
- The default field attributes which specify the default field highlighting and color attributes for different field usages.

Design Features Controlled by the DSNSTD Parameter

Design feature	*CUATEXT	*CUAENTRY	*S38
Default DFNSCRFMT function	CUA Text	CUA Entry	*STD header
Default DFNSCRFMT secondary	CUA Window	-	-
Default command keys	CUA set	CUA set	S/38 set
YSAAFMT model value	*CUATEXT	*CUAENTRY	S/38
YLHSFLL model value	*SAA	*SAA	: . : . . : . . :
YCUAPMT model value (F4)	*YES	*NO	*NO
YHLPCSR model value	*YES	*YES	*NO
YEDTFLDATR field attributes	CUA values	CUA values	S/38

The YSAAFMT model value is initially set by the DSNSTD parameter on YCRTMDU. It controls the following aspects of automatic panel design:

Design Aspects Controlled by the YSAAFMT Model Value

Design Aspects	Handling of Field Trailers	Initial Panel Layout	Instruction Set Form	Condition Value List
*CUATEXT	CUA	CUA Text	Fn= n=	Pop-Window
*CUAENTRY	CUA	CUA Entry	Fn= n=	Panel
*S38	YLHSFLL	S/38	CMD: SEL:	Panel

- Some of the parameters (GENLIB, SQLLIB, GENFLR) support a special value to provide automatic creation of a suffixed instance name. For example, if you specify MYMDL for the MDLLIB parameter and you specify *GEN, the default value, for the GENLIB, SQLLIB, and GENFLR parameters, the instance names will be MYGEN, MYSQL, and MYFLR, respectively.
- The common root used in all the instance names is derived from the MDLLIB parameter value. The rules for deriving the root and appending the respective suffixes are as follows:
  - Look for the letters MDL in the MDLLIB value. If found, substitute the suffix letters to create the instance name. For example, if MDLLIB is Y2MDL, then the name used for GENLIB would be Y2GEN. The name used for GENFLR would be Y2FLR.
  - If the letters MDL are not found in the MDLLIB value, simply append the suffix letters. For example, if MDLLIB is MY, then the name used for GENLIB would be MYGEN. If the model library name is more than seven characters long, the suffix will be truncated. If the model library name is 10 characters long, an escape message will be sent.

Example

To create a set of model libraries for an RPG and CBL application using CUATXT design standards:

```
YCRTMDLLIB  SYSTEXT(My system) + MDLLIB(MYMDL) GENLIB(MYGEN) OBJPFX(MY) +
HLLVNM(*RPGCBL) DSNSTD(*CUATEXT)
```

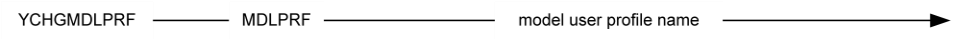
This creates the following:

- Library MYMDL—Library to contain model.
- Library MYGEN—Library to contain generated source.
- Library list MYMDL—Library list for My system model, which contains the following libraries:
  - QTEMP
  - MYMDL
  - MYGEN
  - QGPL
  - CA 2E product library
  - CA 2E Toolkit library

YCRTMDLVSN (Create Model Version) Command

This command lets you copy a model object to create a new version of that object. You can identify the originating object by name or by object surrogate key number. At present, you can create versions of functions or messages only.

Required



Parameters

The following are parameters for the YCRTMDLVSN command.

FRMOBJNAM

The name of the object that is copied. Values for this parameter are described in the following:

- ***FRMOBJSGT**—(default) Single value indicating that the object surrogate number parameter is used to identify the model object.
- **object owner name**—The character name of the object which owns the object. Thus, for a function, the owning file would be entered.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- **object name**—The character name of the object.
- **object type**—The object type of the object.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message. Note that for objects of type *MSG, the owner is the *Messages file.

FRMOBJSGT

Unique number identifier of the model object that is copied. Values for this parameter are described in the following:

- ***FRMOBJNAM**—(default) Use the object name parameter details to identify this model object.
- **object surrogate**—The surrogate number of the model object.

TOOBJNAM

The name of the object to which usage dependencies in the model are redirected. Values for this parameter are described in the following:

- ***GENERATE**—(default) This special value indicates that a new name is generated for the new version. To control how the name generates, you should use the shipped exit program YOBJNAMRIC.
- **object name**—The new object name.

CUROBJ

This parameter allows the developer to choose whether to make the new version current. Values for this parameter are described in the following:

- ***NO**—(default) The new version is not made current.
- ***YES**—The new version is made current.

TFRNAM

This special value indicates whether a transfer of names is to occur as part of processing. Values for this parameter are described in the following:

- ***NO**—(default) No transfer of names is made.
- ***YES**—If the new version is made current following its creation, then the name of the current version is transferred to the new version. Otherwise, the name of the from version is transferred.

CHGTYP

This parameter allows the user to control whether the making of the new version current is treated as a significant change. If a value other than *NONE is specified, then component change processing will be invoked to reflect the change throughout users of the new object. This parameter may only be specified if CUROBJ is set to *YES. Values for this parameter are described in the following:

- ***NONE**—(default) The action of making the new version current is not treated as a significant change in the model; no component change processing is to occur.
- ***PRIVATE**—Component change processing is to occur to apply a private change to the users of the new version.
- ***PUBLIC**—Component change processing is to occur to apply a public change to the users of the new version.
- ***FRMOBJ**—The change type associated with the From object will be examined by component change processing.

RTNEDTSGT

CL variable that receives the surrogate of the resulting edit object. Decimal variable, seven digits packed.

RTNNEWSGT

CL variable that receives the surrogate of the new version. Decimal variable, seven digits packed.

RTNCURSGT

CL variable that receives the surrogate of the current version in the group. Decimal variable, seven digits packed.

Notes

- A search is made for the from model object in the first model library found in the library list.
- If the 25-character object name is transferred from the source object to the target object, the source object must be renamed in the process. The shipped 25-character exit program, YOBJNAMRIC, will be used for this purpose.

Example

To create a new version of the Edit Widget function and call it Edit Sprocket enter the following command:

```
YCRTMDLVSN FRMOBJNAM( 'Widget' 'Edit + Widget' *FUN ) TOOBJNAM( *FRMOBJNAM + 'Edit Sprocket' *FRMOBJNAM )
```

YCRTOBJVSN (Create Object Version) Command

This command provides the means by which an object can be copied to create a new version of that object.

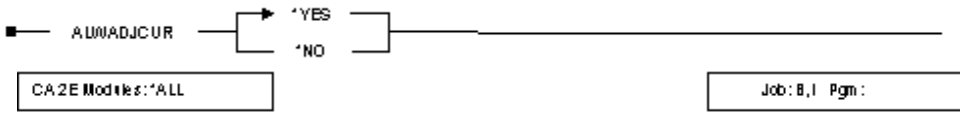
The originating object can be identified by object surrogate number.

At present, you can create versions of functions or messages only.

Note: The processing of this command involves the prompting of the Create Model Version command.

Required

Optional



Parameters

The following are parameters for the YCRTOBJVSN command.

FRMOBJSGT

Unique number identifier of the model object that will be copied. The value for this parameter is as follows:

- **object surrogate**—The surrogate number of the model object.

ALWADJCUR

This parameter indicates whether the user is allowed to adjust the currency status of versions in the group to which the new version will belong. In practical terms, it determines whether the CUROBJ and TFRNAM parameters on the Create Model Version command (YCRTMDLVSN) are prompted. Values for this parameter are described in the following:

- ***YES**—(default) The user will be allowed to access to the CUROBJ and TFRNAM parameters on the prompted Create Model Version command.
- ***NO**—The user will not be allowed access to the CUROBJ and TFRNAM parameters on the prompted Create Model Version command.

Example

To create a new version of the object identified by object surrogate number identifier 1103454:

```
YCRTOBJVSN    FRMOBJSGT( 1103454 )
```

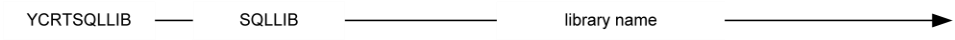
This will result in the Create Model Version panel being prompted where the new name for the object may be entered.

<paratext> <pagenum>

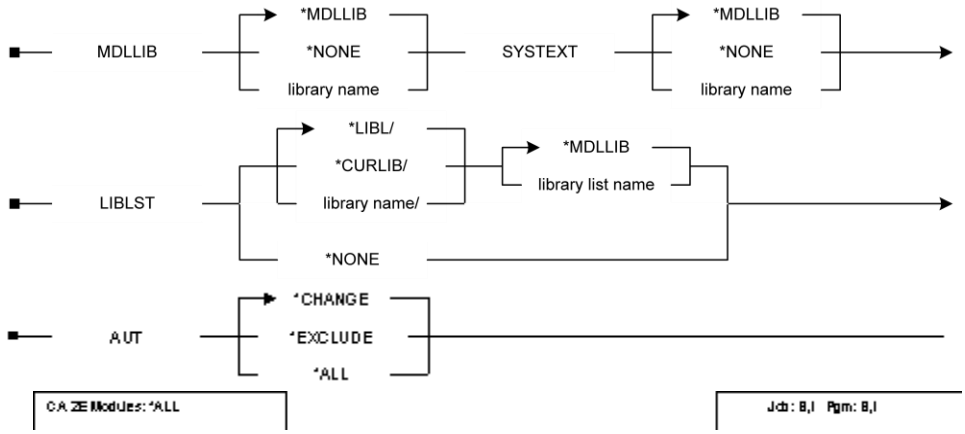
YCRTSQLLIB (Create SQL Library) Command

Creates a library containing an SQL/400 collection into which CA 2E generates SQL/400 tables, views, and indexes.

Required



Optional



Parameters

The following are parameters for the YCRTSQLLIB command.

SQLLIB

Name of library into which SQL/400 collection is placed. Sets the YSQLLIB model value.

MDLLIB

Name of the model library in which to set the YSQLLIB model value. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Default to the first model library name found in the library list.
- ***NONE**—Do not set model value.

SYSTEXT

Text for the SQL library. Values for this parameter are described in the following:

- ***SYSTEXT**—(default) Use text found in model value YMDLTXT in library specified by *MDLLIB.
- ***NONE**—Do not use text for SQL library.

LIBLST

Qualified name of a library list that is built or updated. The library list will contain the libraries necessary to use the model that is being created. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Library list has same name as the model library.
- ***NONE**—Do not create a library list.

AUT

Authorization rights granted to created libraries and source files. Values for this parameter are described in the following:

- ***CHANGE**—(default) Grant operational rights and all data rights.
- ***EXCLUDE**—Do not grant access rights.
- ***ALL**—Grant all rights.

Notes

To use the command, you must be authorized to the following i OS commands:

- Create journal (CRTJRN)
- Create journal receiver (CRTJRNRCV)
- Create data dictionary (CRTDTADCT)

Example

To create a library containing an SQL/400 collection for use by CA 2E to associate with a model:

```
YCRTSQLLIB  SQLLIB(TESTSQL) + MDLLIB(TESTMDL) + SYSTEXT('My test SQL/400 Library')
```

YCRTWS (Create Web Service Instance) Command

Creates a web service instance. The instance can be modeled within a CA 2E model, deployed to an application server, or both. The deployed web service instance exposes a number of operations that correspond to exposed procedures, within modules, within the target Service Program.

Notes:

- If the command fails, detailed errors are written to the YQSHLOG log file in library QTEMP.
- This command requires the user issuing this command to have special authorities *ALLOBJ and *IOSYSCFG. This is due to the underlying IBM IWS scripts requiring those special authorities.

Required

MDLFIL

MDLFUN

SERVER

SERVICE

Optional

UPDMDL

INSTALL

MACHINE

PGMOBJ

USRPRF

RTLIBL

Parameters

The following parameters are for the YCRTWS command:

UPDMDL

Determines whether the CA 2E model information should be updated.

***ADD**

The web service instance definition is added to the CA 2E model. The combination of Web Services Server and Web Service (name) must be unique within a model.

Note: If a Server/Service combination (case insensitive) is already modeled in this model the command will fail.

***NO**

The web service instance definition is not added to the CA model.

***UPDINSSTS**

The installed status of the web service instance, defined within the model, is updated.

Note: If the definition does not already exist in the model, the command will fail.

INSTALL

Determines whether the web service instance is deployed to the application server.

***NO**

The web service instance is not deployed to the application server.

***YES**

The web service instance is deployed to the application server.

MDLFIL

Specifies the name of the CA 2E model file over which the CA 2E model function is defined.

name

Specify the CA 2E model file name.

MDLFUN

Specifies the name of the CA 2E WS model function under which the web service instance is defined in the model.

name

Specify the CA 2E model function name.

MACHINE

Specifies the name of the machine onto which the web service instance is deployed.

***CURRENT**

Refers to the local machine.

name

Specify the machine name. This can be the local machine or a remote machine. The machine name is not validated and the machine need not exist on the local, or any, network.

SERVER

Specifies the name of the application server to which the web service instance will be deployed.

Notes:

- Server name must match (case sensitive) with the name of the actual application server.
- Server/Service/Machine combination (case insensitive) must be unique within a model.

name

Specify the application server name.

SERVICE

Specifies the name of the web service to be deployed.

Note: Server/Service/Machine combination (case insensitive) must be unique within a model.

name

Specify the web service name.

PGMOBJ

Specifies the name of the target service program.

name

Specify the target service program name.

USRPRF

Specifies the name of the user profile under which the web services will run when invoked.

name

Specify the user profile name.

***USRPRF**

The user profile invoking the YCRTWS command will be used to run the web service.

Note: If the service user ID specified here, is different from the server user ID (e.g. QWSERVICE), the server user ID must be given *USE authority to the service user ID.

***SRVID**

The Web services server user ID (e.g. QWSERVICE), is used to run the service.

RTLIBL

Specifies the runtime library list that the web service instance will use when invoked.

name

Specify the list of libraries to be added to the runtime library list.

***NOCHG**

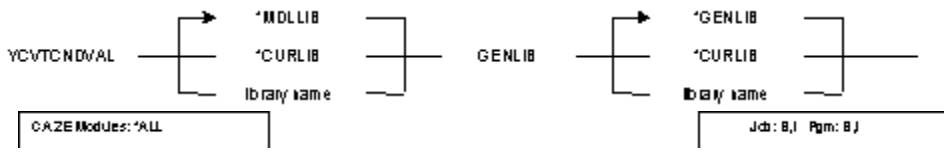
A default library list is used. See the application server documentation for details.

YCVTCNDVAL (Convert Condition Values) Command

Converts the condition value lists held in a model into a database file: the database file is used by the display values program to provide an inquiry function from user programs.

This command must be run in order for new or changed values to be available in the GENLIB for display by an application program. The file must not be in use when you run this command.

Optional



Parameters

The following are parameters for the YCVTCNDVAL command.

GENLIB

MDLLIB

Name of library containing the name of a design model from which the condition values are converted. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Use the first model library found in the library list.
- ***CURLIB**—Use current library for invoking job.

CVTOPT

Determines which condition values should be converted. Values for this parameter are described in the following:

- ***ALL—**Convert all condition values.
- ***MDLLST—**Convert only those condition values where the LST condition is explicitly selected in the model list specified in the Model object list prompt (MDLLST parameter).

MDLLST

Note: This parameter is ignored unless CVTOPT(*MDLLST) is specified.

The qualified name of the model object list that is used. If a LST condition has been explicitly selected in the model list, then the LST condition and all the VAL conditions will be converted. Any other object types in the list (including any VAL conditions where the LST condition has not been explicitly selected) are ignored.

The possible model object list name values are:

- ***MDLPRF—**Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER—**Special value meaning that the user profile name of the current user is used as the model object list name.
- **model-object-list-name**—the name of the model object list to be used.

  Possible library values are:

  - ***MDLLIB—**Special value meaning that the first model library in the current library list is used as the library for the model object list.
  - **library-name—**The name of the model library which contains the model object list.

Notes

- The file must not be in use when you run this command.
- The converted values are placed in a file whose name is constructed from the Value List Prefix model value (YVLSPFX) and the letters VLLSP; for example, XX + VLLSP = XXVLLSP. If a version of the file does not already exist in the library specified by the GENLIB parameter, one will be created.

  Any existing records in the file are deleted, except for those records flagged as user-added (that is, not generated from a CA 2E model). User-added records have a value other than blank in the USRMOD field.
- The condition values converted by this command are displayed to the user when using generated programs if a ? or F4 is entered into a status field that conditions are defined for. The values are displayed by an HLL program whose name will be constructed from the Value List Prefix model value (YVLSPFX) and the letters VLLSR; for example, XX + VLLSR = XXVLLSR. The program will always be duplicated from the product library when the YCVTCNDVAL command is run. The source for the program is available in the shipped source library. If the value list prefix model value (YVLSPFX) is changed, any programs that call the value list display will need to be regenerated.

Example

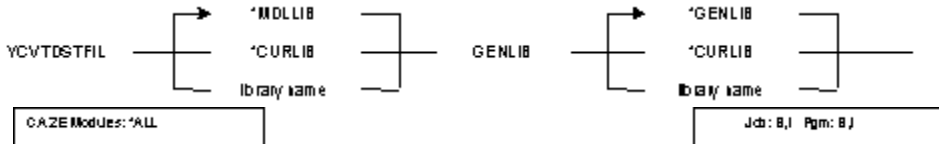
To convert the following values from model library MYMDL to generation library MYGEN:

```
YCVTCNDVAL   MDLLIB(MYMDL)   GENLIB(MYGEN)
```

YCVTDSTFIL (Convert Distributed Files) Command

Adds to and/or deletes from the distributed database file the names of files set as being distributed. All files in the model that have been set to Distributed = Y that do not currently exist in the database file will be added. Any file currently in the database that has been changed to Distributed = N will be deleted. This database file is used in conjunction with the configuration table (which contains locations of distributed databases) to define the network of distributed files which can be accessed by distributed application functions. The configuration tables can be edited by the user using the command Work Distributed Files (YWRKDSTFIL).

Optional



Parameters

The following are parameters for the YCVTDSTFIL command.

MDLLIB

Name of library containing a design model from which the user messages are converted. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Use the first model library found in the library list.
- ***CURLIB**—Use current library for invoking job.

GENLIB

Name of library containing the database file into which distributed file entries are placed. Values for this parameter are described in the following:

- ***GENLIB**—(default) Use the source generation library named by the YGENLIB model value in the model library.
- ***CURLIB**—Use current library for invoking job.

Notes

- Distributed files are those files which have been flagged with Distributed = Y on the Edit File Details panel in the model.
- Entries for all access paths for distributed files are placed in a file Y2DSTFP. If Y2DSTFP does not already exist in the library specified by the GENLIB parameter, it will be created.
- An entry will also be added to the associated configuration file Y2CFGTP for each access path. If Y2CFGTP does not already exist in the library specified by the GENLIB parameter, it will be created.
- Any existing records in Y2CFGTP will be preserved **only** if the associated access path still exists in Y2DSTFP. Otherwise they will be deleted.
- The editor for the configuration file used by YWRKDSTFIL is shipped as an application defined in the model under the files *Configuration Table and *Distributed File. You must generate this application using the design and generator options you choose before you can use YWRKDSTFIL.
- The distributed file entries converted are displayed to the user when the YWRKDSTFIL command is executed.

For more information on working with distributed files and the configuration table, refer to the YWRKDSTFIL command.

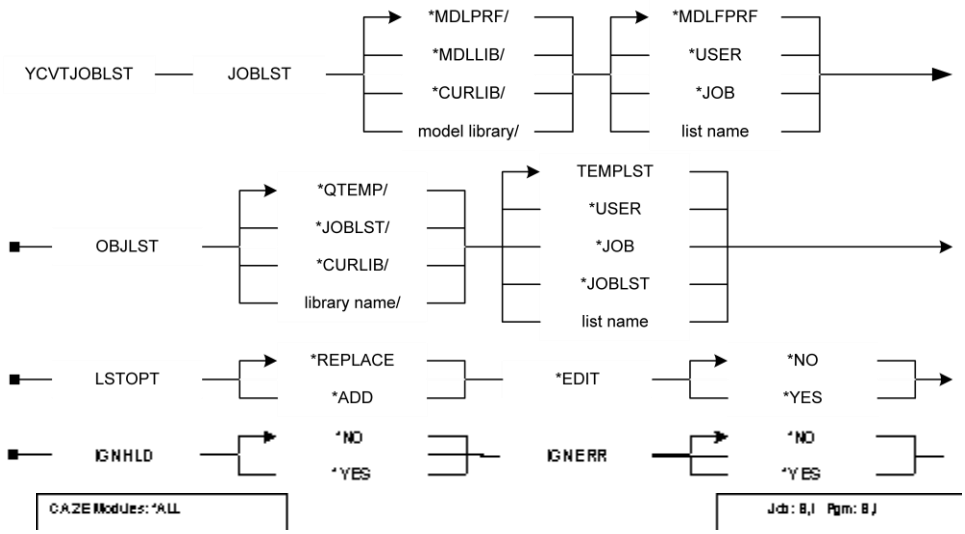
Example

To convert new or changed distributed files from model library MYMDL to generation library MYGEN:

```
YCVTDSTFIL MDLLIB(MYMDL)  GENLIB(MYGEN)
```

YCVTJOBLST (Convert a Job List to CA 2E Toolkit Object List) Command

Optional



Parameters

The following are parameters for the YCVTJOBLST command.

JOBLST

The qualified name of the CA 2E job list which is converted. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the job list name is to be retrieved from the model profile details of the current user.
- ***USER**—Special value meaning that the user profile name of the current user is used as the job list name.
- ***JOB**—Special value meaning that a job list with the same name as the current job name is used.
- **list name**—The name of the used list.
- ***MDLPRF**—(default) Special value meaning that the job list library name is to be retrieved from the model profile details of the current user.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- ***CURLIB**—Special value meaning that the model library is derived from the job's current library.
- **library name**—The name of the library.

OBJLST

The qualified name of the CA 2E Toolkit Object List that is the target of the conversion. Values for this parameter are described in the following:

- **TEMPLST**—(default) Default list name.
- ***USER**—Special value meaning that the user profile name of the current user is used as the object list name for the target of the command.
- ***JOB**—Special value meaning that a job list with the same name as the current job name is used.
- ***JOBLST**—The object list has the same name as the job list.
- **list name**—The name of the object list to be built.
- **QTEMP**—(default) The QTEMP temporary library of the current job is used.
- ***JOBLST**—The object list resides in the same library as that of the JOBLST parameter.
- ***CURLIB**—The object list resides in the current library of the job.
- **library name**—The name of the library to be the target of the command.

LSTOPT

This parameter specifies the action to be taken if the object list already exists. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing object list should be replaced with the output from this command.
- ***ADD**—The existing object list should be augmented with the output from this command.

EDIT

This parameter specifies whether the resulting list is edited as part of processing. Values for this parameter are described in the following:

- ***NO**—(default) Editing of the list is not performed.
- ***YES**—Editing of the list is performed.

IGNHLD

This parameter specifies whether or not HELD entries in the job list should be ignored. Values for this parameter are described in the following:

- ***NO**—(default) Held entries are not ignored.
- ***YES**—Held entries are ignored.

IGNERR

This parameter specifies whether or not ERROR entries in the job list should be ignored. Values for this parameter are described in the following:

- ***NO**—(default) Error entries are not ignored.
- ***YES**—Error entries are ignored.

Notes

- The library specified for the job list must be a valid model library.
- The EDIT parameter is ignored if the job running the command is a batch job.

Example

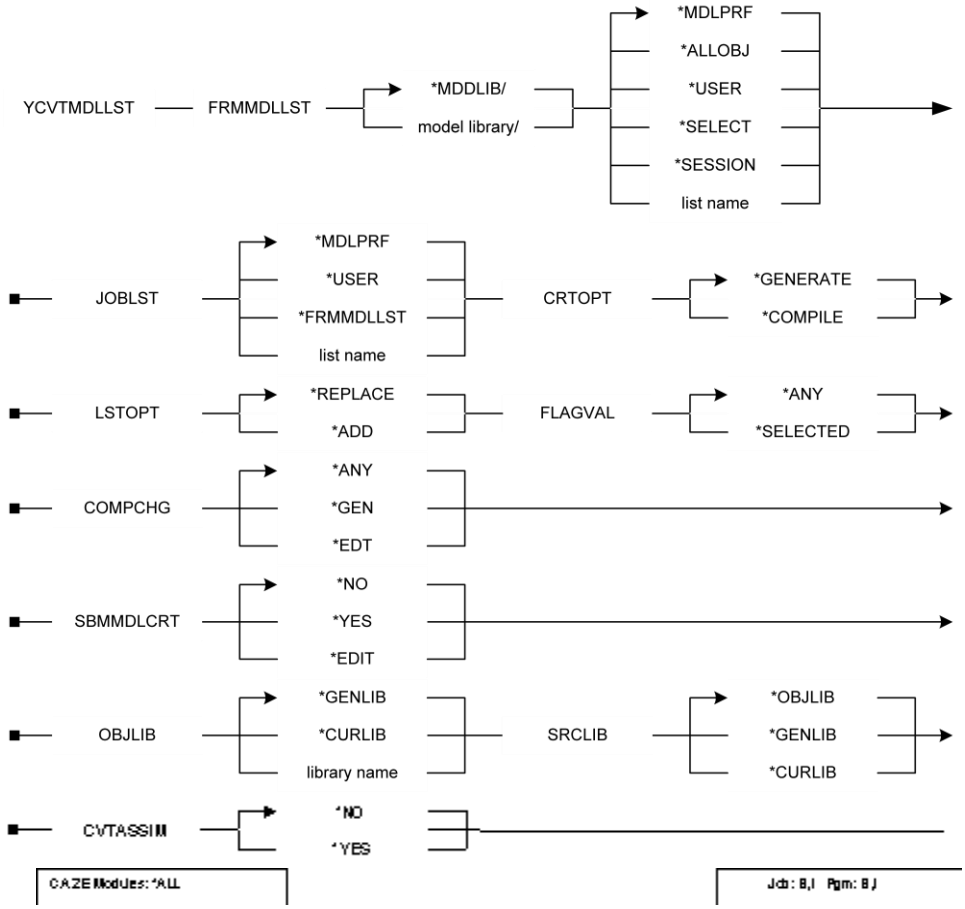
To build object list MOVLST from the current user's job list, to object list MOVLST in QTEMP, ignoring any entries in error on the job list:

```
YCVTJOBLST JOBLST( *MDLLIB/*USER ) +
OBJLST( QTEMP/MOVLST ) LSTOPT( *REPLACE) + IGNERR( *YES )
```

YCVTMDLLST (Convert Model List to Job List) Command

This command allows a user to convert the generatable entries in a model object list to a job list. The target job list is created if it does not exist. If it exists, it may be added to or replaced by taking the appropriate option on the LSTOPT parameter.

Optional



Parameters

The following are parameters for the YCVTMDLLST command.

FRMMDLLST

The qualified name of the model object list that is the source list involved in the conversion. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are included in the command.
- ***USER**—Special value meaning that the user profile name of the current user is used as the model object list name.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- ***SESSION**—Special value meaning that the session list recorded on the current user's model profile is to be used as the model object list name.
- **list name**—The source model object list name may be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the source list.
- **library name**—The model library name for the source list.

JOBLST

The name of the job list that is the target list of the conversion. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The target list is specified on the current user's model profile.
- ***USER**—Special value meaning that the user profile name of the current user is used as the target job list name.
- ***FRMMDLLST**—Special value meaning that the name of the from model object list is used as the target job list name.
- **list name**—The target job list name.

CRTOPT

This parameter specifies the default action for entries in the target job list. Values for this parameter are described in the following:

- ***GENERATE**—(default) The job list entries are ready for object regeneration and recompilation.
- ***COMPILE**—The job list entries are ready for object recompilation only.

LSTOPT

This parameter specifies the action taken if the job list already exists in the target model. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing job list is replaced with the output from this command.
- ***ADD**—The existing job list is augmented with the output from this command.

FLAGVAL

This parameter allows model objects to be filtered by the entry flag value. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on flag value.
- ***SELECTED**—Only explicitly selected model objects will satisfy the filter.

COMPCHG

This parameter allows model objects to be filtered by the component changed flag (action required). If used, model objects in the list will be filtered according to the current status of the component changed flag on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on component changed flag.
- ***GEN**—Model objects with status *GEN will satisfy the condition.
- ***EDT**—Model objects with status *EDT will satisfy the condition.

SBMMDLCRT

This parameter specifies whether the resulting job list is submitted for model generation/creation. Values for this parameter are described in the following:

- ***NO**—(default) The job list is not submitted.
- ***YES**—The job list is submitted after being updated.
- ***EDIT**—The job list is edited prior to submission.

OBJLIB

This parameter specifies the object library name to be used on the Submit Model Create command (YSBMMDLCRT). Values for this parameter are described in the following:

- ***GENLIB**—(default) Special value meaning that the object library is the generation library of the first model found in the current library list.
- ***CURLIB**—Special value meaning that the object library is the current library.
- **library name**—The object library name.

SRCLIB

This parameter specifies the source library name used on the Submit Model Create command (YSBMMDLCRT). Values for this parameter are described in the following:

- ***OBJLIB**—(default) Special value meaning that the source library name is the same as for the OBJLIB parameter.
- ***GENLIB**—Special value meaning that the source library name is the same as for the OBJLIB parameter.
- ***CURLIB**—Special value meaning that the source library name is the same as for the OBJLIB parameter.

CVTASSIM

This parameter specifies whether or not assimilated files are converted to the job list. Values for this parameter are described in the following:

- ***NO**—(default) Assimilated files are not converted.
- ***YES**—Assimilated files are converted.

Notes

- The specified library for the model object list must be a valid model library.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- The FRMMDLLST must exist prior to running the command.
- If the target job list does not already exist, it will be created and the LSTOPT parameter will be ignored.
- Only generatable objects will be placed onto the job list, all other model object list entries will be ignored.
- The CVTASSIM parameter provides the developer with some control over whether assimilated physical files are placed in the resulting job list. Thus, for example, where the physical file is part of some third party software, the developer would wish to avoid attempting to generate and compile the physical file, but would wish any logical files in the model object list to be converted in the normal way.
- When you attempt to convert a model object list into a job list and the model object list contains one or more *DDL-based access paths which satisfy the DDL limitations; such access paths are not added to the job list.
- The current implementation of the DDL generation mode is not valid for the following cases:
  - Access paths that have virtual fields
  - SPN access path
  - QRY access path
  - Multi-member files

  Workaround for Virtual Fields, SPN, and QRY Access Paths: If the earlier generation mode is *DDS, revert to it and regenerate the access path. You need not regenerate the functions that use this access path. If you want to have an SQL type database, regenerate the access path using *SQL generation mode. The functions using this access path must be regenerated.

  Workaround for Multi-Member Files: If you want to have more than one member for the access paths, revert to *DDS generation mode.

Note: If you want to change an access path, which is previously defined as *DDS with a MAXMBR compiler override, to *DDL, you must revert to *DDS generation mode and must remove the compiler override, and then change back to *DDL generation mode.

Example

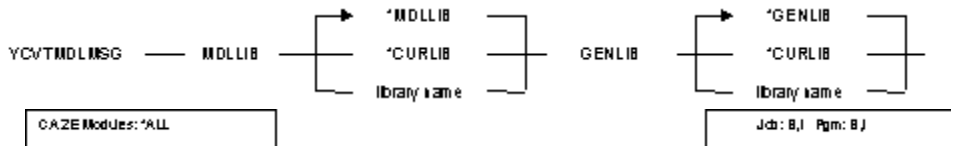
To convert all model objects with a component changed status (action required) of *GEN (requiring generation and compilation) to job list OUTLIST from the model library WRKMDL:

```
YCVTMDLLST FRMMDLLST( WRKMDL/*ALLOBJ ) +  JOBLST( OUTLIST ) CRTOPT( *GENERATE ) +
LSTOPT( *REPLACE ) COMPCHG( *GEN )
```

YCVTMDLMSG (Convert Model Messages) Command

Converts the user message functions held in a design model into i OS message descriptions in one or more message files.

Optional



Parameters

The following are parameters for the YCVTMDLMSG command.

MDLLIB

Name of library containing a design model from which the user messages are converted. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Use the first model library found in the library list.
- ***CURLIB**—Use current library for invoking job.

GENLIB

CVTOPT

Determines which model messages should be converted. Values for this parameter are described in the following:

- ***ALL—**Converts all model messages.
- ***MDLLST—**Convert only those model messages explicitly selected in the model list specified in the Model object list prompt (MDLLST parameter).

MDLLST

Note: This parameter is ignored unless CVTOPT(*MDLLST) is specified.

The qualified name of the model object list that is used. If a model message has been explicitly selected in the model list, it will be converted. Any other object types in the list (or any model messages in the model list which have not been explicitly selected) are ignored.

Values for this parameter are described in the following:

- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the model object list name.
- model-object-list-name—the name of the model object list to be usedPossible library values are:
- ***MDLLIB**—Special value meaning that the first model library in the current library list is used as the library for the model object list.
- **library-name**—The name of the model library which contains the model object list.

Notes

- This command creates i OS message descriptions from CA 2E message function definitions. The name of the message file that the message descriptions are added to is determined for each message function as follows:
  - If there is an override message file name specified for the message function, then the first message file of that name in the invoking job's library list is used.
  - If no override is specified, then the default message file, (as specified by the YMSGVNM model value in the generation library, as specified by the GENLIB model value) is used.
- The message files you use must already exist in the relevant libraries and the user must have the necessary authority to update them.
- The conversion process produces an audit report listing for each user message, the message file used and the success or failure of the conversion.

Example

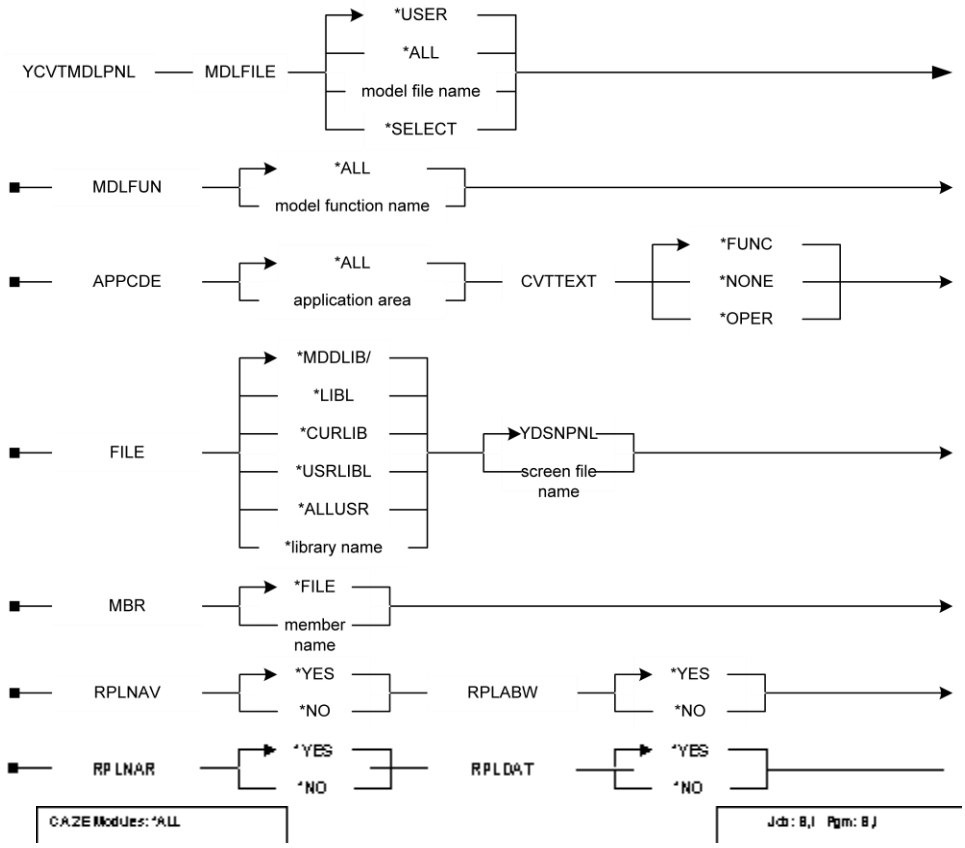
To convert user messages from model library MYMDL to generation library MYGEN:

```
YCVTMDLMSG MDLLIB(MYMDL)  GENLIB(MYGEN)
```

YCVTMDLPNL (Convert Model Panel Designs) Command

Converts the panel designs of functions into CA 2E Toolkit panel designs so that they may be used for prototyping.

Optional



Parameters

The following are parameters for the YCVTMDLPNL command.

MDLFILE

Name of a file to which the functions containing the panel designs to be converted belong. Values for this parameter are described in the following:

- ***USER**—(default) Convert the panel designs of functions that belong to all user-defined files in the model.
- ***ALL**—Convert the panel designs of functions that belong to all files in the model, including system files.
- ***SELECT**—Select file.

MDLFUN

Name of a function whose panel designs are converted. The value for this parameter is as follows:

- ***ALL**—Convert the panel designs of all functions for the specified files.

APPCDE

Name of an application area containing the files of the functions whose panel designs are converted. The value for this parameter is as follows:

- ***ALL**—(default) Convert the panel designs of functions attached to files from all application areas.

CVTTEXT

Specifies whether narrative text for the file is included in the generated panel designs, and if so, which type of text. Values for this parameter are described in the following:

- ***FUNC**—(default) Include functional text.
- ***NONE**—Do not include narrative text.
- ***OPER**—Include operational text.

FILE

Qualified name of file to contain panel designs. The value for this parameter is as follows:

- **YDSNPNL**—(default) File name.

MBR

Name of member in panel design file. The value for this parameter is as follows:

- ***FILE**—(default) The member has the same name as the file.

RPLNAV

Specifies whether to keep or replace the command key navigation that was defined for the CA 2E Toolkit panel design. Values for this parameter are described in the following:

- ***YES**—(default) The navigation is replaced.
- ***NO**—The navigation is not replaced.

RPLABW

Specifies whether to keep or replace the action bar definition that was defined in the CA 2E Toolkit panel design. Values for this parameter are described in the following:

- ***YES**—(default) The action bar definition is replaced.
- ***NO**—The action bar definition is not replaced.

RPLNAR

Used to keep or clear any narrative that was entered for the CA 2E Toolkit panel design. Values for this parameter are described in the following:

- ***YES**—(default) The narrative is cleared.
- ***NO**—The narrative is kept.

RPLDAT

Used to keep or clear any test data that was entered for the CA 2E Toolkit panel design. Values for this parameter are described in the following:

- ***YES**—(default) The test data is cleared.
- ***NO**—The test data is kept.

Notes

- A library containing the model files must be present in the library list of the job that executes the YCVTMDLPNL command.
- The converted panel designs are placed in the nominated panel design file. The file must already exist in the specified library: new files may be created using the CA 2E Toolkit command Create Design File **(**YCRTDSNF**)**.
- Each prototype panel design is given the name of the program source member of the function from which it is derived. A suffix is used to number the panels derived from a single CA 2E function.
- The converted panel designs can be prototyped using the CA 2E Toolkit command Display Panel (YDSPPNL). Refer to the CA 2E Toolkit manuals for further details.

Example

To convert all the user function panel designs to file YDSNPNL:

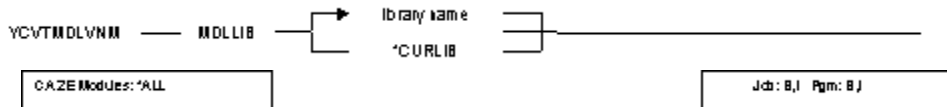
```
YCVTMDLPNL  FILE(YDSNPNL)
```

YCVTMDLVNM (Convert Model Names) Command

Converts the implementation names of CA 2E objects within a design model. The names are updated to comply with the requirements of the HLL in which the model is to be generated.

A report is produced of those names that have been changed (YCVTVNMPP$).

Required



Parameters

The following are parameters for the YCVTMDLVNM command.

MDLLIB

Name of library containing the design model whose implementation names are changed. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

Notes

- The YCVTMDLVNM command examines the following implementation names in the data model, and if necessary, converts them:
  - Access path source member names
  - Access path format names
  - Field or column names
  - HLL program source member names
  - Device file source member names
  - Help text source member names
- The nature of the conversion is controlled by the current value of the YHLLVNM model value. This dictates the rules for deciding whether a name needs converting.
  - If the YHLLVNM model value is *CBL, then any instances of the characters @, # and $ are replaced by the appropriate character specified for the name type in the generation types table. If there is no replacement character for the name type in the table, the letter J will be used.
  - If the value of YHLLVNM is *RPG, then any instances of the underscore character will be replaced by the quotation (") character. Any other valid characters are replaced by "J."

For more information about converting a model from one HLL to another, refer to *Generating and Implementing Applications*, "Preparing for Generation and Compilation".

Example

To convert the model MYMDL:

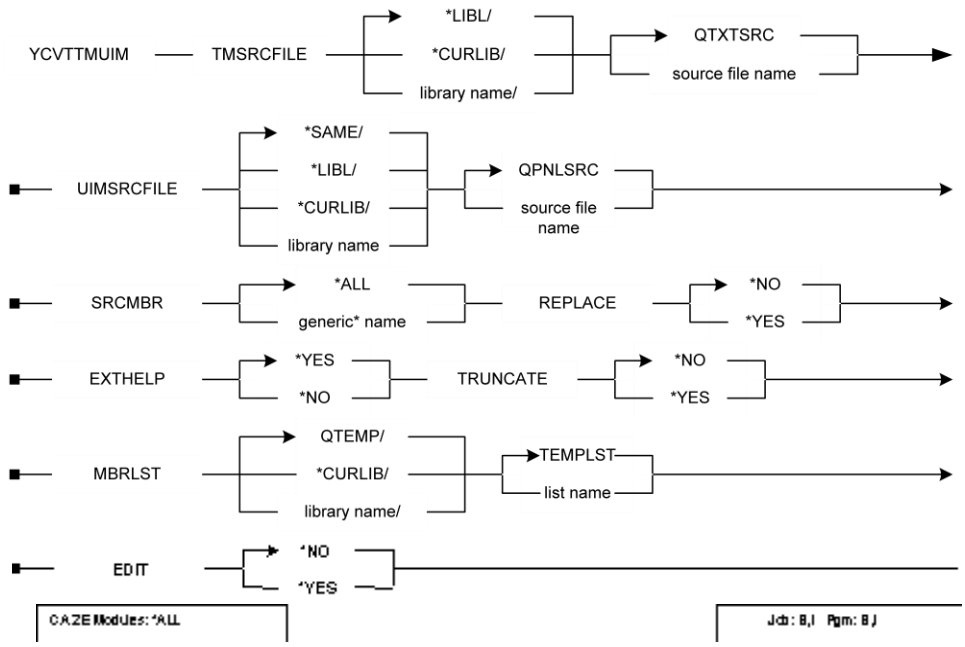
```
YCVTMDLVNM   MDLLIB(MYMDL)
```

YCVTTMUIM (Convert Help Text to UIM Panel) Command

This routine converts source that uses the existing CA 2E Toolkit help support tags (as defined in Text Management or generated by CA 2E) into UIM panel group source.

Due to the difference between TM and UIM, only an approximate conversion is possible. The more your source file differs from the standard generated TM source, the less accurate the conversion will be.

Required



Parameters

The following are parameters for the YCVTTMUIM command.

TMSRCFILE

The qualified name of a source file containing TM help text. Values for this parameter are described in the following:

- ***LIBL**—(default) Use the library list to locate the source file.
- ***CURLIB**—Use current library to locate the source file.

UIMSRCFILE

The qualified name of a source file that the converted UIM help text will be placed in. You must have *ALL rights to this file. Values for this parameter are described in the following:

- ***SAME**—(default) The library name is the same as that specified in TMSRCFILE.
- ***LIBL**—The library list is used to locate the source file.
- ***CURLIB**—The current library for the job is used to locate the source file.

SRCMBR

The generic name of the source members to convert. The name assigned to the UIM member will be the same as that of the TM member. The value for this parameter is as follows:

- ***ALL**—Convert all source members.

REPLACE

Specifies the action to be taken if a member already exists in the UIMSRCFILE. Values for this parameter are described in the following:

- ***NO**—(default) Issue an error and bypass the member.
- ***YES**—Change the member and reconvert.

EXTHELP

Specifies if an extended help panel should be generated for the member. This is functionally equivalent to the existing index panel used by TM. It uses the UIM :LINK tag to provide two-way hypertext links between each panel and the extended help panel. Values for this parameter are described in the following:

- ***YES**—(default) Generate the extended help panel.
- ***NO**—Do not generate the extended help panel.

TRUNCATE

Allows the user to control the action on conversion when a line to be converted is too long. Values for this parameter are described in the following:

- ***NO**—(default) Excess data is continued on the next line and not truncated.
- ***YES**—Excess data is truncated during conversion.

MBRLST

Specifies a target member list to be used to contain an entry for each member converted. Values for this parameter are described in the following:

- **TEMPLST**—(default) Default member name.
- **QTEMP/**—Default library name.
- ***CURLIB/**—The library to contain the member list is the current library.

EDIT

Specifies whether the resulting list is to be edited at the end of the conversion process. Values for this parameter are described in the following:

- ***NO**—(default) The list is not to be edited.
- ***YES**—The list is to be edited.

Notes

- The converted source can be compiled into a panel group (*PNLGRP). It can then be used with existing applications, without the need to regenerate those applications as follows:
  - The help display routine (YDDSHPR) will check for the existence of a panel group with the same name as the help member. If it exists, it will use that panel group to display UIM help panels.
- In order to supply cursor sensitive help, help panels must be related to screen areas.
  - For full UIM, this is supplied by DDS keywords in the associated display file.
  - For converted UIM, this must be supplied by a source member that defines a vector table. This source member can either be the original TM source member or the converted UIM source member, both of which contain the ".*yv" tags that define the vector table. The routine will look for this member in the same file as the original TM source member.

Example

To convert all source members in MYGENLIB\QTXTSRC beginning with UU and place output in MYGENLIB/QPNLSRC, replace any member which already exists. By default the extended help panel will be generated:

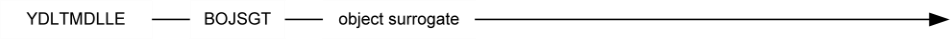
```
YCVTTMUIM TMSRCFILE (MYGENLIB/QTXTSRC) + UIMSRCFILE (*SAME/QPNLSRC) SRCMBR(UU*) +
REPLACE (*YES)
```

- If CSG client functions are on the job list and the client objects do not exist in the folder, entries will not be added to the job list for the client objects (the server object will always be added to the job list).

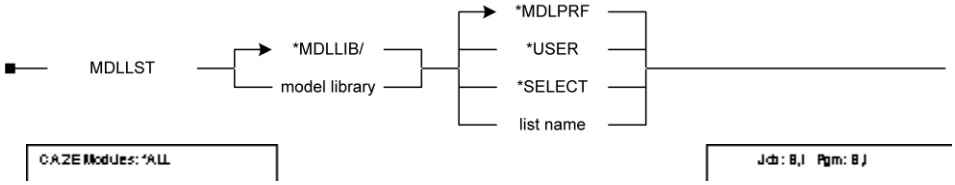
YDLTMDLLE (Delete a Model Object List Entry) Command

This command allows a user to delete a model object list entry.

Required



Optional



Parameters

The following are parameters for the YDLTMDLLE command.

OBJSGT

Unique number identifier of the model object to be deleted. The value for this parameter is as follows:

- **Object surrogate**—The surrogate number of the model object is required.

MDLLST

The qualified name of the model object list in which the entry to be deleted exists. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The name of the list.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the model library.

Note

Both the model object list and the list entry must exist prior to running this command.

Example

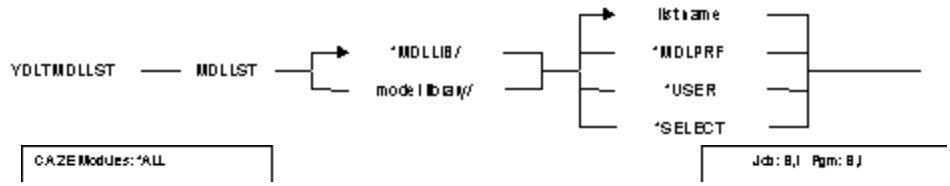
To delete the model object list entry for the object identified by surrogate number 1100547 in model object list DEVRA in the first model library to be found in the current library list:

```
YDLTMDLLE OBJSGT( 1100547 ) + MDLLST(*MDLLIB/DEVRA )
```

YDLTMDLLST (Delete a Model Object List) Command

This command allows a user to delete a model object list.

Required



Parameters

The following are parameters for the YDLTMDLLST command.

MDLLST

The qualified name of the model object list that is to be deleted. Values for this parameter are described in the following:

- **list name**—(default) The model object list name.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—The model library name for the list.

Notes

- The MDLLST must exist prior to running the command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- There is close integration between this command and user-defined fields on the all objects model list. To provide an efficient method for clearing the user-defined fields of objects checked out to a particular model object list, part of the processing of this command is to clear the Checkout List, Checkout User, Checkout Date, Checkout Time and Checkout Status on all model object records on which the list being deleted is recorded as Checkout List.

Example

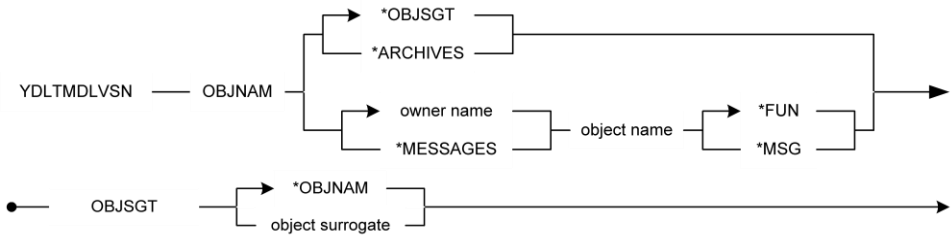
To delete model object list MYLIST from the model contained in the current library list:

```
YDLTMDLLST MDLLST( *MDLLIB/MYLIST )
```

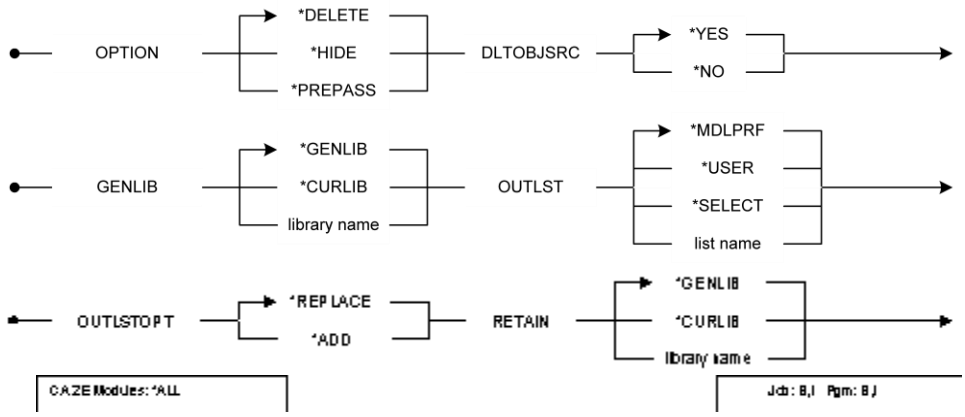
YDLTMDLVSN (Delete Model Version) Command

This command allows the user to delete a function or a message from the model.

Required



Optional



Parameters

The following are parameters for the YDLTMDLVSN command.

OBJNAM

The name of the object that is deleted. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate number parameter is used to identify the model object.
- ***ARCHIVES**—This generic value instructs the processing program to examine all archive versions in the model. In this mode, all entries are written to the specified output list. The RETAIN parameter specifies how many archives are preserved in a particular group.
- **object owner name**—The character name of the object which owns the object. Thus, for a function, the owning file would be entered.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- **object name**—The character name of the object.
- **object type**—The object type of the object.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message. Note that for objects of type *MSG, the owner is the *Messages file.

OBJSGT

Unique number identifier of the model object that is copied. Values for this parameter are described in the following:

- **OBJNAM**—(default) Use the object name parameter details to identify this model object.
- **object surrogate**—The surrogate number of the model object

OPTION

This parameter provides the user with some flexibility when deleting model versions. For a model version, the user can either physically delete it or hide it by making the version non-current. When using the generic delete for archives, the user can perform a prepass, which will build a list of affected versions without actually deleting them.

Only qualified versions will be processed, meaning that the version must not be used by any other model object. Values for this parameter are described in the following:

- ***DELETE**—(default) Special value meaning that versions that qualify for processing are physically deleted.
- ***HIDE**—Special value meaning that versions that qualify for processing are not deleted but made non-current. They will not, by default, be shown on model panels. If a version is already non-current it remains unaffected by this process.
- ***PREPASS**—For the *ARCHIVES processing only. Special value meaning that the archives that qualify for processing should be written to the out list but not physically deleted.

DLTOBJSRC

This parameter allows the user to specify that the object and source corresponding with the model design object are deleted at the same time the design object is deleted from the model. Values for this parameter are described in the following:

- ***YES**—(default) The object and source are deleted.
- ***NO**—The object and source are not deleted.

GENLIB

The name of the generation library that contains the source and object corresponding with the model object that is deleted. Values for this parameter are described in the following:

- ***GENLIB**—(default) Special value meaning that the generation library for the model is used.
- ***CURLIB**—Special value meaning that the generation library is the current library for the job.
- **library name**—The generation library name.

OUTLST

The name of the model object list that is to receive an entry for each archive version which satisfies the RETAIN parameter (i.e., qualifies for the delete operation). This parameter is only prompted if *ARCHIVES is specified for the OBJNAM parameter. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile details for the current user.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name.
- ***SELECT**—Special value meaning that an interactive select program is invoked. The output list will be selected.
- **list name**—The name of the target list.

OUTLSTOPT

This parameter specifies the action to be taken if the output list already exists in the model. This parameter is only prompted if *ARCHIVES is specified for the OBJNAM parameter. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list is to be replaced with the output from this command.
- ***ADD**—The existing model object list is augmented with the output from this command.

RETAIN

This parameter specifies the number of archive versions (version type ARC) that are retained. This parameter is only prompted when the OBJNAM parameter is *ARCHIVES. The archive versions are examined in reverse chronological order, the excess number above this parameter being deleted. Values for this parameter are described in the following:

- ***ALL**—(default) All archive versions are retained.
- ***NONE**—No archive versions are retained.
- **numeric value**—The developer may specify a discreet number of versions that are retained.

Notes

- It is intended that this command be used as part of a change control system. The primary purpose is for the removal of archives from the model as they become redundant with the creation of later versions.
- No version can be deleted if it is used by any other model object.
- When *ARCHIVES are being processed, all versions that are processed are written to the specified output list. If it is not possible to make a current version non-current or to delete a non-current version, the entry is flagged *ERROR.

Example

To delete the model version Edit Customer 3 from the model, accepting the default removal of object and source from the generation library for the model, enter the following:

```
YDLTMDLVSN OBJNAM( 'Customer' 'Edit + Customer 3' '*FUN' )
```

YDLTOBJTBL (Delete Object Table User Space)

This command deletes all User Spaces for a 2E model. They are then rebuilt when a user enters the model again or YSNCMDL is run. This command can be used to resolve model corruption problems, or when any sort of problems with model data are encountered. This command should only be used when instructed to do so by 2E Support.

Parameters

Parameter	Definition	Value and Description
MDLLIB	Library for Data Model This specifies the model for which the user spaces are deleted	*MDLLIB: Use the first model library found in the library list.
model-name	Specify model library name	

Chapter 5: Commands (YDOCMDLACP - YOPRMDLLST)

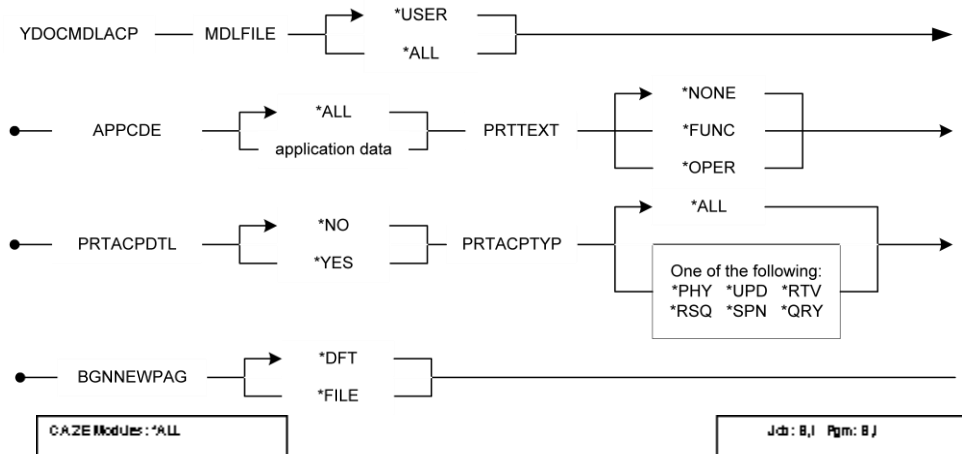
This chapter contains details for CA 2E commands YDOCMDLACP through YOPRMDLLST. These commands appear in alphabetical order and include descriptions of their functions, parameters and allowed values, notes, and examples. Each command is also accompanied by a command diagram.

This section contains the following topics:

YDOCMDLACP (Document Model Access Paths) Command

Documents the access path files within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLACP command.

MDLFILE

Types of files whose access paths are documented. Values for this parameter are described in the following:

- ***USER**—(default) List the access paths of all user-defined files in the model.
- ***ALL**—List the access paths of all files in the model, including system files.

APPCDE

Name of an application area to which the files whose access paths are listed belong. The value for this parameter is as follows:

- ***ALL** —(default) List the access paths of files from all application areas.

PRTTEXT

Specifies whether narrative text for the access path is included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

PRTACPDTL

Specifies whether the details of the access path are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include full details.
- ***YES**—Include full details.

PRTACPTYP

Specifies the types of access path that are included in the generated documentation. Values for this parameter are described in the following:

- ***ALL**—(default) Include all access path types.
- ***PHY**—Include only physical access paths.
- ***UPD**—Include only update access paths.
- ***RTV**—Include only retrieval access paths.
- ***RSQ**—Include only resequence access paths.
- ***SPN**—Include only span access paths.
- ***QRY**—Include only query access paths

BGNNEWPAG

Specifies whether documentation for a function or file starts on a new page or not. Values for this parameter are described in the following:

- ***DFT**—(default) Do not start a new page.
- ***FILE**—Start a new page for each file whose functions are documented.

Notes

A library containing the model files must be present in the library list of the job that executes the commands.

Example

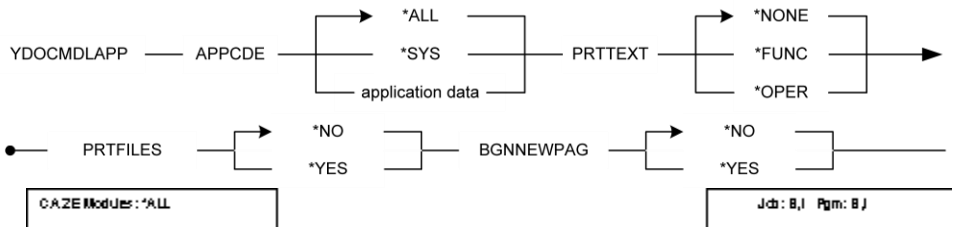
To print all details about the model files in a model library:

```
YDOCMDLACP
```

YDOCMDLAPP (Document Model Application Areas) Command

Documents the application areas within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLAPP command.

APPCDE

Name of an application area that is documented. Values for this parameter are described in the following:

- ***ALL**—(default) List all application areas.
- ***SYS**—List the system application area (this contains all files in the model).

PRTTEXT

Specifies whether narrative text for the file is included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

PRTFILES

Specifies whether the files belonging to the application area(s) specified in the APPCDE parameter are included in the generated documentation. Values for this parameter are described in the following:

- ***YES**—(default) Include files.
- ***NO**—Do not include files.

BGNNEWPAG

Specifies whether documentation for an application area is to start on a new page or not. Values for this parameter are described in the following:

- ***NO**—(default) Do not start a new page.
- ***YES**—Start a new page for each application area documented.

Notes

A library containing the model files must be present in the library list of the job that executes the YDOCMDLAPP command.

Example

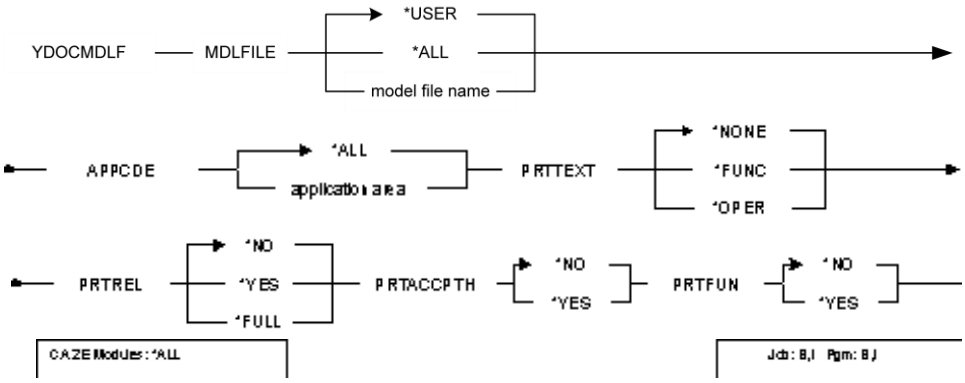
To print all details about the model application areas in a model library, including functional text and files within each area:

```
YDOCMDLAPP PRTTEXT(*FUNC) PRTFILE(*YES)
```

YDOCMDLF (Document Model Files) Command

Documents the files within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLF command.

MDLFILE

Specifies name of files you want to include in the listing. Values for this parameter are described in the following:

- ***USER**—(default) List all user-defined files in the model.
- ***ALL**—List all files in the model, including system files.

APPCDE

Name of an application area to which the files that are listed belong. Values for this parameter are described in the following:

- ***ALL**—(default) List files from all application areas.

PRTTEXT

Specifies whether narrative text for the file is included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

PRTREL

Specifies whether the relations that reference the file are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include relations.
- ***YES**—Include relations.
- ***FULL**—Print relations and associated field entries together with details of the field entry.

PRTACCPTH

Specifies whether the names of the access paths referenced by each file are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include access paths.
- ***YES**—Include access paths.

PRTFUN

Specifies whether the names of the functions that reference the file are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include functions.
- ***YES**—Include functions.

Notes

A library containing the model files must be present in the library list of the job that executes the YDOCMDLF command.

Example

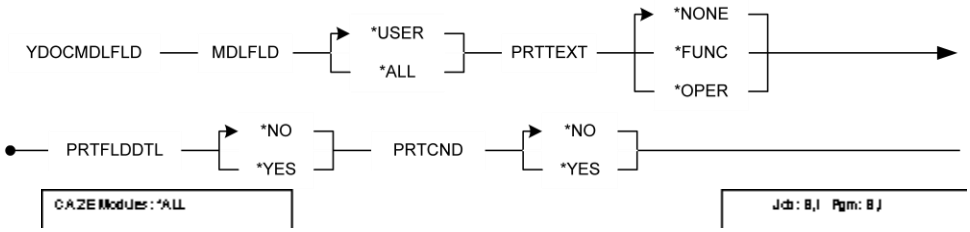
To print all details about the model files in a model library:

```
YDOCMDLF PRTTEXT(*FUNC) PRTREL(*YES) + PRTACCPTH(*YES) PRTMSG(*YES)
```

YDOCMDLFLD (Document Model Fields) Command

Documents the fields within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLFLD command.

MDLFLD

Specifies whether you want to omit system fields from the listing. Values for this parameter are described in the following:

- ***USER**—(default) List all user-defined fields in the model.
- ***ALL**—List all fields in the model, including system fields.

PRTTEXT

Specifies whether narrative text for the fields should be included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

PRTFLDDTL

Specifies whether you want to include full details for the fields in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include field details.
- ***YES**—Include field details.

PRTCND

Specifies whether the names of any field conditions referenced by the fields are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not include conditions.
- ***YES**—Include conditions.

Notes

A library containing the model files must be present in the library list of the job that executes the YDOCMDLFLD command.

Example

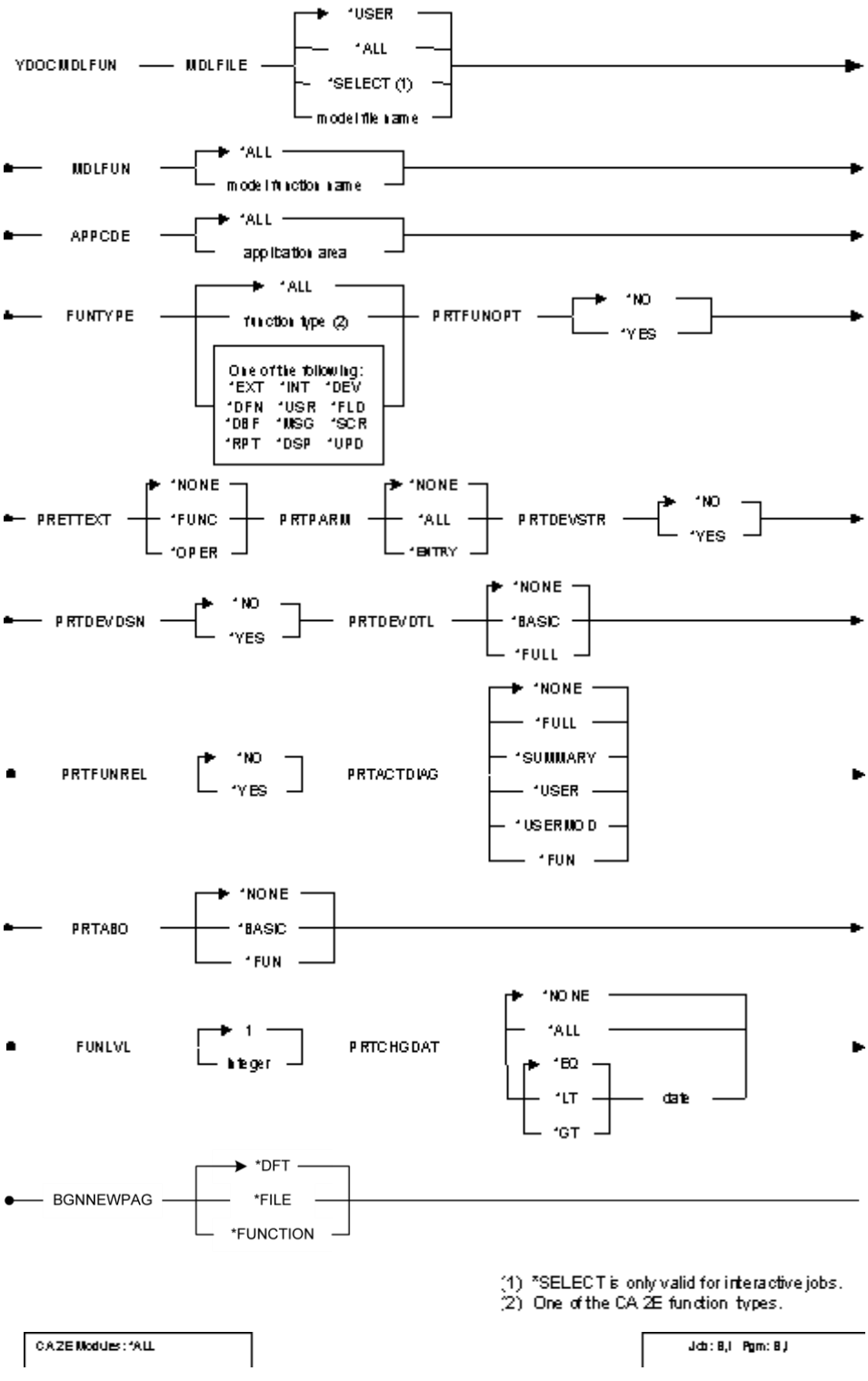
To print all details about the model fields in a model library:

```
YDOCMDLFLD PRTTEXT(*FUNC) PRTFLDDTL(*YES) + PRTCND(*YES)
```

YDOCMDLFUN (Document Model Functions) Command

Documents the functions within a design model.

Optional


YDOCMDLFUN
MDLFILE
*USER
*ALL
*SELECT (1)
model file name
MDLFUN
*ALL
model function name
APPCDE
*ALL
application area
FUNTYPE
*ALL
function type (2)
One of the following:
*EXT *INT *DEV
*DFN *USR *FLD
*DBF *MSG *SCR
*RPT *DSP *UPD
PRTFUNOPT
*NO
*YES
PRETTEXT
*NONE
*FUNC
*OPER
PRTPARM
*NONE
*ALL
*ENTRY
PRTDEVSTR
*NO
*YES
PRTDEVDSN
*NO
*YES
PRTDEVDTL
*NONE
*BASIC
*FULL
PRTFUNREL
*NO
*YES
PRTACTDIAG
*NONE
*FULL
*SUMMARY
*USER
*USERMOD
*FUN
PRTABO
*NONE
*BASIC
*FUN
FUNLVL
1
integer
PRTCHGDAT
*NONE
*ALL
*EQ
*LT
*GT
date
BGNNEWPAG
*DFT
*FILE
*FUNCTION
(1) *SELECT is only valid for interactive jobs.
(2) One of the CA 2E function types.
CA2EModules: *ALL
Job: B,I Pgm: B,I

Parameters

The following are parameters for the YDOCMDLFUN command.

MDLFILE

Name of a file, or generic name of some files, to which the functions that are listed belong. Values for this parameter are described in the following:

- ***USER**—(default) List functions that belong to all user-defined files in the model.
- ***ALL**—List functions that belong to all files in the model, including system files.
- ***SELECT**—Display the files in the current model, one of which may be selected.

MDLFUN

Name of a function that is listed, or generic name of functions that are listed. The value for this parameter is as follows:

- ***ALL**—(default) List all functions for the specified files.

APPCDE

Name of an application area to which the files referenced by the listed functions belong. The value for this parameter is as follows:

- ***ALL**—(default) List functions that reference files from all application areas.

FUNTYPE

Types of functions you want listed. Must be one of the standard function types or a special value. Values for this parameter are described in the following:

- ***ALL**—(default) List all types of functions.
- ***INT**—List only internal functions.
- ***EXT**—List only external functions.
- ***DBF**—List only database functions.
- ***DEV**—List only device functions.
- ***DFN**—List only define screen/report format functions.
- ***FLD**—List only field functions.
- ***MSG**—List only message functions.
- ***USR**—List only user functions.
- ***SCR**—List only display device functions.
- ***RPT**—List only report device functions.
- ***DSP**—List only display functions.
- ***UPD**—List only edit functions.
- **Other values**—See the notes section for function types and classes.

PRTFUNOPT

Specifies whether the function options are included in the generated documentation. Values for this parameter are described in the following:

- ***NO**—(default) Do not print function options.
- ***YES**—Print function options.

PRTTEXT

Specifies whether any narrative text for the function is included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

PRTPARM

Specifies whether the parameters of the listed functions are included in the listing. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any function parameters.
- ***ENTRY**—Include the entry parameters of each function in the listing.
- ***ALL**—Include the *ENTRY parameters of each function in the listing. Include the parameters used to call other functions in each function's action diagram.

PRTDEVSTR

Specifies whether the device structures of the listed functions are included in the listing. Only report design functions (types PRTOBJ and PRTFIL) have device structures. Values for this parameter are described in the following:

- ***NO**—(default) Do not print any function device structures.
- ***YES**—Include the function device structures in the listing.

PRTDEVDSN

Specifies whether the device (panel and report) designs of the listed functions are included in the listing. Values for this parameter are described in the following:

- ***NO**—(default) Do not print any function device designs.
- ***YES**—Include the function device designs in the listing.

PRTDEVDTL

Specifies whether the device details (that is, field details listed by format) of the listed functions are included in the listing. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any function device details.
- ***BASIC**—Print basic details of all fields on each format of the device design.
- ***FULL**—Print details of display attributes of fields present on the device design (that is, not hidden or dropped) as well as basic details of all fields on each format of the device design.

PRTFUNREL

Specifies whether the device design relations of the listed functions are included in the listing. Values for this parameter are described in the following:

- ***NO**—(default) Do not print any function relations.
- ***YES**—Include the function relations in the listing.

PRTACTDIAG

Specifies whether the action diagrams of the listed functions are included in the listing, and if so, to what level of detail the diagrams are reported. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any action diagrams.
- ***FULL**—Print complete action diagram.
- ***SUMMARY**—Print only the summary or top level of the action diagram.
- ***USER**—Print only the user modifiable points in each action diagram.
- ***USERMOD**—Print only the user modifications to action diagrams.
- ***FUN**—Print only the calls to other functions.

PRTABO

Specifies whether the action bar details of the listed functions are included in the listing. Ignored if PRTDEVDSN (*NO) is specified. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any function action bar details.
- ***BASIC**—Print basic details of the action bars on the device design.
- ***FULL**—Print full details of the action bars on the device design.

FUNLVL

Indicates the number of invocation levels of the functions' action diagrams to print. Ignored if PRTACTDIAG(*NONE) is specified. Values for this parameter are described in the following:

- **1**—Print the action diagram of the first level function only.
- **2-9**—Print the action diagrams of the functions called by the first level function, and those called by the second level function, up to the specified level.

PRTCHGDAT

Specifies the change dates printed for the action diagrams of the listed functions. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any change dates.
- ***ALL**—Print all change dates.

Otherwise, PRTCHGDAT is a list parameter made up of two elements:

- Date operator:
  - ***EQ**—(default) Equal to
  - ***LT**—Less than
  - ***GT**—Greater than
- Date (entered in system date format)

BGNNEWPAG

Specifies whether documentation for a function or file is to start on a new page or not. Values for this parameter are described in the following:

- ***DFT**—(default) Start a new page on page overflow.
- ***FILE**—Start a new page for each file whose functions are to be documented.
- ***FUNCTION**—Start a new page for each function documented.

Notes

- A library containing the model files must be present in the library list of the job that executes the YDOCMDLFUN command.
- Specifying either *BASIC or *FULL for the PRTDEVDTL parameter provides a separate summary of any attached device user source functions.

The following table shows the function groups.

Function type	Abbreviation	Class	Sub-class	Implem. Type	Display/Update
Change object	*CHGOBJ	*DBF	-	*INT	-
Create object	*CRTOBJ	*DBF	-	*INT	-
Delete object	*DLTOBJ	*DBF	-	*INT	-
Retrieve object	*RTVOBJ	*DBF	-	*INT	-
Define screen format	*DFNSCRFMT	*DFN	*SCR	-	-

Define report format	*DFNRPTFMT	*DFN	*RPT	-	-
Prompt record	*PMTRCD	*DEV	*SCR	*EXT	*DSP
Display record 1	*DSPRCD	*DEV	*SCR	*EXT	*DSP
Display record 2	*DSPRCD2	*DEV	*SCR	*EXT	*DSP
Display record 3	*DSPRCD3	*DEV	*SCR	*EXT	*DSP
Edit record (1 screen)	*EDTRCD	*DEV	*SCR	*EXT	*UPD
Edit record (2 screens)	*EDTRCD2	*DEV	*SCR	*EXT	*UPD
Edit record (3 screens)	*EDTRCD3	*DEV	*SCR	*EXT	*UPD
Select record	*SELRCD	*DEV	*SCR	*EXT	*DSP
Display file	*DSPFIL	*DEV	*SCR	*EXT	*DSP
Edit file	*EDTFIL	*DEV	*SCR	*EXT	*UPD
Display transactions	*DSPTRN	*DEV	*SCR	*EXT	*DSP
Edit transactions	*EDTTRN	*DEV	*SCR	*EXT	*UPD
Print file	*PRTFIL	*DEV	*RPT	*EXT	-
Print object	*PRTOBJ	*DEV	*RPT	*INT	-
Execute internal funct.	*EXCINTFUN	*USR	-	*INT	-
Execute external funct.	*EXCEXTFUN	*USR	-	*EXT	-
Execute user program	*EXCUSRPGM	*USR	-	*EXT	-
Execute user source	*EXCUSRSRC	*USR	-	*INT	-
Send error message	*SNDERRMSG	*MSG	-	-	-
Send status message	*SNDSTSMSG	*MSG	-	-	-
Send information msg	*SNDINFMSG	*MSG	-	-	-
Send completion msg	*SNDCMPMSG	*MSG	-	-	-

Retrieve message	*RTVMSG	*MSG	-	-	-
Execute message	*EXCMSG	*MSG	-	*EXT	-
Derived field function	*FUNFLD	*FLD	-	*INT	-

Example

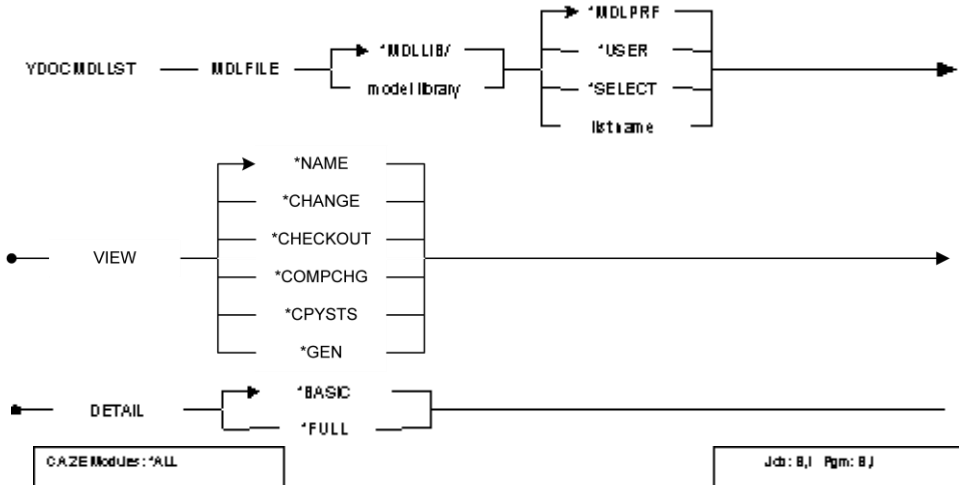
To print all details about the functions attached to all model files in a model library:

```
YDOCMDLFUN PRTTEXT(*FUNC) + PRTFUNPAR(*YES) PRTDEVDSN(*YES) + PRTACTDIAG(*YES)
```

YDOCMDLLST (Document a Model Object List) Command

This command allows a user to print the entries contained in a model object list.

Required



Parameters

The following are parameters for the YDOCMDLLST command.

MDLLST

The qualified name of the model object list that is documented. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The model object list name must be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—The model library name for the list.

VIEW

This parameter enables different data to be displayed for each list entry. It is prompted only if the DETAIL parameter is set to *BASIC. Values for this parameter are described in the following:

- ***NAME**—(default) Details are printed of the object name, attribute, type, and owner.
- ***CHANGE**—In addition to name, the change date and time details are printed.
- ***CHECKOUT**—In addition to name, the check out details are printed.
- ***COMPCHG**—In addition to name, the component change details are printed.
- ***CPYSTS**—In addition to name, the details relating to the copying of the object by the Copy Model Objects (YCPYMDLOBJ) command are printed.
- ***GEN**—In addition to name, the generation details are printed.

DETAIL

The level of detail which is displayed. Values for this parameter are described in the following:

- ***BASIC**—(default) Minimal level of detail.
- ***FULL**—Maximum level of detail.

Notes

- The MDLLST must exist before running the command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.

Example

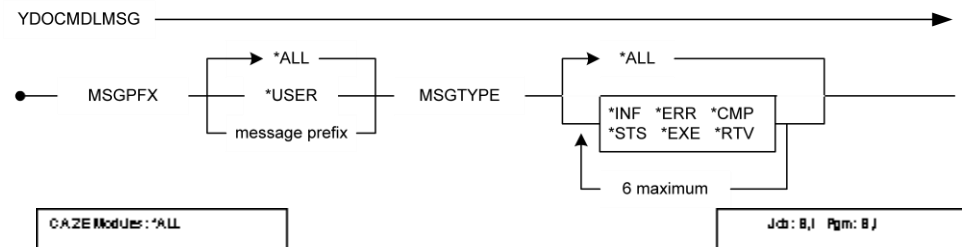
To document the model object list with the name retrieved from the user profile extension record of the current user in the model library contained in the current library list:

```
YDOCMDLLST MDLLST( *MDLLIB/*MDLPRF )
```

YDOCMDLMSG (Document Model Messages) Command

Documents the messages within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLMSG command.

MSGPFX

Prefix of messages which are listed. Values for this parameter are described in the following:

- ***ALL** —(default) List all message functions in the model, including shipped messages.
- ***USER** —List all user-defined message functions in the model.

MSGTYPE

Type of message function to be listed. Values for this parameter are described in the following:

- ***ALL** —(default) List all messages for the specified files.
- **Message type** —One of the six message function types:
  - INF
  - ERR
  - CMP
  - STS
  - EXC
  - RTV

Notes

A library containing the model files must be present in the library list of the job that executes the YDOCMDLMSG command.

Example

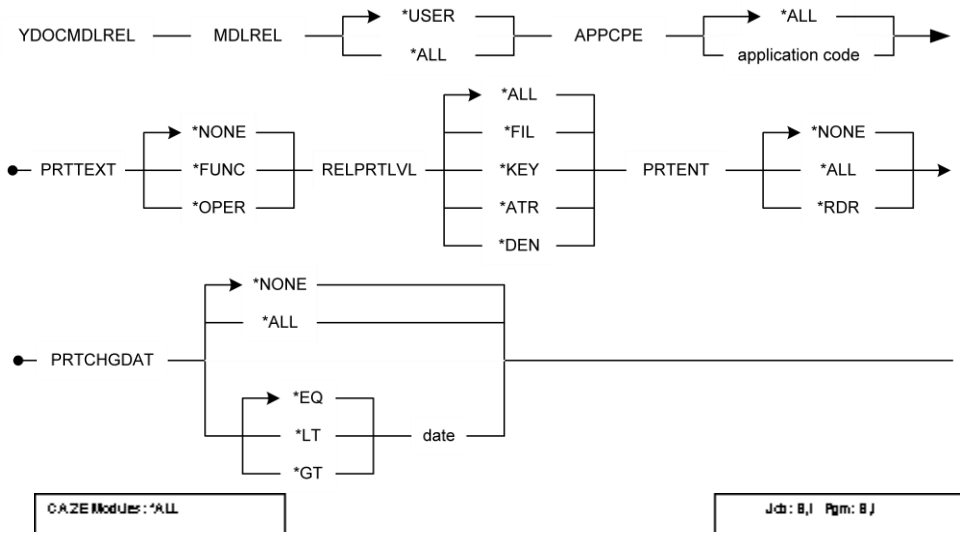
To print all details about the message functions in a model library:

```
YDOCMDLMSG
```

YDOCMDLREL (Document Model Relations) Command

Documents the relations within a design model.

Optional



Parameters

The following are parameters for the YDOCMDLREL command.

MDLREL

Listed relations. Values for this parameter are described in the following:

- ***USER**—(default) List only the user-defined relations in the model.
- ***ALL**—List all relations in the model including the relations that define internal objects.

APPCDE

Name of an application area that the files referenced by the listed relations belong to. The value for this parameter is as follows:

- ***ALL**—(default) Include relations that use files from all application areas.

PRTTEXT

Specifies whether narrative text for the relations is included in the generated documentation, and if so, which type of text. Values for this parameter are described in the following:

- ***NONE**—(default) Do not include narrative text.
- ***FUNC**—Include functional text.
- ***OPER**—Include operational text.

RELPRTLVL

Specifies which types of relations you want to include in the generated documentation. Values for this parameter are described in the following:

- ***ALL**—(default) Include all relations.
- ***FIL**—Include only file level relations.
- ***KEY**—Include only key level relations.
- ***ATR**—Include only field level relations.
- ***DF**—Include only Defined as relations.

PRTENT

Specifies whether the entries from file-to-file relations are included in the generated documentation. Values for this parameter are described in the following:

- ***NONE**—(default) Do not list entries.
- ***RDR**—List only redirected entries.
- ***ALL**—All entries from file-to-file relations (Owned by, Refers to) are listed after the relations that give rise to them.

PRTCHGDAT

Specifies whether the change dates for relations are printed. Values for this parameter are described in the following:

- ***NONE**—(default) Do not print any change dates.
- ***ALL**—Print all change dates.

Otherwise, PRTCHGDAT is a list parameter made up of two elements:

  - Date operator:

- ***EQ**—(default) Equal to.
- ***LT**—Less than.
- ***GT**—Greater than.

  - Date (entered in system date format)

Notes

A library containing the model files must be present in the library list of the job that executes the YDOCMDLREL command.

Example

To print details about all model files in a model library, including the functional narrative text:

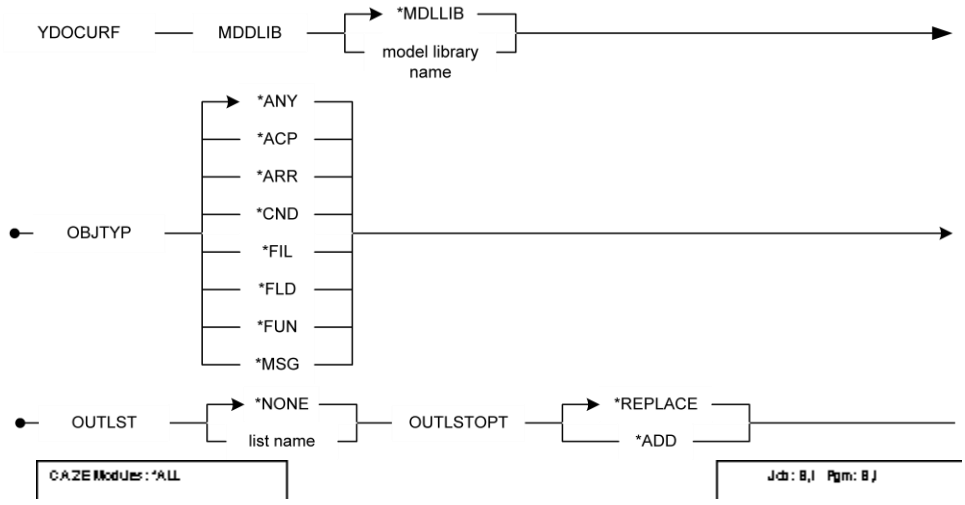
```
YDOCMDLREL PRTTEXT(*FUNC) RELPRTLVL(*ALL)
```

YDOCURF (Document Unreferenced Objects) Command

This command provides a model list of model objects that are unreferenced in the model.

Note: A model object that is identified by the YDOCURF command as unreferenced within a model may be referenced from outside the model by menus, messages, user defined programs, and so on.

Optional



Parameters

The following are parameters for the YDOCURF command.

MDLLIB

The name of the model library whose objects are to be analyzed. Values for this parameter are described in the following:

- ***MDDLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library.
- **library name**—The name of a specific model library.

OBJTYP

A list of up to six special values that may be used to select which object types should be analyzed by the command. Values for this parameter are described in the following:

- ***ANY**—(default) Special value meaning that any object type should be included in the analysis.
- ***ACP**—Include access paths in the analysis.
- ***ARR**—Include arrays in the analysis.
- ***CND**—Include conditions in the analysis.
- ***FIL**—Include files in the analysis.
- ***FLD**—Include fields in the analysis.
- ***FUN**—Include functions in the analysis.
- ***MSG**—Include messages in the analysis.

OUTLST

The name of the model object list that receives output from the command. Values for this parameter are described in the following:

- ***NONE**—(default) Special value meaning that no outlist processing is to be performed.
- **list name**—The name of the model object list to be used as output.

OUTLSTOPT

Specifies the action to take if the output list already exists. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list should be replaced with the output from this command.
- ***ADD**—The existing model object list should be augmented with the output from this command.

Example

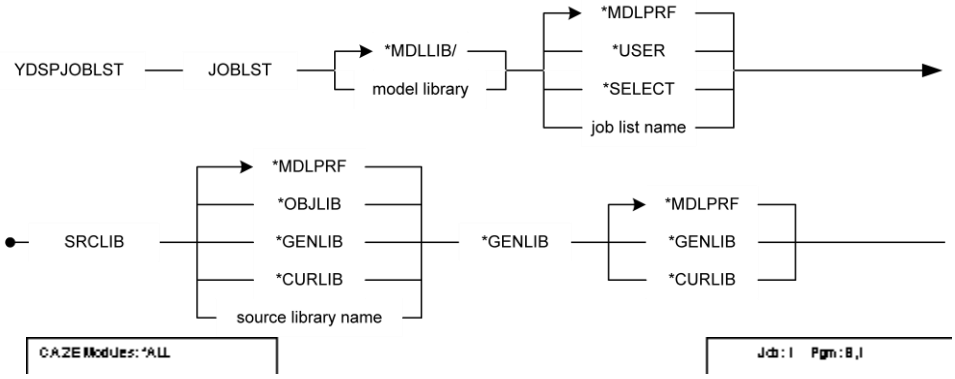
To create a list of unreferenced functions in the current model library in model object list, UNREFFUN.

```
YDOCURF OBJTYP(*FUN) OUTLST(UNREFFUN)
```

YDSPJOBLST (Display a Job List) Command

This command allows a user to display a CA 2E job list.

Optional



Parameters

The following are parameters for the YDSPJOBLST command.

JOBLST

The qualified name of the CA 2E job list that is displayed. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the list name is retrieved from the model profile of the current user.
- ***USER**—Special value meaning that the user profile name of the current user is used as the job list name.
- ***SELECT**—Special value meaning that the list is selected.
- **list name**—The name of the list that is used can be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the library.

SRCLIB

This parameter specifies the library from which source associated with job list entry is edited. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Use the source library defined in the model profile associated with the current user.
- ***OBJLIB**—The source library to use is the same one as that specified for the object library in the model profile associated with the current user.
- ***GENLIB**—Use the name of the generation library from the model value (YGENLIB) as a source library.
- ***CURLIB**—Use the current library as a source library.
- **library name**—The library name for source.

GENLIB

This parameter specifies the library in which to place compiled objects. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***GENLIB**—Retrieve the name of the generation library from the model value (YGENLIB).
- ***CURLIB**—Use the current library for invoking job.

Notes

- The library specified for the job list must be a valid model library.
- Note that the YGENSRC job entries are displayed in chronological order at the top of the subfile, above the generate/compile requests. This allows the user to select a particular YGENSRC. If one is selected, only the requests for that YGENSRC are displayed.
- The status fields shown in the subfile control relate to the job list entries. If a particular YGENSRC request is selected, then the statistics relate just to that YGENSRC.
- Type, Act, Status and YGENSRC can be used to select records in the list for display.

Example

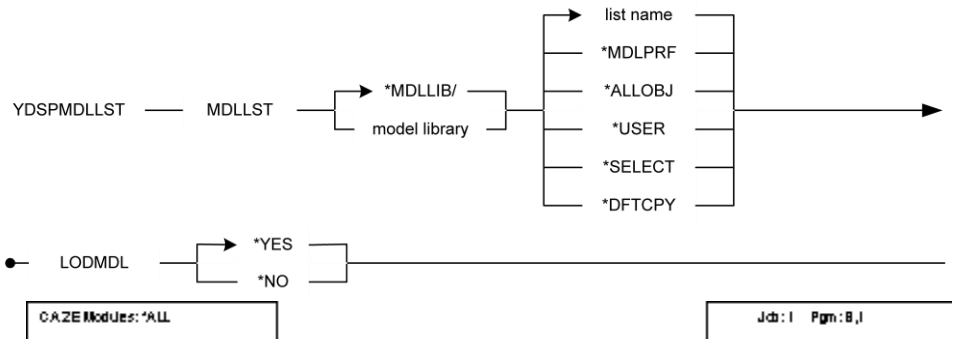
To display the current user's job list:

```
YDSPJOBLST JOBLST( *MDLLIB/*USER )
```

YDSPMDLLST (Display a Model Object List) Command

This command provides access to a display panel to view the contents of a model object list.

Optional



Parameters

The following are parameters for the YDSPMDLLST command.

MDLLST

The qualified name of the model object list that is displayed. Values for this parameter are described in the following:

- **list name**—(default) The model object list name must be entered.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are included in the command.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- ***DFTCPY**—Special value meaning that the model object list name is defaulted to YCPYLSTRFP.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—Model library name for the list.

LODMDL

Indicates whether the model environment is loaded before displaying the panel. Values for this parameter are described in the following:

- ***YES**—(default) Special value meaning that the model environment is to be loaded before displaying the panel.
- ***NO**—The model environment is to be loaded only if required to process a particular subfile option. Note that in this case:
  - A lock is not placed on the model before the panel is displayed.
  - The model will be loaded for each subfile option that requires it and the model will be unloaded when returning to the display. In other words, you incur additional overhead each time you use a subfile option that requires the model to be loaded.

Notes

- The MDLLST must exist before you run the command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.

Example

This section contains an example of the command as it might look using real data. To display model object list LST0001 from the model contained in the current library list:

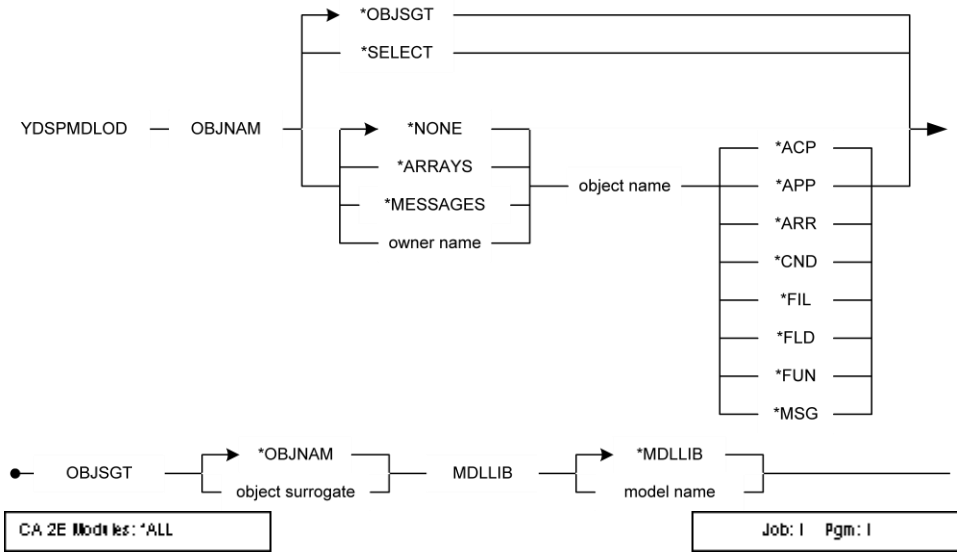
```
YDSPMDLLST MDLLST(*MDLLIB/LST0001)
```

YDSPMDLOD (Display Model Object Description) Command

This command allows a user access to an interactive panel that displays the details for a given model object.

Details relate to the creation and subsequent change of the object. For more information, please refer to the help text for the panel.

Required



Parameters

The following are parameters for the YDSPMDLOD command.

OBJNAM

The name of the object whose details are displayed. This parameter consists of three elements that together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate is to be used to identify the model object that is displayed.
- ***SELECT**—Single value indicating that the displayed object is selected using an interactive display function.
- **object owner name**—The character name of the object that owns the object to be displayed. Thus, for a function, the owning file would be entered.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **object name**—The character name of the displayed object.
- **object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

OBJSGT

Unique number identifier of the model object that is displayed. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the displayed model object.
- **object surrogate**—The surrogate number of the model object.

MDLLIB

The data model in which the object whose description is to be displayed resides. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model is the first one to be found in the current library list.
- **model name**—The name of a specific data model.

Notes

- A value other than *MDLLIB for MDLLIB may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- Model objects can either be identified by object name (OBJNAM) or by object surrogate key number (OBJSGT). If the OBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).

  Model object names are structured as follows:

Type	Name
ACP	File name/Access path name/'ACP'
APP	---/Application area code/'APP'
ARR	*Arrays/Array name/'ARR'
CND	Field name/Condition name/'CND'
FIL	---/File name/'FIL'
FLD	---/Field name/'FLD'
FUN	File name/Function name/'FUN'
MSG	*Messages/Message name/'MSG'

Example

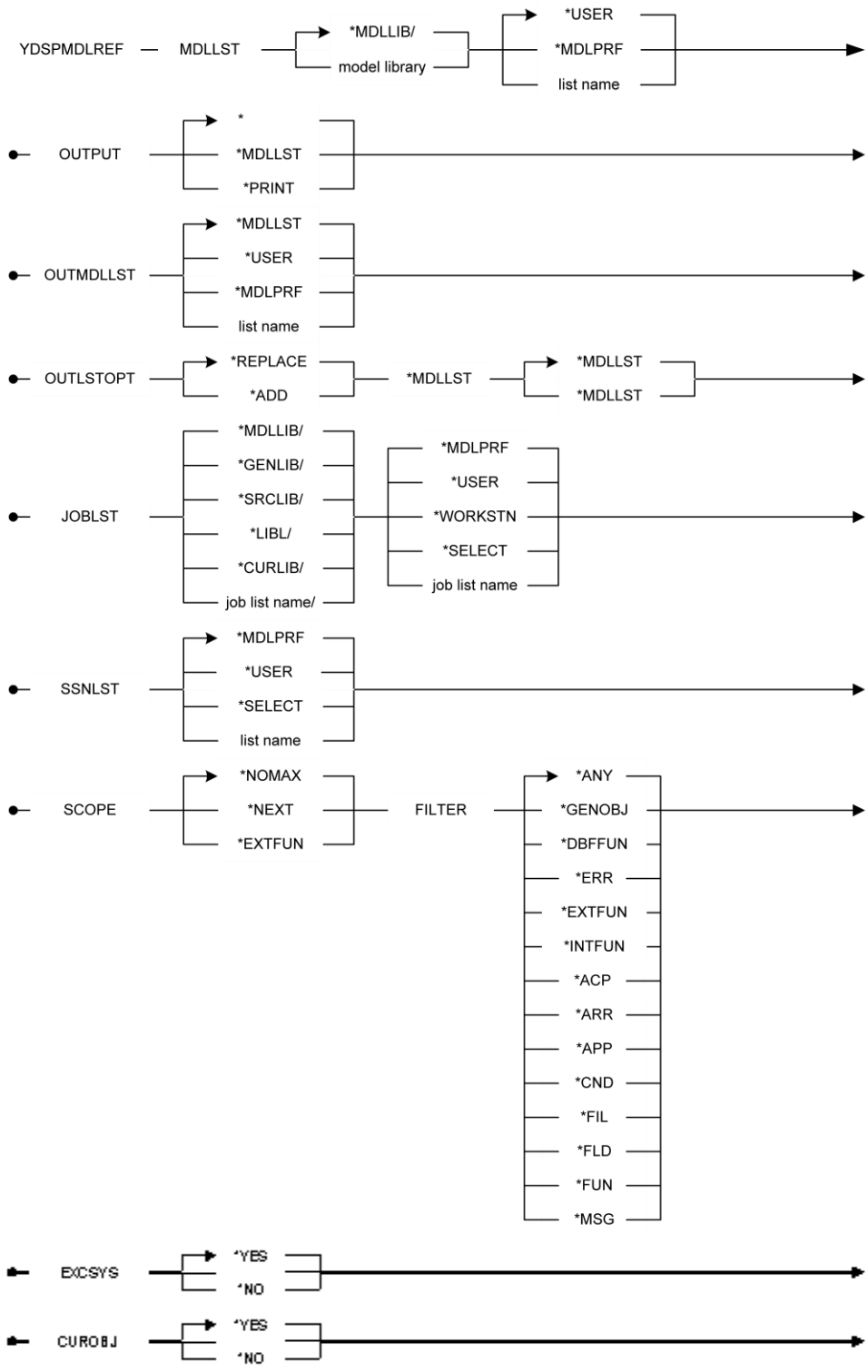
To display the Edit Order Details function, which is owned by the order details file, enter the following:

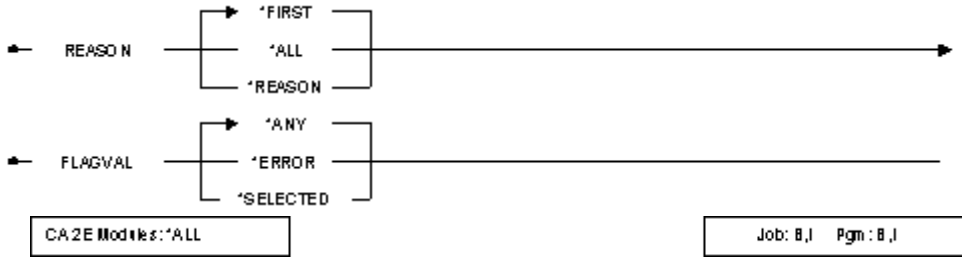
```
YDSPMDLOD OBJNAM( 'Order Details' 'Edit + Order Details' *FUN)
```

YDSPMDLREF (Display Model References) Command

This command allows a user to display references to a list of model objects. References can be displayed, printed or converted to a model object list.

Required


YDSPMDLREF
MDLLST
*MDLLIB/
model library
*USER
*MDLPRF
list name
OUTPUT
*
*MDLLST
*PRINT
OUTMDLLST
*MDLLST
*USER
*MDLPRF
list name
OUTLSTOPT
*REPLACE
*ADD
*MDLLST
*MDLLST
*MDLLST
JOBLST
*MDLLIB/
*GENLIB/
*SRCLIB/
*LIBL/
*CURLIB/
job list name/
*MDLPRF
*USER
*WORKSTN
*SELECT
job list name
SSNLST
*MDLPRF
*USER
*SELECT
list name
SCOPE
*NOMAX
*NEXT
*EXTFUN
FILTER
*ANY
*GENOBJ
*DBFFUN
*ERR
*EXTFUN
*INTFUN
*ACP
*ARR
*APP
*CND
*FIL
*FLD
*FUN
*MSG
EXCSYS
*YES
*NO
CUROBJ
*YES
*NO



Parameters

The following are parameters for the YDSPMDLREF command.

MDLLST

The qualified name of the model object list containing the objects whose references are displayed. Values for this parameter are described in the following:

- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user defaults for the current user in the specified model library.
- **list name**—The model object list name.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library.
- **library name**—The model library name for the list.

OUTPUT

This parameter determines how the references are presented. Values for this parameter are described in the following:

- *****—(default) References are displayed.
- ***MDLLST**—References are directed to a model object list. This option enables users to perform further list operations on the list entries.
- ***PRINT**—References are printed.

Note: If referenced objects are subject to filtering, then the primary object details are only printed if at least one referenced object satisfies the filtering criteria specified. This reduces the volume of output.

OUTMDLLST

The name of the model object list that receives output from the command. Values for this parameter are described in the following:

- ***MDLLST**—(default) Special value meaning that the output should be placed in the input list.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user defaults for the current user in the specified model library.
- **list name**—The target model object list name.

OUTLSTOPT

This parameter specifies the action to be taken if the output list already exists. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list should be replaced with the output from this command.
- ***ADD**—The existing model object list should be augmented with the output from this command.

EDIT

This parameter specifies whether the resulting output list is edited as part of processing. Values for this parameter are described in the following:

- ***NO**—(default) Editing of the list is not performed.
- ***YES**—Editing of the list is performed.

JOBLST

Qualified name of job list that contains the names of source members to be generated and/or compiled. If the nominated job list does not already exist, it will be created. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Retrieve the job list name from the model profile details of the current user.
- ***USER**—The job list name is the same name as the current user.
- ***WORKSTN**—Use device name of current work station as list name.
- ***SELECT**—Display a list of existing job lists, one of which may be selected. The name of the job list may be entered
- ***MDLLIB/**—The job list library is the first model library found in the library list.
- ***GENLIB/**—Use the generation library specified in the first model found in the library list.
- ***SRCLIB/**—Use the source library specified on the model profile of the current user.
- ***LIBL/**—The job list library is the first model found in the library list.
- ***CURLIB/**—The model library is found in the current library for the current job.

The job list library can be entered.

SSNLST

This parameter specifies the session list to use while editing the model. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The session list is to be retrieved from the model profile details for the current user.
- ***USER**—The session list has the same name as the current user.
- ***SELECT**—An interactive display is used to select a model object list to be used as the session list.

The name of the list can be entered.

For more information on the purpose of SSNLST, refer to the Change Model Profile command (YCHGMDLPRF).

SCOPE

This parameter allows the user to control the extent to which references are traced. Values for this parameter are described in the following:

- ***NOMAX**—(default) No maximum level of expansion.
- ***NEXT**—Expansion proceeds to the next level only.
- ***EXTFUN**—This option is intended for use with objects of type FUN. Expansion proceeds up to and including the first external function on each chain of references. This enables the user to determine the functions referenced by a given program.

FILTER

This parameter allows the user to specify filtering on the objects displayed. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering of objects is performed.
- ***GENOBJ**—Only generatable objects are displayed.
- ***DBFFUN**—Only database functions are displayed.
- ***ERR**—Only references to deleted objects are displayed.
- ***EXTFUN**—Expand references down to and including the first generated function.
- ***INTFUN**—Only internal functions are displayed.
- **Object type**—A specific object type may be entered for filtering purposes. They are: *ACP, *ARR, *APP, *CND, *FIL, *FLD, *FUN, and *MSG

CMTCDE

This parameter allows the inclusion of inactive AD code in the analysis:

- ***YES—**Inactive code is included.
- ***NO**—Inactive code is not included.
- ***IGN**—Code is not examined for active/inactive status. The WRN field will not be populated. Processing of *IGN is typically much faster.
- ***MDLVAL**—Retrieve the value from the CA 2E model value (YCMTCDE).

EXCSYS

This parameter allows the exclusion of system objects from the analysis. System objects are the internal objects used by CA 2E such as the '*Standard Header/Footer' internal file. The value for this parameter is as follows:

- ***YES**—(default) System objects are excluded.
- ***NO**—Include system objects.

CUROBJ

This parameter allows the exclusion of non-current versions from the analysis. Currency only applies to objects that are supported for versioning. Values for this parameter are described in the following:

- ***YES**—(default) Only current objects are included.
- ***NO**—Include system objects.

For more information on versions, see the "Working with CA 2E Model Objects" chapter in the *Generating and Implementing Applications* guide.

REASON

This parameter allows the reason for the dependency between objects to be part of the filtering process. Values for this parameter are described in the following:

- ***FIRST**—(default) The processing program will note the first reason when an object is encountered.
- ***ALL**—All subsequent encounters with a given object are included.
- **REASON**—A specific reason may be entered. The processing program will search for dependencies of the specified reason.

FLAGVAL

This parameter allows the input list entries to be filtered according to the value of the object selected indicator for each entry. Values for this parameter are described in the following:

- ***ANY**—No selection processing is performed.
- ***ERROR**—Only process entries flagged in error are considered.
- ***SELECTED**—Only process selected entries are considered.

Notes

- The model library specified must be a valid model library.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is not changed back after execution.
- If the output model list does not exist prior to running the command, it is created.
- If the list is printed, the display model references panel is used. This panel is described below.
- If the resulting list is printed, the Document Model Object list command (YDOCMDLLST) is called.
- The edit parameters are ignored if the running job is a batch job.
- The filtering parameter is applied to the expanded details. Thus, if the output is to display, the filtering may be adjusted without having to rebuild the expansion. If output is to a model list, the filtering is performed when the entries are written to the list.
- An indented output format is used for the *PRINT option whenever this is appropriate. If the default of *FIRST is used for the REASON filter, the objects are sorted in a keyed sequence. It would be inappropriate to indent this sequence because the relationships between the objects are purely arbitrary. However, any other value for REASON will cause the processing program to examine objects strictly in the order of expansion; the indented format here clearly shows the hierarchy of dependencies to assist developers.
- The reason code *DSLDBF signifies that the function object is used or referred as a default database Function. But the option was not selected in the Function options of the referring Function. For example, Create record, Change record, or Delete Record were set to 'N'. You can control the display of this reason code by changing the data area named "YDSLDBFRFA" in the model library. The command to change the data area is as follows:

  ```
  CHGDTAARA DTAARA(model-library/YDSLDBFRFA *ALL) VALUE(N)
  ```

Reference Table

The Ref level value refers to the level of reference by a given object to the original object that is displayed on the subfile control. This value may be changed by the user and will control the data shown in the subfile. By manipulating the range of levels available, all possible reference levels can be viewed individually or in combination.

The Ref type column in the subfile refers to the reason for the reference. The following table shows the possible reasons for references between model objects. Actual references to a given model object will depend upon the model object type.

Obj Type	To Type	Ref Code	Reason	Note
All		*ENTRY	Displayed only on the first panel when you access the YDSPMDLREF panel for a model object list rather than for a single list entry; for example, by using the YDSPMDLREF command or F22 from the YEDTMDLLST panel. It indicates that each list entry displayed has simply been updated to reflect its current state in the model; no references have been expanded. Your original list is not changed. You can now perform impact analysis on single list entries using the selection options. See also the online Help for the panel and the Impact Analysis topic in chapter 1 of Generating and Implementing Applications.	
ACP	ACP	*ASSACP	For associated access path, such as the UPD access path for a given RTV.	
	ACP	*REFACP	For referenced access path, such as the RTV access path on a file-to-file relation of a referring access path.	
	AUX	*ACPAUX	For access path auxiliaries.	1
	CND	*ACPCND	For access path condition.	
	FIL	*REFFIL	For owning FIL.	
	FLD	*PHYENT	For access path physical entries.	2
	FUN	*SELRCD	For select record override function. No references.	
CND	CND	*LSTCND	For list condition.	
	FLD	*REFFLD	For owning field.	
APP	-		No references.	

ARR	ACP	*ARRDTL	For array entry details.	
	FIL	*REFFIL	For associated FIL.	
	FLD	*ARRENT	For array entries.	
FIL	FIL	*REFFIL	For referenced FIL.	
	FIL	*SHRFIL	For file defining a sharing level.	
	FLD	*FILENT	For field appearing as an entry.	
	FLD	*MAPFLD	For field mapping (user-defined field types).	
	FLD	*PARFLD	For field mapping (user-defined field types).	
	FLD	*SRCFLD	For field mapping (user-defined field types).	
	FLD	*RNMFLD	For renamed entry.	
	FLD	*VRTENT	For field as virtual entry.	
	MSG	*RCDEXS	For record exists message.	
	MSG	*RCDNFD	For record not found message.	
FLD	CND	*FLDCND	For condition.	
	FLD	*REFFLD	For domain definition.	
	FUN	*EXTINT	For external/internal conversion function (user-defined field type).	
	FUN	*FLDUSR	Field-attached user source function (enabled)	
	FUN	*INTEXT	For internal/external conversion function (user-defined field type).	
FUN	ACP	*BASED	For based-on access path.	
	ACP	*FUNPAR	For function/message parameter definition.	
	ARR	*BASED	For based-on array.	
	ARR	*FUNPAR	For function/message parameter definition.	
	CND	*ABOCND	For action bar condition.	
	CND	*ACTCND	For action diagram condition.	
	CND	*DEVCND	For screen entry condition.	
	CND	*VLDCND	For field validation condition.	
	DSP	*DSPDTA	For display file details.	1
	FIL	*REFFIL	For owning FIL.	3
	FLD	*ACTION	For action diagram compares.	

FLD	*DEVENT	For device fields.	
FLD	*FUNPDT	For function/message parameter details.	
FLD	*PARAM	For action diagram parameter fields.	
FLD	*REFFLD	For derived field.	4
FLD	*SCRMAP	For screen field mapping (user-defined field types).	
FLD	*SCRPAR	For screen field mapping (user-defined field types).	
FLD	*SCRSRC	For screen field mapping (user-defined field types).	
FLD	*SCRTXT	For screen text field.	5
FUN	*ACTION	For action diagram functions.	
FUN	*DFTDBF	Default database function.	
FUN	*DSLDBF	Deselected database function.	
FUN	*DEVSTR	For device structure reference.	
FUN	*DEVUSR	Device-attached user source function	
FUN	*ENTUSR	Entry-attached user source function	
FUN	*FLDUSR	Field-attached user source function (disabled)	
FUN	*FMTUSR	Format-attached user source function	
FUN	*RPTUSR	Report-attached user source function	
FUN	*SCRUSR	Screen-attached user source function	
FUN	*SELRCD	For select record override function.	
HLP	*HLPDTA	For help file details.	1
MSG	*ACTION	For action diagram message.	
SRV	*FUNAUX	For function auxiliary.	

The Note column indicates whether or not there is a reciprocal entry in the usage table by the referenced object. The numbers in the column refer to the following explanations as to why there will not be a corresponding usage entry:

The referenced object is not a primary object.

Reference is for object redefinition only, the relationship is already noted using *FILENT.

The usage of FIL to FUN is not a useful piece of information. The usage might, however, be discernible via ACP. In other words, usage on a FIL will display any ACP objects attached to the FIL. These in turn will display any using FUN objects.

The relationship between FLD and FUN represents the interdependency of these two objects on each other. The FLD object actually uses the FUN. However, the FUN is not a primary object. It is not treated as a function in its own right and is not accessible except through the FLD object.

The reference of FUN to FLD for screen text is not treated as a bona fide usage since the fields involved are internal product objects only.

Example

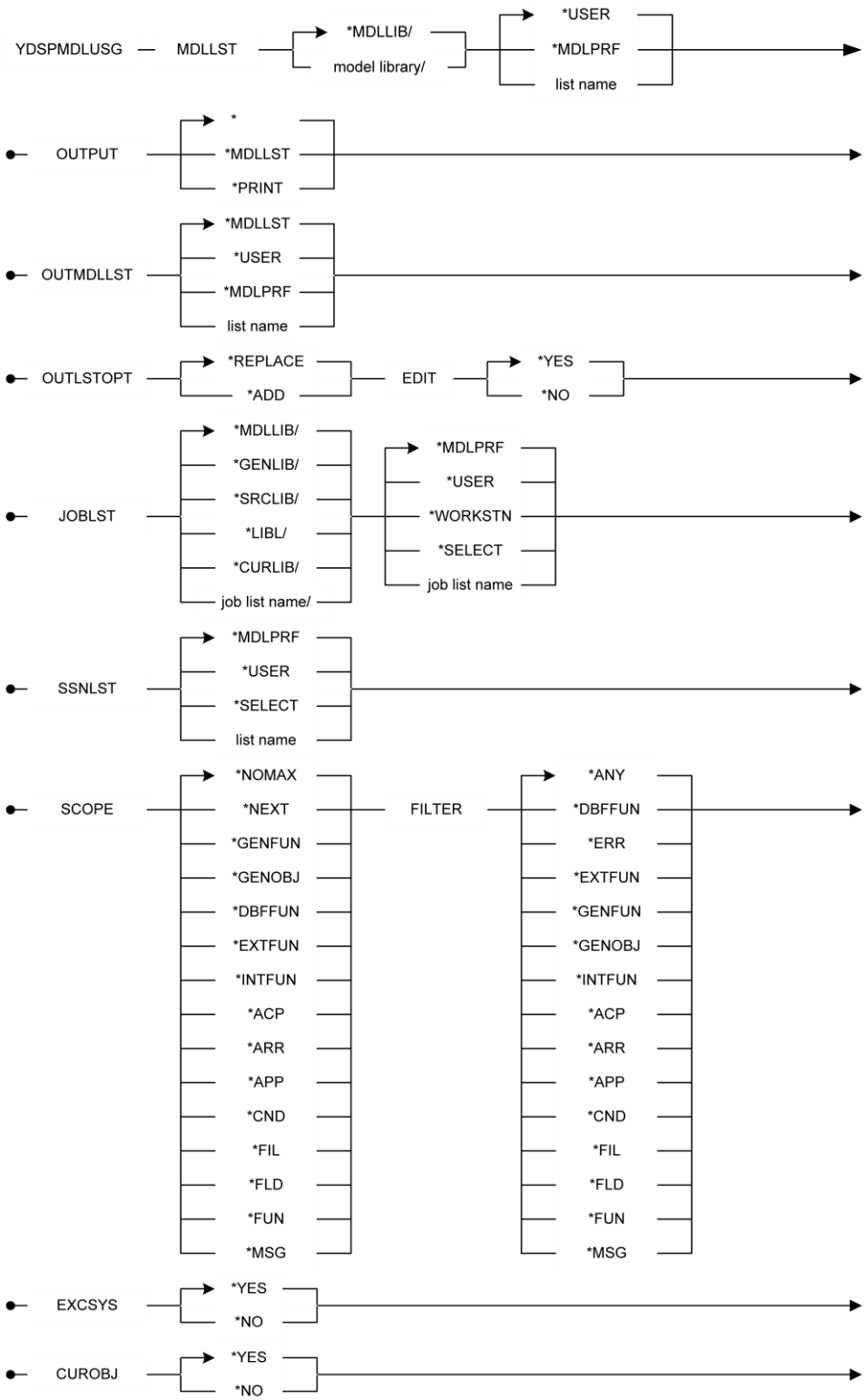
To edit the list of generatable object references to objects existing on model object list TMPLST:

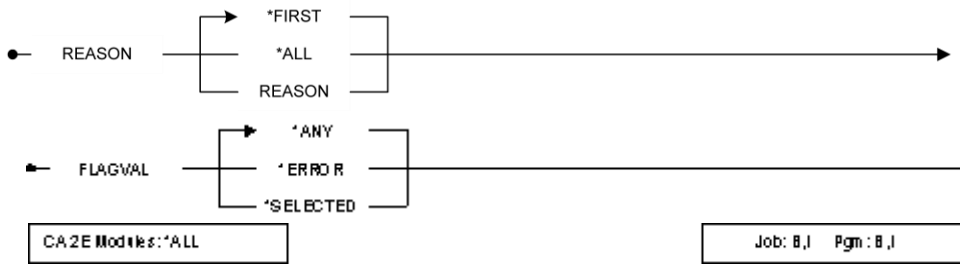
```
YDSPMDLREF MDLLST( *MDLLIB/TMPLST ) + OUTPUT( *MDLLST ) EDIT( *YES ) +
FILTER( *GENOBJ )
```

YDSPMDLUSG (Display Model Usages) Command

This command allows a user to display usages of a list of model objects. Usages can be displayed, printed or converted to a model object list.

Optional


YDSPMDLUSG
MDLLST
*MDLLIB/
model library/
*USER
*MDLPRF
list name
OUTPUT
*
*MDLLST
*PRINT
OUTMDLLST
*MDLLST
*USER
*MDLPRF
list name
OUTLSTOPT
*REPLACE
*ADD
EDIT
*YES
*NO
JOBLST
*MDLLIB/
*GENLIB/
*SRCLIB/
*LIBL/
*CURLIB/
job list name/
*MDLPRF
*USER
*WORKSTN
*SELECT
job list name
SSNLST
*MDLPRF
*USER
*SELECT
list name
SCOPE
*NOMAX
*NEXT
*GENFUN
*GENOBJ
*DBFFUN
*EXTFUN
*INTFUN
*ACP
*ARR
*APP
*CND
*FIL
*FLD
*FUN
*MSG
FILTER
*ANY
*DBFFUN
*ERR
*EXTFUN
*GENFUN
*GENOBJ
*INTFUN
*ACP
*ARR
*APP
*CND
*FIL
*FLD
*FUN
*MSG
EXCSYS
*YES
*NO
CUROBJ
*YES
*NO



Parameters

The following are parameters for the YDSPMDLUSG command.

MDLLST

The qualified name of the model object list containing the objects whose usages are displayed. Values for this parameter are described in the following:

- ***USER**—(default) Special value meaning that the model object list name is the same as the name of the current user.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user defaults for the current user in the specified model library.
- **list name**—The model object list name.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library.
- **library name**—The model library name for the list.

OUTPUT

This parameter determines how the usages are presented. Values for this parameter are described in the following:

- *****—(default) References are displayed.
- ***MDLLST**—Usages are directed to a model object list. This option enables users to perform further list operations on the list entries.
- ***PRINT**—Usages are printed.

OUTMDLLST

The name of the model object list that receives output from the command. Values for this parameter are described in the following:

- ***MDLLST**—(default) Special value meaning that the output should be placed in the input list.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user defaults for the current user in the specified model library.
- **list name**—The target model object list name.

OUTLSTOPT

This parameter specifies the action taken if the output list already exists. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list should be replaced with the output from this command.
- ***ADD**—The existing model object list should be augmented with the output from this command.

EDIT

This parameter specifies whether the resulting output list is edited as part of processing. Values for this parameter are described in the following:

- ***NO**—(default) Editing of the list is not performed.
- ***YES**—Editing of the list is performed.

JOBLST

Qualified name of job list that contains the names of source members to be generated and/or compiled. If the nominated job list does not already exist, it will be created. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Retrieve the job list name from the model profile details of the current user.
- ***USER**—The job list name is the same name as the current user.
- ***WORKSTN**—Use device name of current work station as list name.
- **SELECT**—Display a list of existing job lists, one of which may be selected. The name of the job list may be entered.
- ***MDLLIB/**—The job list library is the first model library found in the library list.
- ***GENLIB/**—Use the generation library specified in the first model found in the library list.
- ***SRCLIB**/—Use the source library specified on the model profile of the current user.
- ***LIBL/**—The job list library is the first model found in the library list.
- ***CURLIB**/—The model library is found in the current library for the current job. The job list library can be entered.

SSNLST

This parameter specifies the session list to use while editing the model. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The session list is retrieved from the model profile details for the current user.
- ***USER**—The session list has the same name as the current user.
- ***SELECT**—An interactive display is used to select a model object list used as the session list. The name of the list can be entered.

For more information on the purpose of SSNLST, see the Change Model Profile command (YCHGMDLPRF) section in this chapter.

SCOPE

This parameter allows the user to control the extent to which usages are traced. Values for this parameter are described in the following:

- ***NOMAX**—(default) No maximum level of expansion.
- ***NEXT**—Expansion proceeds to the next level only.
- ***GENFUN**—Expand objects up to and including the first generatable function.
- ***GENOBJ**—Expand objects up to and including the first generatable object.
- ***INTFUN**—Expand objects until the first internal function is encountered.
- ***EXTFUN**—Expand objects until the first generatable function.
- **object type**—The object type when expansion is to stop. Allowed object types are: *ACP, *APP, *ARR, *CND, *FIL, *FLD, *FUN, and *MSG.

FILTER

This parameter allows the user to specify filtering on the objects displayed. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering of objects is performed.
- ***DBFFUN**—Only database functions are displayed.
- ***ERR**—Only error usages are displayed; namely, usages by deleted objects.
- ***EXTFUN**—Only external functions are displayed.
- ***GENFUN**—Only generatable functions are displayed.
- ***GENOBJ**—Only generatable objects are displayed.
- ***INTFUN**—Only internal functions are displayed.
- **object type**—A specific object type may be entered for filtering purposes. They are: *ACP, *ARR, *APP, *CND, *FIL, *FLD, *FUN, and *MSG.

CMTCDE

This parameter allows the inclusion of inactive AD code in the analysis:

- ***YES—**Inactive code is included.
- ***NO**—Inactive code is not included.
- ***IGN**—Code is not examined for active/inactive status. The WRN field will not be populated. Processing of *IGN is typically much faster.
- ***MDLVAL**—Retrieve the value from the CA 2E model value (YCMTCDE).

EXCSYS

This parameter allows the exclusion of system objects from the analysis. System objects are the internal objects used by CA 2E such as the *Standard Header/Footer internal file. Values for this parameter are described in the following:

- ***YES**—(default) System objects are excluded.
- ***NO**—Include system objects.

CUROBJ

This parameter allows the exclusion of non-current versions from the analysis. Currency only applies to objects that are supported for versioning. Values for this parameter are described in the following:

- ***YES**—(default) Only current objects are included.
- ***NO**—Include system objects.

For more information on versions, see the "Working with CA 2E Model Objects" chapter in the *Generating and Implementing Applications* guide.

REASON

This parameter allows the reason for the dependency between objects to be part of the filtering process. Values for this parameter are described in the following:

- ***FIRST**—(default) The processing program will note the first reason when an object is encountered.
- ***ALL**—All subsequent encounters with a given object are included.
- **REASON**—A specific reason may be entered. The processing program will search for dependencies of the specified reason.

FLAGVAL

This parameter allows the input list entries to be filtered according to the value of the object selected indicator for each entry. Values for this parameter are described in the following:

- ***ANY**—No selection processing is performed.
- ***ERROR**—Only process entries flagged in error are considered.
- ***SELECTED**—Only process selected entries are considered.

Notes

- The model library specified must be a valid model library.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is not changed back after execution.
- If the output model list does not exist prior to running the command, it is created.
- If the list is to be printed, the display model object references panel is used. This panel is described below.
- If the resulting list is to be printed, the Document Model Object list command (YDOCMDLLST) is called.
- The edit parameters are ignored if the running job is a batch job.
- The scope parameter will limit the expansion. This will normally be useful to prevent unnecessary objects from being included in the analysis. Note that a scoped expansion means that an expansion will need to be repeated if additional objects are to be included.
- The filtering parameter is applied to the expanded details. Thus, if the output is to display, the filtering may be adjusted without having to rebuild the expansion. If output is to a model list, the filtering is performed when the entries are written to the list.
- An indented output format is used for the *PRINT option whenever this is appropriate. If the default of *FIRST is used for the REASON filter, the objects are sorted in a keyed sequence. It would be inappropriate to indent this sequence because the relationships between the objects are purely arbitrary. However, any other value for REASON will cause the processing program to examine objects strictly in the order of expansion; the indented format here clearly shows the hierarchy of dependencies to assist developers.
- The reason code *DSLDBF signifies that the function object is used or referred as a default database Function. But the option was not selected in the Function options of the referring Function. For example, Create record, Change record, or Delete Record were set to 'N'. You can control the display of this reason code by changing the data area named "YDSLDBFRFA" in the model library. The command to change the data area is as follows:

  ```
  CHGDTAARA DTAARA(model-library/YDSLDBFRFA *ALL) VALUE(N)
  ```

Usage Table

The Usg level value refers to the level of usage of a given object to the original object that is displayed on the subfile control. This value may be changed by the user and will control the data shown in the subfile. By manipulating the range of levels available, all possible usage levels can be viewed individually or in combination.

The Usg type column in the subfile refers to the reason for the usage. The following table shows the possible reasons for usages between model objects. Actual usages to a given model object depend upon the model object type.

Obj Type	Used by Object Type	Usage Code	Reason	Note
All		*ENTRY	Displayed only on the first panel when you access the YDSPMDLUSG panel for a model object list rather than for a single list entry; for example, by using the YDSPMDLUSG command or F20 from the YEDTMDLLST panel. It indicates that each list entry displayed has simply been updated to reflect its current state in the model; no usages have been expanded. Your original list is not changed. You can now perform impact analysis on single list entries using the selection options. See also the online Help for the panel and the Impact Analysis topic in chapter 1 of Generating and Implementing Applications.	
ACP	ACP	*ACPENT	For joining access path.	
	ACP	*ASSACP	For associated access path.	
	ACP	*REFACP	For referring access path. Subject access path is used as referenced access path due to file-to-file relation.	
	ARR	*ARRDTL	For array detail definition.	
	FUN	*BASED	Function based on access path.	
	FUN	*FUNPAR	For function/message parameter definition.	
	FUN	*RELFUN	For functions using related access path. That is, access path is present as a *REFACP and FUN uses access path, such as for validation.	1
APP	FIL	*APPFIL	For association in application area.	

ARR	FUN	*BASED	Function based-on array.	
	FUN	*FUNPAR	For function/message parameter definition.	
CND	ACP	*ACPCND	For access path condition.	
	CND	*LSTCND	Member of list condition.	
	FIL	*MAPFLD	For field mapping (user-defined field types).	
	FLD	*DFTFLD	Default field condition.	
	FLD	*FLDCND	Field condition.	
	FUN	*ABOCND	Action bar condition.	
	FUN	*ACTCND	Action diagram condition.	
	FUN	*DEVCND	Device entry condition.	
	FUN	*DFTDEV	Default device entry condition.	
	FUN	*INPOVR	Device input override condition.	
	FUN	*OUTOVR	Device output override condition.	
	FUN	*PARAM	Action diagram parameter.	
	FUN	*SCRMAP	For screen field mapping (user-defined field types).	
FIL	ACP	*REFFIL	For owning file.	
	APP	*APPARA	For application area.	2
FLD	ACP	*ACPCND	For access path condition.	3
	ARR	*ARRENT	For array entry.	
	FIL	*ENTAUX	For entry redirection.	
	FIL	*FILENT	For key/attribute entry.	
	FIL	*MAPFLD	For field mapping (user-defined field types).	
	FIL	*VRTENT	For virtual entry.	
	FLD	*REFFLD	For domain definition.	
	FLD	*RNMFLD	For rename by entry.	
	FUN	*ACTION	For action diagram action.	
	FUN	*ACTCND	Action diagram condition.	
	FUN	*ACTCMP	Action diagram compare.	
	FUN	*DEVCND	For device conditioning.	

	FUN	*DEVENT	For device entry.	
	FUN	*FUNPAR	For function/message parameter definition.	
	FUN	*FUNPDT	For function/message parameter detail definition.	
	FUN	*PARAM	For action diagram parameter.	
	FUN	*SCRMAP	For screen field mapping (user-defined field types).	
FUN	ACP	*SELRCD	For select record override function.	
	ACP	*ACPFUN	For access path function.	
	FLD	*EXTINT	For external/internal conversion function (user-defined field type).	
	FLD	*FLDUSR	Field-attached user source function (enabled)	
	FLD	*INTEXT	For internal/external conversion function (user-defined field type).	
	FUN	*ARCVSN	For archived version.	1
	FUN	*ACTION	For action diagram function.	
	FUN	*DFTDBF	Default database function.	
	FUN	*DSLDBF	Deselected database function.	
	FUN	*DEVSTR	For device structure usage.	
	FUN	*DEVUSR	Device-attached user source function	
	FUN	*ENTUSR	Entry-attached user source function	
	FUN	*FLDUSR	Field-attached user source function (disabled)	
	FUN	*FMTUSR	Format-attached user source function	
	FUN	*RPTUSR	Report-attached user source function	
	FUN	*SCRUSR	Screen-attached user source function	
	FUN	*SELRCD	For select record override function.	
MSG	FIL	*RCDEXS	For record exists message.	
	FIL	*RCDNFD	For record not found message.	
	FUN	*ACTION	For action diagram function.	
	MSG	*ARCVSN	For archived version.	1

The Note column indicates whether or not there is a reciprocal entry in the reference table by the using object. The numbers in the column refer to the following explanations as to why there will not be a corresponding reference entry:

The using object is not required by the used object as part of its definition. Thus, it is not shown in the reference table.

FIL objects are not included as defining an application area because this would make the scope of APP references too large. For this reason there is no corresponding entry for the APP object type in the reference table.

The reference of ACP to FLD is accomplished through the Access Path Condition (*ACPCND on the reference table).

Developers will note from the table that there is no direct relationship between objects of type FLD and ACP. That is, a change to a FLD has no direct effect on the ACP objects that use it. The change is propagated through the FIL objects on which the FLD appears as a file entry. This is a valuable feature in the product, where it provides the capacity to accommodate changes to the data model. When an ACP is required for DDS source generation, or for the construction of a device design, the structure is determined dynamically, incorporating any changes that may have occurred since the last time the structure was required. The dynamic nature of ACP objects means, however, that there is no database representation available for interrogation by impact analysis. The result is that changes to a FLD impact the FILs which use it and are propagated to all ACP objects, even if the FLD is not used by each individual ACP.

Example

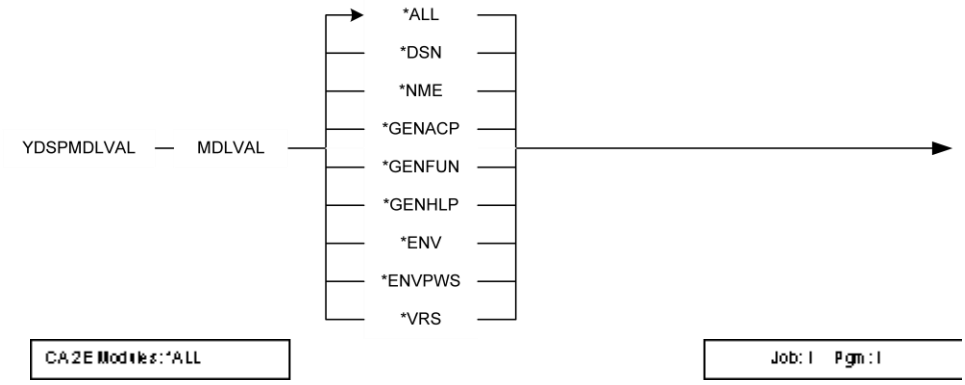
To print usages of objects present on model object list WRKLST in library UUMDL, including usages up to the first external function:

```
YDSPMDLUSG MDLLST( UUMDL/WRKLST ) + OUTPUT( *PRINT ) SCOPE( *EXTFUN )
```

YDSPMDLVAL (Display Model Value) Command

Displays the model values.

Optional



Parameters

The following are parameters for the YDSPMDLVAL command.

MDLVAL

Type of model value displayed. Values for this parameter are described in the following:

- ***ALL**—(default) Display all model values.
- ***DSN**—Display model values controlling design options.
- ***NME**—Display model values controlling naming options.
- **GENACP**—Display model values controlling access path generation options.
- **GENFUN**—Display model values controlling function generation options.
- ***GENHLP**—Display the model values controlling help generation options.
- ***ENV**—Display model values controlling development and run time environment options.
- ***ENVPWS**—Display model values controlling PWS development and run time environment options.
- ***VRS**—Display model values recording product version levels.

For more information on the role of each model value, refer to the Change Model Value command (YCHGMDLVAL).

Example

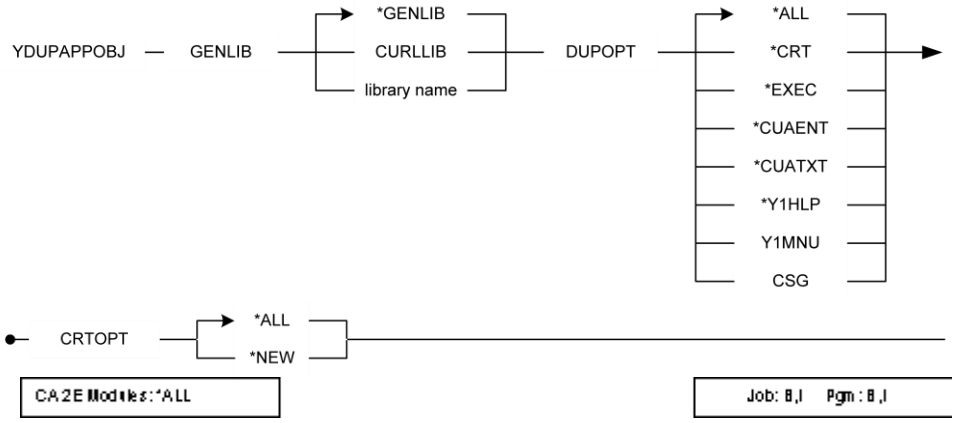
To display all model values:

```
YDSPMDLVAL MDLVAL(*ALL)
```

YDUPAPPOBJ (Duplicate Application Objects) Command

Duplicates into a named library the application objects from Synon/2E that are required to run Synon/2E generated programs. This command is useful when you want to run an application independently of Synon/2E; that is, without the Synon/2E product library in your library list. You can, as an option, duplicate Synon/1E objects necessary to display help text and menus.

Optional



Parameters

The following are parameters for the YDUPAPPOBJ command.

GENLIB

Name of library into which required objects are duplicated. Values for this parameter are described in the following:

- ***GENLIB**—(default) Use the default generated source library name as specified by the YGENLIB model value in the first model found in the library list.
- ***CURLIB**—Use current library for invoking job.

DUPOPT

Types of application objects that are duplicated. Values for this parameter are described in the following:

- ***ALL**—(default) Duplicate all objects.
- ***CRT**—Only duplicate objects needed for compilation.
- ***EXEC**—Only duplicate objects needed for execution, both for CUA Text and CUA Entry.
- **CUAENT**—Duplicate objects needed for execution but without support for CUA Text.
- ***CUATXT**—Duplicate objects needed for the CUA Text subset.
- ***Y1HLP**—Duplicate CA 2E Toolkit objects needed for help text display.
- ***Y1MNU**—Duplicate CA 2E Toolkit objects needed for menu display.
- ***TRG**—Duplicate 2E trigger-related objects.
- ***WS**—Duplicate 2E web service related objects.

CRTOPT

Duplicate existing objects option. Values for this parameter are described in the following:

- ***ALL**—(default) Duplicate all objects. Replace any existing objects with updated versions.
- ***NEW**—Only duplicate objects that do not already exist in the destination library.

Notes

- You should implement this command over your generation library or over a copy of your generation library.
- The source for the CA 2E objects is supplied in the Synon/2E shipped source library.

Example

To duplicate the CA 2E- and CA 2E Toolkit-required objects into generation library MYGEN:

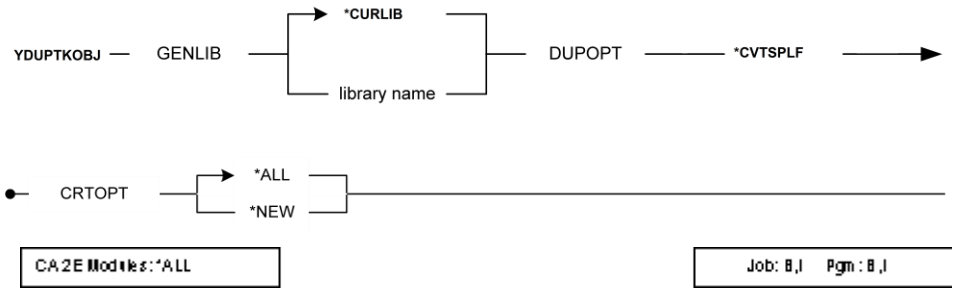
```
YDUPAPPOBJ GENLIB(MYGEN)
```

For a diagram illustrating the entries of the two intersecting lists, see the YOPRMDLLST command in this chapter. You can also invoke this command using its short form, Y2

YDUPTKOBJ (Duplicate Toolkit Objects)

Duplicates into a named library the toolkit objects from CA 2E that are required to run CA 2E generated programs. This command is useful when you want to run a toolkit command independently of CA 2E Toolkit; that is, without the CA 2E product library in your library list. You can, as an option, duplicate CA 1E objects necessary to display help text and menus.

Optional



Parameters

The following are parameters for the YDUPTKOBJ command.

TOLIB

Name of library into which required objects are duplicated. Values for this parameter are described in the following:

- ***CURLIB**—Use current library for invoking job.
- **Named Library**

Important! You should not run YDUPTKOBJ with Y1SY or any product libraries as the target.

DUPOPT

Types of toolkit objects that are duplicated. Value for this parameter is described in the following:

- ***CVTSPLF**— Only duplicated objects needed for converting spooled files.

CRTOPT

Duplicate existing objects option. Values for this parameter are described in the following:

- ***ALL**—(default) Duplicate all objects. Replace any existing objects with updated versions.
- ***NEW**—Only duplicate objects that do not already exist in the destination library.

Example

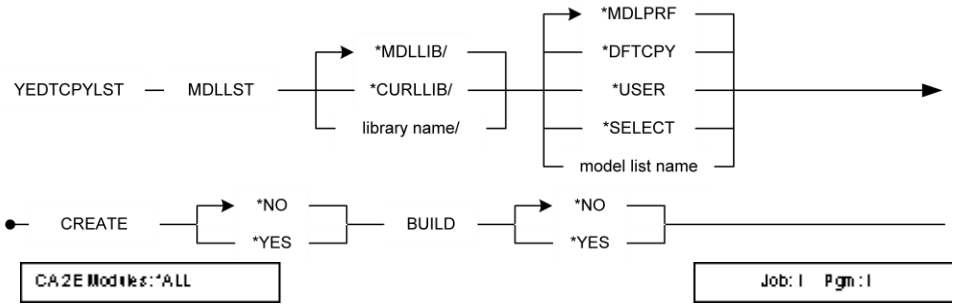
To duplicate the CA 2E Toolkit-required objects into generation library MYGEN:

```
YDUPPKOBJ TOLIB(MYGEN)
```

YEDTCPYLST (Edit Model Object List for Copy) Command

Calls an interactive program to edit a model object list that specifies the model objects that are to be copied between models.

Optional



Parameters

The following are parameters for the YEDTCPYLST command.

BUILD

Indicates whether the specified list is built, if it does not exist, or added to if it does exist. Values for this parameter are described in the following:

- ***NO**—(default) The list is not built.
- ***YES**—The Build Model Object List command (YBLDMDLLST) is prompted to create or augment the list.

Notes

Calls an interactive program to edit a model object list for a design model. The model list can subsequently be used by the command Copy Model Objects (YCPYMDLOBJ):

- To determine which objects from the source model are copied to another model.
- To obtain an alternative name to give to a object in the destination model to which it is copied.

Example

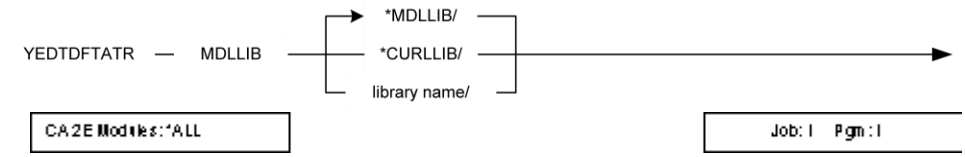
To edit a model object list called MYLIST in model library MYNEWMDL:

```
YEDTCPYLST MDLLST(MYNEWMDL/MYLST)
```

YEDTDFTATR (Edit Default Display Attributes) Command

Calls an interactive program to edit the default display attributes given to fields on screen designs created with CA 2E.

Optional



Parameters

The following are parameters for the YEDTDFTATR command.

MDLLIB

Name of library containing the name of a design model for which the default attributes are changed. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Use the first model library found in the library list.
- ***CURLIB**—Use current library for invoking job.

Notes

The display attributes are used when creating new designs or when adding new fields to existing screen designs. The field attributes of existing fields on existing designs are not changed. You can override the display attributes of an individual field using the screen editor (Edit Screen Field Display Attributes display).

Example

To call the program to change the display attributes:

```
YEDTDFTATR
```

A panel will appear with values that can be altered.

YEDTMDL (Edit Model) Command

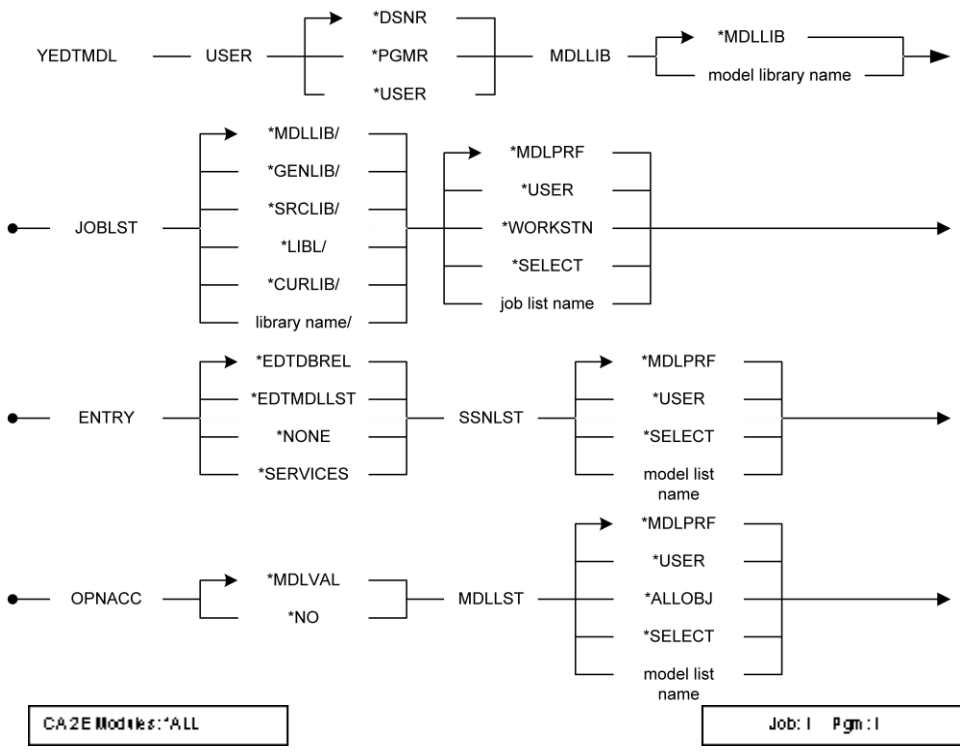
Calls an interactive program to enter and edit the first CA 2E model in your library list.

Short Form

Y2

Long Form

Optional



Parameters

The following are parameters for the YEDTMDL command.

USER

Type of user. Values for this parameter are described in the following:

- ***DSNR**—(default) A designer can change any aspect of the model, including the database.
- ***PGMR**—A programmer can add or change any functions that are in the model, but cannot alter the relations, files or fields.
- ***USER**—A user can view all aspects of the model but cannot change any design objects. This class of user is useful to allow the data model to be examined without the possibility of change.

MDLLIB

This parameter specifies the data model that is edited. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model to be edited is the first one found in the current job's library list.
- **model name**—The model library name.

JOBLST

Qualified name of job list that contains the names of source members to be generated and/or compiled. If the nominated job list does not already exist, it will be created. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Retrieve the job list name from the model profile details of the current user.
- ***USER**—The job list name is the same name as the current user.
- ***WORKSTN**—Use device name of current work station as list name.
- ***SELECT**—Display a list of existing job lists, one of which may be selected. The name of the job list can be entered.
- ***MDLLIB/**—The job list library is to be the first model library found in the library list.
- ***GENLIB/**—Use the generation library specified in the first model found in the library list.
- ***SRCLIB/**—Use the source library specified on the model profile of the current user.
- ***LIBL/**—The job list library is the first model found in the library list.
- ***CURLIB/**—The model library is found in the current library for the current job.

The job list library can be entered.

ENTRY

This parameter provides the user with multiple entry points to the specified model. Values for this parameter are described in the following:

- ***EDTDBREL**—(default) The first panel to be accessed is the Edit Database Relations panel.
- ***EDTMDLLST**—The mode of entry to the model is via the Edit Model List panel. If this value is specified, the MDLLST parameter specifies the model list that is edited.
- ***SERVICES**—The services menu is the first panel accessed.
- ***NONE**—This option can be used to establish a model environment but without any particular entry to the model. In this case the model environment is started and the developer is presented with the Command Entry panel. Numerous commands require the model environment to be active and will check to ensure that it is active when invoked. These commands will adopt an already active environment. Thus, if a series of commands are run, it will be more efficient to use this option before executing such commands. Another advantage of this option is that the lock applied to the model will be established for the entire session, preventing interference by another developer.

SSNLST

This parameter specifies the session list to use while editing the model. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The session list is retrieved from the model profile details for the current user.
- ***USER**—The session list has the same name as the current user.
- ***SELECT**—An interactive display is used to select a model object list to be used as the session list.

The name of the list can be entered.

For more information on the purpose of SSNLST, refer to the Change Model Profile command (YCHGMDLPRF).

OPNACC

This parameter enables the current user to override to *NO (if authorized) the Open Access model value. The intention with this parameter is to provide a *DSNR with the opportunity to gain exclusive access to the data model. Values for this parameter are described in the following:

- ***MDLVAL**—(default) Access to the model is controlled by the current model value for Open Access.
- ***NO**—This value can be used to set the Open Access model value to exclude any concurrent *DSNR activity in the model and to exclude users of any other class.

For more information on Open Access, see the Change Model Value command (YCHGMDLVAL) in this chapter and the "Creating and Managing Your Model" chapter in the CA 2E *Administrator Guide*.

MDLLST

The qualified name of the model object list that is edited. Values for this parameter are described in the following:

- ***USER**—The list to be edited has the same name as the current user.
- ***ALLOBJ**—The all objects list is to be edited.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.

Notes

- The model can either be set up to allow concurrent *DSNRs and *PGMRs/*USERs into the model by setting the model value YOPNACC to *YES, or the model value can be left as *NO where either a single *DNSR or multiple *PGMRs/*USERs can be in the model concurrently.
- To add and remove permanent locks, or to change the Open Access (YOPNACC) model value, you must have all rights to the data model. (Designer with locks capability).
- To edit a model as a user of type *DSNR, you must have at least all rights except for existence to the model. Generally the most convenient way to arrange this is to grant all rights to all the objects in the model library (the default), and then to control access to the model by granting or revoking rights to use the data area YMDLLIBRFA in the model library. The YEDTMDL command checks the user's authority to this data area before allowing entry to the model.
- For example, to revoke all rights to user profile IVAN to edit or view a model MYMDL:

  ```
  RVKOBJAUT OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA) USER (IVAN) AUT(*ALL)
  ```

- Or to grant user profile IVAN rights to an edit model MYMDL as designer with lock capability:

  ```
  GRTOBJAUT OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA) USER(IVAN) AUT(*ALL)
  ```

- To override the model value YOPNACC (represented in the command by OPNACC) by specifying *NO, you must have all rights to the data area YOPNACCRFA. Access rights to YOPNACCRFA can be assigned or revoked in the same way as above. Should you wish to change the YOPNACC value temporarily just for the duration of the session, you must synchronize the model on exit.

Examples

To edit a model as a designer:

```
YEDTMDL
```

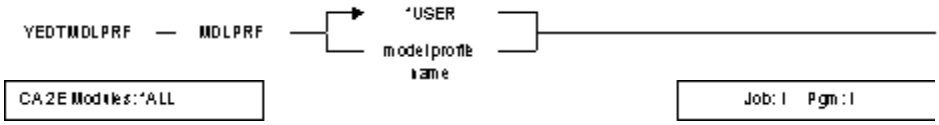
To edit a model as a programmer, using a list named after the current device name:

```
YEDTMDL USER(*PGMR) JOBLST(*WORKSTN)
```

YEDTMDLPRF (Edit Model Profile) Command

A model profile is created for each user in each data model. It records information used by several commands, for control and defaulting purposes. This command allows the user to edit the data stored for a specified user.

Required



Parameters

The following are parameters for the YEDTMDLPRF command.

MDLPRF

The name of the model profile that is edited. Values for this parameter are described in the following:

- ***USER**—(default) Special value indicating that the model profile of the current user is edited.
- **model profile name**—The name of the model profile to be edited can be entered.

MDLLSTA

The qualified name of the model object list that is the first operand list involved in the operation. Values for this parameter are described in the following:

- **list name**—(default) The first model object list name must be entered.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are used as input to the command.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- ***SELECT**—Special value meaning that the model object list is selected using an interactive display function.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the first list.
- **library name**—The model library name for the first list.

Notes

- The first model library in the current library list is used to find the specified model profile.
- If the model profile does not exist, it is created. A special profile, YSYS, is used to provide default values for a new model profile.

Example

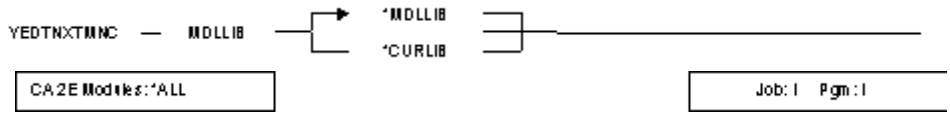
To edit the model profile Y2PGMR enter the following:

```
YEDTMDLPRF MDLPRF( Y2PGMR )
```

YEDTNXTMNC (Edit Next Mnemonics) Command

This command calls an interactive program to edit the next type mnemonics to be used when all autonames are used up for the given types. It requires an exclusive lock on the model. If multiple models use the same naming library, this command will not lock those other models, so it is up to the designer to ensure that no one is using the other models.

Required



Parameters

The following are parameters for the YEDTNXTMNC command.

MDLLIB

Name of library containing the YALCVNMRFP file associated with the model, usually the design model library. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Use the first model library found in the library list.
- ***CURLIB**—Use the current library for invoking the job.

Example

To call an interactive program to edit the type mnemonics for automatic naming in the current model library:

```
YEDTNXTMNC
```

YENDTRGSVR (End Trigger Server) Command

This command allows the user to end a currently running trigger server which is monitoring a Trigger Data Queue in a specified library.

See the Start Trigger Server (YSTRTRGSVR) command for more details on Trigger Data Queues.

Parameters

The following are parameters for the YENDTRGSVR command:

TRGLIB

Specifies the name of the library containing the Trigger Data Queue.

The values are:

***TRGLIB**

The library containing the Trigger Server program is used.

library-name

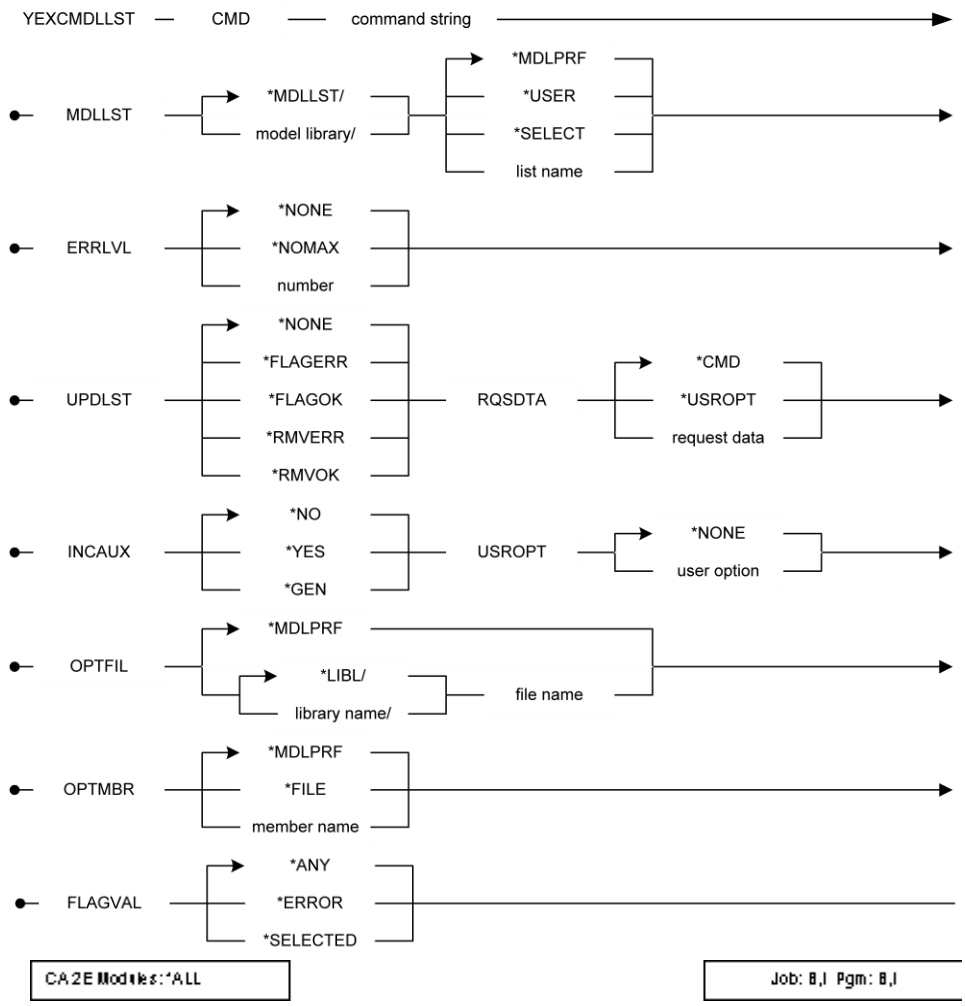
Specify the library name which contains the Trigger Data Queue to be monitored.

YEXCMDLLST (Execute a Model Object List) Command

This command allows a user to perform some action with each of the entries contained in a model object list.

Substitution variables are provided to include object related data from a particular list entry in the command to be executed. For a list of supported values, see details relating to the CMD parameter.

Required



Parameters

The following are parameters for the YEXCMDLLST command.

CMD

The command string containing the details of the command that is executed for each entry in the model object list.

Note: For convenience, this parameter is a command string (*CMDSTR). This allows the command that is to be executed to be prompted and syntax- checked prior to execution. In some instances this prevents substitution variables from being defined for a given parameter, such as making it impossible to specify the object type (&YT) in a parameter that has a fixed set of allowed values. When this happens you must use the RQSDTA parameter to specify the command to be executed.

The following substitution variables are supported:

- **&YN**—Object name, 25 character string.
- **&Y@**—Object surrogate, 7 decimal.
- **&YT**—Object type, 3 character (FIL, FUN, ACP, etc).
- **&YA**—Object attribute, 3 character (RPG, INF, STS, RTV, etc).
- **&YO**—Object owner name, 25 character string.
- **&YW**—Object owner surrogate, 7 decimal.
- **&YY**—Object owner type, 3 character string.
- **&YR**—Object group surrogate, 7 decimal.
- **&YM**—Model list name, 10 character string.
- **&YI**—Object implementation name, 10 character string.
- **&YP**—Object promotion type, 3 character (ADD, CHG, GEN, etc).
- **&YS**—Object SEU type, 10 character (RPG, DSPF, PRTF, etc).
- **&YU**—User name, 10 character.
- **&YL**—Model library name, 10 character.
- **&YC**—User option text. The text is retrieved from the specified user options member.
- **&YG**—Object change type, 8 character (*PRIVATE, *PUBLIC, etc).
- **&YF**—Function type, 10 character (EXCEXTFUN, EDTRCD, etc).
- **&YZ**—Assimilated file, a 1-character string set to Y or N. Indicates whether the FIL or owning FIL object is an assimilated file from an external database.
- **&Y6**—DBFGEN value (LGL ACP only), 1 character string (D or S)
- **&Y7**—DBFACC value (LGL ACP only), 1 character string (G or T)
- **&Y8**—MNTSTS value (LGL ACP only), 1 character string (I, D or R)
- **&Y9**—UNQKEY value (LGL ACP only), 1 character string (U, F, L, C or blank)

MDLLST

The qualified name of the model object list that is executed. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The model object list name must be entered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the list.
- **library name**—The model library name for the list can be entered.

ERRLVL

The number of errors tolerated by the command can be specified. Values for this parameter are described in the following:

- ***NONE**—(default) No errors allowed. The command will stop at the first error encountered.
- ***NOMAX**—There is no limit to the number of errors.
- **number of errors**—The number of errors can be specified.

UPDLST

List update option. Values for this parameter are described in the following:

- ***NONE**—(default) The list entries are not updated.
- ***FLAGERR**—If an error occurs during processing of a list entry, the entry is flagged. The value to use is specified by the OUTFLAGVAL parameter. This value may be the subject of a later filtering operation.
- ***FLAGOK**—If an error does not occur during processing of a list entry, the entry is flagged. The value to use is specified by the OUTFLAGVAL parameter. This value may be the subject of a later filtering operation.
- ***RMVERR**—If an error occurs during processing of a list entry, the entry is removed from the list.
- ***RMVOK**—If the processing of a list entry completes without error, the entry is removed from the list.

RQSDTA

The request data command string containing the details of the command that is executed for each entry in the model object list. For a list of supported substitution variables, see details relating to the CMD parameter. Values for this parameter are described in the following:

- ***CMD**—(default) The command string parameter contains the request to be executed.
- ***USROPT**—The request data command string is retrieved from the user options file/member specified using the user option parameters.

INCAUX

This parameter lets you choose whether or not to include object auxiliaries, such as the display and help members for an interactive function, in the processing. Values for this parameter are described in the following:

- ***NO**—(default) Auxiliaries are not included in processing.
- ***YES**—Auxiliaries are included in processing.
- ***GEN**—Auxiliaries are only included in processing if they are currently generatable as part of the normal generation of the access path or function.

Note: For functions that have help text, which auxiliaries are included in processing when INCAUX is set to *GEN is affected by the setting of the YGENHLP model value (or the corresponding GENHLP function option). The YGENHLP model value specifies whether the function only (*NO), the help text only (*ONLY), or both (*YES) are to be generated.

- If YGENHLP IS *NO, the help auxiliary is excluded from YEXCMDLLST processing.
- If YGENHLP is *ONLY, only the help auxiliary is included in processing.
- If YGENHLP is *YES, all generatable auxiliaries are included. YGENHLP has no effect on functions that do not have help text.

USROPT

Two character user-defined option identifying a record in the specified user-defined options file. This option contains user option text for substitution in the RQSDTA command string. The value for this parameter is as follows:

- ***NONE**—(default) No user-defined option text is used.

OPTFIL

Qualified name of the file containing the user-defined option text. The value for this parameter is as follows:

- ***MDLPRF**—(default) Single value meaning that the value for the user option file is retrieved from the model profile details for the current user.

OPTMBR

Name of the member containing the user-defined option text. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Single value meaning that the value for the user option file is retrieved from the model profile details for the current user.
- ***FILE**—The member has the same name as the file.

FLAGVAL

This parameter allows list entries to be selected for execution. Values for this parameter are described in the following:

- ***ANY**—(default) Execute all list entries.
- ***ERROR**—Only entries flagged in *ERROR are executed.
- ***SELECTED**—Only *SELECTED entries are executed.

Notes

- The MDLLST must exist prior to running the command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- If *USROPT is specified for parameter RQSDTA, then the USROPT parameter cannot be *NONE.
- Substitution variables are available to insert details relating to a particular model object list entry into the command string. The substitution prefix & or @ can be used and an alternative character can be defined in the Synon/1E data area YPEXCHA (note that the Synon/1E Edit Data Area command (YEDTDTAARA) may be used to change this value).
- Note that object owner name will be returned as *Arrays for Array objects, *Messages for Message objects, the object name for model file objects and *NONE for objects that do not have an owner, such as FLD objects.
- When you to try to execute a model list using YCRTJOBLE command, and the model list has one or more *DDL-based access paths in it, and the access path has either of the four DDL limitations, the entry is not added to the job list. The access path source is not generated.
- The current implementation of the DDL generation mode is not valid for the following cases:
  - Access paths that have virtual fields
  - SPN access path
  - QRY access path
  - Multi-member files

  Workaround for Virtual Fields, SPN, and QRY Access Paths: If the earlier generation mode is *DDS, revert to it and regenerate the access path. You need not regenerate the functions that use this access path. If you want to have an SQL type database, regenerate the access path using *SQL generation mode. The functions using this access path must be regenerated.

  Workaround for Multi-Member Files: If you want to have more than one member for the access paths, revert to *DDS generation mode.

Note: If you want to change an access path, which is previously defined as *DDS with a MAXMBR compiler override, to *DDL, you must revert to *DDS generation mode and must remove the compiler override, and then change back to *DDL generation mode.

Example

To print each function by executing model object list IM0033 from the model contained in the current library list with the Document Model Functions command (YDOCMDLFUN):

```
YEXCMDLLST CMD( YDOCMDLFUN + MDLFILE(&YO) MDLFUN(&YN) ) MDLLST + (*MDLLIB/IM0033 )
```

YEXCOVR (Execute with preprocessor) Command

The YEXCOVR command executes a user-specified command that invokes the CA 2E Toolkit compile preprocessor program. The preprocessor processes preprocessor directives held in a source member. The source lines are identified with Z*, Y*, T*, X* or P*.

There are two parameters you can use to specify the request to be executed: CMD and RQSDTA. Each parameter allows the request to be specified, so they are mutually exclusive. Alternatively, a source member that contains preprocessor directives can be specified in the SRCFILE and SRCMBR parameters.

Parameters

The following are parameters for the YEXCOVR command.

CMD

The command to be executed. This parameter is a *CMDSTR, allowing the command to be prompted and validated before execution.

RQSDTA

This parameter allows the command to be entered as a string. The details of the request will not be validated until run time.

***CMD**

Use the command specified in the CMD parameter.This is the default.

***SRCMBR**

Processes the preprocessor directives in the source member specified in the SRCFILE and SRCMBR parameters.

request-data

Processes the specified data.

SRCFILE

Specifies the name of the source file that contains the source member to be processed and the library where the source file is stored.

Note: This parameter is ignored unless RQSDTA(*SRCMBR) is specified.

Possible library values are:

***LIBL**

The system searches the library list to find the library where the source file is stored. This is the default.

***CURLIB**

The current library is used to find the source file. If you have not specified a current library, QGPL is used.

library-name

Enter the name of the library where the source file is stored.

SRCMBR

Specifies the name of the member in the source file that contains the preprocessor directives to be processed.

Note: This parameter is ignored unless RQSDTA(*SRCMBR) is specified.

Usage

The YEXCOVR command is used to ensure that any Toolkit preprocessor directives (Z*, Y*, T*, X* and P* source lines) stored in a source member are processed when the source member is compiled. All source members compiled from within a CA 2E model automatically invoke the preprocessor, but unless you have changed your system to automatically invoke the preprocessor when compiling source members from PDM, it will not be invoked.

To set up your system to use the preprocessor, open the CA 2E Toolkit main menu by running the following command:

```
YG0 *Y1
```

Take option 6 (Installation and Authorization) to display the CMDINZ menu, and then take option 7 (CA 2E Toolkit Compile Pre-processor). This will take you through the necessary steps to set up your system to automatically use the preprocessor.

Examples

- To compile *PGM source member RPGPGM1 in source file QRPGLESRC in library DEVLIB:

  ```
  YEXCOVR CMD(CRTBNDRPG PGM(RPGPGM1) SRCFILE(DEVLIB/QRPGLESRC))
  ```

  Note: If you configured your system to use the compile preprocessor automatically, you can use option 14 against the source member from the WRKMBRPDM screen.

- To execute preprocessor directives stored in source member SRVPGM1 in source file QSRVSRC in library DEVLIB:

  ```
  YEXCOVR RQSDTA(*SRCMBR) SRCFILE(DEVLIB/QSRVSRC) SRCMBR(SRVPGM1)
  ```

YEXCSQL (Execute SQL Statements) Command

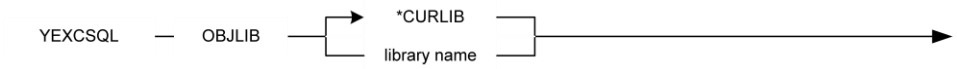
The Execute SQL statements (YEXCSQL) command prepares and executes SQL/400 statements that are contained in a source file member.

Earlier, YEXCSQL worked only for the target object library being an SQL collection. But now, the target object library can be either an SQL collection or a normal library (non-SQL collection).

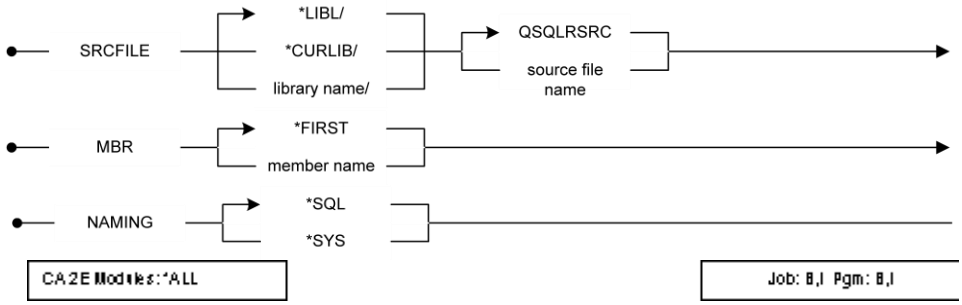
If an SQL collection is specified as the target object library, the tables, indexes, and views are created in the specified SQL collection, as before. If a normal library is specified as the target object library, the tables, views and indexes are created in that library but are not journaled. While the tables, views, and indexes are being created, you may receive warning messages that the library is a non-SQL collection and that the tables/views/indexes are not journaled. These messages can be ignored for generation purposes.

Warning: If a non-SQL Collection/Library is used as a target object library, then the tables/views/indexes created in it are not journaled automatically.

Required



Optional



Parameters

The following are parameters for the YEXCSQL command.

OBJLIB

Target library in which to place the SQL statements. This can either be an SQL Collection or a Library. Values for this parameter are described in the following:

- ***CURLIB**—(default) Use current library for invoking job.
- **library name**—Specific library name.

SRCFILE

Qualified name of file from which SQL statements are processed. Values for this parameter are described in the following:

- ***LIBL**—(default) Libraries named in the job library list to be processed.
- ***CURLIB**—Current library for the job to be processed.
- **library name**—Specific library name.
- **QSQLRSRC**—(default) SQL statements are in the QSQLRSRC source file.
- **source file name**—Specific name of source file containing the SQL statements.

MBR

Member name of source file to be executed. Values for this parameter are described in the following:

- ***FIRST**—(default) Use first member in source file.
- **member name**—Name of specific member in source file.

NAMING

Naming convention used for naming the objects in SQL statements. Values for this parameter are described in the following:

- ***SQL**—(default) Use SQL naming convention (collection-name.table-name).
- ***SYS**—Use system naming convention (library-name/file-name).

Example

To apply SQL/400 statements, stored in file QSQLSRC, to SQL/400 collection, MYLIB:

```
YEXCSQL OBJLIB(MYLIB) SRCFILE(QSQLSRC)
```

YEXCWSIPDD (Execute WSIPDD file) Command

Executes a Web Service Instance Portable Deployment Data file (WSIPDD), to deploy web service instances. The WSIPDD file should have been populated by the CA 2E YPOPWSIPDD command. Typically the WSIPDD file is then moved to a different machine to be executed by the YEXCWSIPDD command.

Notes:

- Portable deployment does not require CA 2E or 1E to exist on the remote machine on which the YEXCWSIPDD command is running. However, the YEXCWSIPDD command does require certain application objects to exist on the machine on which the command is running. These objects can be created in a target library using the YDUPAPPOBJ command parameter, *WS argument. The YEXCWSIPDD command takes a WSIPDD file as an input.
- To run this command the YCA/CAWS/UserData and YCA/CAWS/ProdData/YQSHLOG folders need to exist in the IFS. Take extra care with this in the case where the command is being used on a remote machine that does not have 2E installed.

  For more information on how to restore the YCA structure, see the section "Web Services Support" in the *Installation Guide.*
- The input WSIPDD file is always called YWSIPDDRFP, but the location is specified on the WSIPDDLIB parameter. For each record in the WSIPDD file with an ACTION flag of 'I' a web service instance will be deployed by the YCRTWS command. The Target parameters on the YEXCWSIPDD command allow the WSIPDD web service instance data to be overridden when deploying.
- If a web service instance is successfully deployed as a result of the YEXCWSIPDD command instance record's Action flag is updated to BLANK. The YINZWSIPDD command can be used to reset the WSIPDD Action flag.

Parameters

The following are parameters for the YEXCWSIPDD command.

WSIPDDLIB

The name of the library in which the input WSIPDD file resides. Values for this parameter are described in the following:

- ***GENLIB**—Special value representing the model's generation library.
- **name**—Specify the library containing the YWSIPDDRFP/00L files.

Note: The WSIPDD file will typically have been populated by the YPOPWSIPDD command.

TMACHINE

Indicates whether WSIPDD web service instance information should be overridden when deploying. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- ***CURRENT**—The WSIPDD web service instance Machine name is overridden to *CURRENT when deploying.
- **name**—The WSIPDD web service instance Machine name is overridden to the named value, when deploying.

TSERVER

Indicates whether WSIPDD web service instance information should be overridden when deploying. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- **character value**—The WSIPDD web service instance Server name is overridden to the named value, when deploying.

Note: Server name must match (case sensitive) with the name of the actual application server.

TSERVICE

Indicates whether WSIPDD web service instance information should be overridden when deploying.

- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- **name**—The WSIPDD web service instance Service name is overridden to the named value, when deploying.

TGTOBJLIB

Indicates whether WSIPDD web service instance information should be overridden when deploying. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- **name**—The WSIPDD web service instance target object Library name is overridden to the named value, when deploying.

TUSRPRF

Indicates whether WSIPDD web service instance information should be overridden when deploying. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- **name**—The WSIPDD web service instance target user Profile name is overridden to the named value, when deploying.

TRTLIBL

Indicates whether WSIPDD web service instance information should be overridden when deploying. Values for this parameter are described in the following:

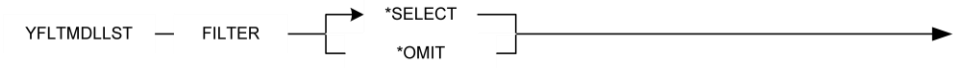
- ***INSTANCE**—No overriding is applied. WSIPDD web service instance information is used to deploy a web service instance.
- **name**—The WSIPDD web service instance target runtime library list is overridden to the name value, when deploying.

YFLTMDLLST (Filter a Model Object List) Command

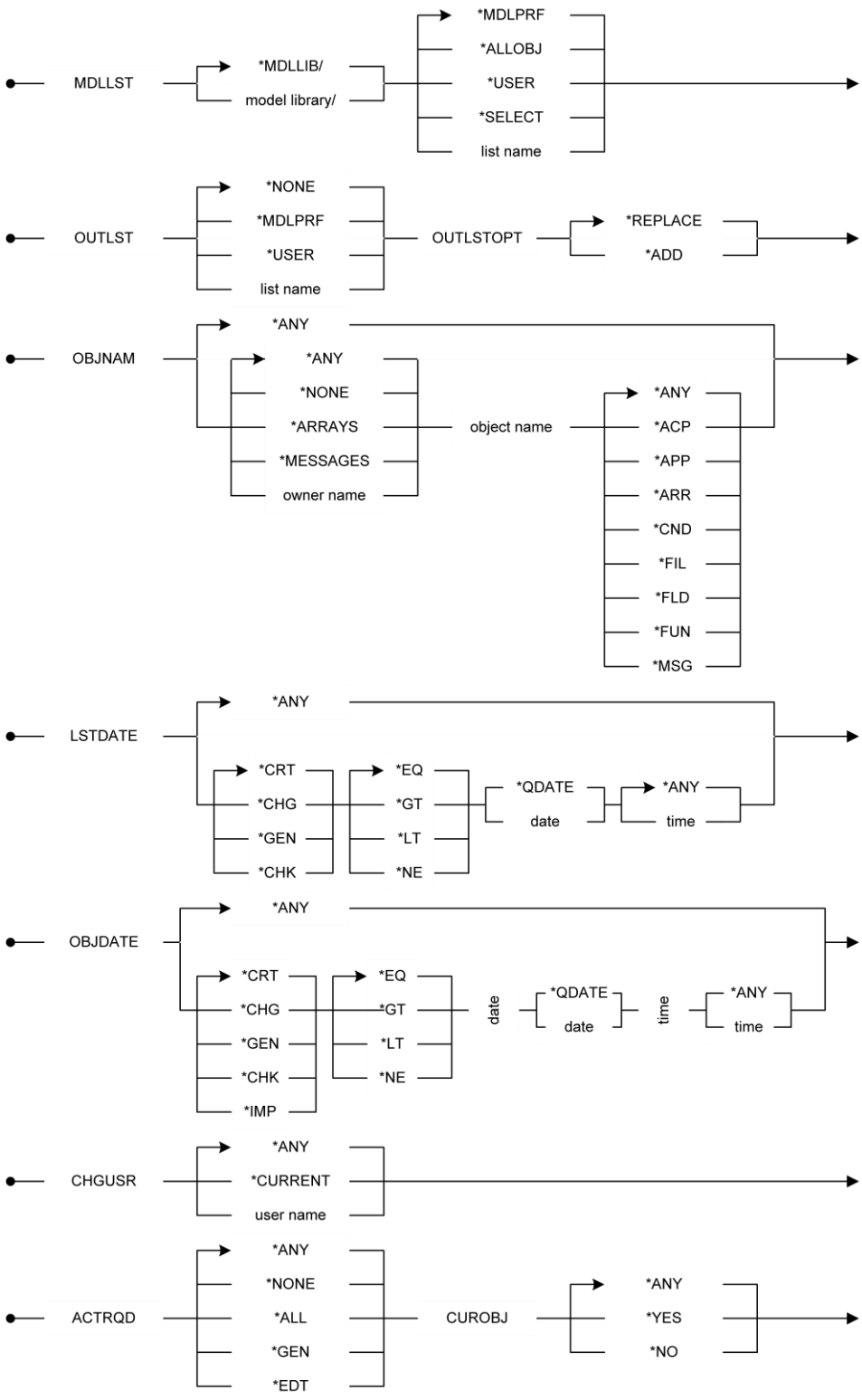
This command allows a user to remove unwanted entries from a model object list.

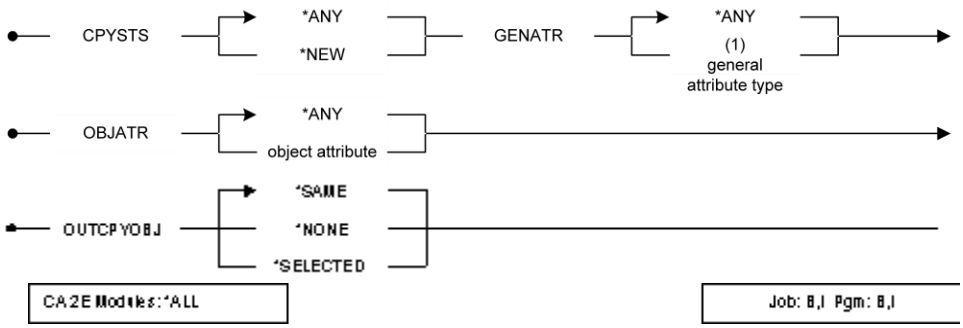
An outlist capability allows users to filter entries to an alternative list leaving the input list unchanged by the filtering process.

Required



Optional


MDLLST
*MDLLIB/
model library/
*MDLPRF
*ALLOBJ
*USER
*SELECT
list name
OUTLST
*NONE
*MDLPRF
*USER
list name
OUTLSTOPT
*REPLACE
*ADD
OBJNAM
*ANY
*ANY
*NONE
*ARRAYS
*MESSAGES
owner name
object name
*ANY
*ACP
*APP
*ARR
*CND
*FIL
*FLD
*FUN
*MSG
LSTDATE
*ANY
*CRT
*CHG
*GEN
*CHK
*EQ
*GT
*LT
*NE
*QDATE
date
*ANY
time
OBJDATE
*ANY
*CRT
*CHG
*GEN
*CHK
*IMP
*EQ
*GT
*LT
*NE
date
*QDATE
date
time
*ANY
time
CHGUSR
*ANY
*CURRENT
user name
ACTRQD
*ANY
*NONE
*ALL
*GEN
*EDT
CUROBJ
*ANY
*YES
*NO



Parameters

The following are parameters for the YFLTMDLLST command.

DLLST

The qualified name of the model object list that is filtered. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name for the target of the command.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are included in the command. If this value is specified for MDLLST, then an OUTLST must also be specified.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.
- **list name**—The name of the list to be filtered.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used.
- **library name**—The name of the model library.

OUTLST

The name of the target model object list for entries that satisfy the filter. Values for this parameter are described in the following:

- ***NONE**—(default) Special value meaning that no outlist processing is performed. In this case, all changes are made to the input list.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The outlist value is retrieved.
- ***USER**—Special value meaning that the user profile name of the current user is used as the list name.
- **list name**—The name of the target list.

OBJNAM

The object name details on which to filter. This parameter consists of up to five sets of three elements which together identify the model objects. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering by model object name.
- ***ALL**—All model object owners are included.
- **owner name**—Name mask of the file to which model objects to be included must belong. That is, the condition is satisfied if a given object is owned by the specified file.
- ***ARRAYS**—Special value for the product internal file *ARRAYS.
- ***MESSAGES**—Special value for the product internal file *MESSAGES.
- **object name**—Name mask of the object that is satisfied for a given object to be included.
- ***ALL**—(default) All objects are added.
- ***ACP**—Objects of type access path.
- ***APP**—Objects of type application area.
- ***ARR**—Objects of type array.
- ***CND**—Objects of type condition.
- ***FIL**—Objects of type file.
- ***FLD**—Objects of type field.
- ***FUN**—Objects of type function.
- ***MSG**—Objects of type message.

LSTDATE

The date stored on the model object list entry on which to filter. The value for this parameter is as follows:

- ***ANY**—(default) No filtering on list entry date.

Otherwise, LSTDATE is a list parameter made up of the following elements:

- Date type on which to filter:
  - ***CHG**—(default) Filter on last change date of object.
  - ***CRT**—Filter on create date of object.
  - ***GEN**—For generatable objects, filter on last successful generation date of object.
  - ***CHK**—Filter on the check out date of object.
- date operator:
  - ***EQ**—(default) Equal to.
  - ***NE**—Not equal to.
  - ***GT**—Greater than.
  - ***LT**—Less than.
- date (Entered in system date format):
  - ***QDATE**—(default) Use current system date.
- time (Entered in *hhmmss* format):
  - ***ANY**—(default) Any time will satisfy the condition.

OBJDATE

The date stored on the model object record on which to filter. The value for this parameter is as follows:

- ***ANY**—(default) No filtering on list entry date.

Otherwise, OBJDATE is a list parameter made up of the following elements:

- Date type on which to filter:
  - ***CHG**—(default) Filter on last change date of object.
  - ***CRT**—Filter on create date of object.
  - ***GEN**—For generatable objects, filter on last successful generation date of object.
  - ***CHK**—Filter on the check out date of object.
  - ***IMP**—Filter on the import date of the object.
- Date operator:
  - ***EQ**—(default) Equal to.
  - ***NE**—Not equal to.
  - ***GT**—Greater than.
  - ***LT**—Less than.
- Date (Entered in system date format):
  - ***QDATE**—(default) Use current system date.
- Time (Entered in *hhmmss* format):
  - ***ANY**—(default) Any time will satisfy the condition.

CHGUSR

The name of the user who made the change to model objects on which to filter. If used, model objects in the list are filtered according to the current value of the change user field on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on change user.
- ***CURRENT**—The name of the current user is used to filter model objects.
- **user name**—The user name on which to filter.

ACTRQD

The model object action required flag on which to filter. If used, model objects in the list are filtered according to the current status of the action required flag on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on action required flag.
- ***NONE**—Model objects with no action required flag set.
- ***ALL**—Model objects with action required flag set.
- ***GEN**—Model objects with status *GEN satisfy the condition.
- ***EDT**—Model objects with status *EDT satisfy the condition.

SYSOBJ

Indicates whether or not the object is a system object to be included in the filtering process. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on system object status.
- ***YES**—Only system objects satisfy the condition.
- ***NO**—Only non-system objects satisfy the condition.

CHGTYP

The change type value assigned to a model object on which to filter. If used, model objects in the list are filtered according to the status of the change type flag on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on change type flag.
- ***PUBLIC**—Model objects with status *PUB satisfy the condition.
- ***PRIVATE**—Model objects with status *PRV satisfy the condition.

IPCPRC

Impact processed indicator shows whether the impact of a change to the object has been applied to the using objects of the changed object. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on impact processed indicator.
- ***YES**—Model objects with impact processed *YES will satisfy the condition.
- ***NO**—Model objects with impact processed *NO will satisfy the condition.

PRMTYP

The promotion type value assigned to a model object list entry on which to filter. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on promotion type flag.
- ***ADD**—Model object list entries with status *ADD satisfy the condition.
- ***CHG**—Model object list entries with status *CHG satisfy the condition.
- ***GEN**—Model object list entries with status *GEN satisfy the condition.

VSNTYP

The model object version type flag on which to filter. If used, model objects in the list are filtered according to the current status of the version type flag on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on version type flag.
- ***DEV**—Model objects with status *DEV satisfy the condition.
- ***PRD**—Model objects with status *PRD satisfy the condition.
- ***ARC**—Model objects with status *ARC satisfy the condition.

VSNSNC

The model object version synchronized flag on which to filter. If used, model objects in the list are filtered according to the current status of the version synchronized flag on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on version synchronized flag.
- ***YES**—Model objects with status *YES satisfy the condition.
- ***NO**—Model objects with status *NO satisfy the condition.

CHKUSR

The user who checked out the list entry on which to filter. If used, model objects in the list are filtered according to the check out user value on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on check out user.
- ***CURRENT**—The current user name is used.
- **user name**—The user name on which to filter.

CHKLST

The list name on which a given list entry is currently checked out. If used, model objects in the list are filtered according to the check out list value on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on check out list.
- ***CURRENT**—The current user name is used for list name filtering.
- ***NONE**—Only those objects that have no value specified for check out list satisfy the condition.
- **list name**—The list name on which to filter.

CHKSTS

The check out status on which to filter. If used, model objects in the list are filtered according to the check out status value on the model object record corresponding with each list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on check out status.
- **status value**—The value on which to filter.

CPYOBJ

The status of the copy object indicator on the list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering.
- ***SELECTED**—Selected entries satisfy the condition. Selected means explicitly selected for the purposes of the Copy Model Objects command (YCPYMDLOBJ).
- ***NONE**—Entries that are not selected satisfy the condition.

CPYRQD

The status of the copy required indicator on the list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering.
- ***SELECTED**—Selected entries satisfy the condition. Selected means implicitly selected for the purposes of the Copy Model Objects command (YCPYMDLOBJ).
- ***NONE**—Entries that are not selected satisfy the condition.

CPYSTS

The status of the copy status indicator on the list entry. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering.
- ***NEW**—Entries with a value of *NEW satisfy the condition.

GENATR

This parameter allows up to five different general attributes of the object to be used in the filtering process. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering.
- ***OBJDLT**—Only entries that refer to deleted objects satisfy the condition.

Note: When filtering for deleted objects it is not possible to add entries for deleted objects to an output list because the corresponding object does not exist. To process these entries, copy the input list to the output list first, then filter *SELECT the deleted objects.

- ***GENFUN**—Only generatable functions satisfy the condition.
- ***GENOBJ**—Only generatable objects satisfy the condition.
- ***DBFFUN**—Only database functions satisfy the condition.
- ***DEVFUN**—Only device based functions satisfy the condition.
- ***DSPFUN**—Only display functions satisfy the condition.
- ***EDTFUN**—Only edit functions satisfy the condition.
- ***EXTFUN**—Only external functions satisfy the condition.
- ***INTFUN**—Only internal functions satisfy the condition.
- ***PRTFUN**—Only print functions satisfy the condition.
- ***CHGOBJ**—Only change object database functions satisfy the condition.
- ***CRTOBJ**—Only create object database functions satisfy the condition.
- ***DLTOBJ**—Only delete object database functions satisfy the condition.
- ***DSPFIL**—Only display file functions satisfy the condition.
- ***DSPREC**—Only display record functions satisfy the condition.
- ***DSPTRN**—Only display transaction functions satisfy the condition.
- ***EDTFIL**—Only edit file functions satisfy the condition.
- ***EDTRCD**—Only edit record functions satisfy the condition.
- ***EDTTRN**—Only edit transaction functions satisfy the condition.
- ***EXCEXT**—Only execute external functions satisfy the condition.
- ***EXCINT**—Only execute internal functions satisfy the condition.
- ***PMTRCD**—Only prompt record functions satisfy the condition.
- ***PRTFIL**—Only print file functions satisfy the condition.
- ***PRTOBJ**—Only print object functions satisfy the condition.
- ***SELRCD**—Only select record functions satisfy the condition.
- ***USRPGM**—Only user program functions satisfy the condition.

- ***USRSRC**—Only user source functions satisfy the condition.

OBJATR

This parameter allows model objects to be filtered by up to five different model object attributes. Values for this parameter are described in the following:

- ***ANY**—(default) No filtering on object attribute value is performed.
- **object attribute**—A list of three character object attributes may be entered.

For example, if physical access paths were required, specifying FILTER(*SELECT) OBJATR(PHY) will select any such entries in the input list.

OUTFLAGVAL

This parameter specifies the initial value to be placed in the object selected flag associated with each list entry. Values for this parameter are described in the following:

- ***SAME**—(default) No flag value is used. New entries are written with the flag indicating that the entry is not selected. No change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing list entries are flagged as selected. This flag can be used by other list commands when selecting list entries.

OUTCPYOBJ

This parameter specifies the initial value to be placed in the copy object flag associated with each list entry. This flag is used by the Copy Model Object command (YCPYMDLOBJ) when selecting objects to copy to a target model. Values for this parameter are described in the following:

- ***SAME**—(default) No flag value is used. New entries are written with the flag indicating that the entry is not selected. No change to the selection status of existing entries.
- ***NONE**—New and existing entries are flagged as not selected.
- ***SELECTED**—New and existing list entries are flagged as selected.

Notes

- The input model object list must already exist prior to running this command.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- If *ANY is used for a parameter, that parameter is not used for filtering.
- The various parameters are ANDed together. Thus, the following request would select all functions from the input list changed today:

```
YFLTMDLLST OBJNAM( *ALL *ALL *FUN ) +   DATE(*CHG *EQ *QDATE *ANY )
```

- Where a list of values may be specified for a parameter the elements are ORed together. Thus, the following would select access paths or functions that have had an action required since they were last generated:

```
YFLTMDLLST OBJNAM((*ALL *ALL *ACP) (*ALL + *ALL *FUN)) ACTRQD(*ALL)
```

- When *ALLOBJ is specified for MDLLST, meaning that all current model objects are examined, then the OUTLST parameter must specify a target model object list.
- When filtering on LSTDATE, list entries are filtered according to the value recorded on the list entry. Note that this is historic data relating to the time that the entry was written. Use the Check Model Object list command (YCHKMDLLST) to refresh list entries.
- The OUTLSTOPT parameter is ignored if *NONE is specified for OUTLST or if the target model object list does not already exist.
- It should be noted that if an OUTLST is specified, there will be no changes to the input list.
- Where a name mask is supported, users may specify wild card characters to filter object names. Upper and lower case differences are ignored in these fields. **?** may be used as a wild card character in any position in the string and means match on any character. ***** anywhere within the mask indicates a floating scan. ***** at the end of the mask indicates a generic name. For example:
  - ***INV001***—Would select all objects with INV001 anywhere in the name.
  - ***INV001**—Would select all objects ending with INV001.
  - ***INV???**—Would select all objects ending with INV at the 4th, 5th and 6th positions from the end.
  - **INV*UPD***—Would select objects starting with INV and containing UPD.

Examples

To filter model object list WRKOBJS to out list WRKLST so that it contains all objects requiring a visit due to updates made to component objects:

```
YFLTMDLLST ( *EDT ) MDLLST +
 (*MDLLIB/WRKOBJS ) OUTLST +
 (*MDLLIB/WRKLST) OUTLSTOPT +
 ( *REPLACE ) ACTRQD
```

To filter model object list FUNLST so that it contains only objects relating to Costing (note that a reliable naming convention is required for this type of filtering):

```
YFLTMDLLST ( '*COST*' ) MDLLST + (*MDLLIB/FUNLST) OBJNAM
```

To filter model object list ALLOBJLST so that it contains only non-current objects:

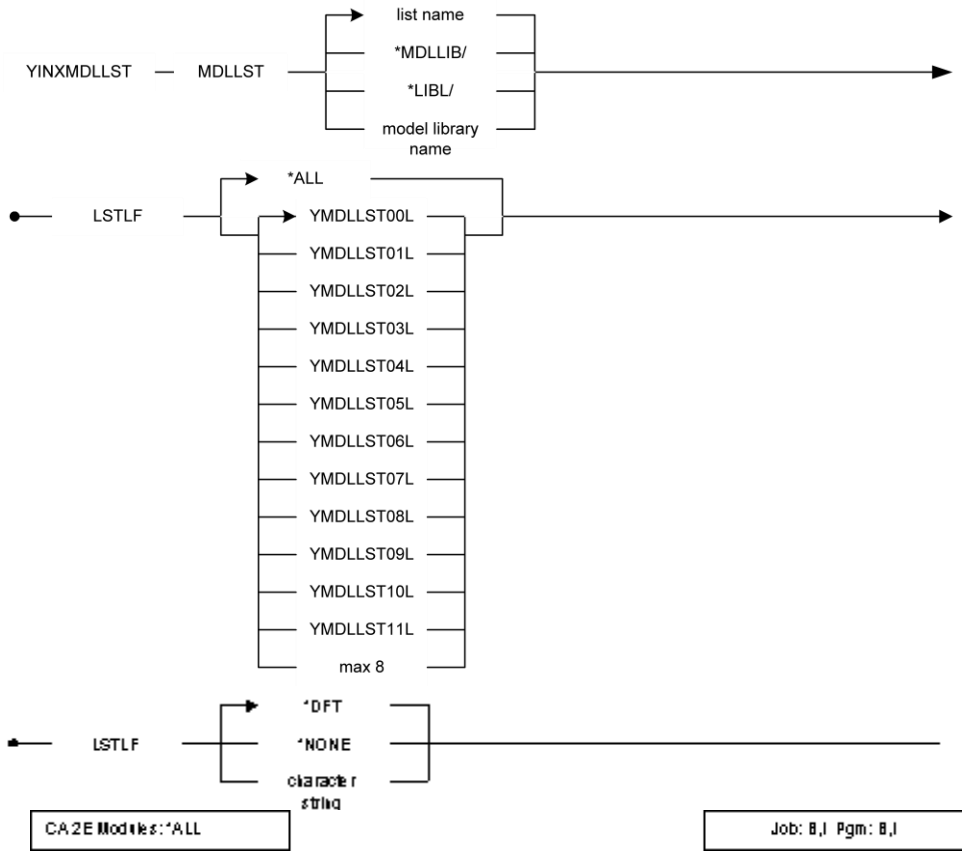
```
YFLTMDLLST FILTER( *OMIT ) ( *YES ) + MDLLST(ALLOBJLST ) CUROBJ
```

YINXMDLLST (Index a Model List) Command

This command checks for the existence of a model object list and its accompanying logical file members. If the list does not exist, it is created during processing.

The command is mainly intended for use internally by Synon/2E, but should be used by developers if an empty list is required for user-defined processing.

Required



Parameters

The following are parameters for the YINXMDLLST command.

MDLLST

The qualified model list name that is created, or whose indexes are checked. Values for this parameter are described in the following:

- **model list name**—(default) The list name must be entered.
- ***MDLLIB/**—The library, in which the model list file to be checked exists, is the first model library found in the current job's library.
- ***LIBL/**—The library list is searched to find the first model library.
- **model library name**—The model library name.

LSTLF

The name of the logical file that is checked. Values for this parameter are described in the following:

- **YMDLLST00L**—(default) The member in file YMDLLST00L is checked for existence. If it does not exist, it is created.
- **YMDLLST01L - 11L**—Processed as for YMDLLST00L.
- ***ALL**—Single value indicating that all logical files are checked. A new logical file member is added to any file that does not already have one for the specified model list.

TEXT

Up to fifty characters can be entered to describe the list. This is stored as member text for the particular list member specified. Values for this parameter are described in the following:

- ***DFT**—(default) A formatted string is used. It contains the model, list and current user id.
- ***NONE**—No text is specified.
- **character string**—The text can be entered.

Notes

For user-defined processing, it is recommended that when creating a new list, the minimum indexes to be checked using this command is YMDLLST00L. This ensures the integrity of the list, since this member is uniquely keyed on the primary key of the file, the object surrogate number.

Example

To check for the existence of list TSTLST, and to check/create the YMDLLST00L and YMDLLST06L indexes, enter the following command:

```
YINXMDLLST MDLLST( TSTLST ) LSTLF +
 ( YMDLLST00L YMDLLST06L ) TEXT( 'Test list.' )
```

YINZWSIPDD command

The YINZWSIPDD command can be used to reset the Action flag on records in a WSIPDD file. See YEXCWSIPDD for more information.

Parameters

The following are parameters for the YINZWSIPDD command.

WSIPDDLIB

The name of the library in which the target WSIPDD file resides. Values for this parameter are described in the following:

- ***GENLIB**—Special value representing the model's generation library.
- **name**—Specify the library containing the YWSIPDDRFP file.

ACTION

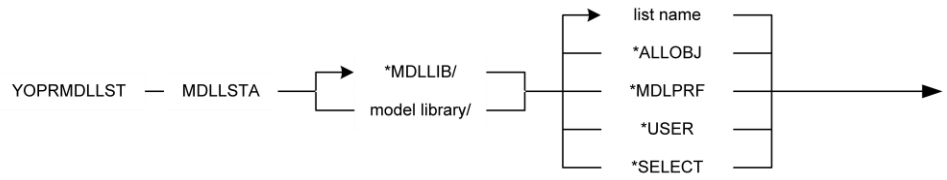
The value to set the ACTION flag to, for all instances in the WSIPDD file. Value for this parameter is described in the following:

- ***INSTALL**—Special value representing the Action field should be set to 'I'.

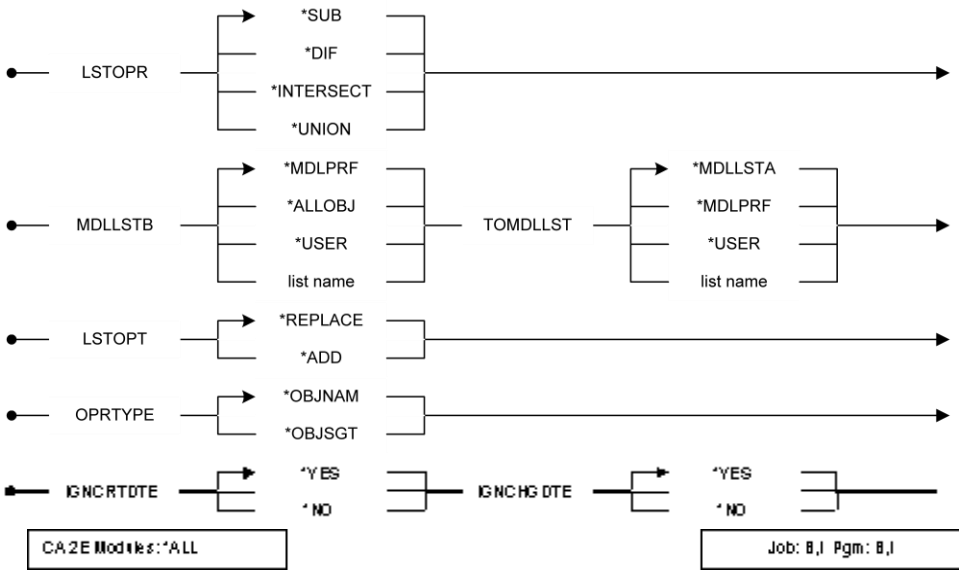
YOPRMDLLST (Operate on a Model Object List) Command

This command allows a user to perform set operations on two input model object lists and to optionally direct the resulting output to a third list. The target list may or may not exist. If it exists, it can be added to or replaced by taking the appropriate option on the LSTOPT parameter.

Required



Optional



Parameters

The following are parameters for the YOPRMDLLST command.

MDLLSTA

The qualified name of the model object list that is the first operand list involved in the operation. Values for this parameter are described in the following:

- **list name**—(default) The first model object list name must be entered.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are used as input to the command.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- ***SELECT**—Special value meaning that the model object list is selected using an interactive display function.
- ***MDLLIB**—(default) Special value meaning that the first model library found in the user's current library list is used as the model library for the first list.
- **library name**—The model library name for the first list.

LSTOPR

The operation to be performed on the two lists (sets). See the notes that follow for more details. Values for this parameter are described in the following:

- ***SUB**—(default) Subtract the contents of MDLLSTB from MDLLSTA.
- ***DIFF**—Calculate the difference between the two input lists.
- ***INTERSECT**—Calculate the intersection of the two input lists.
- ***UNION**—Calculate the union of the two input lists.

MDLLSTB

The name of the model object list that is the second operand list involved in the operation. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The change list value is retrieved.
- ***ALLOBJ**—Special value meaning that a model object list is not used, but that all model objects are used as input to the command.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- **list name**—The second model object list name must be entered.

TOMDLLST

The qualified name of the model object list that is the target list involved in the operation. Values for this parameter are described in the following:

- ***MDLLSTA**—(default) Single value meaning that the target list of the operation is the first input list.
- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. The outlist value is retrieved.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- **list name**—The target model object list name.

LSTOPT

This parameter specifies the action taken if the list already exists in the target model. Values for this parameter are described in the following:

- ***REPLACE**—(default) The existing model object list is replaced with the output from this command.
- ***ADD**—The existing model object list is augmented with the output from this command.

OPRTYPE

Operation method. Values for this parameter are described in the following:

- ***OBJNAM**—(default) The list entries are compared by object name.
- ***OBJSGT**—The list entries are compared by object surrogate number. This method of comparison may be useful when comparing lists containing objects that have been renamed since the list was created, and the developer does not wish to refresh the list entry prior to the comparison.

IGNCRTDTE

This parameter specifies that create dates on the records are ignored when comparing list entries. This parameter is intended for use when comparing objects from different models. Values for this parameter are described in the following:

- ***YES**—(default) Ignore create dates when comparing objects.
- ***NO**—Do not ignore create dates when comparing objects.

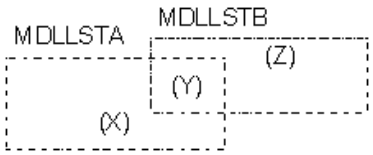
IGNCHGDTE

This parameter specifies that change dates on the records are ignored when comparing list entries. Although possible when comparing lists within a particular model, this parameter is most useful when comparing lists of the same objects between different models. Values for this parameter are described in the following:

- ***YES**—(default) Ignore change dates when comparing objects.
- ***NO**—Do not ignore change dates when comparing objects.

Notes

- The model library specified must be a valid model library.
- A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.
- Both MDLLSTA and MDLLSTB must exist prior to running the command.
- The MDLLSTA and MDLLSTB lists cannot be the same.
- List operations are as follows:
  - This diagram illustrates the entries of the two intersecting lists:



  - The results of the list operations are:

Operand 1	LSTOPR	Operand 2	TOMDLLST
MDLLSTA	*UNION	MDLLSTB	Area: X,Y and Z
MDLLSTA	*DIFF	MDLLSTB	
MDLLSTA	*INTERSECT	MDLLSTB	
MDLLSTA	*UNION	MDLLSTB	

- The LSTOPT parameter is ignored if the target list does not already exist.

Examples

To output a model object list WRKLST to contain all objects from input list INLSTA that are not in input list INLSTB:

```
YOPRMDLLST MDLLSTA( *MDLLIB/INLSTA ) +
LSTOPR( *SUB ) MDLLSTB( INLSTB ) TOMDLLST +
 ( WRKLST )  LSTOPT( *REPLACE )
```

To combine two model object lists to the first input list:

```
YOPRMDLLST MDLLSTA( *MDLLIB/INLSTA ) +
LSTOPR( *UNION ) MDLLSTB( INLSTB ) +
TOMDLLST ( *MDLLSTA )
```

To compare list entries between two models to detect whether objects have been changed in one of the models:

```
YCPYMDLLST FRMMDLLST(YOURMDL/WRKLST)+
 TOMDLLST( MYMDL/*USER ) LSTOPT(*REPLACE)
 YOPRMDLLST MDLLSTA( MYMDL/*USER ) +
LSTOPR( *DIFF ) MDLLSTB( *ALLOBJ )+
 TOMDLLST( *MDLLSTA )  IGNCHGDTE( *NO )
```

Chapter 6: Commands (YPOPWSIPDD - Y2CALL)

This chapter contains details for CA 2E commands YPOPWSIPDD through Y2CALL. These commands appear in alphabetical order and include descriptions of their functions, parameters and allowed values, notes, and examples. Each command is also accompanied by a command diagram.

This section contains the following topics:

YPOPWSIPDD (Populate WSIPDD File) Command

Populates a Web Service Instance Portable Deployment Data file (WSIPDD). Once populated, a WSIPDD file can be moved to a remote machine, where the related YEXCWSIPDD (Execute WSIPDD) command can process the file to deploy web service instances on that remote machine.

Notes:

- Portable deployment does not require CA 2E or 1E to exist on the remote machine. See YEXWSIPDD for details. The YPOPWSIDD command takes a model list as input. The model list is processed and for each function of type Web Service, additional processing occurs (items in the list that are not Web Service functions are ignored). For each Web Service function, all its web service instances are processed. Where a web service instance is not excluded due to filtering arguments on the YPOPWSIPDD command, an instance record will be created in the target WSIPDD file.
- The target WSIPDD file is always called YWSIPDDRFP, but the location is specified on the WSIPDDLIB parameter. The Target parameters on the YPOPWSIPDD command allow the modeled web service instance data to be overridden when populated to the WSIPDD file.

Parameters

The following are parameters for the YPOPWSIPDD command.

MDLLST

The qualified name of the model object list that is edited. Values for this parameter are described in the following:

- ***MDLPRF**—Special value meaning that the model object list name is retrieved from the user profile extension record for the current user in the specified model library. Create web service instance.
- ***USER**—Special value meaning that the model object list name is the same as the name of the current user.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.

WSIPDDLIB

The name of the library in which the target WSIPDD file resides. Values for this parameter are described in the following:

***GENLIB**— Special value representing the model's generation library

name—Specify the library containing the YWSIPDDRFP/00L files.

Note: The WSIPDD file can be created in a target library using the YDUPAPPOBJ command (see the *WS argument for the DUPOPT parameter).

MBROPT

Determines whether output replaces or is appended to any existing data in the WSIPDD file. Values for this parameter are described in the following:

- ***REPLACE**—Replace existing member.
- ***ADD**—Add to the contents of any existing member.

FMACHINE

Indicates filtering that should be applied to each web service instance to determine if it should be added to the target WSIPDD file. Values for this parameter are described in the following:

- ***ALL**—No filtering is applied, all web service instances, discovered via the input model list, are populated to the WSIPDD file.
- **name**—Any web service instance, discovered via the input model object list, whose Server name does NOT match this Filter Server name will NOT be populated to the WSIPDD file.

FSERVICE

Indicates filtering that should be applied to each web service instance to determine if it should be added to the target WSIPDD file. Values for this parameter are described in the following:

- ***ALL**—No filtering is applied, all web service instances, discovered via the input model list, are populated to the WSIPDD file.
- **name**—Any web service instance, discovered via the input model object list, whose Service name does NOT match this Filter Service name will NOT be populated to the WSIPDD file.

TMACHINE

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. Modeled web service instance information is used to populate the WSIPDD file.
- ***CURRENT**—The modeled web service instance Machine name is overridden to *CURRENT in the WSIPDD target file.
- **name**—The modeled web service instance Machine name is overridden to the named value in the WSIPDD target file.

TSERVER

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. Modelled web service instance information is used to populate the WSIPDD file.
- **character value**—The modelled web service instance Server name is overridden to the named value in the WSIPDD target file.

Note: Server name must match (case sensitive) with the name of the actual application server.

TSERVICE

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. Modelled web service instance information is used to populate the WSIPDD file.
- **name**—The modelled web service instance Service name is overridden to the named value in the WSIPDD target file.

TGTOBJLIB

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. Modelled web service instance information is used to populate the WSIPDD file.
- **name**—The modelled web service instance target object Library name is overridden to the named value in the WSIPDD target file.

TUSRPRF

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file. Values for this parameter are described in the following:

- ***INSTANCE**—No overriding is applied. Modelled web service instance information is used to populate the WSIPDD file.
- **name**—The modelled web service instance target user Profile name is overridden to the named value in the WSIPDD target file.

TRTLIBL

Indicates whether modelled web service instance information should be overridden when populated to the target WSIPDD file.

- ***INSTANCE**—No overriding is applied. Modelled web service instance information is used to populate the WSIPDD file.
- **name**—The modelled web service instance target runtime library list is overridden to the name value in the WSIPDD target file.

FILTYP

Type of access path that is subsetted on the Work With Distributed Files panel. Values for this parameter are described in the following:

- **TABLE**—(default) Subset to show only tables.
- **VIEW**—Subset to show only views.
- ***ALL**—Do not subset. Show all tables and views.

YPRCSFLSEL (Process Subfile Selection) Command

This command allows you to perform certain actions on a model object, such as view object details, edit the action diagram for the object (if it is a function with an action diagram), display object locks and so forth. Most of the options are available using the same subfile selector as is used in model panels.

Parameters

The following are parameters for the YPRCSFLSEL command.

OBJSGT

The 7-digit object surrogate for the object.

SFLSEL

The subfile-selection operator you wish to use. Note that not all selectors are available for all object types.

- L, L1, 31, 32 (Locks)—Display locks for the object.
- U, U1, 91, 92 (Usages)—Display usages for the object.
- R, 81, 82 (References)—Display references for the object.
- Z, 2, 5, (Details)— Display object details (only applies to objects of type FUN, MSG, ACP, FLD, CND, ARR, APP or FIL).
- C, 3 (Copy)— Copy the object (only applies to objects of type FUN and MSG).
- D, 4 (Delete)— Delete the object (only applies to objects of type FUN, MSG, ACP, ARR, CND or APP).
- F, 10 (Action diagram)— Edit the action diagram for the object (only applies to functions which have an action diagram and to function fields). If used for a file, this will display the EDIT FUNCTIONS panel for that file.
- P, 13 (Parameters)— Edit the object parameters (only applies to functions, function fields and messages).
- 16 (Y2CALL)— Call the object using the Y2CALL interface (only applies to external functions).
- S, 17 (Device design)— Edit the device design for the object (only applies to functions which have a device design).
- T, 18 (Device structure)— Edit the device structure for the object (only applies to print functions).
- 19 (Versions)— Work with object versions (only applies to objects of type FUN and MSG).
- A, 20 (Access path)— Work with file access paths (only applies to functions).
- N, 21, 22 (Narrative text)— Edit the narrative text for the object.
- E, 23 (STRSEU)— Edit the object source using SEU (only applies to objects which have source).
- 25 (Document model function)— Document the model function using the YDOCMDLFUN interface.
- 28 (Check out)— Check out the object (only applies if the model is set up for change control).
- O, 30 (Open function object)— Open the function object (only applies to functions).
- 8 (Model object details)— Display the model object details.
- 94 (Simulate private change)— Simulate a private change to the object.
- 95 (Simulate public change)— Simulate a public change to the object.

- 12 (Resolve conflicts)— Resolve object conflicts (only applies if the model is set up for change control).
- RA (Refresh Action Diagram)— Checks the action diagram of the specified function and refreshes the action titles to use any changed object names as necessary (only applies to functions which have an action diagram).

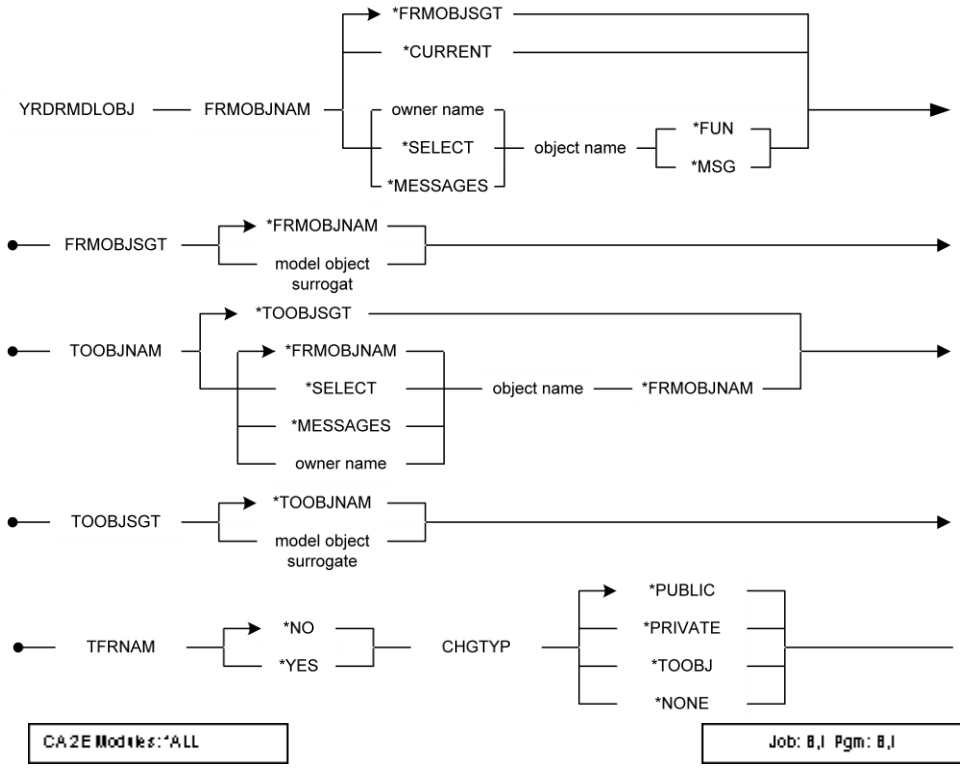
YRDRMDLOBJ (Redirect Model Object) Command

This command provides the means by which the usages of one object can be redirected to another object of the same type. For example, using this command, it is possible to switch from one version of a function to another.

Objects can be identified by name or by object surrogate number.

At present, only functions or messages can be redirected with this command.

Required



Parameters

The following are parameters for the YRDRMDLOBJ command.

FRMOBJNAM

The name of the object from which usage dependencies in the model are redirected. Values for this parameter are described in the following:

- ***FRMOBJSGT**—(default) Single value indicating that the object surrogate number parameter is used to identify the model object.
- ***CURRENT**—Single value indicating that the To model object is made current regardless of whichever version in its group is current.
- **object owner name**—(default) The character name of the object that owns the object. Thus, for a function, the owning file would be entered.
- ***SELECT**—Special value indicating that the object to be redirected from is selected using an interactive display function.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- **object name**—The character name of the object.
- **object type**—The object type of the object.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message. Note that for objects of type *MSG, the owner is the *Messages file.

FRMOBJSGT

Unique number identifier of the model object from which usage dependencies in the model are redirected. Values for this parameter are described in the following:

- ***FRMOBJNAM**—(default) Use the object name parameter details to identify this model object.
- ***SELECT**—Single value indicating that the object to be redirected to is selected using an interactive display function.
- **object surrogate**—The surrogate number of the model object.

TOOBJNAM

The name of the object to which usage dependencies in the model are redirected. Values for this parameter are described in the following:

- ***TOOBJSGT**—(default) Single value indicating that the object surrogate number parameter is used to identify the model object.
- **object owner name**—The character name of the object that owns the object. Thus, for a function, the owning file would be entered.
- ***SELECT**—Special value indicating that the object to be redirected from is selected using an interactive display function.
- ***MESSAGES**—The internal file *Messages is the owner of the object (note that this value is only allowed for objects of type *MSG).
- **object name**—The character name of the object.
- **object type**—The object type of the object.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message. Note that for objects of type *MSG the owner is the *Messages file.

TOOBJSGT

Unique number identifier of the model object to which model usage dependencies are redirected. Values for this parameter are described in the following:

- ***TOOBJNAM**—(default) Use the object name parameter details to identify this model object.
- **object surrogate**—The surrogate number of the model object.

TFRNAM

This parameter allows the developer to choose whether or not to transfer the 25-character object name at the same time as redirecting model usages. Values for this parameter are described in the following:

- ***NO**—(default) The from object name is not transferred.
- ***YES**—The from object name is transferred.

CHGTYP

This field may be used to specify the type of change that the processing programs are to consider the redirection of references to be. Changing this field can cause the component changed flag (COMPCHG) on objects that use this object to be updated also. This will indicate the action required by the user to accommodate the change. Values for this parameter are described in the following:

- ***PUBLIC**—(default) The object is considered to have been publicly changed. Appropriate component change processing is performed.
- ***PRIVATE**—The object is considered to have been privately changed. Appropriate component change processing is performed.
- ***TOOBJ**—The change type associated with the to object will be examined by component change processing.
- ***NONE**—The redirection of references is not to be considered a change to the object. No component change processing is performed.

Notes

- A search is made for the two model objects in the first model library found in the library list.
- Redirecting references involves repointing model objects. Since the primary key of model objects is the internal surrogate number, it is this value that is involved in the repointing mechanism. In effect, repointing involves changing the foreign key references of model objects from the source surrogate number to the target surrogate number. Note the following points:
  - Only objects of type FUN and MSG can be repointed.
  - It is not possible to repoint between a MSG and a FUN.
  - Any usages of the source object are switched to use the target object.
  - Archive objects will not be updated during the repointing process. This has repercussions for partial rollback. See the Synon/CM Enhancements section on Rollback procedure for more details.
  - The from object must be the current member of its group.
  - If the target of the command is not a member of the same group, then it must be the current member of its group.
  - If the target of the command is not a member of the same group, then the from object remains current after the transfer, and the implementation name (if there is one) is not transferred. This is necessary to ensure that there is a current member in the from group.
- The implementation name of the object is transferred to the new current object. This is the message identifier for MSG type objects and the source member name for generatable FUN type objects.
  - Note that for EXCUSRSRC and EXCUSRPGM there is special processing due to the fact that these types are corrupted by concurrent development unless the previous source member is preserved. The source member name is transferred to the target function. However, if the source member exists, a new source member name is automatically generated to contain a copy of the original source prior to the transfer.
- Model objects can either be identified by object name (FRMOBJNAM) or by object surrogate key number (FRMOBJSGT). If the FRMOBJNAM parameter is used, the processing program must convert to surrogate key number internally. Thus, it will normally be more efficient to use the surrogate number if this value is available. The surrogate number for an object can be obtained using the Retrieve Model Object command (YRTVMDLOBJ).
- Note that if TFRNAM (*YES) is specified, the object names are swapped between the two objects. Developers should take note of this change and also that model object lists are not automatically updated to reflect this change to the objects referred to by the entries. Use the Check Model Object List command (YCHKMDLLST) to refresh model object list entries.

Example

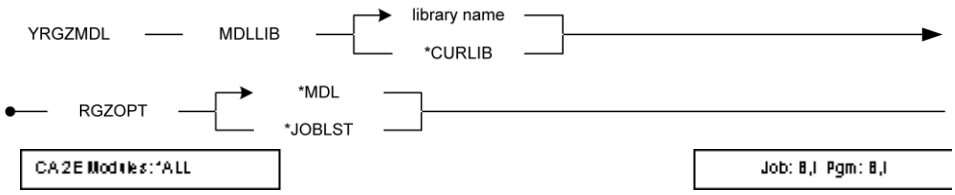
To redirect model object dependencies from Edit Owner to Edit Owner (new ver) enter the following command:

```
YRDRMDLOBJ FRMOBJNAM +
 ( 'Owner' 'Edit Owner' *FUN ) TOOBJNAM +
(*FRMOBJNAM 'Edit Owner (new ver)' + *FRMOBJNAM )
```

YRGZMDL (Reorganize Model) Command

Reorganizes a model or job list to recover space used by deleted records. This command should be run regularly for both a model and a job list for the purposes of housekeeping.

Required



Parameters

The following are parameters for the YRGZMDL command.

MDLLIB

Name of library containing a design model or job list that is reorganized. The value for this parameter is as follows:

- ***CURLIB**—Use current library for invoking job.

RGZOPT

Option to reorganize model or job list contained in library specified by MDLLIB parameter. Values for this parameter are described in the following:

- ***MDL**—(default) Reorganize model.
- ***JOBLST**—Reorganize job list.

Notes

The model or job list cannot be used while it is being reorganized.

Example

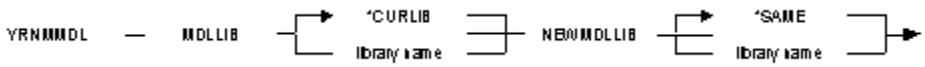
To reorganize model MYMDL:

```
YRGZMDL MDLLIB(MYMDL)
```

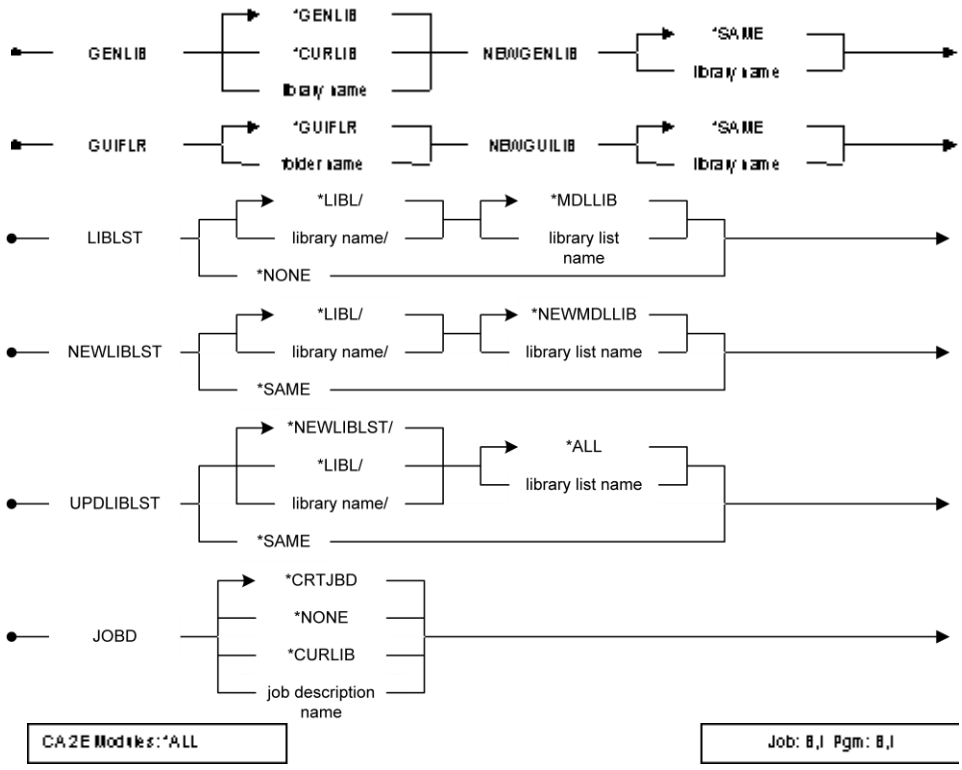
YRNMMDL (Rename Model) Command

Renames a model library. Updates model values and library lists.

Required



Optional



Parameters

The following are parameters for the YRNMMDL command.

MDLLIB

Name of library to contain the design model that is renamed. Must be a model library. The value for this parameter is as follows:

- ***CURLIB**—(default) Current library invokes job.

NEWMDLLIB

New name for the model library. The value for this parameter is as follows:

- ***SAME**—(default) Do not rename model library.

GENLIB

Name of generation library to be renamed. Values for this parameter are described in the following:

- ***GENLIB**—(default) Name is specified by the YGENLIB model value of the model named by the MDLLIB parameter.
- ***CURLIB**—Current library invokes job.

NEWGENLIB

New name for generation library. The value for this parameter is as follows:

- ***SAME**—(default) Do not rename the generation library.

GUIFLR'

Name of the shared folder used by Synon/TC to transfer information to the PC that is to be renamed. Values for this parameter are described in the following:

- ***GUIFLR**—(default) Name is specified by the YGUIFLR model value of the model named by the MDLLIB parameter.
- **folder name**—Valid shared folder name.

NEWGUIFLR

New name for the shared folder used by Synon/TC to transfer information to the PC. Values for this parameter are described in the following:

- ***SAME**—(default) Do not rename the Synon/TC shared folder.
- **folder name**—Valid shared folder name.

LIBLST

Qualified name of a library list that is renamed. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Library list has the same name as the model library.
- ***NONE**—Do not rename the library list.

NEWLIBLST

New name for library list. Values for this parameter are described in the following:

- ***NEWMDLLIB**—(default) Rename library list to have same name as that specified by the NEWMDLLIB parameter.
- ***SAME**—Do not rename library list.

UPDLIBLST

Qualified name of library lists that are updated to reflect the new names of the model and/or generation libraries. Values for this parameter are described in the following:

- ***NEWLIBLST/*ALL**—(default) All library lists in the library list file of the library specified by the NEWLIBLST parameter are updated.
- ***NONE**—Do not update any library lists.

JOBD

Job description name in the library given by the MDLLIB parameter whose initial library list is updated. Values for this parameter are described in the following:

- ***CRTJBD**—(default) Name is specified by YCRTJBD model value of model named by MDLLIB parameter.
- ***NONE**—Do not update job description.
- ***CURLIB**—Current library.

Notes

- You cannot rename a model or a generation library while they are in use.
- Prior to running the command, all journals and journal receivers must be removed from each library to be renamed.
- If specifying a LIBL to be updated, the LIBL must exist at the start of running the command. If the only LIBL the user wants to update is the one that gets renamed from LIBLST to NEWLIBLST, the user should enter LIBLIST name for UPDLIBLST.
- The command does the following:
  - If NEWMDLLIB is not *SAME, renames the model library, and changes the YMDLLIB model value to the value specified by NEWMDLLIB parameter.
  - If NEWGENLIB is not *SAME, renames the generation library, and changes the YGENLIB model value to the value specified by NEWGENLIB parameter.
  - If NEWLIBLST is not *SAME, and LIBLST is not *NONE, renames the library list to the value specified by NEWLIBLST parameter.
  - Updates library lists specified by the UPDLIBLST parameter for the renamed libraries.
  - If JOBD is not *NONE, changes the initial library list of the named job description to the value specified by NEWLIBLST parameter.
- If you have an SQL collection, be aware that YRNMMDL will not rename the SQL collection.

Example

To rename library MYMDL to YOURMDL, and library MYGEN to YOURGEN, and library list MYMDL to YOURMDL, updating the initial library list of the appropriate job description:

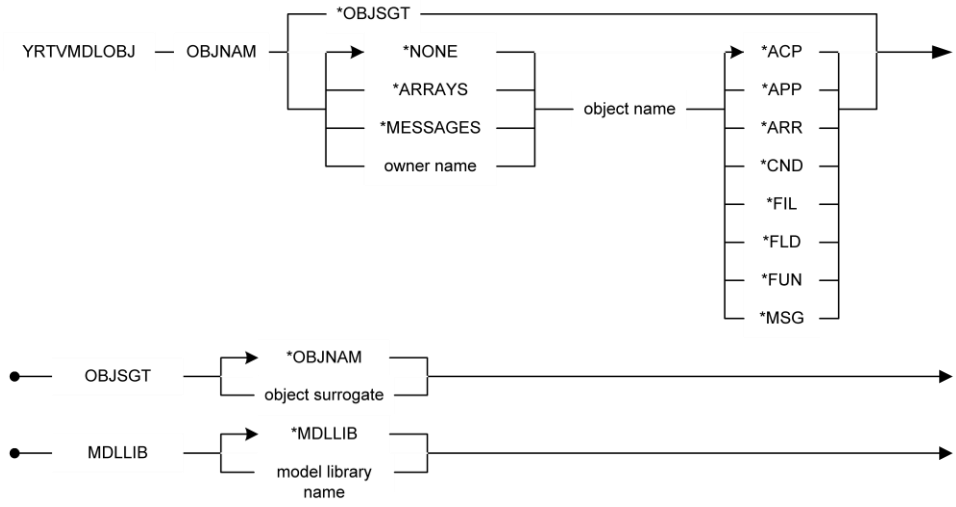
```
YRNMMDL MDLLIB(MYMDL) NEWMDLLIB + (YOURMDL) GENLIB(MYGEN) + NEWGENLIB(YOURGEN)
```

YRTVMDLOBJ (Retrieve Object Description) Command

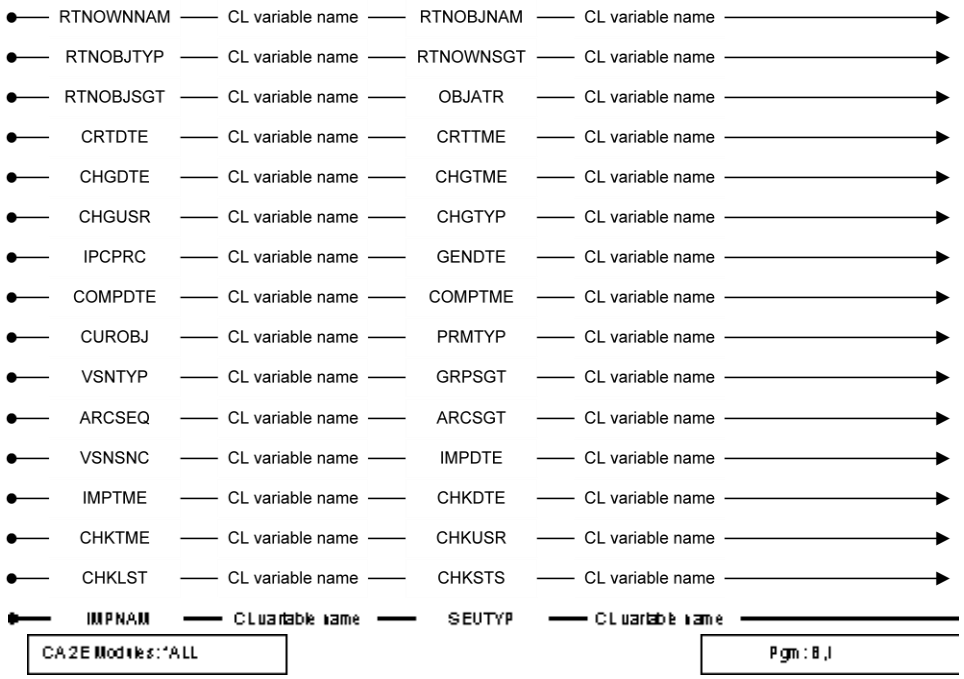
This command provides access to general and certain user-definable object data that is supported in the data model for most model object types.

It is intended that these data be used to support a user-defined change control facility operating upon model objects.

Required



Optional



Parameters

The following are parameters for the YRTVMDLOBJ command.

OBJNAM

The object name of the object whose details are retrieved. This parameter consists of three elements that together identify a model object. Values for this parameter are described in the following:

- ***OBJSGT**—(default) Single value indicating that the object surrogate is used to identify the model object.
- **object owner name**—The character name of the object that owns the object. Thus, for a function, the owning file would be entered.
- **object name**—The character name of the object.
- **object type**—The object type of the object.
- ***ACP**—Object is of type access path.
- ***APP**—Object is of type application area.
- ***ARR**—Object is of type array.
- ***CND**—Object is of type condition.
- ***FIL**—Object is of type file.
- ***FLD**—Object is of type field.
- ***FUN**—Object is of type function.
- ***MSG**—Object is of type message.

OBJSGT

Unique number identifier of the model object whose details are retrieved. Values for this parameter are described in the following:

- ***OBJNAM**—(default) Use object name to identify the model object.
- **object surrogate**—The surrogate number of the model object.

RTNOWNNAM

CL variable that receives the name of the owner of the object. If the object has no owner, then *NONE is returned. Character variable, twenty-five bytes long.

RTNOBJNAM

CL variable that receives the name of the object. Character variable, twenty-five bytes long.

RTNOBJTYP

CL variable that receives the object type of the object. Character variable, three bytes long.

TNOWNSGT

CL variable that receives the surrogate key of the owner of the object. Decimal variable, seven digits packed.

RTNOBJSGT

CL variable that receives the surrogate key of the object. Decimal variable, seven digits packed.

OBJATR

CL variable that receives the object attribute of the object. Character variable, three bytes long.

CRTDTE

CL variable that receives the value of the creation date of the object. Character variable, eight bytes long.

CRTTME

CL variable that receives the value of the creation time of the object. Character variable, six bytes long.

CHGDTE

CL variable that receives the value of the last change date of the object. Character variable, eight bytes long.

CHGTME

CL variable that receives the value of the last change time of the object. Character variable, six bytes long.

CHGUSR

CL variable that receives the name of the user who last changed the object. Character variable, ten bytes long.

CHGTYP

CL variable that receives the current value of the change type that has occurs for the object. Character variable, eight bytes long. Values for this parameter are described in the following:

- **blank**—No changes have occurred to this object since it was last updated for non-generatable objects or last generated for generatable objects.
- ***PRIVATE**—A private change has been made to the object, such as a change to the panel design of an interactive function type.
- ***PUBLIC**—A public change has been made to the object, such as a change to the parameters of a function.
- ***GEN**—A change has been made to the object requiring generation/compilation of the implementation objects only, such as a change to the source member text.
- ***OBJ**—A change has been made to the object which affects no other objects and does not require regeneration of any source.

IPCPRC

CL variable that receives the component change processed value for the object. Character variable, four bytes long.

Values for this parameter are described in the following:

- ***YES**—Component change processing has been performed for the object.
- ***NO**—Component change processing has not been performed for the object.

GENDTE

CL variable that receives the value of the last successful generation date of the object. Character variable, eight bytes long.

ENTME

CL variable that receives the value of the last successful generation time of the object. Character variable, six bytes long.

ACTRQD

CL variable that receives the current value of the component change indicator for the object. Character variable, four bytes long. Values for this parameter are described in the following:

- **blank**—No components of this object have been changed since it was last edited for non-generatable objects or last generated for generatable objects.
- ***EDT**—A component of the object has been changed requiring that the object be revisited and edited to resolve the changes. If this is a generatable object, it must also be regenerated.
- ***GEN**—A component of the object has changed requiring that this generatable object be regenerated.

OMPDTE

CL variable that receives the last date that the object or one of its components was changed. Character variable, eight bytes long.

COMPTME

CL variable that receives the last time that the object or one of its components was changed. Character variable, six bytes long.

CUROBJ

CL variable that receives the current value of the current object indicator for the object. Character variable, four bytes long. Values for this parameter are described in the following:

- ***YES**—This is a current object, which means that it is visible on model panels and can be used extensively throughout the data model.
- ***NO**—This is not a current object. This means that it is not visible on normal model panels. It is visible on model object lists and on the Display Versions panel. This does not necessarily mean that the object is not used.

PRMTYP

CL variable that receives the current value of the promotion type that occurs for the object.

Character variable, four bytes long. Values for this parameter are described in the following:

- **blank**—No promotion type set.
- ***ADD**—The object is added to a target environment.
- ***CHG**—The object replaces an existing version of itself in the target environment.
- ***GEN**—The object is generated from the target environment. The design object is not promoted.

SNTYP

CL variable that receives the current value of the version type indicator for the object.

Character variable, four bytes long. Values for this parameter are described in the following:

- **blank**—The object is not part of a group.
- ***PRD**—This is a production object. There can only be one member of a group with this status.
- ***DEV**—This is a development version of the object.
- ***ARC**—This is an archive version of the object.

RPSGT

CL variable that receives the current value of the group object surrogate key number. This value will be zero if the object does not currently belong to a group.

Numeric variable, seven bytes long.

ARCSEQ

CL variable that receives the current value of the archive sequence number of the object. This value is only set for archive versions.

ARCSGT

CL variable that receives the current value of the archive surrogate number. This value is only set for production versions that have an archive version in existence.

VSNSNC

CL variable that receives the current value of the version synchronized indicator for the object. This field is intended for use as part of a change control system.

Character variable, four bytes long. Values for this parameter are described in the following:

- ***YES**—The object is synchronized.
- ***NO**—The object is not synchronized.

MPDTE

CL variable that receives the last import date for the object. This value is set by the import program when the object is created in the model as part of an import process from another model or another product. Thus, for example, this date might represent the date the object was created in the model as part of processing by the Copy Model Object command (YCPYMDLOBJ).

Character variable, eight bytes long.

IMPTME

CL variable that receives the last import time for the object. This value is set by the import program when the object is created in the model as part of an import process from another model or another product. Thus, for example, this time might represent the time the object was created in the model as part of processing by the Copy Model Object command (YCPYMDLOBJ).

Character variable, six bytes long.

CHKDTE

CL variable that receives the value of the checkout date of the object. Character variable, eight bytes long.

CHKTME

CL variable that receives the value of the checkout time of the object. Character variable, six bytes long.

CHKUSR

CL variable that receives the name of the user who last checked out the object. Character variable, ten bytes long.

CHKLST

CL variable that receives the name of the list on which the object checked out. Character variable, ten bytes long.

CHKSTS

CL variable that receives the checkout status of the object. Character variable, seven bytes long.

IMPNAM

CL variable that receives the implementation name of the object. Character variable, ten bytes long. Values for this parameter are described in the following:

- **ACP**—source member name
- **APP**—application area code
- **ARR**—blank
- **CND**—blank
- **FIL**—generation prefix
- **FLD**—DDS name
- **FUN**—source member name, or blank (internal functions)
- **MSG**—message identifier

SEUTYP

CL variable that receives the system SEU type of the object. This value is blank for non-source based objects. Character variable, ten bytes long.

ASSOJ

CL variable that identifies model objects associated with assimilated objects. Character variable, four bytes long. Values for this parameter are described in the following:

- ***YFIL**—Any file that is flagged as assimilated.
- ***YACP**—Physical access path that is based on an assimilated file.
- ***YFUN**—Execute User Program (EXCUSRPGM) or Execute User Source (EXCUSRSRC) functions.

Notes

Object data can be retrieved using either the object name or the object surrogate identifier. Whichever method is used, all return variables contain the appropriate data.

Example

To find out the date and time that the Customer file was last changed, and who made that change:

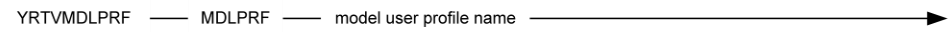
```
YRTVMDLOBJ OBJNAM( *NONE 'Customer' +
*FIL ) CHGDTE( &CHGDTE ) CHGTME +
 ( &CHGTME ) CHGUSR( &CHGUSR )
```

This information could then be used for comparison with the same object (by name) in another data model.

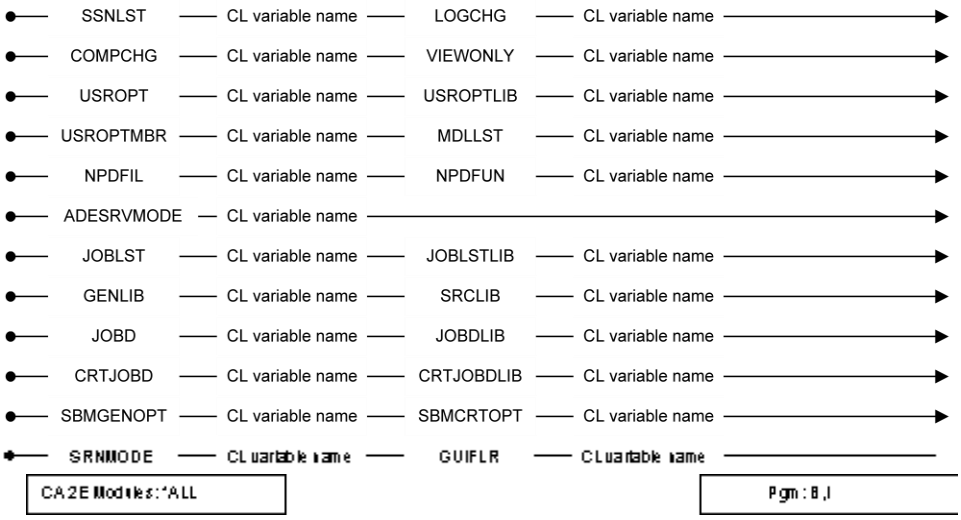
YRTVMDLPRF (Retrieve Model Profile details) Command

Certain data is stored in each data model associated with the user profiles of developers. This command allows these values to be retrieved in command language programs.

Required



Optional



Parameters

The following are parameters for the YRTVMDLPRF command.

MDLPRF

The name of the model user profile that is retrieved. The value for this parameter is as follows:

- **user profile name**—The name of the user profile must be entered.

SSNLST

CL variable that receives the name of the session list for the specified user. Character variable, ten bytes long.

LOGCHG

CL variable that receives the log changed objects indicator. Character variable, four bytes long.

COMPCHG

CL variable that receives the perform component changed processing indicator. Character variable, four bytes long.

MDLLST

CL variable that receives the name of the model object list that is retrieved when *MDLPRF is specified on list and other commands. Character variable, ten bytes long.

VIEWONLY

CL variable that receives the view only indicator. Character variable, four bytes long.

USROPT

CL variable that receives the name of the user options file. Character variable, ten bytes long.

USROPTLIB

CL variable that receives the name of the user options file library. Character variable, ten bytes long.

USROPTMBR

CL variable that receives the name of the user options file member. Character variable, ten bytes long.

NPDFIL

CL variable that receives the name of notepad function owning file. Character variable, twenty-five bytes long.

NPDFUN

CL variable that receives the name of notepad function. Character variable, twenty-five bytes long

ADESRNMODE

CL variable that receives the default value for the action diagram full screen mode. Character variable, four bytes long.

JOBLST

CL variable that receives the name of the job list to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

JOBLSTLIB

CL variable that receives the name of the job list library to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

GENLIB

CL variable that receives the name of the generation library to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

SRCLIB

CL variable that receives the name of the library where source is placed. Character variable, ten bytes long.

JOBD

CL variable that receives the name of the job description to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

JOBDLIB

CL variable that receives the name of the job description library to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

CRTJOBD

CL variable that receives the name of the create job description library to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

CRTJOBDLIB

CL variable that receives the name of the create job description library to be defaulted in the Submit Model Creates command (YSBMMDLCRT). Character variable, ten bytes long.

SBMGENOPT

CL variable that receives the default value for the submit generation option parameter on the Submit Model Creates command (YSBMMDLCRT). Character variable, four bytes long.

SBMCRTOPT

CL variable that receives the default value for the submit compilation option parameter on the Submit Model Creates command (YSBMMDLCRT). Character variable, six bytes long.

SRNMODE

CL variable that receives the default value for full screen mode. It is used by the Edit Model List command (YEDTMDLLST). Character variable, one byte long.

GUIFLR

CL variable that receives the default value for the GUI folder parameter on the Submit Model Creates command (YSBMMDLCRT). Character variable, sixty-three bytes long.

Notes

The model profile name must exist prior to running this command.

Example

To retrieve the model update details for user GEORGE:

```
YRTVMDLPRF MDLPRF( GEORGE ) CHGLST +
 ( &CHGLST ) LOGCHG( &LOGCHG ) COMPCHG +
 ( &COMPCHG )
```

YRTVMDLVAL (Retrieve Model Value) Command

Retrieves a model value.

Required

(1) One of the allowed values.

Parameters

The following are parameters for the YRTVMDLVAL command.

MDLVAL

Name of model value that is retrieved.

One of the model values.

See the YCHGMDLVAL (Change Model Value) command in this chapter for further details about the allowed model values.

VALUE

CL variable to receive model value. (80 characters.)

Notes

- See the YCHGMDLVAL command section in this chapter for further information about the role of the various model values.
- This command can only be executed within a CL program.

Example

To retrieve the YGENLIB model value:

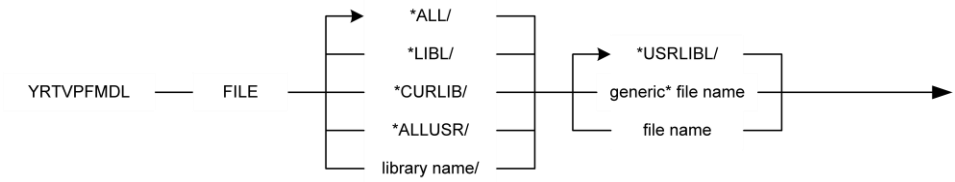
```
DCL VAR(&VALUE) TYPE(*CHAR) LEN(80)
 DCL VAR (&GENLIB) TYPE(*CHAR) LEN(10)
 YRTVMDLVAL MDLVAL(YGENLIB) +
VALUE(&VALUE) CHGVAR &GENLIB &VALUE
```

YRTVPFMDL (Retrieve Physical Files Into Model) Command

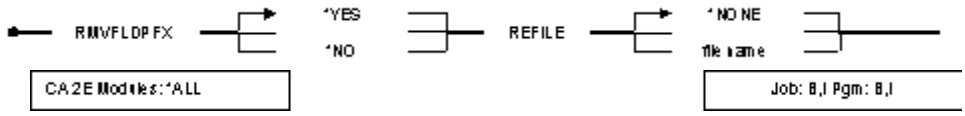
Retrieves file descriptions of compiled database physical files into a design model. All developers and programmers should be out of the model before you execute this command.

For more information on the YRTVPFMDL command, see the "Assimilation" chapter in the *Defining a Data Model* guide.

Required



Optional



Parameters

The following are parameters for the YRTVPFMDL command.

FILE

Qualified generic name of physical files whose descriptions are retrieved into the model. The value for this parameter is as follows:

- ***ALL**—Retrieve all files.

RMVFLDPFX

Specifies whether any action is taken to remove a prefix from the DDS field names of the fields in the retrieved physical files. Values for this parameter are described in the following:

- ***YES**—(default) The first two characters of each field name are considered to be a prefix that is dropped from the name. For example, field name XXCUCD will be translated to XX+CUCD.
- ***NO**—Do not remove the first two characters from the field names.

REFFILE

Name of file to treat as the field reference file. No file or Has relations are created for this file but all fields are added to the data dictionary. File should be among the list of files specified by the FILE parameter. The value for this parameter is as follows:

- ***NONE**—(default) No file is treated as a field reference file.

Notes

The YRTVPFMDL command retrieves into a model all file and field definitions not already present in the model. For each named database file, the command proceeds in the following manner.

- The i OS command Display File Field Description (DSPFFD) is used to obtain the object's file, format and field definitions.
- The existing model is examined to determine if a file already exists in the model with a source member name that is the same as the i OS object name of the file. (The source member names for files are specified with the Edit Access Path Details display: the name is held on the PHY (Physical) access path).

  If an access path of type PHY with the same member name already exists in the model, the retrieved file is ignored. If one does not exist, then a new file is added to the model as follows:

  - The text (if any) of the database file is used as the name: for example, Customer file. If another file already exists with the text, then a number is appended to the text of the new file name to make it unique. For example, Customer file 01265. If the file has no text then the i OS object name is used as the Synon/2E name.
  - The new file has a file type of REF.
  - An access path of type Physical (PHY) is defined. The source member name held for the access path is the i OS object name of the retrieved file.
  - The Assimilated file flag on the file details is set to yes.
  - A Defined as relation is created for the file.
- For each field in the database file definition or column in the table definition, the YRTVPFMDL command proceeds in the following manner:

  The existing model is examined to determine if a field already exists in the model with a DDS name that is the same as that of the retrieved field. The comparison is done as follows:

  - If RMVFLDPFX(*YES) was specified, the field prefix is dropped from the retrieved DDS field name for purposes of comparison. For example, JJCUCD is compared with CUCD. The field names, against which the retrieved field name is compared, are the values (i.e., those shown on the Edit Field Details and the Display Field References displays).
  - If RMVFLDPFX(*NO) was specified, the full DDS name of the retrieved field is used for the comparison. For example, JJCUCD is compared with JJCUCD. The field names against which the retrieved field name is compared are not the values, but rather old field names.
  - If the name matches, a further comparison is made between the length and i OS data type (packed, zoned, or alphanumeric) of the retrieved field and the existing model fields.

- If a field of the correct name and type exists in the model, no new field is added. If one does not exist, then a new field is added to the model as follows:
- The text of the retrieved field is used as the name: for instance, "Customer code". If another field already exists with the text, then a number is appended to the text of the new name to make it unique. For example, Customer code 012654. If the retrieved field has no text, then the retrieved DDS name is used as the Synon/2E name.
- The new field is given a data type according to the i OS data type of the retrieved field as follows:

i OS Type	Length in Bytes	CA 2E Type
A	1	STS
A	2 - 10	CDE
A	11 or more	TXT
P	Any: integer	NBR
P	Any: real	VAL
S	Any: integer	
S	Any: real	VAL

- If RMVFLDPFX(*YES) was specified, the truncated field name is stored as the field name. For example, JJCUCD is stored as CUCD.
- If RMVFLDPFX(*NO) was specified, Synon/2E allocates a new name as the DDS field name. For example, CUSNUM might be stored as ZQNB.
- A Has relation is added to connect every field with the file, unless the file is a field reference file, as specified by the REFFILE parameter.

- A physical file format entry is created. This is used to store:
  - The old field name used in the particular i OS physical file.
  - The sequence of the field within the retrieved PF record.
  - The i OS data type and length of the old field.

  This information can be used by Synon/2E to regenerate source for an i OS physical file that exactly matches the retrieved version. You can use the Edit Physical File Format entries display to examine and change the physical file format entries.
- Amending the retrieved file.

  Having retrieved a database file into a model, you will need to add extra information to specify the key relations and to establish the file-to-file relations among the various files that you have retrieved. This should be done as follows:
  - Change the field attributes of code fields that are the keys of files to be CDE. (Select Z2 to invoke the Edit Field Details display for the key fields, and use F8 to change the attributes).
  - Alter the relation type of the relations that connect Synon/2E files with their own key fields from Has to Known by or Qualified by. (Type over the existing relation using the Edit Database Relations display).
  - Alter the relation type of files containing fields that are the keys to other files from Has to Refers to, or Owned by, and change the referenced field of such relations to be the appropriate referenced Synon/2E file. (Type over the existing relation using Edit Database Relations display. Also delete superfluous relations using the D option).

  You should also amend the field details in order to specify additional data types and to add field conditions:
  - Change the field attributes of non-key fields to be the appropriate type (for example, TXT, STS). (Select Z2 to invoke the Edit Field Details display for the non-key fields, and use F8 to change the attributes).
  - Use the Edit Condition Details display to add conditions. Use the Edit Field Details display to change the field attributes to specify a check condition.
- The removing of field prefixes that takes place if RMVFLDPFX(*YES) is specified is done by a CL program, YRTVPHYR1C, the source of which is supplied. You may modify this program if you wish. An example would be if you need to retrieve file descriptions from systems where a convention of appending a prefix to the end of field names, rather than the beginning, has been used.

- The use of the reference file parameter is not related to the Synon field reference files specified by the model value YFRFVNM.

Example

To retrieve all the physical file descriptions of files whose names begin with "C" and reside in library QGPL, removing field prefixes from the DDS fields:

```
YRTVPFMDL FILE(QGPL/C*)
```

For example, if you have three files to retrieve:

Company	CMPDTAP	XCOCD	Company code	4
		XXCONM	Company name	50
Customer	CUSDTAP	PPCOCD	Company code	4
		PPCUCD	Customer code	4
		PCUNM	Customer name	50
		PPCULM	Credit limit	7.2 P
		PPCUCD	Customer type cd	1
Customer type	CUSTYPP	QQCUCD	Customer type cd	1
		QQCUNM	Customer type nm	50

They will be retrieved as follows:

FIL	**Company**	REF	Has	FLD	**Company code**	CDE	
FIL	**Company**	REF	Has	FLD	**Company name**	TXT	
FIL	**Customer**	REF	Has	FLD	**Company code**	CDE	
FIL	**Customer**	REF	Has	FLD	**Customer code**	CDE	
FIL	**Customer**	REF	Has	FLD	**Customer name**	TXT	
FIL	**Customer**	REF	Has	FLD	**Credit limit**	NBR	
FIL	**Customer**	REF	Has	FLD	**Customer type cd**	STS	
FIL	**Customer type**	REF	Has	FLD	**Customer type cd**	STS	
FIL	**Customer type**	REF	Has	FLD	**Customer type nm**	TXT	

You will probably want to change them to the following set of statements:

FIL	**Company**	REF	Known by	FLD	**Company code**	CDE
FIL	**Company**	REF	Has	FLD	**Company name**	TXT

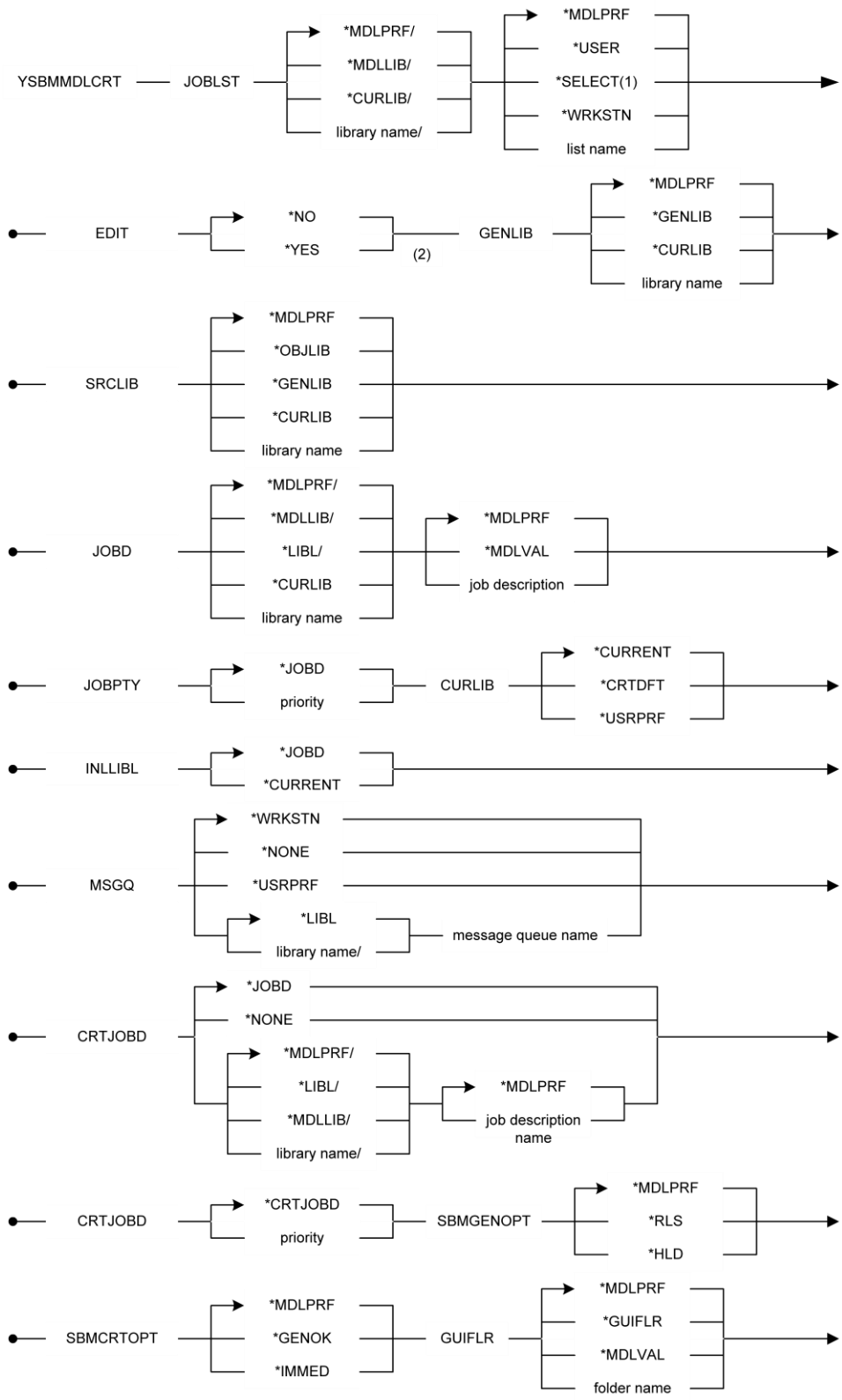
FIL	**Customer**	REF	Owned by	FIL	**Company**	REF
FIL	**Customer**	REF	Known by	FLD	**Customer code**	CDE
FIL	**Customer**	REF	Has	FLD	**Customer name**	TXT
FIL	**Customer**	REF	Has	FLD	**Credit limit**	VAL
FIL	**Customer**	REF	Refers to	FIL	**Customer type**	REF
FIL	**Customer type**	REF	Known by	FLD	**Customer type**	CDE
FIL	**Customer type**	REF	Has	FLD	**Customer type nm**	TXT

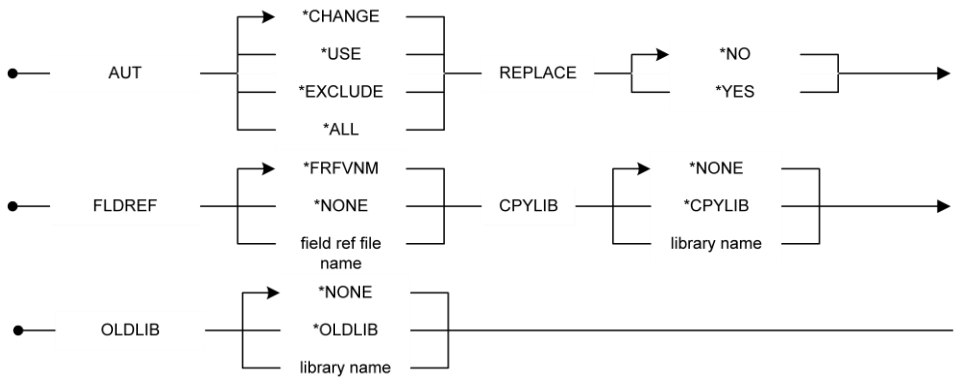
Note: For more information about this example, see the YRTVPFMDL command in this guide.

YSBMMDLCRT (Submit Create Requests from Model) Command

Generates and compiles the source members named in a CA 2E job list.

Optional


YSBMMDLCRT
JOBLST
*MDLPRF/
*MDLLIB/
*CURLIB/
library name/
*MDLPRF
*USER
*SELECT(1)
*WRKSTN
list name
EDIT
*NO
*YES
(2)
GENLIB
*MDLPRF
*GENLIB
*CURLIB
library name
SRCLIB
*MDLPRF
*OBJLIB
*GENLIB
*CURLIB
library name
JOBD
*MDLPRF/
*MDLLIB/
*LIBL/
*CURLIB
library name
*MDLPRF
*MDLVAL
job description
JOBPTY
*JOBD
priority
CURLIB
*CURRENT
*CRTDFT
*USRPRF
INLLIBL
*JOBD
*CURRENT
MSGQ
*WRKSTN
*NONE
*USRPRF
*LIBL
library name/
message queue name
CRTJOBD
*JOBD
*NONE
*MDLPRF/
*LIBL/
*MDLLIB/
library name/
*MDLPRF
job description name
CRTJOBD
*CRTJOBD
priority
SBMGENOPT
*MDLPRF
*RLS
*HLD
SBMCRTOPT
*MDLPRF
*GENOK
*IMMED
GUIFLR
*MDLPRF
*GUIFLR
*MDLVAL
folder name



(1) *SELECT and (2) EDIT are valid only in an interactive environment.

Parameters

The following are parameters for the YSBMMDLCRT command.

JOBLST

Qualified name of a job list generated by CA 2E.Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***MDLLIB/*USER**—Default job list name to user profile name, and store list in model library.
- ***WRKSTN**—Default the job list name to job name of invoking job.
- ***SELECT**—Display a list of existing lists, one of which may be selected.

EDIT

Edit list option. Values for this parameter are described in the following:

- ***NO**—(default) Do not invoke list edit function.
- ***YES**—Invoke the list edit function before proceeding with compilations.

GENLIB

Library in which to place compiled objects. Values for this parameter are described in the following:

- ***MDLPRF**—The value is retrieved from the model profile details associated with the current user.
- ***GENLIB**—Retrieve the name of the generation library from the model value (YGENLIB).
- ***CURLIB**—Use the current library for invoking the job.

SRCLIB

This parameter specifies the library into which source is generated or which contains the source for a create object request. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***OBJLIB**—The library to use is the same one as that specified for the GENLIB parameter.
- ***GENLIB**—Retrieve the name of the generation library from the model value (YGENLIB).
- ***CURLIB**—Use current library for invoking job.
- **library name**—The library name for source.

JOBD

Qualified name of job description used when submitting job to carry out source generation. Values for this parameter are described in the following:

- *MDLPRF—(default) The value is retrieved from the model profile details associated with the current user.
- *MDLVAL—Retrieve the name of the job description from the CA 2E model value (YCRTJBD).
-

JOBPTY

Job scheduling priority; 1 (high) to 9 (low). The value for this parameter is as follows:

- ***JOBD**—(default) Use priority on job description.

CURLIB

This parameter specifies the current library for the submitted job. Values for this parameter are described in the following:

- ***CURRENT**—(default) The current value for the current library is used.
- ***CRTDFT**—No current library is specified.
- ***USRPRF**—The current library for the user profile is used.
- **library name**—The current library name.

INLLIBL

Initial library list for submitted jobs. Values for this parameter are described in the following:

- ***JOBD**—Use library list of job description specified by JOBD parameter.
- ***CURRENT**—Use library list of current job.

MSGQ

Qualified name of a message queue to which completion messages from submitted jobs are sent. Values for this parameter are described in the following:

- **WRKSTN**—(default) Send messages to the workstation message queue.
- **USRPRF**—Send messages to the user's default message queue.
- ***NONE**—No messages are sent.

CRTJOBD

Qualified name of job description used for compilations. Values for this parameter are described in the following:

- ***JOBD**—(default) Use the job description specified by the JOBD parameter.
- ***NONE**—No compilations are submitted and no objects are deleted.

CRTJOBPTY

Job scheduling priority for compilations; 1 (high) to 9 (low). The value for this parameter is as follows:

- ***CRTJOBD**—(default) Use priority on job description specified by CRTJOBD parameter.

SBMGENOPT

Option to specify whether or not the job containing the functions to be generated should be released after being submitted. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***RLS**—Release the job containing functions to be generated immediately.
- ***HLD**—Leave the job containing the functions to be generated as HELD on the job queue.

SBMCRTOPT

Option to specify when compilations should be submitted. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***GENOK**—Compilations of source members in the job list are submitted only after the source has been successfully generated.
- ***IMMED**—Compilations of source members are submitted immediately; that is, by the job which submits the job to generate the source.

GUIFLR

Name of folder to contain GUI implementation objects created from source generated by CA 2E. Values for this parameter are described in the following:

- ***MDLPRF**—(default) The value is retrieved from the model profile details associated with the current user.
- ***GUIFLR**—The name of the folder is retrieved from the model value (YGUIFLR).
- ***MDLVAL**—The name of the folder is retrieved from the model value (YGUIFLR).
- **folder name**—The folder name can be entered.

AUT

Authorization rights to be granted for compiled objects. Values for this parameter are described in the following:

- ***CHANGE**—(default) Grant operational rights and all data rights.
- ***USE**—Grant operational rights and read data rights.
- ***EXCLUDE**—Do not grant access rights.
- ***ALL**—Grant all rights.

REPLACE

Option to use the REPLACE option on the appropriate i OS CRTxxxPGM command (only applies to programs). Only applies if you are creating IBM i native objects; that is, if the model value YCRTENV of your model is QCMD. Values for this parameter are described in the following:

- ***NO**—(default) Do not use the REPLACE option on the CRT*xxx*PGM command being used to create the program. If the program already exists, CA 2E deletes it before the compilation is submitted. Hence if the compilation fails, no version of the program will exist.
- ***YES**—Specify REPLACE(*YES) on the i OS CRT*xxx*PGM command being used to create the program. If the program already exists, CA 2E will not delete it until the new version has been successfully compiled. If the compilation is not successful, the old version of the program will still exist.

FLDREF

Name of field reference file member. If a member of the specified name is found in the member list, it is compiled before all other source members. Values for this parameter are described in the following:

- ***FRFVNM**—Retrieve the name of the field reference file from the model value (YFRFVNM).
- ***NONE**—No field reference file member is compiled.

CPYLIB

Name of library from which to copy data to add to new versions of existing physical files. Values for this parameter are described in the following:

- ***NONE**—(default) Data is not copied for existing files.
- ***CPYLIB**—Retrieve the name of the data library from the model value (YCPYLIB).

OLDLIB

Library into which previous versions of physical files are placed. When a new version of an i OS object is created, the old version from the nominated GENLIB is placed in the given OLDLIB. Any previous old version is deleted. Values for this parameter are described in the following:

- ***NONE**—(default) Old versions of the physical files are not saved.
- ***OLDLIB**—Retrieve the name of the data library from the model value (YOLDLIB).

Notes

The Job List

The job list must already exist. It can be created either from within a model, or by using the Build Job List (YBLDJOBLST) command.

A single job will be submitted to carry out any source generation. Each compilation of a source member will normally be submitted as a separate job. You can specify whether the job is to be submitted immediately, or after the generation has completed successfully.

For each source member named in a job list, the YSBMMDLCRT command can invoke up to three different processing steps:

- Generation of source
- Compilation of source into an object
- For physical files only, back copying of data from an existing version of the file into the newly compiled file

Generation of source

A single job will be submitted to carry out source generation.

The job will be released by default SBMGENOPT(*RLS) unless SBMGENOPT(*HLD) is specified.

If you have specified that a field dictionary be used; that is, the YFRFVNM model value is other than *NONE, source for the field reference file will be regenerated before the compilation list is shown.

Compilation of Source Into an Object

Source members will be processed in the correct order for compilation:

- Field reference physical file
- Physical files
- Logical files
- Display files
- Programs

Each compilation will be submitted as a separate job. The compilations can be submitted in two ways:

- SBMCRTOPT(*GENOK)—The job which generates the source will submit the jobs to compile the generated objects, provided the generation is successful. The submitting of the compilations is thus a batch process.
- SBMCRTOPT(*IMMED)—The job which submits the source generation will also submit the compilations. Thus, the submitting of compilations will usually be interactive. Compilations for objects requiring source generation will be submitted as held jobs. If an error occurs when generating a source member, the corresponding compilation job will be canceled by the source generation job. If a compilation is canceled, this is recorded as an error on the job list.

Completion messages are returned to the message queue specified by the MSGQ parameter.

All the elements of a function or an access path (QRY access paths only) must be generated successfully before compilation is attempted. Thus, if any part fails to generate, no part will be submitted for compilation if SBMCRTOPT (*GENOK), or the compilations for all parts will be canceled if SBMCRTOPT(*IMMED).

If CRTJOBD(*NONE) is specified, no objects will be deleted, and no compilations will be submitted.

Back copying of data

Physical files that are recompiled will be archived if a value is specified for the OLDLIB parameter.

Data is copied into physical files that have been recompiled if a value is specified for the CPYLIB parameter. If CPYLIB is the same as OLDLIB, the YSBMMDLCRT command copies the original data back into the physical files as far as possible.

When CRTJOBD(*NONE) is specified, no data will be copied and no files will be archived.

Example

To generate and compile all the source members in your job list and to then submit the compilations after the generation is complete:

```
YSBMMDLCRT
```

YSETCPYNME (Set Model Object Copy Name) Command

This command examines objects in the CA 2E data model and, depending on whether the object is versionable, either checks or resets the Copy name in the object's model object description.

The Copy name field stored on the Model Object file is used by the Copy Model Objects (YCPYMDLOBJ) command to determine object matches between the source and target models. You can use the Copy name field to force YCPYMDLOBJ to match Object A in the source model with Object B in the target model. The YSETCPYNME command allows all objects in the model to be reset back so that an object's Copy name is equal to its Object name. A report documents changes made by the program for non-versionable objects and lists warnings for versionable objects.

You must have *DSNR authority to use this command.

Notes

- For versionable objects (functions and messages), the Copy name of all members of the group of versions must be the same. This is because all versions in the group should match with the same object in a target model, otherwise, the concept of versions does not exist. A copy of any one of the versions in the group using YCPYMDLOBJ should match with the same object in the target model.

  The Copy name of each versionable object is validated against all other versions in that object's group. If it differs from any of the other versions a report notifies you that the group is not set up correctly. To recover, you must adjust the Copy name of one of the versions with the Change Model Object Description (YCHGMDLOD) command, which updates all versions in the group to have the same Copy name.
- For non-versionable objects, such as access paths, fields, and conditions the program compares the Copy name with the Object name of the object. If they differ, the Copy name is set to be equal to the Object name and the update action is reported.

Example

To check and/or change the copy names associated with model objects contained in model library HSTMDL:

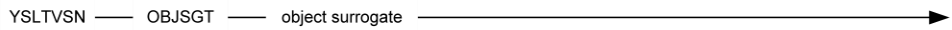
```
YSETCPYNME  MDLLIB( HSTMDL )
```

Deleted GENFLR, OS2NODE, WINFLR, UNXDIR, UNXNDE

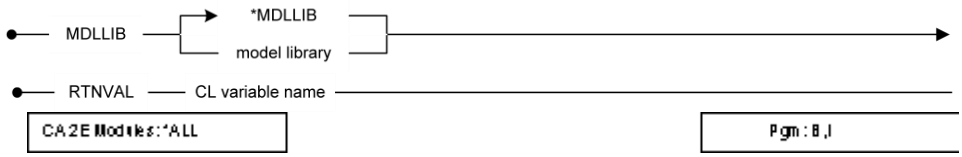
YSLTVSN (Select Version) Command

This command provides an interactive panel for the selection of a version from a group of versions.

Required



Optional



Parameters

The following are parameters for the YSLTVSN command.

OBJSGT

The object surrogate number of the object from whose group the version is to be selected.

This is a required parameter.

RTNVAL

The CL variable that receives the object surrogate number of the selected model object version. The value for this parameter is as follows:

- **CL variable name**—Numeric variable, seven bytes long.

Example

To select a version from the group to which the object identified by object surrogate number 1102597 belongs:

```
YSLTVSN  OBJSGT( 1102597 )
```

This command provides a return parameter containing the object surrogate number identifier of the selected version. It must, therefore, be called from an interactive CL program.

YSNCMDL (Synchronize Model) Command

This command allows the user to synchronize a model outside of the model environment.

Required

Parameters

The following are parameters for the YSNCMDL command.

MDLLIB

The name of the model to be synchronized. Values for this parameter are described in the following:

- ***MDLLIB**—(default) Special value indicating the model library in the current library list.
- ***CURLIB**—(default) Special value indicating the current library of the job.
- **model name**—The name of the model to be synchronized.

Notes

Each CA 2E file in the model specified is expanded and the model synchronization flag is set.

Example

To synchronize the model Y2PGMR enter the following:

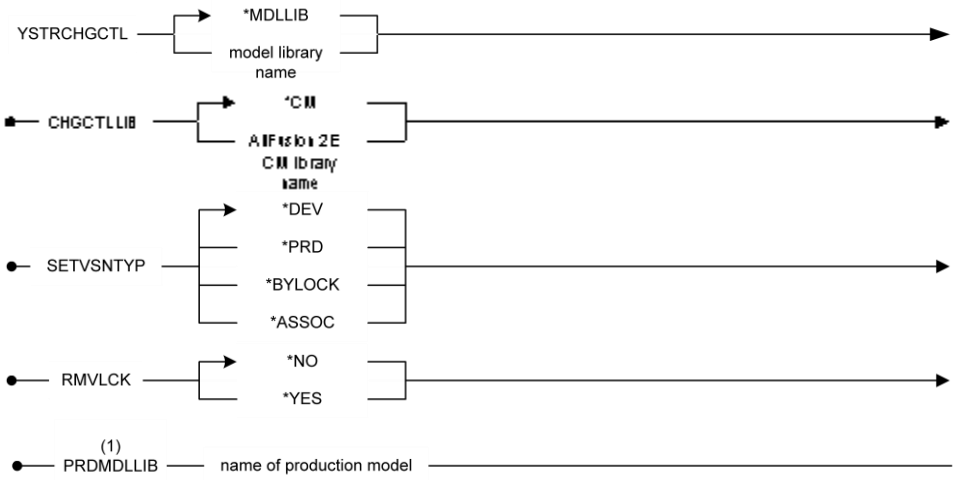
```
YSNCMDL MDLLIB(Y2PGMR)
```

YSTRCHGCTL (Start Change Control) Command

This command sets up Synon/CM for the model library in the current job's library list. If there is an existing change control system in place, it is possible to access those details during processing so that the initial starting position for a particular model object can be determined.

Administrators are advised to examine the options of this command and to plan its use carefully as it may be important to establish the starting position of model objects correctly.

Required



(1) PRDMDLLIB is prompted when SETVSNTYP is set to *ASSOC.

Parameters

The following are parameters for the YSTRCHGCTL command.

MDLLIB

The name of the model library in which change control is to be started. Values for this parameter are described in the following:

- ***MDDLIB**—(default) Use the first model library in the job's current library list.
- **model library**—The name of the model library to use.

CHGCTLLIB

This parameter specifies the name of the Synon/CM product library installed on your machine. Values for this parameter are described in the following:

- ***CM**—(default) Use default name (Y2SYCM).
- **Synon/CM library**—Name of Synon/CM library.

SETVSNTYP

This parameter specifies the starting version type for model objects. Values for this parameter are described in the following:

- ***DEV**—(default) All model objects are initialized to a version type of DEV. This indicates that the model is a development model.
- ***PRD**—All model objects are initialized to a version type of PRD. This indicates that the model is a production model.
- ***BYLOCK**—Normally models will consist of a combination of development and production objects. A common way of differentiating between these two types is to place a permanent lock on production objects inside the model (lock type *PERM). This parameter value will examine the lock status of each object. If a permanent lock exists for an object, the version type for that object is set to *PRD, otherwise it is set to *DEV.
- ***ASSOC**—Use this value to invoke the Associate Production Model (YASCPRDMDL) command. You will be prompted to enter the name of the target production model library. See the PRDMDLLIB parameter below.

RMVLCK

This parameter specifies the action to take for locks within the model. Values for this parameter are described in the following:

- ***NO**—(default) Model locks are not deleted.
- ***YES**—As part of processing, if a permanent lock is detected for an object, it is deleted.

PRDMDLLIB

The name of the production model that is to be used for comparing model objects using the Associate Production Model (YASCPRDMDL) command. See the command diagram or the online Help for more details relating to this command.

This parameter is prompted only if *ASSOC is specified for the Set Version Type parameter (SETVSNTYP). The value for this parameter is as follows:

- **model library name**—The name of the model library. You must specify this name explicitly.

Notes

Regardless of the value specified for the SETVSNTYP parameter, you can set the version type in an exit program called YSTRCHGR1C. The interface for this program is as follows, and the source is shipped in QCLSRC in library Y2SYSRC:

```
Parameters :                Object surrogate     :              7 decimal
                            Derived version type :              3 character
                            Return version type  :              3 character
```

The derived version type is arrived at during processing dependent upon the value specified for SETVSNTYP. However, if a valid value is returned by the exit program, it will be used in preference. Developers may take this opportunity to examine details of the object, identified by its surrogate number key (in an external database, for example).

The CA 2E Change Management product is required to run with Change Management.

See your CA 2E Change Management documentation for more information.

Example

To start change control in model library J4DEVMDL, setting all objects to *DEV and removing any permanent locks:

```
YSTRCHGCTL SETVSNTYP(*DEV) RMVLCK(*YES )
```

YSTRTRGSVR (Start Trigger Server) Command

The YSTRTRGSVR command allows you to start one or more Trigger Server jobs to monitor a Trigger Data Queue. Trigger Data Queues are used to provide asynchronous trigger support, allowing Trigger Functions to be run within a separate job to the one that actually caused the trigger to fire.

When a trigger fires due to a database file change and the trigger program is specified as the CA 2E Trigger Router, the Trigger Router will check in the Trigger Control file for any Trigger Functions specified for the trigger. If any are found, they will either be called directly or an entry will be added to the Trigger Data Queue, containing all the data related to the trigger. This entry will then be processed by a Trigger Server job, which will itself call the Trigger Function directly.

This allows potentially long-running Trigger Function processing to be 'off-loaded' to another job from the one that made the database change which caused the trigger to be fired, thus ensuring that the application program can continue without having to wait for all processing to finish.

Some examples of long-running processing contained in a Trigger Function which you may want to be processed asynchronously are:

- Processing which involves printing
- Processing which involves possible object allocation contention
- Processing which involves accessing resources external to the iSeries on which the database transaction is occurring
- Processing which involves the invoking of a web service or similar function

When a Trigger Function is processed asynchronously, there is no way for the Trigger Function to signal to IBM i or to the application program if errors occur. Thus the asynchronous processing of Trigger Functions is not recommended where errors in the Trigger Function would require the database change to be canceled.

Note: Trigger Data Queues are always called YTRIGGERQ. If a Trigger Data Queue does not exist in the library specified in the TRGLIB parameter, one will be created. Trigger Data Queues are unkeyed data queues with a maximum entry length of 32859.

Parameters

The following are parameters for the YSTRTRGSVR command.

TRGLIB

Specifies the name of the library containing the Trigger Data Queue. If a Trigger Data Queue does not exist in the specified library, one is created.

The values are:

***TRGLIB**

The library containing the Trigger Server program is used.

library-name

Specify the library name which contains the Trigger Data Queue to be monitored.

JOBD

Specifies the job description that should be used for the Trigger Server job or jobs.

Note: A Trigger Server runs as a continuous batch process until you end it with the End Trigger Server (YENDTRGSVR) command. Due to this, the processor overrides your job description with the system-generated JOBQ(QSYS/QSYSNOMAX). This ensures that the Trigger Server is submitted using a job queue that which multiple active jobs. All other job definition attributes are taken from the job description specified in this parameter.

The values are:

***USRPRF**

The trigger server job uses the job description for the user profile used by the job that is currently running.

job-description-name

Specify the name (library-name/job-description-name) of the job description used for the trigger server job.

NBRSVR

Specifies the number of Trigger Server jobs that should be started by this command. All Trigger Server jobs use the same job description, as specified in the JOBD parameter. All jobs monitor the same Trigger Data Queue in the library you specify in the TRGLIB parameter.

The Trigger Data Queue is a FIFO (first-in, first-out) data queue. If multiple Trigger Server jobs are running concurrently, each trigger request is selected from the Trigger Data Queue by the first available Trigger Server job. There is no guarantee of the trigger request processing order, since this depends on many factors affecting the speed at which each Trigger Server job runs. If you must process trigger requests in the same order as the original trigger fired, you have two options:

- Process the trigger requests synchronously as a direct call by the Trigger Router
- Ensure that only a single Trigger Server job monitors a specified Trigger Data Queue.

Note: Running more than one Trigger Server job concurrently can improve system performance where many asynchronous trigger requests may appear at once. However, it will not affect the performance of the job in which the trigger was fired.

Values are:

***DFT**

A single Trigger Server job is started to monitor the Trigger Data Queue in the library specified in the TRGLIB parameter.

***MAX**

9 Trigger Server jobs are started to monitor the Trigger Data Queue in the library specified in the TRGLIB parameter. The maximum number of Trigger Server jobs that can be started is 99, but we retained the value for this parameter as 9 to preserve existing functionality.

number-of-trigger-server-jobs

Between 1 and 99 Trigger Server jobs can be started to monitor the Trigger Data Queue in the library specified in the TRGLIB parameter.

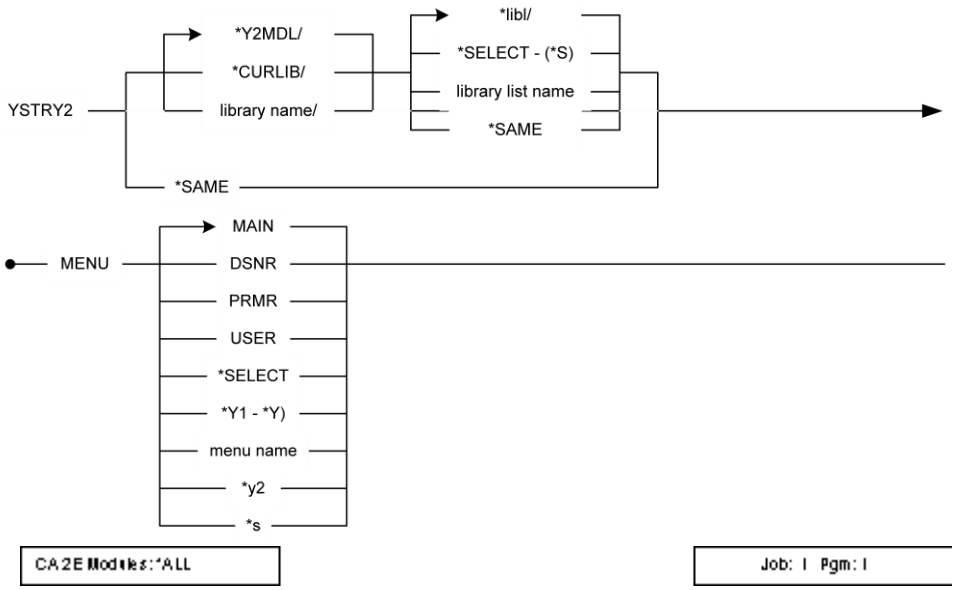
CLEAR

Specifies whether data should be cleared from the Trigger Data Queue prior to starting the Trigger Server job or jobs.

YSTRY2 (Start CA 2E) Command

Sets library list for CA 2E and displays the CA 2E menus.

Optional



Parameters

The following are parameters for the YSTRY2 command.

LIBLST

Name of library list used to replace the current job's library list before displaying the help menus. When you create a model using the command Create Model Library **(**YCRTMDLLIB**)**, an appropriate library list to allow you to use the model is created automatically. By default it will have the same name as the model. The library list is stored using the CA 2E Toolkit library list facilities. Values for this parameter are described in the following:

- ***Y2MDL**—(default) CA 2E model library list name.
- ***SELECT**—Display library lists in the specified library, one of which may be selected.
- ***SAME**—Do not change the current library list.

MENU

Name of first menu you want to display. Values for this parameter are described in the following:

- **MAIN**—(default) Display master menu.
- **DSNR**—Display menu for designers.
- **PGMR**—Display menu for programmers.
- **USER**—Display menu for users.
- ***SELECT**—Display a list of help menus, one of which may be selected.
- ***Y1**—Display CA 2E Toolkit master menu.

Notes

The library list of the job is replaced with the specified list.

Additional library lists can be created using the Synon/1E command Build Library List **(**YBLDLIBLST**)**. Existing library lists can be altered using the Synon/1E command Edit Library List **(**YEDTLIBLST**)**.

Example

To enter CA 2E and edit model MYMDL:

```
YSTRY2 LIBLST(MYMDL)
```

YUNSWS (Uninstall Web Service Instance) Command

Uninstalls a web service instance. The instance can be deleted from within a CA 2E model and/or uninstalled from an application server.

Notes:

- If the command fails, detailed errors will be written to the YQSHLOG log file in library QTEMP.
- This command requires the user issuing this command to have special authorities *ALLOBJ and *IOSYSCFG. This is due to the underlying IBM IWS scripts requiring those special authorities.

Required

SERVER

SERVICE

Optional

MACHINE

UPDMDL

UNINSTALL

STPSRV

Parameters

The following are parameters for the YUNSWS command.

MACHINE

Specifies the name of the machine onto which the web service instance to be uninstalled.

***CURRENT**

Refers to the local machine.

name

Specify the machine name. This can be the local machine or a remote machine. The machine name is not validated and the machine need not exist on the local, or any, network.

UPDMDL

Determines whether the CA 2E model information should be updated.

***UPDINSSTS**

The installed status of the web service instance, defined within the model, is updated.

***DELETE**

The web service instance definition is deleted from the model.

***NO**

The web service instance definition is not deleted from the model.

UNINSTALL

Determines whether the web service instance is uninstalled from the application server.

***NO**

The web service instance is not uninstalled from the application server.

***YES**

The web service instance is uninstalled from the application server.

Note: A Web Service instance can only be uninstalled from an application server located on the local machine.

SERVER

Specifies the web service name to be uninstalled.

name

Specify the web service name.

SERVICE

Specifies the web service name to be uninstalled.

name

Specify the web service name.

STPSRV

Is an indication as to whether the service should be stopped before an uninstall.

***YES**

If the service is running, it will be stopped before uninstall.

***NO**

The service will not be stopped before uninstall. An error will be returned if the service is active.

YWRKDSTFIL (Work Distributed Files) Command

Allows the user to specify for each distributed file the names of databases to which it applies plus the default sequence of databases by which to access it. This command invokes the distributed file configuration table editor which allows users to tailor the configuration of RDBs for tables and views. This RDB configuration information is used by distributed applications generated from the model. This application is shipped in the null model and must be generated by the users.

Optional

Parameters

The following are parameters for the YWRKDSTFIL command.

FILTYP

Type of access path that is subsetted on the Work With Distributed Files panel. Values for this parameter are described in the following:

- **TABLE**—(default) Subset to show only tables.
- **VIEW**—Subset to show only views.
- ***ALL**—Do not subset. Show all tables and views.

Notes

The YWRKDSTFIL command invokes the program Y2DSTFR (Work With Distributed Files) which in turn will invoke Y2CFGTR (Work With Configuration Table Entries), when option 5 is selected. These programs allow a user to tailor the RDBs for tables and views that will be accessed by distributed applications.

The command and the associated programs are application objects that form part of the objects created when the command YDUPAPPOBJ is run.

For more information about YDUPAPPOBJ, refer to the Duplicate Application Objects (YDUPAPPOBJ) command.

Example

To work with distributed files, subsetting to show only tables:

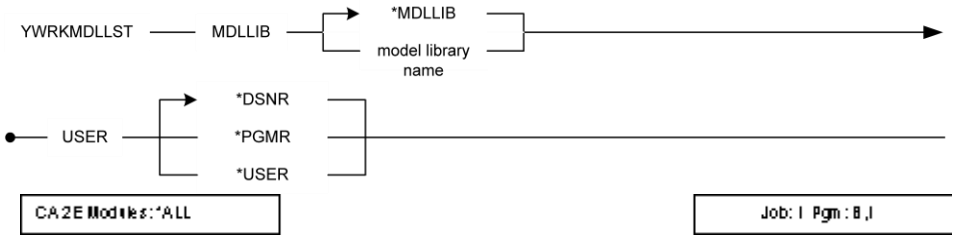
```
YWRKDSTFIL FILTYP (TABLE)
```

YWRKMDLLST (Work with Model Object Lists) Command

This command provides access to an interactive panel displaying all model object lists that exist in the specified model.

The work with panel provides a number of options to manipulate a model object list. For more information on the available options, please refer to the help text for the panel.

Required



Parameters

The following are parameters for the YWRKMDLLST command.

MDLLIB

The name of the model library in which the lists reside. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model library to use is the first library in the current job's library list.
- **model library name**—The name of the model.

USER

Type of user. Must be one of the following: Values for this parameter are described in the following:

- ***DSNR**—(default) A designer may change any aspect of the model, including the database.
- ***PGMR**—A programmer can add or change any functions that are in the model, but may not alter the relations, files or database fields.
- ***USER**—A user may view all aspects of the model but may not change any design objects. This class of user is useful to allow the data model to be examined without the possibility of change.

Notes

A value other than *MDLLIB for MDLLST may result in the library list being changed. If the user is currently editing a model, the switching of the library list will not occur and the command will fail. If changed during processing, the library list is changed back after execution.

Example

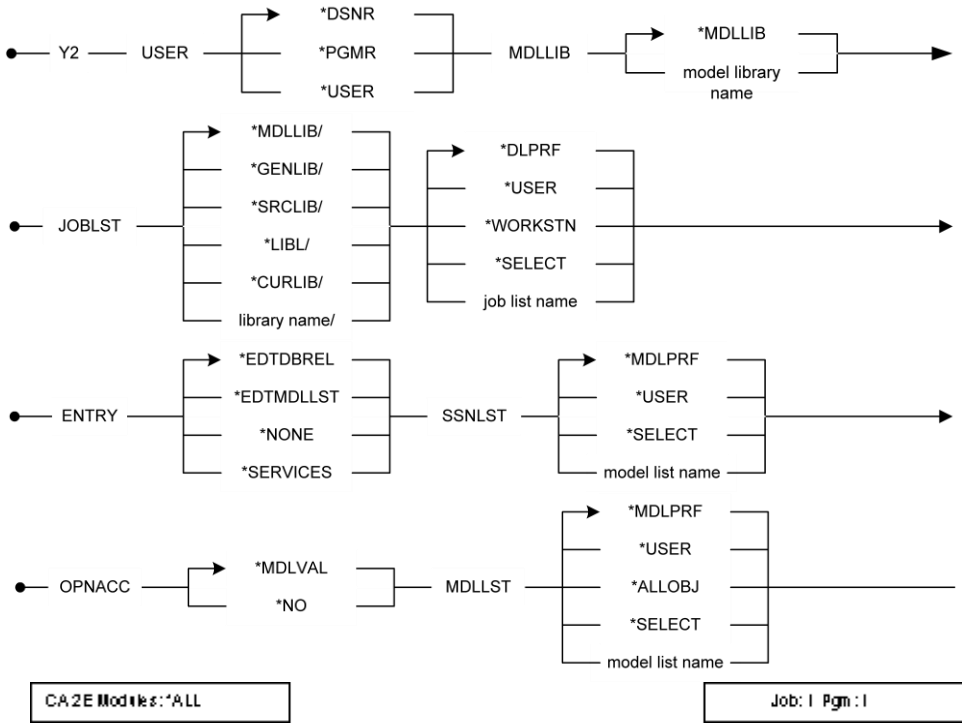
To work with model object lists in model library ATSTMDL, enter the following command:

```
YWRKMDLLST MDLLIB(ATSTMDL)
```

Y2 (Edit Model) Command

Calls an interactive program to enter and edit the first CA 2E model in your library list. Y2 is the short form of YEDTMDL.

Optional



Parameters

The following are parameters for the Y2 command.

USER

Type of user. Values for this parameter are described in the following:

- ***DSNR**—(default) A designer can change any aspect of the model, including the database.
- ***PGMR**—A programmer can add or change any functions that are in the model, but cannot alter the relations, files or fields.
- ***USER**—A user can view all aspects of the model but cannot change any design objects. This class of user is useful to allow the data model to be examined without the possibility of change.

MDLLIB

This parameter specifies the data model that is edited. Values for this parameter are described in the following:

- ***MDLLIB**—(default) The model to be edited is the first one found in the current job's library list.
- **model name**—The model library name.

JOBLST

Qualified name of job list that contains the names of source members to be generated and/or compiled. If the nominated job list does not already exist, it will be created. Values for this parameter are described in the following:

- ***MDLPRF**—(default) Retrieve the job list name from the model profile details of the current user.
- ***USER**—The job list name is the same name as the current user.
- ***WORKSTN**—Use device name of current workstation as list name.
- ***SELECT**—Display a list of existing job lists, one of which may be selected.
- **job list name**—The name of the job list can be entered.
- ***MDLLIB/**—The job list library is the first model library found in the library list.
- ***GENLIB/**—Use the generation library specified in the first model found in the library list.
- ***SRCLIB/**—Use the source library specified on the model profile of the current user.
- ***LIBL**/—The job list library is the first model found in the library list
- ***CURLIB/**—The model library is found in the current library for the current job.
- **job list library**—The job list library can be entered.

ENTRY

This parameter provides the user with multiple entry points to the specified model. Values for this parameter are described in the following:

- ***EDTDBREL**—(default) The first panel to be accessed is the Edit Database Relations panel.
- ***EDTMDLLST**—The mode of entry to the model is via the Edit Model List panel. If this value is specified, the MDLLST parameter specifies the model list that is edited.
- ***SERVICES**—The services menu is the first panel to be accessed.
- ***NONE**—This option can be used to establish a model environment but without any particular entry to the model. In this case the model environment is started and the developer is presented with the Command Entry panel. Numerous commands require the model environment to be active and will check to ensure that it is active when invoked. These commands will adopt an already active environment. Thus, if a series of commands are run, it will be more efficient to use this option before executing such commands. Another advantage of this option is that the lock applied to the model is established for the entire session, preventing interference by another developer.

SSNLST

This parameter specifies the session list to use while editing the model.

For more information:on the purpose of SSNLST, refer to the Change Model Profile command (YCHGMDLPRF). Values for this parameter are described in the following:

- ***MDLPRF**—(default) The session list is retrieved from the model profile details for the current user.
- ***USER**—The session list has the same name as the current user.
- ***SELECT**—An interactive display is used to select a model object list to be used as the session list.
- **model list name**—The name of the list can be entered.

OPNACC

This parameter enables the current user to override to *NO (if authorized) the Open Access model value. The intention with this parameter is to provide a *DSNR with the opportunity to gain exclusive access to the data model.

For more information:on Open Access, see the Change Model Value command (YCHGMDLVAL) in this chapter and the "Creating and Managing Your Model" chapter in the CA 2E *Administrator Guide*. Values for this parameter are described in the following:

- ***MDLVAL**—(default) Access to the model is controlled by the current model value for Open Access.
- ***NO**—This value can be used to set the Open Access model value to exclude any concurrent *DSNR activity in the model and to exclude users of any other class.

MDLLST

The qualified name of the model object list that is edited. Values for this parameter are described in the following:

- ***USER**—The list to be edited has the same name as the current user.
- ***ALLOBJ**—The all objects list is edited.
- ***SELECT**—Special value indicating that the model object list is selected using an interactive display function.

Notes

- The model can either be set up to allow concurrent *DSNRs and *PGMRs/*USERs into the model by setting the model value YOPNACC to *YES, or the model value can be left as *NO where either a single *DNSR or multiple *PGMRs/*USERs can be in the model concurrently.
- To add and remove permanent locks, or to change the Open Access (YOPNACC), you must have all rights to the data model. (Designer with locks capability).
- To edit a model as a user of type *DSNR, you must have at least all rights except for existence to the model. Generally the most convenient way to arrange this is to grant all rights to all the objects in the model library (the default), and then to control access to the model by granting or revoking rights to use the data area YMDLLIBRFA in the model library. The YEDTMDL command checks the user's authority to this data area before allowing entry to the model.
- For example, to revoke all rights to user profile IVAN to edit or view a model MYMDL:

RVKOBJAUT OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA) USER (IVAN) AUT(*ALL)

- Or to grant user profile IVAN rights to an edit model MYMDL as designer with lock capability:

GRTOBJAUT OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA) USER(IVAN) AUT(*ALL)

- To override the model value YOPNACC (represented in the command by OPNACC) by specifying *NO, you must have all rights to the data area YOPNACCRFA. Access rights to YOPNACCRFA can be assigned or revoked in the same way as above.

  Should you wish to change the YOPNACC value temporarily just for the duration of the session, you must synchronize the model on exit.

Examples

To edit a model as a designer:

```
YEDTMDL
```

To edit a model as a programmer, using a list named after the current device name:

```
YEDTMDL USER(*PGMR) JOBLST(*WORKSTN)
```

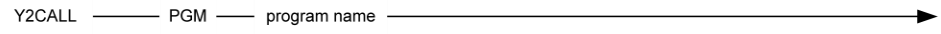
Y2CALL (Call a Program) Command

This command determines the parameters required by a model function from details contained in the data model. Each field, comprising a parameter or part of a parameter structure, is processed and optionally presented to the user for preloading of parameter values.

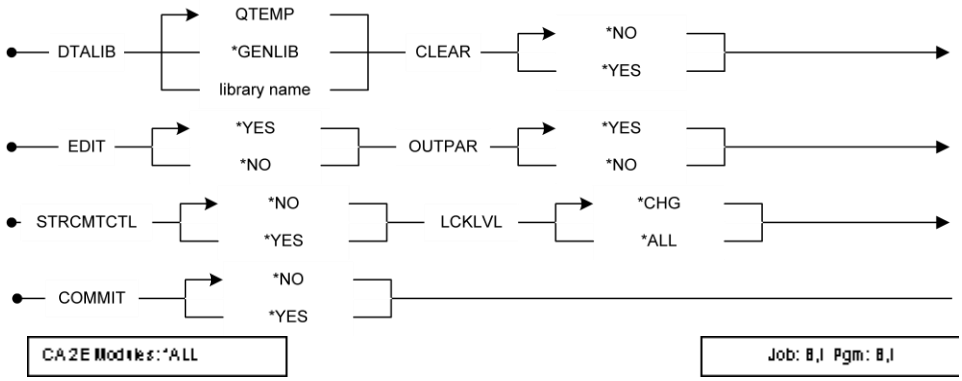
The purpose of this command is to allow developers to test low level functions requiring numerous parameters without the need to code layers of intervening programs which construct those parameters. Values can be supplied for all input capable fields and can be reused for subsequent invocations.

Note: This command does not support function calls that contain multiple-instance array parameters.

Required



Optional



Parameters

The following are parameters for the Y2CALL command.

PGM

The name of the program to be called. This is the same as the source member name of the function in the data model.

DTALIB

The library name in which parameter values are stored. Values for this parameter are described in the following:

- **QTEMP**—(default) Use temporary library QTEMP to store parameter values.
- ***GENLIB**—The generation library is used to store parameter values.
- **library name**—Another library name can be entered.

CLEAR

Indicates whether the parameter values are cleared before editing or the call to the program. Values for this parameter are described in the following:

- ***NO**—(default) Existing values for parameters are not cleared.
- ***YES**—Parameter values are cleared.

EDIT

Indicates whether the developer wishes to edit the parameters prior to the call. Values for this parameter are described in the following:

- ***YES**—(default) Parameter entries are edited.
- ***NO**—Parameters are not edited prior to the call.

OUTPAR

Specifies whether output parameters defined in the called program, should be retrieved when it terminates. The values from the output parameters will be displayed on the edit screen. Values for this parameter are described in the following:

- ***YES**—(default) Output parameters are retrieved when the called program terminates.

  Note: If output parameters are retrieved, the called program will be placed in debug. This is because the retrieval of the parameters is done using debug functions.
- *NO—Output parameters are not retrieved.

Note: Output parameters cannot be retrieved for EXCUSRPGMs or action bars. They also cannot be retrieved if EDIT is *NO.

STRCMTCTL

This parameter specifies whether commitment control is started before the call. This may be necessary for a function that normally would operate as a *SLAVE. Values for this parameter are described in the following:

- ***NO**—(default) Commitment control is not started.
- ***YES**—Commitment control is started prior to the call.

COMMIT

Indicates whether a commit is executed following the call. This parameter is only shown if *YES is specified for the STRCMTCTL parameter. Values for this parameter are described in the following:

- ***NO**—(default) A commit is not issued.
- ***YES**—A commit is issued following the call.

LCKLVL

The level of locking to be applied when performing I/O. This parameter is only shown if *YES is specified for the STRCMTCTL parameter. Values for this parameter are described in the following:

- ***CHG**—(default) Only changed records are locked.
- ***ALL**—All records read on update access paths are locked.

Notes

- If the program name exists in the model, the parameters defined for the function are displayed if EDIT(*YES) is specified.
- The program object must exist prior to the call.
- If parameter values are not defined prior to the call, character parameters will default to blanks and numeric parameters to zero.
- If commitment control is started as part of the processing, it is also ended before the command processing programs return.
- Users should ensure that parameter details are defined correctly in the model. This is especially true of Execute User Program functions (EXCUSRPGM), particularly when they are the first program from a menu option, when it is not necessary to define parameters in the model at all.

Example

To call the model function Prc Allocate Stock Item, clearing any existing parameter values and to edit the parameters:

```
Y2CALL PGM(UUAKEXR) CLEAR(*YES) EDIT(*YES)
```

Appendix A: Appendix A: Commands Grouped by Functional Area

The following shows CA 2E commands grouped by functional area.

Upgrade Commands

- **YAPYMDLCHG**—Apply Model Changes
- **YAPYSYSMDL**—Apply Model System Objects Changes

Model Setup Commands

- **YAPYTRNMDL**—Apply Translation to Model
- **YCHGMDLPRF**—Change Model Profile details
- **YCHGMDLVAL**—Change Model Values
- **YCLRMDL**—Clear a Model
- **YCRTMDLLIB**—Create Model Library
- **YCRTSQLLIB**—Create SQL Library
- **YCVTMDLVNM**—Convert Model Names
- **YDSPMDLVAL**—Display Model Value
- **YEDTDFTATR**—Edit Default Display Attributes
- **YEXCSQL**—Execute SQL Statements
- **YRGZMDL**—Reorganize a Model
- **YRNMMDL**—Rename a Model
- **YRTVMDLPRF**—Retrieve Model Profile details
- **YRTVMDLVAL**—Retrieves a Model Value
- **YRTVPFMDL**—Retrieve Physical Files into Model
- **YSNCMDL**—Synchronize Model
- **YSTRCHGCTL**—Start Change Control

Model Objects Commands

- **YCHGMDLOBJ**—Change Model Object
- **YCHGMDLOD**—Change Model Object Description
- **YCHKMDLOBJ**—Check existence of Model Object
- **YCMPMDLOBJ**—Compare Model Objects
- **YCRTMDLVSN**—Create Model Version
- **YCRTOBJVSN**—Create Model Version
- **YDLTMDLVSN**—Delete Model Version
- **YDSPMDLOD**—Display Model Object Description
- **YRDRMDLOBJ**—Redirect Model Object
- **YSLTVSN**—Select Model Object Version
- **YRTVMDLOBJ**—Retrieve Object Description

Edit Commands

- **YEDTMDL**—Enter/Edit a Model
- **YEDTMDLLST**—Edit Model Object List
- **YEDTMDLPRF**—Edit Model Profile
- **YSTRY2**—Start CA 2E
- **Y2**—Edit Model

Create Application Commands

- **YBLDJOBLST**—Build Job List
- **YCHKJOBLE**—Check Job List Entries
- **YCRTGENOBJ**—Create Generation Objects
- **YCRTJOBLE**—Create Job List Entry
- **YCVTCNDVAL**—Convert Condition Values
- **YCVTJOBLST**—Convert Job List to CA 2E Toolkit Object List
- **YCVTMDLLST**—Convert a Model Object List to a Job List

Model Object Lists Commands

- **YADDMDLLE**—Add a Model Object List Entry
- **YBLDMDLLST**—Build a Model Object List
- **YCHGMDLLE**—Change a Model Object List Entry
- **YCHKMDLLST**—Check a Model Object List
- **YCLRMDLLST**—Clear a Model Object List
- **YDLTMDLLE**—Delete a Model Object List Entry
- **YDLTMDLLST**—Delete a Model Object List
- **YDSPMDLLST**—Display a Model Object List
- **YEXCMDLLST**—Execute a Model Object List
- **YFLTMDLLST**—Filter a Model Object List
- **YINXMDLLST**—Index a Model List
- **YOPRMDLLST**—Operate on a Model Object List
- **YWRKMDLLST**—Work with Model Object Lists

Document Commands

- **YDOCMDLACP**—Document Model Access Paths
- **YDOCMDLAPP**—Document Model Application Areas
- **YDOCMDLF**—Document Model Files
- **YDOCMDLFLD**—Document Model Fields
- **YDOCMDLFUN**—Document Model Functions
- **YDOCMDLLST**—Document a Model Object List
- **YDOCMDLMSG**—Document Model Messages
- **YDOCMDLREL**—Document Model Relations
- **YDOCURF**—Document Unreferenced Objects

Copy Commands

- **YCPYMDL**—Copy a Model
- **YCPYMDLLST**—Copy Model Object List
- **YCPYMDLOBJ**—Copy Model Objects
- **YEDTCPYLST**—Edit a Model Object List for Copy

Miscellaneous Commands

- **YAPYCMPCHG**—Apply Component Changes
- **YCHKMDL**—Check Model
- **YCHKDTAMDL**—Check Data Model
- **YCHKFUNACT**—Check Function Action Diagram
- **YCVTDSTFIL**—Convert Distributed Files
- **YCVTMDLPNL**—Convert Model Panels to Prototypes
- **YCVTTMUIM**—Convert Help Text to UIM
- **YDSPMDLREF**—Display Model References
- **YDSPMDLUSG**—Display Model Usages
- **YEDTNXTMNC**—Edit Next Autoname Type Mnemonic
- **YSETCPYNME**—Set Model Object Copy Name
- **YWRKDSTFIL**—Work Distributed Files
- **Y2CALL**—Call a Program

Index

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Y